National College of Ireland

COMPUTATION OF NUMBERS USING HOMOMORPHIC ENCRYPTION

MSc Research Project

MSc in Cyber Security

Sunil Gangula

X20189800

School of Computing

National College of Ireland

Supervisor: MICHAEL PRIOR

National College of Ireland

Project Submission Sheet – 2021/2022

Student Name: ……SUNIL GANGULA..……

Student ID: ……X20189800……

Programme: ……MSC IN CYBERSECURITY…… **Year:** ……2022……

Module: ……MSC RESEARCH PROJECT…..……

Lecturer: ……MICHAEL PRIOR……

Submission Due Date: ……19-09-2022……

Project Title: COMPUTATION OF NUMBERS USING HOMOMORPHIC ENCRYPTION

Word Count: 7480

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the references section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature: ……SUNIL G……

Date: ……18-09-2022……

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. Projects should be submitted to your Programme Coordinator.
3. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
4. You must ensure that all projects are submitted to your Programme Coordinator on or before the required submission date. **Late submissions will incur penalties.**
5. All projects must be submitted and passed in order to successfully complete the year. **Any project/assignment not submitted will be marked as a fail.**

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

COMPUTATION OF NUMBERS USING HOMOMORPHIC ENCRYPTION

SUNIL GANGULA
X20189800

Abstract

The fundamentals of homomorphic encryption has recently received wide acceptance in the domain of encryption. The idea of being able to perform various mathematical operations on arithmetic data and later encrypting it with generated keys in such a manner that only a legit user can access to the encrypted data has been fascinating to the research scholars. Therefore, the presented thesis utilizes the homomorphic scheme and proposes a method to perform mathematical operations such as addition and multiplication on numbers being provided by the user. The process includes obtaining a plain text from the user and converting it into a cipher text that can be decrypted on the user end. The primary focus is to eliminate the noise that comes along with the input and attach it with a secret key so that the plain text is converted into a cipher text. On the other hand, the cipher text can be decrypted back to plain text by using the public key to get the approximated value on determination of addition and multiplication. The thesis also includes techniques of learning with error and ring learning with error. The final evaluation of the thesis is done on the basis of output so generated by the model.

Keywords: Cryptography, Homomorphic encryption, Cloud security, Partial Homomorphic Encryption

Contents

1. Introduction

In the advent of technical aspects, the usage of computer aided automation has had an immense effect in data protection and data storage. This process of protection and storage needs to be efficient in nature so that the data can be accessed by only legit users. Giving this access to legit users refers to providing a controlled access of data to the user in a secured manner using the process of authentication. However, there are three major aspects of securing information:

- Authenticating the process to legit user
- Giving him access and control of the data
- Encrypting the entire process of data transfer

Apart from this, the creation and generation of large amounts of data not only makes this data less secure, but also makes it exposed to third party attacks. Such attacks raise security concerns to end users and put them in a vulnerable situation by making it difficult for them to trust organizations that tend to secure data for them. In addition to this, the end users can't afford the cost of securing their data using private organizations or local trusted companies. This leaves them with no choice, but to gather and collect their data and use cloud computing services. It has been lately observed 80 percent of company employees tend to make use of cloud sources to store and secure their data [1]. This percentage is most likely to increase in the years coming ahead. Such a high acceptance ratio of cloud usage has been witnessed due to the benefits its offers. Some of its advantages include:

- Providing a flexible environment to store data
- Enhancing and working towards improving disaster recovery in times of single point of failure
- Accessing increased amounts of collaboration to utilize maximum resources

Such benefits and advantages being provided by cloud services make people attract to utilize their resources. However, it is worthy to note here that, providing confidentiality in a cloud environment is something which is not guaranteed by the developers. This makes it less adaptable to store large amounts of data with full security. In addition to this, cloud services also undergoes an issue encrypting and decrypting data before an application needs to be downloaded on a system server. For this process, a user would need to encrypt, decrypt, upload and re-upload data, every time he would use cloud facilities. This makes the entire process of securing data a tedious and time consuming job. In addition to these limitations, all the computations being performed on the cloud are outsourced to another public cloud; thereby leading to a raising concern of data privacy amongst individuals. This impact the trust factors of the end user in utilizing the services provided by a cloud and also make their data vulnerable to intruder interventions like hacking and breaching of a server system.

Therefore in such a scenario, wherein a user is restricted to store and secure data on cloud, the concepts of encryption and cryptography comes in to picture and serves as a driving motivation to the authors to contribute their work in this domain.

A fundamental aspect which adheres to issues mentioned above are further resolved using cryptography. Cryptography is often called as an art of concealing and hiding information so that it can be accessed only by a specific user. This hiding of information prevents an intruder from attacking the system and decrypting the shared information. For this purpose, secret codes are used to convert texts into cipher texts. In earlier times, the process of cryptography was mainly used to send messages confidentially to protect their privacy, unlike now that includes a specific process of encryption using a computational process. This computational process majorly includes arithmetic and logical operations to be performed on numbers and data. However, the visual representation of this data is done through gates and circuits. Homomorphic encryption is one such type of encryption that uses a plain text and converts it into a cipher text using secret and public keys on both the ends of data transfer.

The origin of the word "Homomorphism" is from ancient Greek culture which translates to "same-shape" [2]. The usage of this word was first observed in algebra; wherein certain transformations were done on algebraic sets so that they looked similar in structure. This transformation however involved preserving all the existing relations between the arithmetic operators and operands. In the field of cryptography, the concept of homomorphism is used along with encryption. Therefore, a homomorphic encryption can be used to conduct mathematical operations by encrypting the input and decrypting the result using specific keys. This concept is however considered to be as the new realm in the field of cryptography that gives the data an ability to securely store its data and perform mathematical computation without the necessary to decrypt it in the initial phase. The technique is followed by providing a computational operation to be performed on the data while the data remains encrypted. Hence, this enables the usage of high computational power without the need to sacrifice on user privacy issues.

Thus, by implementing this concept, computations can be carried out to maintain the privacy on both the ends of data communication. Next, depending on the number of arithmetic operations being performed on the data a homomorphic encryption can be categorised as:

- Fully Homomorphic (FE)
- Some-what Homomorphic (SHE)
- Partial Homomorphic (PHE)

On the other hand, it can be observed that a major chunk of homomorphic encryptions are based on lattice cryptography which involves the concepts of Learning with Errors (LWE) and Ring Learning with Errors (RLWE). With all the advantages being provided by HE, the cipher texts involved in the process of encryption also includes a variant of noise in it. Due to this noise and errors such as LWE and RLWE are introduced.

1.1 Background

The idea of cryptography has been understood as a secretive writing technique since the beginning of time. Experts believed that the early documents were all written using cryptic techniques and were considered as a way of sending and transferring data without the involvement of any other person. In order to transmit secret information between two people, cryptography was thought to have been developed. The concept was primarily used in applications ranging from battles to covert business transactions [3]. As a result, this format was also widely used in

computer applications whenever information had to be transmitted over an unreliable medium or even the internet. The fundamental operations of cryptography are as follows:

- Confidentiality: Using this method, it is assumed that only the intended recipient can read the message
- Authentication: This technique considers validating the identity of a real person
- Integrity: This method is in charge of ensuring that the message sent to the user has not been altered in any way
- Non-repudiation: This mechanism has been proven to make sure that the information has only been transferred legitimately by the sender
- Through the process of encryption and decryption, the key exchange mechanism makes sure that the derived key is only decided to be shared only between the server and the client

The plaintext, or unencrypted data, is typically used as the starting point for the cryptographic process. This plaintext is used to encrypt the data into cipher text, which is then further decrypted to restore the original plaintext [4]. The $C = E_k(P)$ preferred techniques that are appropriate for the model's requirements are $P = D_k(C)$ used to carry out the encryption and decryption process. Through the creation of a secret key, the encryption and decryption process is also communicated between the client and the user. Additionally, an algorithm is used to generate the secret key. The following is a common formula to create these keys:

Where;

- P = Plaintext
- C = Cipher text
- E = Encryption Method
- D = Decryption Method
- K = Generated Key

However, the technical analysts have organised the cryptographic algorithms in a variety of ways. The cryptographic algorithms are grouped in this thesis according to the keys that are produced.

