Implementation of Homomorphic Encryption Technique

Apurva Sachan

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India

Implementation of Homomorphic Encryption Technique

Thesis submitted in partial fulfilment of the requirements for the degree of

Master of Technology

in

Computer Science and Engineering

(Specialization: Information Security)

by

Apurva Sachan

(Roll No. 212CS2115)

under the supervision of

Prof. A. K. Turuk

Department of Computer Science and Engineering

National Institute of Technology Rourkela

Rourkela, Odisha, 769 008, India

June 2014

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India.

Certificate

This is to certify that the work in the thesis entitled ***Implementation Of Homomorphic encryption technique*** by ***Apurva Sachan*** is a record of an original research work carried out by her under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Master of Technology with the specialization of Information Security in the department of Computer Science and Engineering, National Institute of Technology Rourkela. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Place: NIT Rourkela
Date: June 2, 2014

Dr. A. K. Turuk
Professor, CSE Department
NIT Rourkela, Odisha

Acknowledgment

I owe my personal gratitude to the people who support me to build this thesis with the direct and indirect help from the lots of people.
First and foremost, I am very much thankful for my supervisor Dr. A. K. Turuk, for his endless support and advice during the work. He appreciate my ideas and let me work in my own way. He is available for the entire time when I need to discuss about the work.
I am also thankful to our HOD Prof. S. K. Rath to provide the lab for the entire duration of my work. He is always encouraging and supportive for all his students.
I am also acknowledging the resources provided by our institute NIT Rourkela.
At last but not least, I am dedicating my work to my parents. Their love, support and encouragement provide me the way to pursue my interest.

Apurva Sachan
Roll: 212CS2115

Declaration

I, Apurva Sachan (Roll No. 212CS2115) understand that plagiarism is defined as any one or the combination of the following :

1. Uncredited verbatim copying of individual sentences, paragraphs or illustrations (such as graphs, diagrams, etc.) from any source, published or unpublished, including the internet.
2. Uncredited improper paraphrasing of pages or paragraphs (changing a few words or phrases, or rearranging the original sentence order).
3. Credited verbatim copying of a major portion of a paper (or thesis chapter) without clear delineation of who did or wrote what. (Source: IEEE, the Institute, Dec. 2004)

I have made sure that all the ideas, expressions, graphs, diagrams, etc., that are not a result of my work, are properly credited. Long phrases or sentences that had to be used verbatim from published literature have been clearly identified using quotation marks.

I affirm that no portion of my work can be considered as plagiarism and I take full responsibility if such a complaint occurs. I understand fully well that the guide of the thesis may not be in a position to check for the possibility of such incidences of plagiarism in this body of work.

Place: NIT Rourkela
Date: June 2, 2014

Apurva Sachan
M.Tech., CSE Department
NIT Rourkela, Odisha

Abstract

Fully homomorphic encryption has long been viewed as cryptography's prized "holy grail" amazingly helpful yet rather subtle. Starting from the breakthrough invention of FHE in 2009 by Craig Gentry, numerous schemes are presented then by various authors following the Gentry's blueprint.

We discuss the basic homomorphic encryption given by the DGHV over the integers. It is modification of the Gentry's scheme which is based on the ideal lattices. The main idea of the DGHV scheme is its simplicity for the arithmetic operations. Our plan is to reduce the size of the public key which ultimately reduces the space complexity of the algorithm. We then further introduces the concept of the approximate common divisor problem on the DGHV scheme.

We propose the GACD attack over the modulus switching and public key compression technique of DGHV scheme. The overall contribution of this work is analysis, design and performance of the scheme.

Keywords: *Homomorphic encryption; Fully Homomorphic Encryption; DGHV scheme; leveled DGHV scheme; approximate common divisor problem*

Contents

List of Tables

List of Algorithms

Chapter 1

Introduction

Chapter 1

Introduction

The essential necessity for a cryptosystem is that adversaries must be prohibited from learning messages which is confidential. An encryption scheme is said to be homomorphic scheme that can be defined as the operation performed on ciphertext and generate the result in an encrypted form which is same as the operation performed on the respective plain text. The term privacy homomorphism is firstly, introduced by the Rivest, Adleman and Dertouzous [2] in a little while after the discovery of RSA algorithm [3]. RSA scheme was multiplicative homomorphic scheme that is it computes the product of ciphertext and equals to the product of the plain texts originally. It does not support the addition operation. The first fully homomorphic encryption was introduced by the Gentry [4] in 2009. Gentry originally achieved " Somewhat Encryption Scheme " limited to few operations to perform in ciphertext. The operations are limited because the noise is attached with the ciphertext and increases with each operation. He makes a cryptosystem with the usual encryption and decryption functions, which change bits from ciphertext from plaintext and vice-versa. He also gave an idea of evaluate function that accepts a description of a operation to be performed on the ciphertext. The problem is that ciphertext data are corrupted with numerical "noise" slight discrepancies from their absolute values. Every arithmetic operation increases the noise and it needs to refresh the ciphertext as the noise crosses the certain threshold. Homomorphic encryption would address the worry about protecting the data against adversaries and even hiding it from the cloud service provider. At the point when encryption is utilized just to make a protected interchanges channel, it has no immediate impact on the efficiency of calculations done at either end of the connection. Homomorphic encryption is defined where cryptosystem becomes the computing platform, and any inefficiency slows the entire process.

1.1 Homomorphic Encryption in Cloud computing

A number of areas are there in cloud computing, such as medical, financial and advertising sector where the services of the cloud computing can be implemented. Large amount of data is stored in the cloud database just because the user doesn't have the large space capacity and computational platform. The data stored is so large, so that user does not want to store and perform any computation locally. So the user prefers to use cloud storage and computation. Here the homomorphic plays very important role as the user want to use the cloud services, but does not want the cloud provider to access user's data. Homomorphic encryption technique provides the way to perform the arithmetic operation like addition and multiplication on encrypted data.

