12-20-2018

Efficient Computation and FPGA implementation of Fully Homomorphic Encryption with Cloud Computing Significance

Qiang Zeng
University of Windsor

Efficient Computation and FPGA Implementation of Fully Homomorphic Encryption with Cloud Computing Significance

by

Qiang Zeng

A Thesis

Submitted to the Faculty of Graduate Studies

through Electrical and Computer Engineering

in Partial Fulfillment of the Requirements for

the Degree of Master of Applied Science at the

University of Windsor

Windsor, Ontario, Canada

2018

Efficient Computation and FPGA Implementation of Fully Homomorphic Encryption with Cloud Computing Significance

by

Qiang Zeng

APPROVED BY:

H. Hu

Department of Mechanical, Automotive & Materials Engineering

C. Chen

Department of Electrical and Computer Engineering

H. Wu, Advisor

Department of Electrical and Computer Engineering

Oct 11, 2018

AUTHOR'S DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyones copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Homomorphic Encryption provides unique security solution for cloud computing. It ensures not only that data in cloud have confidentiality but also that data processing by cloud server does not compromise data privacy. The Fully Homomorphic Encryption (FHE) scheme proposed by Lopez-Alt, Tromer, and Vaikuntanathan (LTV), also known as NTRU(Nth degree truncated polynomial ring) based method, is considered one of the most important FHE methods suitable for practical implementation.

In this thesis, an efficient algorithm and architecture for LTV Fully Homomorphic Encryption is proposed. Conventional linear feedback shift register (LFSR) structure is expanded and modified for performing the truncated polynomial ring multiplication in LTV scheme in parallel. Novel and efficient modular multiplier, modular adder and modular subtractor are proposed to support high speed processing of LFSR operations. In addition, a family of special moduli are selected for high speed computation of modular operations. Though the area keeps the complexity of $O(Nn^2)$ with no advantage in circuit level. The proposed architecture effectively reduces the time complexity from $O(NlogN)$ to linear time, $O(N)$, compared to the best existing works.

An FPGA implementation of the proposed architecture for LTV FHE is achieved and demonstrated. An elaborate comparison of the existing methods and the proposed work is presented, which shows the proposed work gains significant speed up over existing works.

DEDICATION

To my loving parents:

Father: Heping Zeng

Mother: Qing Ping

ACKNOWLEDGMENTS

I would like to express my faithful gratitude to everyone who helped me. First of all, I appreciate my parents' deep love and full support, as well as the encouragement and financial support from them. Without them, I could not overcome all difficulties and accomplish my study.

Furthermore, I am quite grateful to my supervisor Dr. Huapeng Wu from department of Electrical and Computer Engineering at University of Windsor. He has instructed me throughout my research and this thesis. As one of best teachers I have ever had, Dr. Wu impressed upon me that a brilliant teacher edifies students in matters far beyond those in books and academy. His extensive knowledge and logical thinking are invaluable; without his elaborate and constructive comments on my research, this thesis could be impossible.

I thank my friends, Ruiqing Dong, Liyuan Liu, Xiaolin Duan, Siyu Zhang and Pengzhao Song. They gave me their help and time during the adversity of my study.

Ultimately, I hope to show my appreciation to the faculties of Electrical and Computer Engineering at University of Windsor since their efforts during my study for the master degree. Furthermore, I pretty appreciate the financial support from the University of Windsor and my supervisor Dr. Huapeng Wu.