(A) Private / Secret Key

A private key involves the generation of a single key both the ends of encryption and decryption. The sender and receiver of the encrypted sensitive data share this key. Due to the fact that both parties share it, the private key is also referred to as "symmetric". The creation of keys, which are then used to encrypt the plaintext and send it to the recipient as a cipher text, is shown in Figure 1.1 (a). The recipient then uses the same key to decrypt the message and restore the message's original plaintext. The process of secret key cryptography is known as symmetric encryption because it takes place when there is only one key present. On the other hand, a public key cryptography is comparably slower than private key cryptography. A private key is typically a long, impossible-to-guess string of bits generated in a random manner.

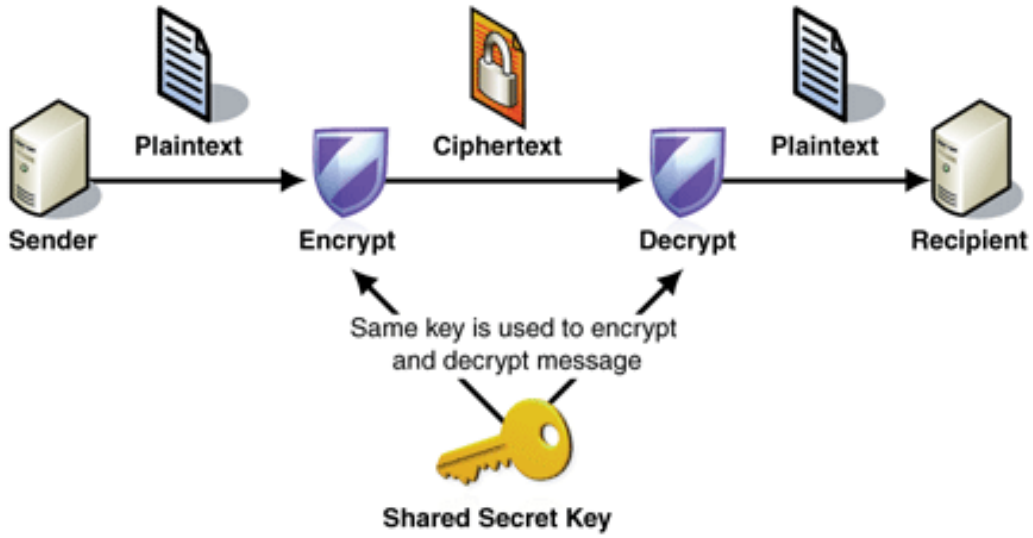


Figure 1.1 (a): Secret Key Cryptography [6]

(B) Public Key

Public key cryptography is used in a manner that differs noticeably from how symmetric cryptography operates. Symmetric cryptography is typically used on a much larger scale by financial corporations and government organisations. In terms of key management, numerous problems with the deployment of symmetric keys were found. As a result, public key cryptography was created [5]. It is necessary to convert the plaintext of a message sent between two parties—known as an interaction—into cipher text. The encryption phase generally refers to this process of conversion. This encryption process is carried out two keys so that only the parties involved in the communication are aware of the generated key.

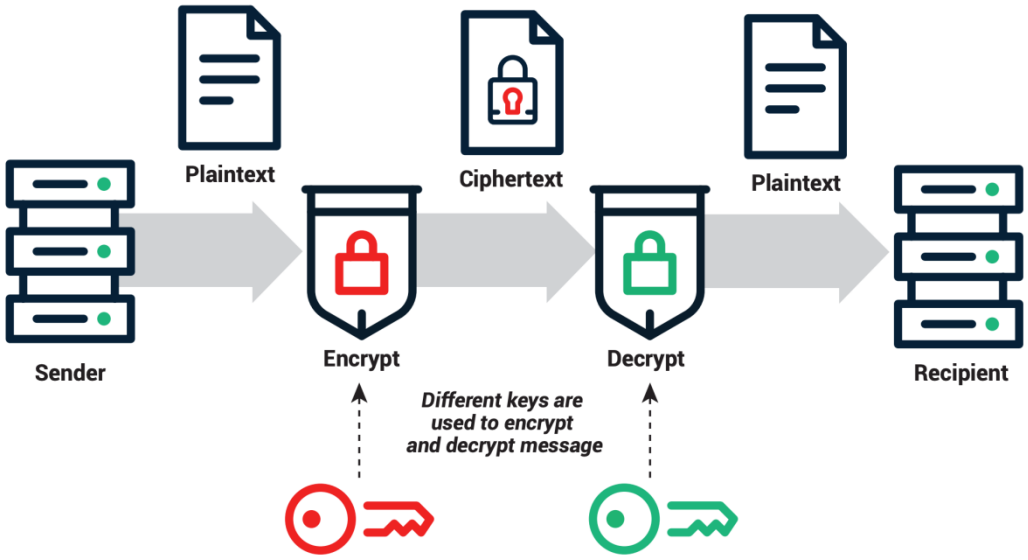


Figure 1.1 (b): Public Key Cryptography [7]

1.2 Research Questions

The primary aim of the thesis is to conduct the executional process of Homomorphic Encryption by performing mathematical operations such as addition and multiplication. The executional process is proposed to be carried out using private and public keys on the ends of the sender and the receiver. To complete a secure information transfer process, the generated keys are then passed to the appropriate client and receiver ends where they are further encrypted and decrypted. Before the model is put into use, though, there are some research questions that need to be resolved. The following are the narrated research questions of the thesis:

- Will the proposed model generate desired and optimized results?
- Will the implementation of secret keys result into less time consumption of resources?
- What are the existing works in the same domain?
- Will the encryption occur after the input is provided by the user?
- Will the execution of homomorphic encryption work on integers?
- What mathematical operations are to be performed on the keys so generated?
- How will the model be evaluated?
- How will the conversion of plain text to cipher text occur?
- What are the functionalities of the polynomials in the thesis?
- How will the keys be generated?
- What is the process of encryption and decryption?

2. Literature Survey

This section of the thesis highlights on the research work being done by other scholars in the same domain.

2.1 Securing Data using LWE and RLWE based Encryption

In a research paper by authors in [8], they proposed a method to detect the cipher texts using HE. In this process the cipher texts were taken as inputs and were further reduced to its dimensional size with respect to the base rings. The dimensional reduction was done using a trace function and a mathematical computation was calculated to improve the efficiency of the overall system. In this system, the trace function was later used to evaluate the plain texts homomorphically and the RLWE was used to generate Rq from Zq factors. The method proved to immensely enhance the overall functioning of the model helped to generate optimized results. The mathematical computations performed by the author were followed by addition and multiplication using RLWE based homomorphic encryption.

In another study by authors [9], they converted LWE plain texts to a single row of RLWE randomly generated numbers of cipher texts. The LWE plain text was given as $\{(b_j , a_j)\}_{j\in[n]}$ and a cipher text was to be generated using a polynomial $b = P_{j\in[n]} b_j \cdot X_j$ and $a_i = P_{j\in[n]} a_j [i]\cdot X_j$ for $i \in [N]$. Homomorphic Encryption was further performed on the polynomial and an RLWE scheme was developed. However, the implementation of this method was combined with the concepts of LWE and RLWE which took input as plain texts and generated polynomial based outputs in the form of

cipher texts. In a similar study by Boura et.al in [10], the author worked on the concepts of homomorphic encryption and presented a detailed summary on various types of the same. He also mentioned how the encryption could be performed on both the ends of the communicating channel. However, the primary aim of the author was to convert plain texts into cipher texts using RLWE based homomorphic encryption. The author also proposed the entire working of secret keys and public keys and mentioned the effects of each keys on the user end. It was observed that the private key was however slower in implementation as compared to the working of the public key. In the later phases the author also mentioned the working algorithm of LWE and RLWE. The research work was further contributed to highlight the importance of HE in any system and how its execution can preserve the entire phase of encoding a plain text. In addition to this, his work also included a detailed explanation on key management system (KMS). The proposed KMS enhanced the overall working off the model and helped to generate optimized results.