1.2 Homomorphic Encryption Technique

The real problem of the somewhat homomorphic encryption is the "noise" is attached with the ciphertext. The source of the noise lies in the probabilistic encryption process. Every arithmetic operation amplifies the noise and produces error during the decryption.That is why Somewhat Homomorphic Encryption technique supports only few operations. As the number of operations, i.e. multiplication and addition are increased, the noise related to the ciphertext is increased. Decryption is failed when the noise exceeds the certain threshold of the noise. Gentry proposed that the functions which compute the operation on the ciphertext are polynomials of small and bounded degree. Approximately speaking, each homomorphic addition operation doubles the noise in the ciphertext, and each multiplication squares it. Therefore number of operations on the ciphertext must be limited or decryption operation produces the incorrect result. The technique is said to be

“somewhat homomorphic” when there is limit on ciphertext depth. To avoid the problem of noise in the ciphertext, Craig introduces the technique “bootstrappable” to convert the scheme into the Fully Homomorphic Scheme.
When arithmetic operation such as addition and multiplication can be performed implicitly on the ciphertext then the technique is called fully homomorphic. In Gentry technique, there is usual *encrypt* and *decrypt* for the encryption and decryption of data respectively. There is also one more function called*evaluate* to perform the arithmetic operation on ciphertext. The *evaluate* function is having a circuit, where input symbols are given through the cascade of logic gates which perform operation on the ciphertext. In principal, any computable function can be expressed in terms of the boolean circuit of arbitrary depth. The depth of the circuit can be defined as the longest path from the input to the output. The main idea behind the depth limit of the circuit is that when the noise associated with the ciphertext cross the certain threshold, then it is decrypted and again, it is encrypted so that the noise again comes to the original level. In this way computer can perform any number of arithmetic operations and can handle the circuit of any depth. The resulting scheme is called Fully Homomorphic Encryption.
The major application of FHE is cloud computing. By this way, user can store his/her data in encrypted form in public cloud without letting know the real data. Cloud is having more storage and computing capabilities then user’s system. So the computation can be done in cloud with the help of FHE without the knowledge of secret key to the cloud administrator. More precisely, FHE is having the following property whenever f is a function composed of addition and multiplication operation in the ring:

$$Decrypt\,(f\,(c_1,\ldots,c_t)) = f\{m_1,\ldots,m_t\} \tag{1.1}$$

On the off chance that the cloud (or an adversary) can proficiently compute $f(c_1,\ldots,c_t)$ from ciphertexts $c_1,\ldots,c_t$, without realizing any data about the relating plaintexts $m_1,\ldots,m_t$, then the framework is proficient and secure.
An another prerequisite for FHE is that the ciphertext sizes stay remain bounded, independent of the function f ; this is known as the ”compact ciphertexts” pre-

requisite.

1.3 Motivation

Cloud processing security difficulties and its additionally an issue to numerous researchers; first necessity was to center on security which is the greatest concern of organizations that are recognizing a move to the cloud. Our proposal is to provide the scheme to perform the arithmetic computation on the encrypted data present in the cloud without any knowledge accessed to the cloud service provider. In this paper we tried to achieve the fully homomorphic encryption which can perform unlimited arithmetic operations on the ciphertext.

1.4 Outline of Thesis

The rest of the thesis is organized as follows:
Chapter 2 contains the brief description of the work already done in the homomorphic encryption field. We describe the recent development of homomorphic encryption and security issues are also studied.
Chapter 3 contains the brief review related to the field of fully homomorphic encryption. First we described the basic of the fully homomorphic encryption which is explained in Gentry's [4] work. Then we discuss about the DGHV [6] FHE scheme over integers given Dijk, Gentry, Halevi, and Vaikuntanathan and further modification done in this scheme [7] given by the Coron, Mandal, Naccache, and Tibouchi. We also describe the security issues in this scheme.
Chapter 4 contains the results and performance of the existing work done in the field of the homomorphic encryption using the DGHV scheme [5] using Modulus Switching and Public Key Compression technique. Then we implemented the GACD attack of Chen and Nguyen [8] on the existing scheme as my proposed work. Finally i present the results of my implementation on the SageMath.
Chapter 5 summarizes the contribution in the field of the homomorphic encryption technique. There is also the comparison of our work with the DGHV scheme.
Chapter 6 We additionally depict the possible future expansions to our work.

Chapter 2

Literature Review

Chapter 2

Literature Review

Rivest, Adleman, and Dertouzos [2]were the first to give the idea of fully homomorphic encryption, which they termed as "privacy homomorphism", and they suggested a few applicant plans. Basic RSA is the first homomorphic scheme, given that $c_1 = p_1^e modN$ and $c_2 = p_2^e modN$, and one can compute $c = c_1c_2 = (p_1p_2)^e modN$ which encrypts the product of the original plaintexts. However, RSA is not semantically secure but it is deterministic algorithm . Despite the fact that RSA is not semantically secure but still it's multiplicative property still is used in many applications.

The first scheme that gives the idea of semantic security by Goldwasser-Micali [9] in 1982. In this paper he introduced the notion of Probabilistic encryption instead of trapdoor function. The GM encryption scheme supports addition of encrypted bits mod 2 (that is, the exclusiveOR function). It becomes easy to decrypt the data at the receiver end but difficult for the adversary. Some other additively homomorphic schemes are also proposed with proof of semantically security such as Benaloh [10], Naccache-Stern [11] and Paillier [12].

RSA [2] is multiplicative homomorphic encryption technique while Elgamal [13] is additive homomorphic technique. This paper presents frameworks that depend on the difficulty of computing logarithms over finite fields. The security of scheme is equivalent to that of the distribution scheme. It also provide the comparison of the Elgamal to the RSA scheme. Some other schemes such as BonehGoh-Nissim [14] are proposed that are semantically secure as well as can perform computation of both addition and multiplication. This scheme allows the