Qiang Zeng

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ACRONYMS

$FPGA$	Field Programmable Gate Array
$\mathbb{Z}$	The set of integers
$\mathbb{Q}$	The set of rational numbers
$\mathbb{R}$	The set of real numbers
R_q	Polynomial ring taken modulo q
$\mathbb{Z}[x]$	the set of polynomial of arbitrary degree with integer coefficients
q	Integer coefficient modulo
R	A polynomial ring
N	The degree of a polynomial
R	Ring
χ	Discrete Gaussian distribution
LFSR	Linear Feedback Shift Register
m	message encoded as polynomial (plaintext)
LTV	Lopez-Alt, Tromer, and Vaikuntanathan cryptosystem
c	Encryption result represented as polynomial (ciphertext)
HE	Homomorphic Encyption
FHE	Fully Homomorphic Encyption
PHE	Partially Homomorphic Encyption
SWHE	Somewhat Homomorphic Encyption

RSA	Ron Rivest, Adi Shamir, and Leonard Adleman public key cryptosystem
LWE	Learning with error problem
R-LWE	Ring-Learning with error problem
SVP	Shortest vector problem
NTRU	Nth degree truncated polynomial ring cryptosystem
BE	Bit Encoder
BS	Bit Selector

1 INTRODUCTION

1.1 Motivation

Internet is a worldwide electronic network providing access to millions of informational resources. The ubiquity of Internet proves that it plays an indispensable role in our daily life. Internet has revolutionized the computer and communication world like nothing before. While telegraph, telephone, radio, and computer had set stage for the unprecedented integration of capabilities, subsequent medium as wireless handheld electronic devices such as smartphone, tablet, PDA, wearable tech, are also in need of internet for next generation and future world.

With the rapid increase in the amount of Internet information and data in recent years, the processing and computing ability of current information technology infrastructure has also grown simultaneously. As of September 2018 there were over 1.8 billion web servers in the world reported estimated and reported by Netcraft[3]. According to data and marketing services company IDC, scale of Internet data is going to reach 63ZB by 2025[4]. One of the reasons why web data grow rapidly is that they are increasingly stacked up by low budget and numerous number of devices such as mobile devices, cameras, radio-frequency-identification(RFID) readers and wireless sensor networks. These small Internet of Things devices capable of information sensing and data processing provide great convenience and unlimited contents for people to access from anywhere at anytime.

To deal with the explosively growing amount of data, Internet communication bandwidth and capability of Internet based data processing are being expanded and optimized. Researchers at Bell labs have reached Internet speed of over 100 petabit $\times$ kilometer per second using fiber optic communication[5]. In addition, quantum computing technologies continues to advance, which is aimed to significantly improve upon the data processing capabilities of today's most formidable electronic machines.

With that said, people nowadays have growing needs for access of more data contents and higher amount of information along with the development of technologies such as video of 4K resolution and VR games.

When people are using their mobile phones and wearable gadgets, they also require to have access to worldwide resourses and powerful computational ability to cope with their data on handheld devices. That is when the concept of cloud computing inevitably came to draw public's attention in early 2000s[6]. In the history of network technology and communication engineering development, computing paradigm transformed from massing big load of work for large processor scheme, to distributed assignment processing scheme based on computer network, to demand processing cloud computing scheme recently. Cloud computing virtualizes and integrates service resources on Internet and has specialist responsible for coordination, management and maintenance, while clients are not necessarily knowledgeable about the internal realization of cloud. Therefore, cloud computing is a brand new effective computing paradigm.

Recent development in cloud computing field demonstrates that such a system providing massive computation and remote data management service has brought us into a revolutionary digital era. Cloud service has become a major source for personal users and enterprises when requesting a reliable system to deal with their data, usually sensitive and private. The cloud offers numerous advantages in costs and functionality with huge potential and while it is gaining rapidly rising popularity, on the other hand it also raises grave questions of cloud security and data confidentiality, since data stored in the cloud could be vulnerable to snooping by the cloud provider or even by other cloud clients[7].

The vulnerability of cloud computing security results from following reasons: Multi-tenancy and elasticity are two important feature of cloud model. Multi-tenancy is the reason why different tenants could share the same service instance. Elasticity

enables scaling up and down resources allocated to service based on current service demands. These two properties increase the attackers' interests in continually finding and taking advantage of the existing vulnerbilities for the cloud computing model. When transfering processing data of user to a third party, it is essential to be aware the responsibility associated with data privacy and compliance and bond a trust connection between user and cloud provider when exploiting benefits of reduced costs, easy maintainance and reprovisioning of resources.