Cheon and Kim [11] contributed their study in the domain of homomorphic encryption using ElGamal. This method was observed to work on a similar structure to that of a traditional public key encryption and further involved the decrypting of information through the secret key of the recipient. The overall time consumption of the model was comparatively less, but however it took high computational power to perform small mathematical operations. Therefore the author could perform only the addition of arithmetic numbers using LWE and RLWE. The method was however used to highlight the working implementation of errors and noise. This work was further extended by authors in [12], wherein they proposed a framework that enhanced the overall functioning of the model. The authors used the concepts of HE to convert plain texts to cipher texts and further provided an optimal rate of 1 – o. However, the model was built to focus on low latency communications that enhanced low time consumption with less computational power involved. The author also extended the same research work to cloud environment wherein the generated keys could be stored in a secure manner. The overall implementation also included RLWE and LWE based homomorphic encryption scheme. Using this method, the overhead communication cost was immensely reduced on the clients end.

2.2 Securing Data using Secret Key and Public Key based Encryption

As it was mentioned in the sections above, a single key is typically used in the working implementation of a symmetric encryption algorithm. Therefore data encryption and decryption are handled by the same key on both the ends. However, there are numerous algorithms, including DES, AES, and Blowfish, to identify this type of encryption. In such a scenario, each algorithm is in charge of gathering and storing various kinds of data in accordance with their respective sizes. Additionally, the size of the key involved is fixed by this block size. Therefore, the output produced in this manner takes different form of cipher texts. The components of symmetric cryptography are listed in below:

- *Plaintext* is regarded as the initial information that the sender wishes to send to the recipient. In most cases, this plaintext serves as the input for the encryption algorithm
- A set of procedures known as an *encryption* algorithm are carried out when plaintext is changed into cipher text
- The *plaintext* is combined with the secret key to create cipher text, which is the generated value that is used to do this

- *Cipher text* is used as the encryption algorithm's input and is regarded as the original plaintext
- *Decryption* algorithm: this is a set of procedures used to decipher the cipher text while using the secret key

To create a single key for data encryption and decryption in symmetric encryption, a well-known technique for private key encryption is used. Most commonly used algorithms in secret key are AES and DES algorithms.

The authors of a study [13] suggested an AES-based method to secure data stored in the cloud. The main idea behind this method for transferring data from one end to the other was permutations. The authors used a set of keys, such as AddRoundKey and ShiftRowKey, to finish the encryption process. On the basis of bit size, these keys were iterated on a regular basis. This bit size was thought to be sufficient for the receiver to decrypt the cipher text. Although the key management system only needed a small amount of storage space, the mechanism still managed to achieve high encryption standards. The process effectively protected the system's overall confidentiality and integrity while encrypting and decrypting the data. In a different study by [14], the authors suggested a simple cryptographic method that could get around cloud storage problems. In later stages, the authors also suggested a hybridised approach that could combine symmetric and asymmetric techniques and lead to a successful implementation of cloud data security. AES and RSA were however included with this model. However, the model with RSA algorithm implementations was successfully implemented in comparison to other existing techniques like DES and 3DES.

As mentioned in the previous section of the thesis, two keys are used in the implementation of a public key and an asymmetric encryption algorithm is used to carry out the execution. While using this method, two generated keys are used: a public key and a private key. The private key is only known by the user of the communication channel, while the public key is accessible to everyone. However, there is a mathematical relationship between the two keys and they are connected to one another. Calculating private keys from public keys requires a lot of time in later stages. Since it is impossible, the private key cannot be derived from the public key. However, it has been noted that the two keys behave differently depending on the application being used. When data is encrypted, the public key is used to encrypt the data. However, when the message needs to be decrypted, the use of a private key enters the picture. Private keys can, however, occasionally be used to generate digital signatures that are then verified with the aid of a public key. The commonly used algorithms of public key encryption include DSA and RSA algorithms.

The authors of a study by [15] suggested using an asymmetric key encryption algorithm that was based on the idea of linear geometry. To enable the information transmission process to take place covertly over a risky internet platform, both the substitution and transposition techniques were employed. The idea behind this model was to protect images rather than texts that were sent over the communication channel. The technique was fully responsible to use random number generator and further developed a random matrix that could cipher the bytes from secret files. The asymmetric encryption algorithm, which was based on the ideas of public key cryptography, was used by the authors in another work by [16]. This system model was responsible to transfer data over neural networks and later provide an authenticated and secured method to protect the data on

an untrusted platform. The model generated a non-linear clustering algorithm and used 32 bit registers to operate effectively. The method later employed a pseudo-random number generator to produce a string of random numbers that could be constructed using round keys.

The diagram below explains the common workflow of a cryptosystem model as observed through the literature survey so conducted:

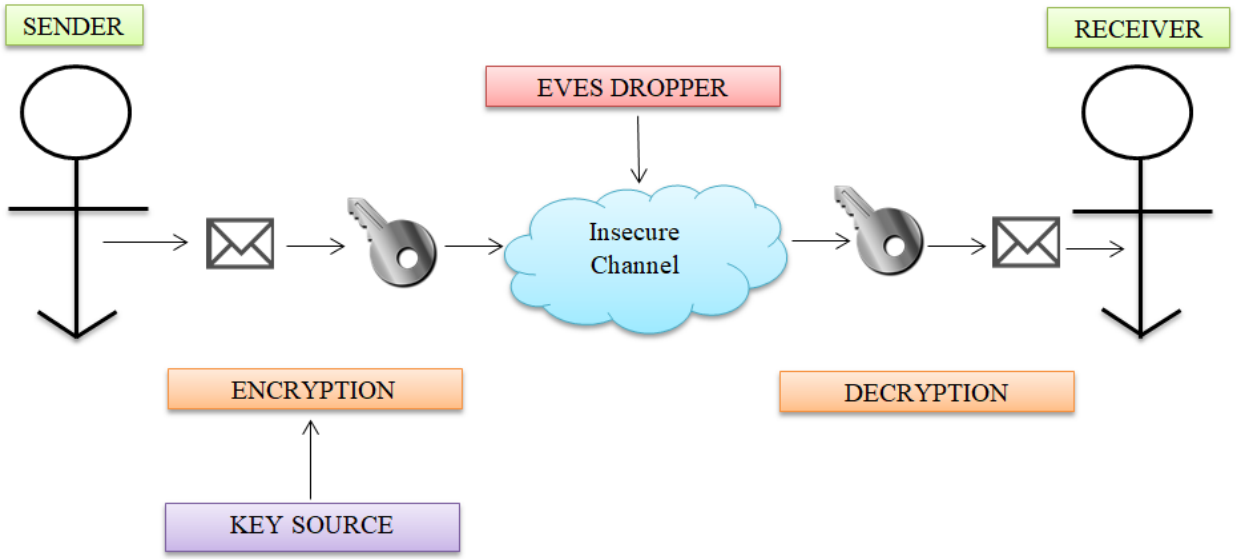


Figure 2.2: Common Workflow as observed in the Literature Survey

2.3 Securing Data using Homomorphic Encryption

According to research by Maya and Hyotaek [17], the authors concentrated on the use of homomorphic encryption and asserted that it offered better security when tested on arithmetic numbers. They claim that this algorithm has improved the system's general security and privacy. In a related piece of work, Jian. et. al. [18] implemented the principles of a fully homomorphic algorithm and proposed to create an arithmetic algorithm based on modules that could fulfil the requirement of maintaining privacy in a third-party cloud. The effectiveness and performance measure of this algorithm were later demonstrated to be superior to that of a similar study published at Cryptology EUROCRYPT 2010 by Marten van Dijk et al. In [19], Na. et. al. contributed their research in order to create applications based on homomorphic encryption (HME). The authors of this paper conducted a review of all the current systems and made improvements to the model. A thorough explanation of the significance and restrictions of HME was put forth by Monique et al. in [20]. The author reviewed several research papers and further summarised them in his work, and came to the conclusion that HME has advantages over RSA. His work was further developed to study the algorithm and was later assessed for performance measures. Tebaa et al [21] proposed a method in the same domain and were further able to increase accuracy by fine-tuning the hyper-parameters of the existing algorithm. However, the author's objective was to improve the system's overall accuracy.