computation of quadratic formulas over the ciphertext. The other scheme by Fellows and Koblitz [15] which also allows the computation of arbitrary circuit over the ciphertext. But the problem with the both schemes is exponentially expansion the ciphertext with the increase in depth of the circuit.
It is also proved that one can develop additively homomorphic encryption scheme from lattices [16], [17], [18]. In lattice based scheme [18], we define a chained encryption scheme which permit an effective evaluation of polynomials of degree d over encrypted data. This system also permits the generation of the ciphertext at the monetary value of the exponential increase of the ciphertext. These schemes are different from the other conventional scheme because of the "noise" attached with the ciphertext and it grows as the as operations are performed on the ciphertext. The exponential growth of the noise as the operations are performed on the ciphertetxt makes these schemes inefficient. There should be an algorithm that must bound the ciphertext growth.
A MIT CSAIL technical report "Interval Obfuscation" [19] to be published in 2009 which can be considered as symmetric homomorphic encryption. The brief description of this report is given to the Fully Homomorphic Encryption technique [4] given by Craig gentry. It uses a secret integer modulus M and a secret integer s that is relatively prime to M. A $s.xmodM$ is an encryption of '0' for some $x \in [1, a]$ where $'a'$ is a small integer. While $s.xmodM$ is an encryption of '1' for some $x \in [b+1, b+a]$ where $'b'$ is a large integer. The receiver can decrypt c by setting $c' \leftarrow c/s^d modM$ and then $\lfloor c'/b^d \rceil$ as output.This idea is also termed as ideal lattice of one dimensional.The somewhat homomorphic encryption is based same idea, but instead using one dimensional lattice it uses n-dimensional lattice.
Ishai and Paskin [20] gave an idea to evaluate branching programs with smaller ciphertext. It is based on public key encryption scheme where there is encrypted based on branching program P. The plain text x is having the ciphertext c then it is easy to compute c' from which we can easily decode the $P(x)$ using the decrypted with the use of public key. The length of the ciphertetxt is directly proportional to the branching program P and the plaintext x.

There are so many homomorphic encryption schemes are proposed since after the discovery of "privacy homomorphism" but it all were hindered due to the exponential growth of the ciphertext.The major breakthrough was achieved in 2009 by the Craig gentry in his PhD thesis fully homomorphic scheme [4] affirmed by IBM on June 25, 2009. It was based on the lattice based cryptography. His scheme is to solve the problem of arbitrary depth circuits while performing the unlimited operations on the. The development of scheme begins from a to a somewhat homomorphic encryption scheme utilizing ideal lattices that is constrained to evaluating low-degree polynomials over encrypted data. He then demonstrates to enhance his scheme to make it bootstrappable specifically, he indicates that by modifying homomorphic encryption slightly, it can evaluate its own particular decrypting circuit which is a self-referential property. He also proved that any bootstrappable function to some degree homomorphic encryption scheme could be changed over into a completely homomorphic encryption. In the specific instance, gentry gave the idea that if the noise associated with the ciphertext crosses the certain threshold then the ciphertext needs to be refreshed. The refresh process brings the noise to the original level and allows to perform the further addition and multiplication on the new refreshed ciphertext. Fully homomorphic scheme is based on the security of his plan on the assumed hardness of two problems: certain worst-case scenario problems over ideal lattices, and the sparse (or low-weight) subset sum problem.
Gentrys scheme turn out to have inherent efficiency limitations. It turns out that the barrier in practical deployments of FHE is the per-gate evaluation time, defined as the ratio of the time it takes to evaluate a circuit C homomorphically to the time it takes to evaluate C on plaintext inputs. The approximate time taken to evaluate is $\omega(\lambda^4)$, which is fairly long time complexity. After the Craig's work, other schemes are also published on the basis of Craig's idea. Other work done which further reduces the FHE problem to the integers given by van Dijk, Gentry, Halevi and Vaikuntanathans(DGHV scheme) [6] over the integers. This scheme is simpler than the Gentry's scheme because it works on integers rather than lattices.

A little while ago, scheme was proposed in [7] that the public key elements are reduced to small subset. Further the elements of public key can be obtained by combining the small subset elements multiplicatively. In particular Brakerski and Vaikuntanathan [21] shows that the hardness of the FHE can also be implemented using the "learning with error" introduced by Regev [22]. It is hard as to solve the problem of shortest vector problem on arbitrary lattice in worst case. Our paper presents new reduction technique and compression of the key and ciphertext [5] to reduce the space and time complexity of the decryption algorithm.

The simplest scheme among what we discussed till now is DGHV [6] published at EUROCRYPT'10. The security of this scheme is based on hardness of approximate integer common divisors problems in 2001 by Howgrave-Graham [23]. There are two versions of the problem is given GACD and PACD. GACD is defined as the general version of the problem as well as PACD is defined as partial version of the problem.

In GACD, the goal of the problem is to recover the secret number p, given polynomially many near multiples $x_0, \cdots, x_m$ of p, that is, each integer x_i is of the hidden form $x_i = pq_i + r_i$ where each q_i is vey large integer and each r_i is very small integer. Whereas in PACD, the setting is exactly same, except that x_0 is chosen as an exact multiple of p, namely $x_0 = pq_0$ where q_0 is very large integer chosen such that no non-trivial factor of x_0 can be found efficiently; for instance, [7] selects q_0 as a rough number, i.e. without small prime number.

The hardness of the approximate integer common divisors lies on the strength how q_i and r_i is generated. For the generation of [6] and [7], noise of $r_i's$ should be small. Because of the small value of the noise, the best known attack is gcd exhaustive search. In [7] and [6], GACD will try every noise of pair (r_0, r_1) and check whether $\gcd(x_0 - r_0, x_1 - r_1)$ is sufficiently large and allows to recover the secret key. There are various approaches are given to break the FHE over the integers.

Chapter 3

Fully Homomorphic Encryption over integers

Chapter 3

Fully Homomorphic Encryption over integers

3.1 Introduction

Our parameters and definitions are adapted by Gentry [4]. The encryption method is homomorphic with respect to the boolean circuits with the addition and multiplication mod 2. The scheme S consists of four algorithms as Key Generation, Encryption, Decryption and Evaluate. The Evaluate algorithm inputs parameters public key pk, circuit C and tuple of ciphertext $\{c_1, \ldots, c_t\}$ as input and gives another ciphertext c as output.

Definition 1 (Homomorphic Encryption). The scheme S=(KeyGen, Encrypt, Decrypt, Evaluate) is homomorphic for a problems P of circuits if it is satisfy for all circuits C $\in \mathcal{C}$. ε is fully homomorphic if it is satisfy for all boolean circuits. *Circuit-privacy* and *compactness* are two important properties of the homomorphic encryption scheme.

Circuit privacy communicates the property that the ciphertext transformed by Evaluate should not give any thought regarding the plaintext or the circuit that evaluate beyond the output value of that circuit even the client who have the learning of the private key. Compactness communicates the property that the ciphertext prepared by Evaluate ought not rely on the circuit C.

3.2 Bootstrappble Encryption

The definition adapted from the Gentry [4] described that circuit of any depth is able to evaluate the perform arithmetic operation as well as the decryption circuit.

Definition 2 (Augmented Decryption Circuit.) Let ε=(KeyGen, Encrypt, Decrypt, Evaluate) be an encryption scheme, circuit $\mathcal{C}$ is used to implement the decryption algorithm and it is depend upon the security parameter λ. We denote this set by $D_{\varepsilon}(\lambda)$.

3.3 A Somewhat Homomorphic Encryption Scheme

Parameters. A somewhat homomorphic encryption has many parameters which is calculated based on the constraints to avoide various attacks. It controls the number of bits in public key, secret key and other other variables. Specifically, there are four parameters which are suggested by Dijk, Gentry, Halevi, and Vaikuntanathan [7] in the DGHV scheme are as follows:

λ is the bit-length of the integers in the public key,

η is the bit-length of the secret key (which is the hidden approximate-gcd of all the public-key integers),

ρ is the bit-length of the noise (i.e., the distance between the public key elements and the nearest multiples of the secret key), and

τ is the number of integers in the public key.

The former parameters are having the following constraints:

- $\rho = \omega(log\lambda)$, to protect against brute-force attacks on the noise;
- $\eta \geq \rho.\theta(\lambda log^2\lambda)$, in order to support homomorphism for deep enough circuits to evaluate the "squashed decryption circuit".
- $\gamma = \omega(\eta^2 log\lambda)$, to thwart various lattice-based attacks on the underlying approximate-gcd problem.

- $\tau \geq \gamma + \omega(log\lambda)$, in order to use the leftover hash lemma in the reduction to approximate gcd.

We also use a secondary noise parameter $\rho' = \rho + \omega(log\lambda)$. For a specific η-bit odd positive integer p, we use the following distribution over γ-bit integers:

$$D_{\gamma,\rho}(p) = \{choose\ q \overset{\$}{\leftarrow} \mathbb{Z} \cup [0, 2^{\gamma}/p),\ r \overset{\$}{\leftarrow} \mathbb{Z} \cup (-2^{\rho}, 2^{\rho}) \colon x = pq + r\}$$

This distribution is clearly efficiently sampleable.
Construction. The construction of the scheme is given as follows:

KeyGen(1^{λ}): The public key is p which is random prime number of η bits. We have to sample $D_{\gamma,\eta}(p)$ to sample the values of x_i for $i = 0, 1, \ldots, \tau$. Recalculate x_i so that x_0 is the largest. and $[x_0]$ $mod\ p$ is even. Now $\{x_0, x_1, \ldots, x_{\tau}\}$ is public key called pk and p is the secret key called sk.

Encrypt($pk, m \in \{0, 1\}$): The output c for encryption of the plaintext bit $m \in \{0, 1\}$ is

$$c = [m + 2r + 2\sum_{i \in S} x_i]_{x_0} \tag{3.1}$$

where $S \in \{1, 2, \ldots, \tau\}$ and $r \in (-2^{\rho'}, 2^{\rho'})$ is a random integer.

Evaluate(pk,C,$c_1, c_2, \ldots, c_t$): The public key and the tuple of ciphertext are given to the circuit C to perform the arithmetic operations such as addition and multiplication on the integers. The circuit-privacy and compactness property of circuit are maintained.

Decrypt(sk, c): m is the output and can be obtained as follows:

$$m \leftarrow [c]_p\ mod\ 2 \tag{3.2}$$

This gives the idea of the DGHV scheme [7]. The scheme is somewhat homomorphic encryption and it is limited to few operations such as addition and multiplication only a few times. According to the DGHV scheme, ciphertext's noise must remain less than p for accurate decryption of the ciphertext so that the scheme roughly follows η/ρ' multiplications on the cipheretxt.

3.4 Limitations of DGHV scheme

The DGHV scheme [7] is over the integers. But still it has many limitations such as the large memory requirement for storing the public key. We have to improve the efficiency of the scheme as well as preserving the hardness of the approximate-GCD problem.

3.5 The new DGHV scheme

Compression of Public Key The main aim of our scheme to reduce the public key and ciphertext for better space complexity [5]. In DGHV scheme, public key is the set of $1, 2, \ldots, \tau$ tuples as follows:

$$x_i = p.q_i + r_i$$

In new DGHV scheme [5], we store the small set of η- bit integers instead of storing γ-bit integers which is comprised of set of x_i elements. The scheme can also described as generating $x_i's$ of $\gamma - \eta$ bits. The overall reduction of memory requirement is 4.6 MB from 802 MB. During the encryption process, again we can use private key p to obtain the remaining bits from the η-bits. We also maintain the constraint that $x_i \ mod \ p$ is small to avoid the noise.

Higher degrees of Ciphertext The efficiency of the scheme can be improved by the computing the ciphertext in quadratic form rather than linear form for masking the message. So, the ciphertext can be computed as

$$c = m + 2r + 2 \sum_{1 \leq i,j \leq \beta} b_{ij}.x_{i,0}.x_{j,1} \ mod \ x_0 \tag{3.3}$$

The resulting ciphertext is quadratic than linear. The given scheme is semantically secure. The main implementation is to reduce the size of the public key. The complexity is reduced to $\mathcal{O}(\lambda^{1.5})$ from $\mathcal{O}(\lambda^3)$. There is also the number of elements $\tau = x_i$ down to $2\beta = x_{i,b}$. It is also proved by the DGHV [5] authors that by increasing the degree of the public key elements to the cubic or more of degree arbitrary fixed size d than the quadratic degree is semantically secure and further reduce the size of public key size.