Mihai and Cezar specifically highlighted the HME model in their study [22] and attempted to create applications that could protect all the sensitive data stored in the cloud. In a different paper by Ciara et al., [23], he presented real-time PHE applications and later suggested that they be

implemented on GPUs. In a similar research work by Li et al. in [24], the author suggested a straightforward method for carrying out Full Homomorphic Encryption (FHE), using matrix addition and multiplication as the mathematical operation to carry out error-free plaintext encryption. A secure channel of communication between user parties was thus created using this technique.

In a related piece of work, Yannan et al. in [25] concentrated on problems with conventional cryptography and attempted to encrypt data in the cloud by transferring storage space incrementally. He suggested a multi-cloud architecture with N data centres that could repeat the data and divide it appropriately for this purpose. To secure the entrustment of computation to a third party, the author carried out the experimental studies using homomorphic encryption in conjunction with public key cryptography. Additionally, he observed conventional cryptographic techniques like multiplicative and additive homomorphism. In another work by Ahmed et al in [26] he implemented fully homomorphic encryption (FHE) and offered services for outsourcing calculations that could encrypt user data from a distance. The user data was majorly stored in data centers and retrieved on demand. This concept of outsourcing an HME was named as Verifiable Fully Homomorphic Encryption (VFHE). Many existing schemes have shown that the concept is feasible, but the performance needs to be dramatically increased in order to make it practical for real time based uses. One subtle challenge is figuring out how to deal with the noise effectively. In addition, the cloud's storage capacity and processing time affected the service's ability to be implemented quickly. His work also provided a well-structured and symmetrically verifiable FHE based on a noise-free, noise-permanent statistical fundamentals design that is independent of homomorphic cipher-text evaluation.

3. Research Methodologies

This section of the thesis highlights on the fundamentals of homomorphic encryption and further mentions the methodology of the proposed system.

3.1 Types of Homomorphic Encryption

Homomorphic Encryption includes the process of securing data after a set of mathematical operations are performed on it. Such mathematical operations enable the conversion of cipher text to plain text without revealing and decrypting the original data. Hence, this encryption scheme is observed to be a sub category of cryptography that helps to transfer data by hiding information in it. The concept of HE is briefly explained in [29], wherein the author explains the codes of cryptography using errors. However, the execution of HE majorly involves conversion of plain texts to cipher texts by adding a binary operation and further executing them. However, it is important to note here that on performance of such operations the storage and retrieval of user data is the most crucial part and hence needs computational analysis to be done on it. In such a scenario, the usage and implementations of HE based encryption and decryption proves to be fruitful. The process of HE allows encryption to be done with less amount of time consumption, less computational complexity involved and more security being provided to the user data. One of the most important characteristic of an HE is that it allows the execution of mathematical operations to be done on encrypted data without the need of decrypting it first. This characteristic serves as an added advantage and an alternative to traditional cryptography. The executional process of an HE is simple in working and

consists of only binary operations such as AND and XOR. Hence, the addition and multiplication of arithmetic numbers can take place as:

$$\text{AND}\ (b1, b2) = b1 \cdot b2$$

and XOR can be seen as the addition of two bits modulo 2:

$$\text{XOR}\ (b1, b2) = (b1 + b2) \bmod 2$$

The implementation of a generic HE occurs in four stages as:

- Generation of the key
- Encryption of the input
- Decryption of the output
- Mathematical operations being taken place on the numbers

The figure below explains the involved four steps:

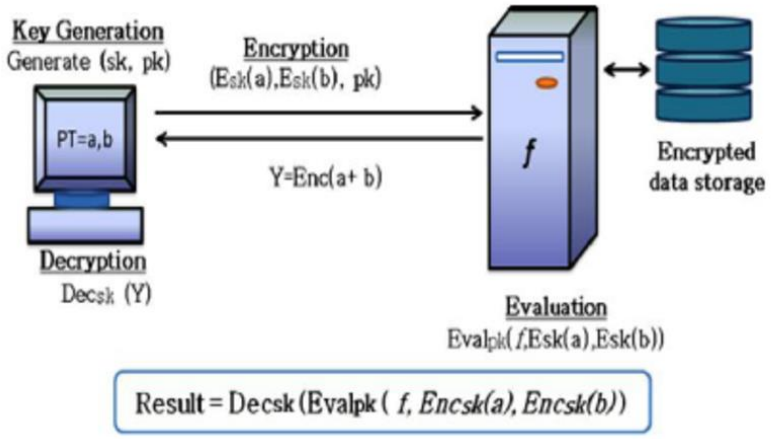


Figure 3.1: Steps of Homomorphic Encryption [28]

Where:

- pk: input taken using the public key
- C: the involved circuit on which operations occur

The working implementation of a traditional HE is determined by the number of operations involved in the process. Hence, on that basis an HE can be categorised as:

- **Fully Homomorphic Encryption (FHE)**
  An FHE is believed to perform arbitrary functions on the encrypted data that further enables mathematical computations to be performed on the input. However, it is important to note that executional working of an FHE follows asymmetric encryption method and performs

mathematical operation on the input at the bit level. For a given set of cipher texts (c_1,c_2…..c_n) and encryption is performed on the messages (m_1,m_2….m_n). In such a scenario, an FHE offers its functionality as f (m_1, m_2……m_n). This functionality is responsible to securely convert plain text into cipher texts and finally provides the result by decrypting the given input. However, throughout the process of encryption, it is necessary that the input components remain encrypted and can only be decrypted by the private key of the receiver. It is worthy to note here, that the original HE concept was developed to work on polynomials, but its implementation raised issues with regards to Ring Learning with Errors (RLWE). Hence, it was then that the FHE was used on arithmetic numbers.

- **Partial Homomorphic Encryption (PHE)**
  When the implementations of computations were required to be done on large datasets the PHE scheme of HE was used. Large scaled data that was used in big corporative companies demanded high level of security. In such a scenario, Partial Homomorphic Encryption was taken into consideration as a solution to this issue because they enabled arithmetic operations to be carried out on encrypted data by employing an analysis that can be performed in parallel. However, the most important feature of an FHE is its infinite chaining of mathematical functions in the cipher space, which allows for the addition and multiplication of encrypted operands indefinitely. Since the HE schemes are built on fractionally inaccurate depiction of plaintext values, PHE is introduced as a mechanism scheme that reduces the noise of cipher values.
- **Somewhat Homomorphic Encryption (SHE)**
  SHE is considered to be as an extension of a PHE that involves the arbitrary operations to be performed either on multiplication or addition. This eventually means that an SHE is restricted to a certain boundary and cannot perform beyond that. It can compute only small additions and multiplications such as calculation of average numbers and addition of two numbers. The SHE scheme is further bootstrappable. This characteristic of an SHE allows the encryption to be evaluated only on its decryption circuit. Hence with this structure, the encryptions can perform arithmetic operations using specific libraries and carrying out the process of key generation and decryption.