Construction of new DGHV scheme: There are few modifications are done in DGHV scheme to reduce the size of the ciphertext and to improve the efficiency of the DGHV scheme. The scheme is as follows:

KeyGen$((1^\lambda))$. Choose a random prime number p and random odd integer $q_0 \in [0, 2^\gamma/p)$.
Then $x_0 = p.q_0$. Initialize the pseudo random number number generator f with seed se. $f(se)$ is used to generate the set of integers $\chi_i \in [0, 2^\gamma)$ for $i = 1, 2, \ldots, \tau$.
For $i = 1, 2, \ldots, \tau$ compute,

$$\delta_i = \langle \chi_i \rangle_p + \xi_i.p - r_i \tag{3.4}$$

where $r_i \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho)$ and $\xi_i \leftarrow \mathbb{Z} \cap [0, 2^{\lambda+\eta}/p)$. By storing the values of δ_i and the knowledge of seed se we can easily again compute the x_i. Then,

$$x_i = \chi_i - \delta_i \tag{3.5}$$

Encrypt$(pk, m \in \{0, 1\})$: The x_i is recovered by knowing the value of the seed se and δ_i. Choose a random integer $\mathbf{b} = (b_i)_{1 \le i \le \tau} \in [0, 2^\alpha)$ and a random integer $r \in (-2^{\rho'}, 2^{\rho'})$. Then output of the ciphertext as follows:

$$c = m + 2r + 2\sum_{i=1}^{\tau} b_i.x_i \ mod \ x_0 \tag{3.6}$$

Evaluate(pk,C,$c_1, c_2, \ldots, c_t$) and Decrypt(sk, c): The algorithm for Evaluate and Decrypt is following the DGHV scheme except that modulo is x_0 for the ciphertext.

3.6 Fully Homomorphic Scheme using DGHV scheme

The scheme does not include the "bootstrapping" technique. The parameters are evaluated polynomially on the basis of depth of the circuit.
SwitchKeyGen This algorithm defines the compression technique of the ciphertext and keys. The input parameters are pk, sk, pk', sk'. The two secret keys p and p' are taken of size η and η'.

1. Let $k = 2\gamma + \eta$ where γ is size of public key integers.
2. Compute a vector Y of θ random numbers modulo $2^{\eta'+1}$ with k bits of precision after the binary point, and a random vector S of θ bits. Then calculate the expanded secret-key $S' = Powerof2(S, \eta')$
3. Calculate the encryption of d of S' under sk'. d can be computed as
$$d = p'.q + r + \lfloor S'.\frac{p'}{2^{\eta'+1}} \rceil \tag{3.7}$$
4. Output $\tau_{pk \to pk'} = (Y, d)$

Fully Homomorphic Scheme

The depth of the circuit is defined as L and security parameter is defined as λ.

KeyGen(1^λ,1^L) For each level of L in the circuit compute L decreasing moduli of size η_i. For each $\eta_i = (i+1)\mu$ from L to 1, compute KeyGen(1^λ) from DGHV scheme. From $j = 1, \ldots, 2$ compute SwitchKeyGen($pk_j, pk_{j-1}, pk'_j, pk_{j-1}$) for $\tau_{pk_j \to pk_{j-1}}$. Now the public key for the full scheme is $pk = (pk_L, \tau_{pk_L \to pk_{L-1}}, \ldots, \tau_{pk_2 \to pk_1})$ and secret key is $sk = (p_1, \ldots, p_L)$.

Encrypt($pk, m \in \{0, 1\}$) Apply Encrypt(pk_L, m) for a level L.

Decrypt(sk,c) Suppose the moduli for that level is p_j. Then ciphertext is calculated as $m \leftarrow [c]_{p_j}\ mod\ 2$.

Add(pk, c_1, c_2) Suppose that two ciphertexts c_1 and c_2 are encrypted under p_j then apply the add operation otherwise apply the Refresh function below to make it so.

Refresh($\tau_{pk_j \to pk_{j-1}}$), c unless both the ciphertexts are encrypted are under p_j and if it so then it simply output ciphertext c.

Mult(pk, c_1, c_2) Suppose that two ciphertexts c_1 and c_2 are encrypted under p_j then apply the multiply operation otherwise apply the Refresh function below to make it so.

Refresh($\tau_{pk_j \to pk_{j-1}}$), c unless both the ciphertexts are encrypted are under p_j and if it so then it simply output ciphertext c.

Refresh($\tau_{pk_{j+1} \to pk_j}$), c output $c' \leftarrow SwitchKey(\tau_{pk_{j+1} \to pk_j}, c)$

Chapter 4

Attacks on Homomorphic Encryption Technique

Chapter 4

Attacks on Homomorphic Encryption Technique

The Dijk, Gentry, Halevi and Vaikuntanathan presented FHE scheme based on the hardness of the approximate integer common divisor problems. There are two versions of this problems: the partial version(PACD) and general version(GACD). The hardness of the PACD and GACD depends upon the $q_i's$ and the $r_i's$ are generated.

4.1 New Square Root Algorithm for PACD

In this section, we describe the new square-root algorithm for the PACD problem, which is based on the univariate polynomials at many points.

4.1.1 Overview

Consider $x_0 = pq_0$ and $x_i = pq_i + r_i$ where $0 \leq r_i \leq 2^\rho$, $1 \leq i \leq m$. We start with the following:

$$p = gcd(x_0, \prod_{i=0}^{2^\rho-1}(x_1 - i)(mod\ x_0) \tag{4.1}$$

This allows 2^ρ gcd computations with essentially 2^ρ modular multiplications. We define the polynomial $f_i(x)$ of degree j, with coefficients modulo x_0 :

$$f_i(x) = \prod_{i=0}^{j-1}(x_1 - (x+i))(mod\ x_0) \tag{4.2}$$

Letting $\rho' = \lfloor \rho/2 \rfloor$,we notice that :

$$\prod_{i=0}^{2^{\rho}-1} (x_1 - 1) \equiv \prod_{k=0}^{2^{\rho'+(\rho mod 2)}-1} f_{2^{\rho'}}(2^{\rho'}k)(mod\ x_0)$$

We can thus write(4.1) as :

$$p = gcd(x_0, \prod_{k=0}^{2^{\rho'+(\rho mod 2)}-1} f_{2^{\rho'}}(2^{\rho'}k)(mod\ x_0)) \tag{4.3}$$

Clearly, (4.3) allows to solve PACD using one gcd, $2^{\rho'+(\rho mod\ 2)} - 1$ modular multiplications, and multi-evaluation of the polynomial of degree $2^{\rho'}$ at $2^{\rho'+(\rho\ mod\ 2)}$ points, where $\rho' + (\rho\ mod\ 2) = \rho - \rho'$. It claims at the cost of the $\mathcal{O}(2^{\rho'}) = \mathcal{O}(\sqrt{2^{\rho}})$ operations modulo x_0, which is essentially the square root of gcd exhaustive search.

4.1.2 Description

The following algorithm to solve PACD, given as Algo 1 :

Algorithm 4.1 Solving PACD by multipoint evaluation of univariate polynomials

```
1: Input: An instance $(x_0, x_1)$ of the PACD problem with noise size $\rho$.
2: Output: The secret number p such that $x_0 = pq_0$ and $x_1 = pq_1$ with appropriate sizes.
3: set $\rho' \leftarrow \lfloor \rho/2 \rfloor$
4: Compute the polynomial $f_{2^{\rho'}}(x)$ defined by 4.2, using Alg.2.
5: Compute the evaluation of $f_{2^{\rho'}}(x)$ at the $2^{\rho'+(\rho\ mod\ 2)}$ points, $0, 2^{\rho'}, \cdots, 2^{\rho'}(2^{\rho'+(\rho\ mod\ 2)} - 1)$, using $2^{\rho\ mod\ 2}$ times.

   Alg.3 with $2^{\rho'}$ points. Each application of Alg. 3 requires the computation of a product tree, using Alg. 2.
```

Alg. 1 relies on two classical subroutines :

- a subroutine to (efficiently) compute a polynomial given as a product of n terms, where n is a power of two: Alg. 2 does this in $\mathcal{O}(n)$ ring operations, provided that quasi-linear multiplication of polynomials is available, which can be achieved in our case using Fast Fourier technique. This subroutine is used in Step 2. The efficiency of Alg. 2 comes from the fact that when the algorithm requires a multiplication, it only multiplies polynomials of similar degree.

- a subroutine to (efficiently) evaluate a univariate degree-n polynomial at n points, where n is a power of two: Alg. 3 does this in $\mathcal{O}(n)$ ring operations, provided that quasi-linear polynomial remainder is available, which can be achieved in our case using Fast Fourier techniques. This subroutine is used in Step 3, and requires the computation of a tree product, which is achieved by Alg. 2. Alg. 3 is based on the well-known fact that the evaluation of a univariate polynomial at a point α is the same as its remainder modulo $X - \alpha$, which allows to factor computations using a tree.

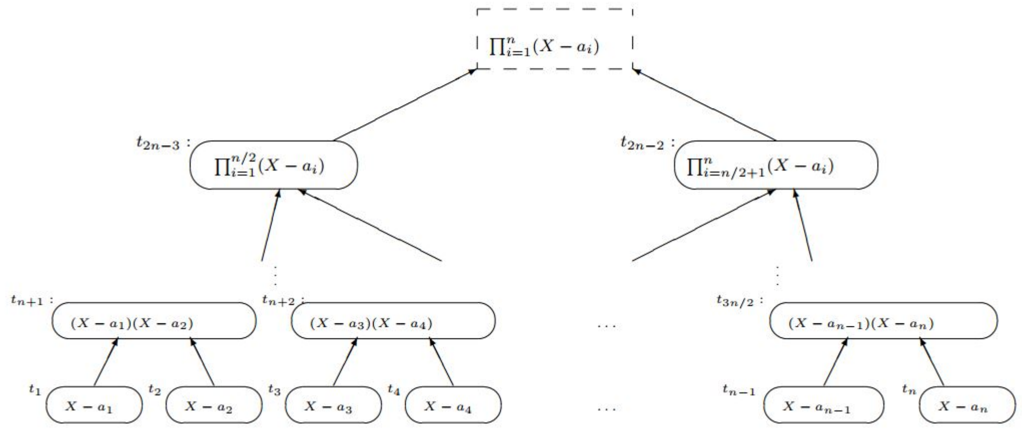

Figure 4.1: Polynomial Product Tree $T = t_1, \cdots, t_{2n}$ for $a_1, \cdots, a_n$

Algorithm 4.2 $[T, D] \leftarrow TreeProduct(A)$

```
1: Input: A set of $n = 2^l$ numbers $a_1, a_2, \cdots, a_n$.
2: Output: The polynomial product tree $T = t_1, t_2, \cdots, t_{2n-1}$, corresponding to
   the evaluation of points $A = a_1, a_2, \cdots, a_n$ as shown in figure 4.1. $D =$
   $[d_1, \cdots, d_{2n-1}]$ descendant indicates for non-leaf nodes or 0 for leaf nodes.
3: for $i = 1 \cdots n$ do
4:     $t_i \leftarrow X - a_i$Initializing leaf nodes
5:     $d_j \leftarrow 0$
6: end for
7: $i \leftarrow 1$Index of lower level
8: $j \leftarrow n + 1$Index of upper level
9: while $j \leq 2n - 1$do
10:     $t_j \leftarrow t_i \cdot t_{i+1}$
11:     $d_j \leftarrow i$
12:     $i \leftarrow i + 2$
13:     $j \leftarrow j + 1$
14: end while
```

Algorithm 4.3 $V \leftarrow RecursiveEvaluation(f, t_i, D)$

```
1: Input: A polynomial $f$ of degree $n$.    A polynomial product tree rooted at
   $t_i$, whose leaves are $X - a_k, \cdots, X - a_m$
2: Output: $V = f(a_k), \cdots, f_(a_m)$
3: if $d_i = 0$ then
4:     return $\{f(a_i)\}$ When $t_i$ is a leaf, we apply an evaluation directly.
5: else
6: $g_i \leftarrow f\ mod\ t_{d_i}$  {left subtree}
7: $V_i \leftarrow RecursiveEvaluation(g_1, t_d, D)$
8: $g_2 \leftarrow f\ mod\ t_{d_i+1}$
9: $V_2 \leftarrow RecursiveEvaluation(g_2, t_d, D)$
10: $returnV_1 \cup V_2$
```

It is concluded that the running time of Alg. 1 is $\mathcal{O}(2^{\rho'}) = \mathcal{O}(\sqrt{2^\rho})$ operations modulo x_0, which is essentially the “square root” of gcd exhaustive search.