3.2 Proposed Methodology

The primary aim of the thesis is deploying the working implementation of homomorphic encryption from scratch. For this purpose, the authors initially went through the research work in the same domain and surveyed multiple existing techniques that supported HE. It is worthy to note here that the primary inspiration of the encryption system was derived from the concepts of BGV. BGV is an encryption scheme that works on matrices and generate vector outputs in the form of results. Its similar working is implemented in homomorphic encryptions as well. However, for the implementation of the thesis, the authors have used the concepts of Partial Homomorphic Encryption (PHE) along with RLWE

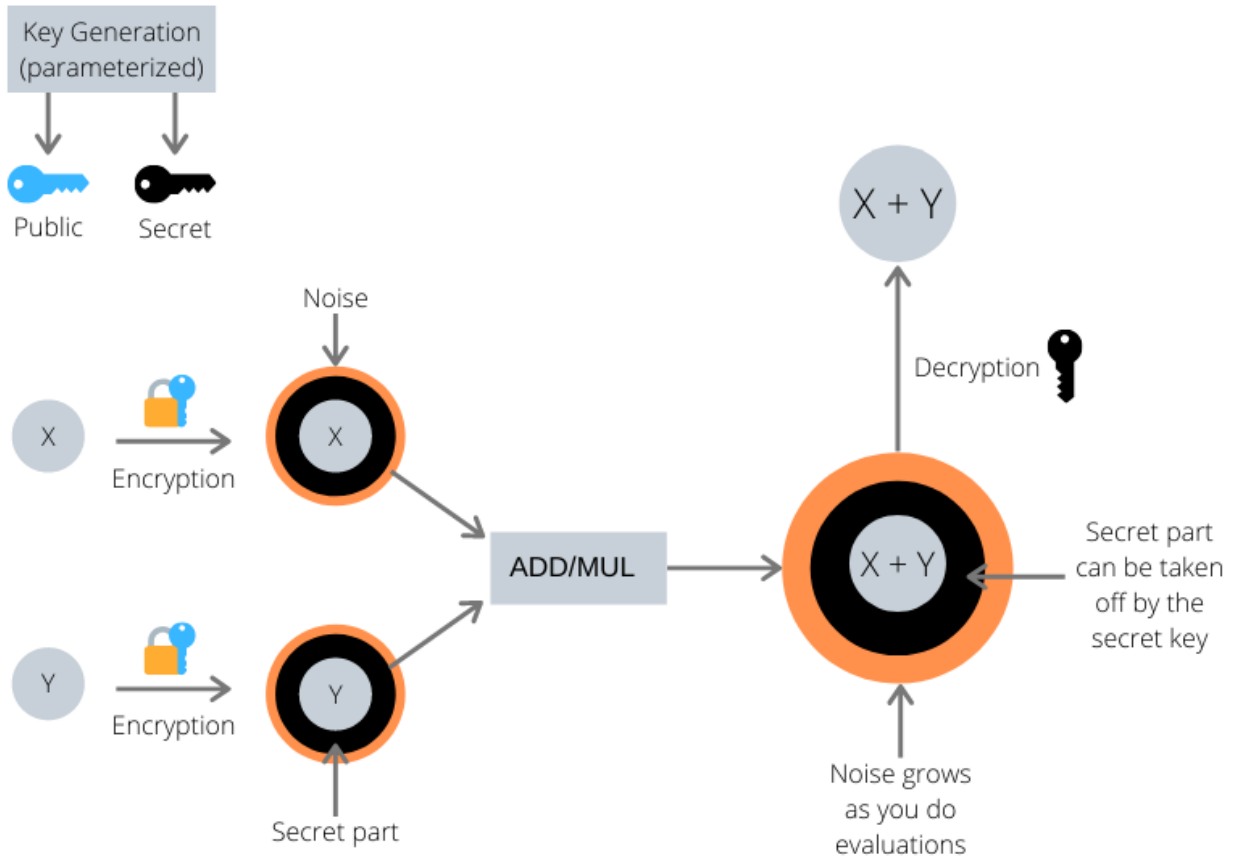


Figure 3.2: Workflow of the Proposed System [27]

Ring Learning with Error (RLWE) is believed to be a variant of Learning with Errors (LWE). An LWE is known to work on integers and tends to select specific pairs s from (a_i, b_i) and further perform the function of encryption. Here, (a_i and b_i) are considered to be input integers and *s* is a pair selected from the input on which the final operation is to be performed. Since, the presented thesis aims to implement on polynomials; the authors executed the fundamentals of RLWE. RLWE works on polynomials and considers the same input of polynomial vectors as (a_i, b_i). Further, it selects a pair *s* and performs encryption using a polynomial modulus. The formula to calculate the polynomial modulus is given as:

$$d<n, \text{ and } x^n \equiv -1 \ (mod \langle x^n+1 \rangle)$$

Where;

- d is the degree of polynomial
- n is considered to be a power of two
- x represents a positive polynomial so entered

Using the concepts of PHE based RLWE encryption the authors perform arithmetic operations of addition and multiplication on the numbers so provided by the user. The presented homomorphic operations are as follows:

- ADD (A, B): A and B are two polynomial inputs given by the user. An encryption of 'N' is performed on each polynomial and the output is generated in the form of A+B. Hence the formula for performing addition would be:

$$A_{N\times 2} + B_{N\times 2}$$

  Here the binary digit 2 represents the degree of polynomial.

- MUL (A, B): A and B are two polynomial inputs given by the user. An encryption of 'N' is performed on each polynomial and the output is generated in the form of A x B. Hence the formula for performing addition would be:

$$A_{N\times 2} + B_{N\times 2}$$

  Here the binary digit 2 represents the degree of polynomial.

As per the concepts and methodologies mentioned above, the implementation of PHE can be broken down into the following steps:

- Key Generation

The process of key generation begins by secretly generating a random key *sk* and performing uniform probability distribution on it represented by $R_{2.}$ Since *sk* is a secretly generated random key, it is considered to be a polynomial with coefficients of 0 or1. In a similar way, a public key is also generated for a polynomial input *a*. This randomly generated public key is represented as R_q. In the next stage of key generation a tuple is created and employed over R_q. The formula of tuple creation is given as:

$$pk=([-(a\cdot sk+e)]_q,a)$$

- Encryption

Since the thesis supports encryption of polynomials using RLWE, the formula used for the same shall be represented as:

$$R_t=Z_t/\langle x^n+1\rangle$$

Where:

- t is the plain text modulus
- R is the RLWE factor
- Z is a set of integers
- n is considered to be a power of two
- x represents a positive polynomial so entered

The encryption however includes encoding an integer, converting it into a polynomial and finally decrypting the same on the receivers end. For this purpose, the encryption algorithm takes the input as public key $pk \in R_q \times R_{q,}$ polynomial as $m \in R_t$ and decrypts the output on the receivers end as $c_t \in R_q \times R_{q.}$

- Decryption

The process of decryption takes place on the receivers end and involves the same formulas as mentioned above. However, it is in this process that the generated encryption is decoded using the public key of the sender and the private key of the receiver.

- Evaluation

The process of evaluation is considered to be as a major part of any system model. Since this contains the parameters on which the entire implementation would depend on, this stage of evaluation is very critical. However there are certain factors on which the evaluation of an encryption depends on. These factors are:

- Security: securing a data is the most important factor to be considered while transferring the information on a public platform such as the Internet. If the users feel their data is safe, it makes them comfortable and easy to trust the process and the third parties so involved.
- Time complexity: when a data is under the process of encryption, the time it requires to encrypt the data on the senders end and the time it would require to decrypt the data on the receivers end, sums up to another important factor which decides the overall performance of the system. In such a scenario, it is also necessary to comprehend the amount of data that needs to be encrypted. If there is large amount of data involved, the time for processing the same would eventually increase and vice versa. Hence, for this factor fulfil successfully it is important to select appropriate encryption schemes as per the requirement of the model.
- Computational complexity: larger the data more is the amount of complexity involved in the implementation. Hence it is necessary that the built model provides an optimized result with less computational overhead involved.