4.2 Limitations

The main limitation of implementing Alg 1. is memory. Consider the Large FHE-challenge from [7] : there, ρ= 40, so the optimal parameter is ρ'= 20, which implies that $f_{2^{\rho'}}$ is a polynomial of degree 2^20 with coefficients of size 19×10^6 bits. In other words, simply storing $f_{2^{\rho'}}$ already requires $2^20 \times 19 \times 10^6$ bits, which is more than 2Tb, while we also need to perform various computations. This means that in practice, we will have to settle for suboptimal parameters. More precisely, assume that we select an additional parameter d, which is a power of two less than $2^{\rho'}$. We rewrite 4.3 as :

$$p = gcd(x_0, \prod_{k=0}^{2^\rho/d-1} f_d(dk)(mod\ x_0)) \tag{4.4}$$

This gives rise to the another version of Alg. 1, given as

Algorithm 4.4 $V \leftarrow RecursiveEvaluation(f, t_i, D)$

```
1: Input: An instance $(x_0, x_1)$ of the PACD problem with noise size $\rho$, and a polynomial degree $d$.
2: Output: The secret number $p$ such that $x_0 = pq_0$ and $x_1 = pq_1 + r_1$ with appropriate sizes.
3: Compute the polynomial $f_d(x)$ defined by 4.2, using Algo 2.
4: Compute the evaluation of $f_d(x)$ at the $2^\rho/d$ points $0, d, 2d, \cdots, d(2^\rho/d-1)$, using $2^\rho/d^2$ times Alg. 3 with $d$ points. Each application of Alg. 3 requires the computation of a product tree, using Alg. 2.
```

The running time Alg. 4 is $\frac{2^\rho \mathcal{O}(d)}{d^2}$ elementary operations modulo x_0, and the space requirements is $\mathcal{O}(d)$ polynomially many bits.

Chapter 5

Performance and Optimization

Chapter 5

Performance and Optimization

In this chapter we implemented the compression technique of DGHV's scheme given by van Dijk, Gentry, Halevi and Vaikuntanathan. The compression technique reduces the time complexity from $\tilde{O}(\lambda^7)$ to $\tilde{O}(\lambda^5)$. We also implemented the GCD attack on the compression technique of DGHV [5]. We then discuss about the attacks on RSA and our proposed method.

5.1 Implementation

In this section we described our scheme based on the idea of DGHV scheme in section 3.5.

Modulus switching technique used to keep the “noise” small is adapted from the [21]. More precisely, the decryption of the ciphertext c calculated from the plain text $m = \{0, 1\}$ is $[c]_p \ mod\ 2$. The term $[c]_p mod 2$ is refer as “noise” associated with the ciphertext c. In the leveled DGHV scheme, ciphertext c which is encrypted under p which can be also be efficiently encrypted ciphertext c' under p'. The resulting noise must be multiplied by p/p' to reduce the size. But for the secure FHE the value of p and p' must not reveal. So first we should compute “virtual ciphertext” as follows $c' = 2^k.q' + r'$ with $q' = q\ mod\ 2$. We use the variant with $x_0 = p.q_0$.

5.2 Parameters

Preliminaries. We use λ as a security parameter.For a real number a, we denote by $\lceil a \rceil$ for rounding up, $\lfloor a \rfloor$ for down and $\lfloor a \rceil$ for the nearest integer respectively. All logarithms are base 2 unless otherwise stated.

We use the following variant with $x_0 = pq_0$. To prevent the sparse subset sum problem in lattice based attack, we have the following constraints $\theta_2 = \gamma \cdot \omega(log\lambda)$. The constraints on other parameters are $\rho = \lambda$,$\alpha = \mathcal{O}(\lambda^2)$, $\theta = \mathcal{O}(\lambda^3)$, $\tau = \mathcal{O}(\lambda^3)$ and $\gamma = \mathcal{O}(\lambda^5)$

The concrete parameters are given by [7]. For these parameters, we take θ =15. We obtain the parameters are as follows :

Parameters	λ	ρ	η	γ	β	θ
Toy	42	16	1088	$1.6{\cdot}10^5$	12	144
Small	52	24	1632	$0.86{\cdot}10^6$	23	153
Medium	62	32	2176	$4.2{\cdot}10^6$	44	1972
Large	72	39	2652	$19{\cdot}10^6$	88	7897

Table 5.1: Concrete parameters as calculated for our implementation to protect from various attacks.

5.3 Result

We have implemented the scheme of DGHV using the compression technique of modulus-switching described in section 3.5, with an optimization of ciphertext expansion procedure. The time complexity resulting after the execution of the scheme are as follows :

Instance	KeyGen	Encryption	Decryption	Evaluate	Recryption
Toy	0.042s	0.058s	0.001s	0.01s	0.42s
Small	1.33s	1.12s	0.001s	0.14s	4.58s
Medium	29.5s	20.97s	0.03s	2.69s	55s
Large	10m 2s	7m 13s	0.09s	51s	11m 35s

Table 5.2: Time complexity of the scheme with Sage 6.1.1 [1]
(Desktop system of dual core with an Intel Core2 duo n5100 processor at 3 GHz each.)

The proposed scheme is better than the DGHV scheme in terms of time complexity. The graph in figure 5.1 is showing the following comparison :

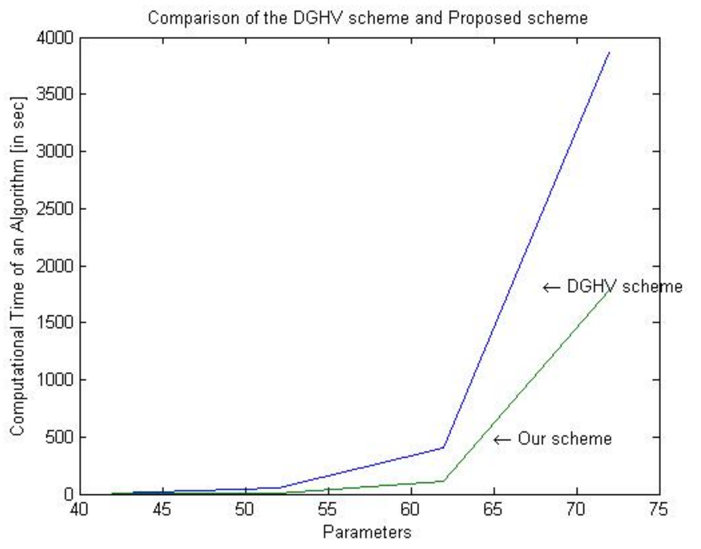

Figure 5.1: Comparison between Our proposed scheme and DGHV scheme

The proposed scheme is better than the DGHV scheme in terms of space complexity. The graph in figure 5.2 is showing the following comparison :

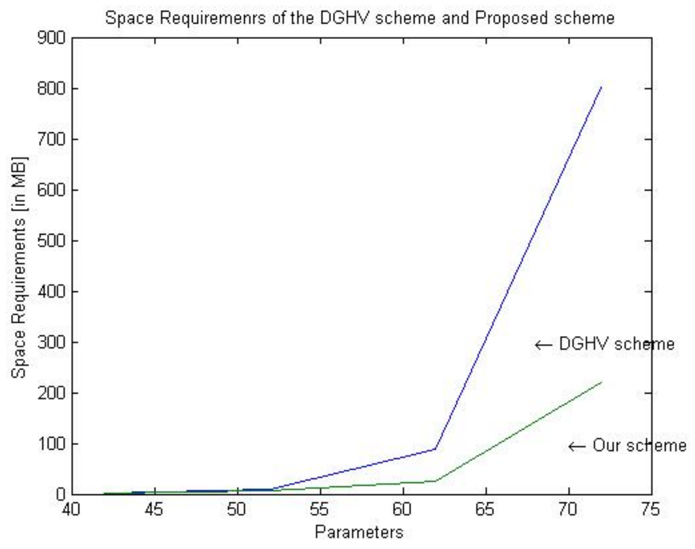

Figure 5.2: Comparison between Our proposed scheme and DGHV scheme

Chapter 6

Conclusion and Future Work

Chapter 6

Conclusion & Future Work

6.1 Conclusion

The cloud computing security built on the light of Fully Homomorphic encryption, will be another thought of security which enables giving conclusions of figurings on encrypted data without knowing the raw data on which the calculation was carried out, in profound respect of the data confidentiality. But still there practical implementation is need to be done in the future.

6.2 Future Work

Our future work is compromised of increasing the efficiency of our proposed scheme so that the computational time will be reduced.

Bibliography

[1] W. Stein, "Sage mathematics software (version 6.1.1)." `http://www.sagemath.org.`, 2010.

[2] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of secure computation*, vol. 4, no. 11, pp. 169–180, 1978.

[3] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[4] C. Gentry, *A fully homomorphic encryption scheme.* PhD thesis, Stanford University, 2009.

[5] J.-S. Coron, D. Naccache, and M. Tibouchi, "Public key compression and modulus switching for fully homomorphic encryption over the integers," in *Advances in Cryptology–EUROCRYPT 2012*, pp. 446–464, Springer, 2012.

[6] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Advances in Cryptology–EUROCRYPT 2010*, pp. 24–43, Springer, 2010.

[7] J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi, "Fully homomorphic encryption over the integers with shorter public keys," in *Advances in Cryptology–CRYPTO 2011*, pp. 487–504, Springer, 2011.

[8] Y. Chen and P. Q. Nguyen, "Faster algorithms for approximate common divisors: Breaking fully-homomorphic-encryption challenges over the integers," in *Advances in Cryptology–EUROCRYPT 2012*, pp. 502–519, Springer, 2012.

[9] S. Goldwasser and S. Micali, "Probabilistic encryption & how to play mental poker keeping secret all partial information," in *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pp. 365–377, ACM, 1982.

[10] J. Benaloh, *Verifiable secret-ballot elections.* PhD thesis, PhD thesis, Yale University, 1987.

[11] D. Naccache and J. Stern, "A new public key cryptosystem based on higher residues," in *Proceedings of the 5th ACM conference on Computer and communications security*, pp. 59–66, ACM, 1998.

[12] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in cryptologyEUROCRYPT99*, pp. 223–238, Springer, 1999.

[13] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Advances in Cryptology*, pp. 10–18, Springer, 1985.

[14] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-dnf formulas on ciphertexts," in *Theory of cryptography*, pp. 325–341, Springer, 2005.

[15] M. Fellows and N. Koblitz, "Combinatorial cryptosystems galore!," *Contemporary Mathematics*, vol. 168, pp. 51–51, 1994.

[16] S. Goldwasser and D. Kharchenko, "Proof of plaintext knowledge for the ajtai-dwork cryptosystem," in *Theory of Cryptography*, pp. 529–555, Springer, 2005.

[17] A. Kawachi, K. Tanaka, and K. Xagawa, "Multi-bit cryptosystems based on lattice problems," in *Public Key Cryptography–PKC 2007*, pp. 315–329, Springer, 2007.

[18] C. A. Melchor, P. Gaborit, and J. Herranz, “Additively homomorphic encryption with d-operand multiplications,” in *Advances in Cryptology–CRYPTO 2010*, pp. 138–154, Springer, 2010.

[19] M. van Dijk and S. Devadas, “Interval obfuscation,” *as an MIT-CSAIL Technical Report in*, 2009.

[20] Y. Ishai and A. Paskin, “Evaluating branching programs on encrypted data,” in *Theory of Cryptography*, pp. 575–594, Springer, 2007.

[21] Z. Brakerski and V. Vaikuntanathan, “Efficient fully homomorphic encryption from (standard) lwe,” in *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pp. 97–106, IEEE, 2011.

[22] O. Regev, “On lattices, learning with errors, random linear codes, and cryptography,” *Journal of the ACM (JACM)*, vol. 56, no. 6, p. 34, 2009.

[23] N. Howgrave-Graham, “Approximate integer common divisors,” in *Cryptography and Lattices*, pp. 51–66, Springer, 2001.