4. System Design

The entire implementation is proposed to take place on a Python setup. The system design of the proposed model is depicted in figure below:

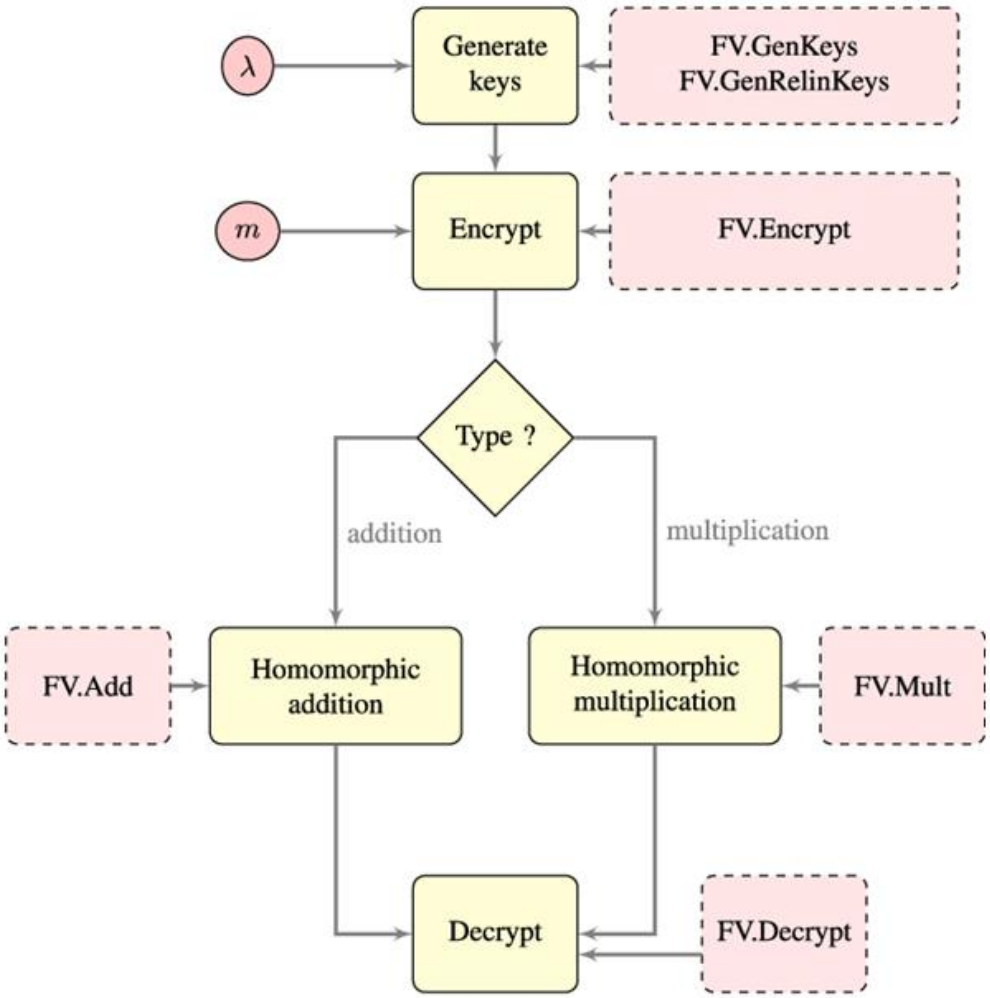


Figure 4: System Design of the model [30]

The system of the proposed model comprises of three modules as observed in the diagram above. This majorly includes key generation, encryption and decryption. A detailed explanation of each step is mentioned in the next section.

5. Data Implementation

The entire process of data implementation has been done using the Python language. The system proposes the working execution of addition and multiplication of two numbers being entered by the user. The authors have tend to use RLWE based PHE to carry out the same. The diagram below depicts the implementation of the model on the sender and the receiver side.

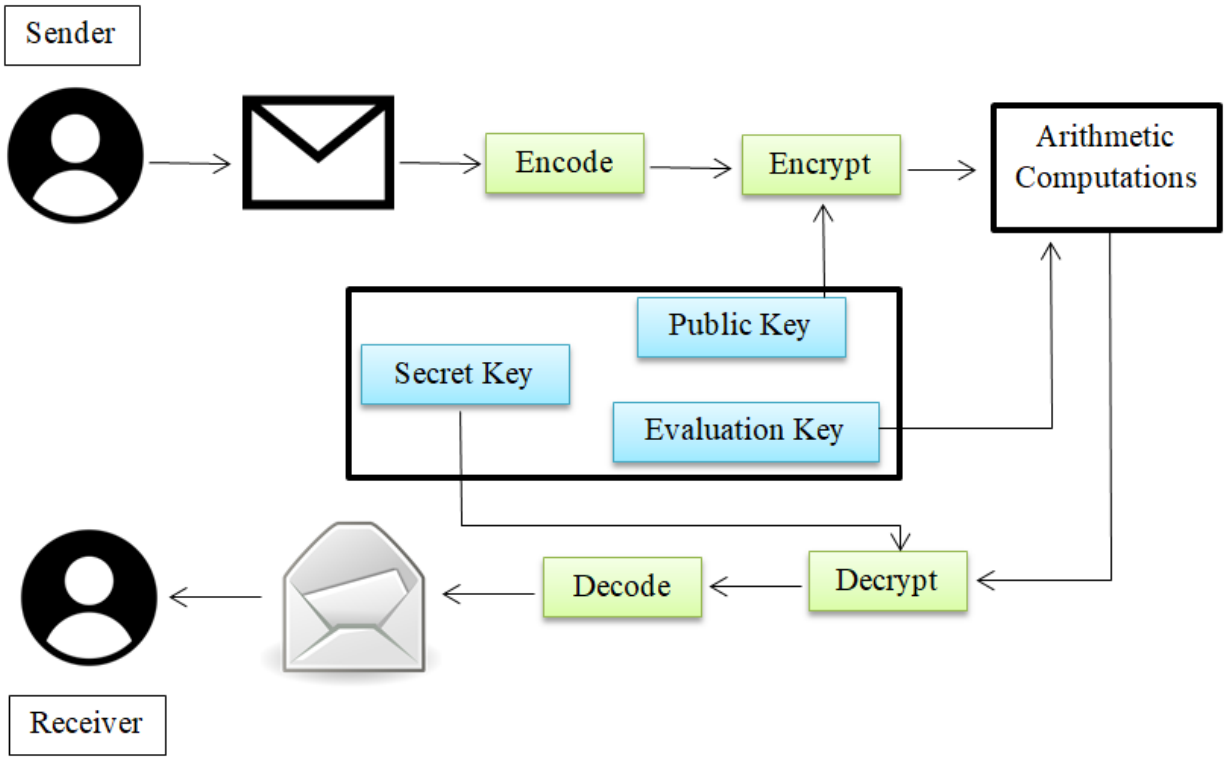


Figure 5: Architecture of the Proposed System

The process initially begins by the user entering the first number that is to be operated. Considering in this case, let the number be 25 as depicted in Snippet 1. The number is taken as input on the back end by the sender and is further processed to encode the given message. This message or the input undergoes the process of encryption. At this stage the encryption is performed using the public key. A public key is used as it is known to both the communicating parties i.e. the sender and the receiver. Next, arithmetic computations are performed on this encrypted data and the number 25 is converted into a cipher text of arrays. This array is considered to be a 16 digit encrypted array.

In the next stage, the user inputs the second number that is to be operated along with the previous number so entered. In this case, the second number so entered is 47. Due to the advantageous feature of HE, the encrypted code of 47 is directly added to the already existing encrypted format of 25. This eliminates the need to make changes in the source and hence operates and encrypts on the already existing encryption. Therefore, an encrypted format of 47 is added to the encrypted format of 25 and two-16 digit arrays are formed and simultaneously added on the back end.

Once the computation of addition is performed and completed, the final answer of 25 + 47 is calculated and still kept in the encrypted format. Further, this needs to be decrypted so that it can be present in a readable format to the receiver. Hence, the process of decryption involves the usage of the secret key which is known only to the sender; unlike the public key which was initially shared between both the parties in stage 1.

Once the secret key of the receiver is used to decrypt the addition of two numbers, the resulting format is readable to the user and is displayed on the screen as addition of two number to be 72 (25 + 47) as depicted on Snippet 1. However, the same procedure is followed for multiplication of

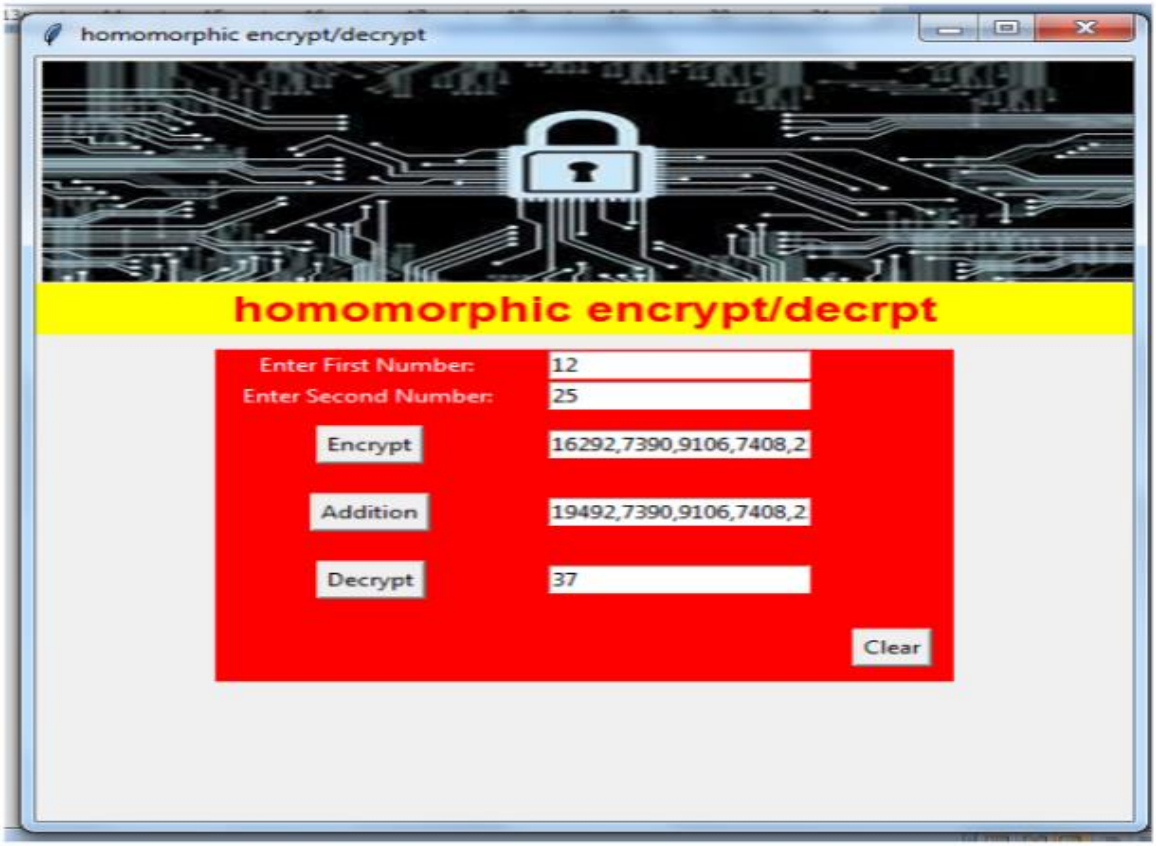
two numbers as well. This multiplication can be further viewed in Snippets 3 and 4 which are mentioned in the results section.

6. Results and Evaluation

The evaluation of a typical HE based encryption occurs on the basis of performance speed of the encryption, the security provided by the encryption and the time taken to perform the mathematical operation. In the proposed thesis, we have implemented Partial Homomorphic Encryption (PHE) on arithmetic numbers. The mathematical operation being performed on they are addition and multiplication. However it is worthy to note here that, this process of encryption allows altering the data on the encrypted format and thereby eliminates the need to continuously update the input data. The below snippets depict the results so obtained after performing mathematical operations on them.

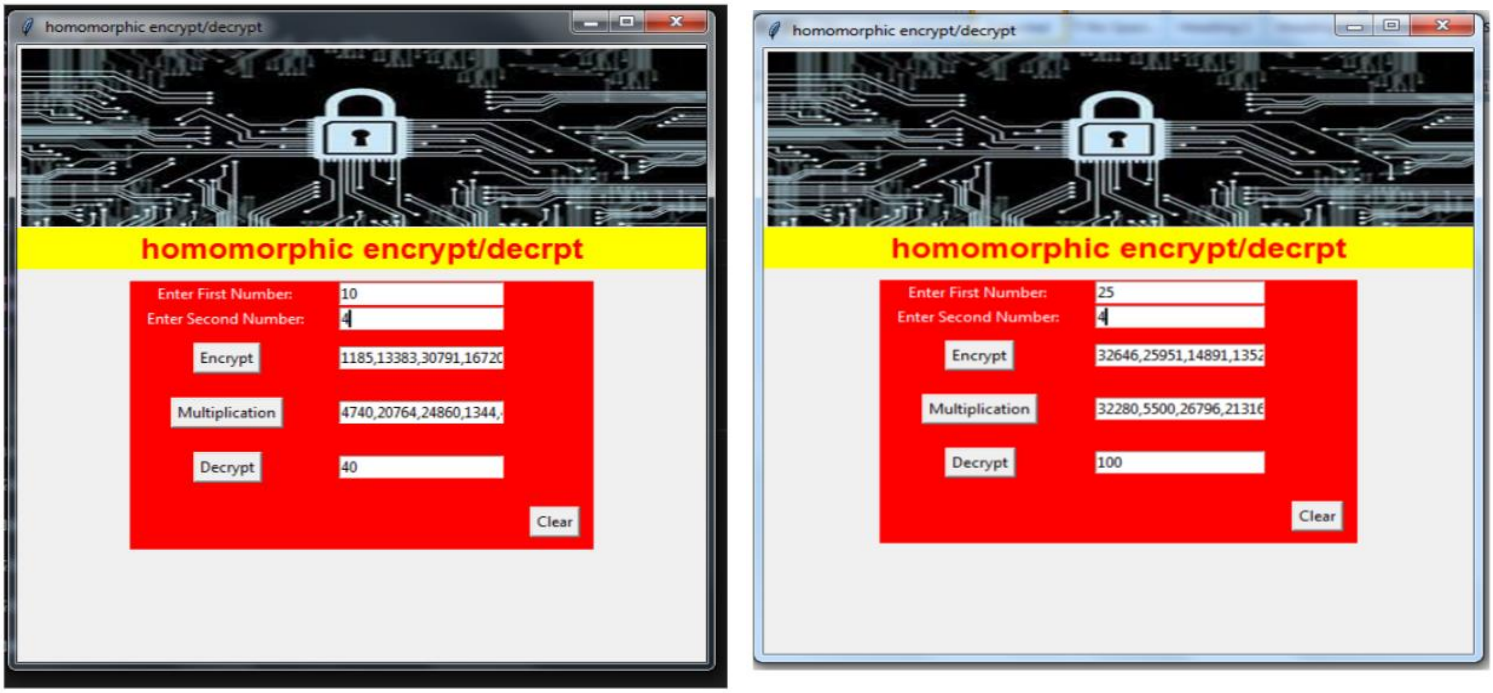
Snippet 1 and snippet 2 illustrates addition of two numbers using HE; wherein the user enters the first number. This number is initially encrypted and converted into a 16 digit array. The user then enters the second number, which is further added to encrypted format of the first number. In the space provided for the addition row, the mathematical operation of two numbers take place in an encrypted format. Next, the additions of the two numbers are generated as output in a decrypted format that is readable to the user. However, the same process is applied to the mathematical operation of multiplication.

Snippet 1: Addition of two numbers



Snippet 2: Addition of two numbers

The above snippet depicts the addition of 12 and 25, in such a way that the user initially enters the number 12 as the first number, which is then encrypted to a 16 digit array. The user then enters the second number as 25, the encrypted format of which is added simultaneously to the existing encrypted format of 12. The addition of encrypted data takes place and the final answer as 37 (12+25) is displayed in a readable format to the user. However, the same process is applied to the mathematical operation of multiplication.



Snippet 3&4: Multiplication of two numbers

7. Conclusion and Discussions

This section puts an end to the proposed thesis and highlights the involved conclusions followed by challenges and limitations.

7.1 Conclusions

The confidentiality of information is more important than ever in today's, Internet-driven world. It is essential to safeguard user profiles and resources from malicious third parties for highly sensitive systems. However, it is now a common practice to encrypt data and exchange keys with service providers, cloud operators, etc. This model gives up control over the sensitive data's privacy. The only people who can access the data are users or service providers. Even after the user has ended their relationship with the services, unreliable cloud operators may continue to hold onto sensitive data and the identifying information of users. For this purpose, utilizing homomorphic encryption (HE) schemes is a promising approach for maintaining the data's privacy. HE is a unique kind of encryption technique that enables any third party to manipulate the encrypted data without first decrypting it. As a result, the authors of the thesis examined the HE and FHE schemes in this paper. In particular, the specifics of the well-known Partially HE (PHE) and Somewhat HE (SWHE), which are significant pillars of achieving FHE, were presented starting from the fundamentals of HE. The major FHE schemes were then presented with this classification after we divided the FHE schemes in the literature into three different categories.

On the other hand, for any encryption based system, it is important to evaluate the process on the basis of security and speed. For that purpose, the authors of this thesis have conducted Homomorphic Encryption to perform addition and multiplication of two numbers. The primary aim of the authors was to focus on advancing key arithmetic operations based on RLWE-HE. The implementation was

carried out using Python and arithmetic library. In the library, we provide both a quick parallel implementation and a hardware cost-effective serial implementation for each of these operations. In order to show the speedup possible in hardware, we also described a modular and hierarchical deployment of a software accelerator using the modules of the suggested arithmetic library. The accelerator and the modules' parameterized design implementation framework by giving the modules the flexibility to be used in additional schemes. It's critical to comprehend how homomorphic schemes operate because there has been a significant amount of study conducted in this area. For this reason, the authors described how to build a fully homomorphic encryption system. The authors demonstrated a plan for a mathematical operation to be carried out on two numbers. The homomorphic property and the accuracy of the decryption are both non-trivial even in this mechanism. To make the most of the plaintext and cipher text, space and variables must be carefully chosen. However, every modification to the cipher text has the potential to make the mathematical operations farther away, which will immediately make the decryption process impossible.

7.2 Challenges and Limitations

Making an implementation that is both effective and secure is currently a huge challenge. Large dimension lattices are required for the confidentiality of the fully homomorphic scheme, which means there will be more operations. However, there are two sides to this improvement. To fulfil that purpose we worked on the implementation of partial homomorphic scheme. It is still less effective though, and the programme could be made more effective. For instance, it would be better if the key generation process were more random. Considering how many multiplication and additions the programme performs, some functions could still be made simpler.

Following are the major challenges experienced by a model:

- Need of Mathematical Innovations: By enhancing the mathematical foundations of FHE, a speedup factor of x1 can be attained
- Need of Software Innovations: By looking into potential parallelization opportunities for FHE when used on multi-core processor architectures like graphical processing units (GPUs); the overall process can be enhanced
- Need of Hardware and Architectural Innovations: By using a specially designed high performance hardware accelerator, the full parallelism of FHE can be obtained

7.3 Future Challenges

Future work should aim to implement and optimise the bootstrapping based arithmetic operations. One of the crucial processes for achieving fully homomorphic encryption is the bootstrap operation, but it is still very expensive to carry out. The bootstrap procedure can be improved to make it more useful. Another future work that involves HE is to take advantage of the inherent benefits that other RLWE-based homomorphic encryption schemes, like BGV, offer and further incorporate them in the library.

References

[1] Applications of Modern cryptography Technologies, applications and choices. https://www.surf.nl/binaries/content/assets/surf/en/knowledgebase/ 2010/rapport_201009_SNcryptoWEB.pdf Accessed July 13, 2017

[2] Arita, S., Nakasato, S.: Fully homomorphic encryption for point numbers. In: Chen, K., Lin, D., Yung, M. (eds.) Inscrypt 2016. LNCS, vol. 10143, pp. 253–270. Springer, Cham (2017

[3] Data Encryption Standard. http://searchsecurity.techtarget.com/ definition/Data-Encryption-Standard Accessed May 12, 2017

[4] Python Libraries. https://www.python.org/ Accessed July 16, 2017

[5] Nabihah Ahmad, Rezaul Hasan, and Warsuzarina Mat Jubadi. Design of aes s-box using combinational logic optimization. In Industrial Electronics & Applications (ISIEA), 2010 IEEE Symposium on, pages 696–699. IEEE, 2010

[6] https://www.researchgate.net/figure/Private-Key-Cryptosystem_fig1_317044668

[7] https://sectigo.com/resource-library/public-key-vs-private-key

[8] Gentry, C., Halevi, S., Smart, N.P.: Better bootstrapping in fully homomorphic encryption. In: Public Key Cryptography–PKC 2012, pp. 1–16. Springer (2012)

[9] Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: Fast fully homomorphic encryption over the torus. Journal of Cryptology pp. 1–58

[10] Boura, C., Gama, N., Georgieva, M., Jetchev, D.: Chimera: Combining ring-lwebased fully homomorphic encryption schemes. Journal of Mathematical Cryptology 14(1), 316–338 (2020)

[11] Cheon, J.H., Kim, J.: A hybrid scheme of public-key encryption and somewhat homomorphic encryption. IEEE Transactions on Information Forensics and Security 10(5), 1052–1063 (2015)

[12] Gentry, C., Halevi, S.: Compressible fhe with applications to pir. In: Theory of Cryptography Conference. pp. 438–464. Springer (2019)

[13] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2):120–126, 1978

[14] Rainer A Rueppel. Stream ciphers. In Analysis and Design of Stream Ciphers, pages 5–16. Springer, 1986

[15] Gurpreet Singh. A study of encryption algorithms (rsa, des, 3des and aes) for information security. International Journal of Computer Applications, 67(19), 2013

[16] Yosef Stein, Haim Primo, and Joshua A Kablotsky. Galois field multiplier system, July 20 2004. US Patent 6,766,345

[17] M. Louk and Hyotaek Lim, "Homomorphic encryption in mobile multi cloud computing," Information Networking (ICOIN), pp. pp.493-497, 12-14 , Jan. 2015

[18] Jian Li, Danjie Song, Sicong Chen, and Xiaofeng Lu, "A simple fully homomorphic encryption scheme available in cloud computing," Jian Li; Danjie Song; Sicong Chen; Xiaofeng Lu, "A simp Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference , vol. 01, pp. 214-217, 2012

[19] Baohua Chen and Na Zhao, "Fully homomorphic encryption application in cloud computing," Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 11th International Computer Conference, pp. 471-474, 2014

[20] Claude Turner, Pushkar Dahal Monique Ogburn, "Homomorphic Encryption ," Procedia Computer Science , pp. 502-509, 2013

[21] M. Tebaa, S. El Hajji, and A. El Ghazi, "Homomorphic encryption method applied to Cloud Computing," Network Security and Systems, pp. 86-89, 2012

[22] Mihai Togan and Cezar Plesca, "Comparison-Based Computations Over Fully Homomorphic Encrypted Data," Communications (COMM), pp. 1-6,29-31, 2014

[23] C. Moore, M. O'Neill, E. O'Sullivan, Y. Doroz, and B. Sunar, "Practical homomorphic encryption: A survey," Circuits and Systems (ISCAS), pp. 2792-2795, 2014

[24] Xing Li, Jianping Yu, Peng Zhang, and Xiaoqiang Sun, "A (Leveled) fully homomorphic encryption scheme based on error-free approximate GCD," Electronics Information and Emergency Communication (ICEIEC), pp. 224-227, 14-16 , 2015

[25] R.Manjua, A.Shajin Nargunam and A. Rajendran,"Multimodal Biometric Authentication system Based Performance Scrutiny", Medwell Journal 2014

[26] Ahmed El-yahyaoui and Mohamed Dafir Ech-Cherif EL kettani," A Verifiable Fully Homomorphic Encryption Scheme for Cloud Computing Security", Technologies, 7, 21, 2019

[27] "Build an Homomorphic Encryption Scheme from Scratch with Python", OpenMined Blog, 2022. [Online]. Available: https://blog.openmined.org/build-an-homomorphic-encryption-scheme-from-scratch-with-python/. [Accessed: 13- Aug- 2022].

[28] 2022. [Online]. Available: https://www.researchgate.net/figure/Homomorphic-Encryption-functions_fig1_263022874. [Accessed: 13- Aug- 2022].

[29][3]Cims.nyu.edu, 2022. [Online]. Available: https://cims.nyu.edu/~regev/papers/qcrypto.pdf. [Accessed: 13- Aug- 2022].

[30] "CSDL | IEEE Computer Society", Computer.org, 2022. [Online]. Available: https://www.computer.org/csdl/journal/tc/2018/03/07797469/13rRUxAASSA. [Accessed: 13- Aug- 2022].