Because cloud computing is distributed, the location of data stored is sometimes unknown, different user could share storage or computing resources, which might compromise the security and expose the information to other user in the same cloud model. When the data is tranfered to the cloud, standard encryption methods are used to secure the operations and the storage of data. The conventional security idea is to encrypt the data before sending it to cloud provider while handing over private key to the server to decrypt data before performing the calculations as requested by the user, the assumption that the cloud provider is trustworthy might affect the confidentiality and privacy of data stored in the cloud.

What is unique about cloud security is that clients or users demand security against cloud provider or server. Since the cloud service provider could be any third party company or enterprise, users intend to keep the data private and manage to not compromise sensitive information to cloud server. Therefore, a new strategy or countermeasure is required to provide solutions for the exclusive cloud security problem.

Homomorphic encryption systems were proposed to allow operations on encrypted data without decrypting the ciphertext, with user the only owner and holder of secret key. After decryption by the user, the result is same as the operations carried out on unencyprted data. In this way, a user gets the utmost out of cloud server's computation capability as well as obaining the expected calculation results, while cloud

provider correctly processed the data as requested under the premise of not knowing the content of user data, as well as receiving the finantial support from user.

The term *homomorphism* appeared first time in history from paper by Rivest,Adleman and Dertouzous[8] as a possible solution to computation on encrypted data and remained encrypted the whole process. This innovative approach is called homomorphic encryption. Since then, countless research works towards homomorphic encryption piled up as trending in cryptography. Homomorphic encryption provides a novel way to allow computation performed on encrypted data without having access of any portion of plain text. Homomorphic encryption scheme offers a unique security measure working decently against possible attacks and well suited with the exclusive features of cloud computing.

In order to achieve practical homomophic encryption scheme, high speed and efficient algorithm and implementation are demanded. Since the proposal of first plausible Fully homomophic Encryption scheme were given by Graig Gentry[9], numerous papers and researchs were introduced aiming the efficient realization of different method of fully homomorphic encryption system. This paper explore one of the most researched and practical fully homomophic schemes, LTV[10] proposed by Lopez-Alt, Tromer and Vaikuntanathan, and present an efficient architecture and FPGA implementation of the scheme. In addition, cloud computing system with application of fully homomophic encryption is discussed.

In this thesis we focus on efficient computation and FPGA implementation of LTV method. A new algorithm for truncated polynomial ring multiplication algorithm is proposed. Then a LFSR based architecture is designed for high speed processing of truncated polynomial multiplication. Finally a FPGA implementation of the proposed architecture is achieved with complexity compared favorably to the recent existing works.

1.2 Contribution

The contributions of this thesis are summarized below:

Conventional linear feedback shift register (LFSR) structure is expanded and modified for computation of the truncated polynomial ring multiplication in LTV cryptosystem. The proposed structure enables parallel computation while keeps compact and pipelined structure. In addition, a set of unique moduli were chosen to be used in the computation of modular operations. Calculation speed is much increased results from the highly efficient modulo computation introduced by the proposed moduli. Novel and efficient modular multiplier, modular adder and modular subtractor are proposed for high speed processing of LFSR operations. While parallel prefix adder structure provide efficient modular addition, Booth algorithm are used to perform modular multiplication. Our proposed architecture and algorithm provides better time complexity of linear time $O(N)$ compared to existing works in the literature, while maintain more or same space complexity. Time and sapce complexity of our work are analyzed and presented in details. Comparison of FPGA implementation performance and results of other related works are also given.

1.3 Organization of the Thesis

The rest of thesis is organized as follows: In Chapter 2, mathematical primitives and fundamentals are introduced, which includes ring, polynomial rings, and modular arithmetic in polynomial ring. In addition, the basic parameter sets of LTV Fully Homomorphic Encyption scheme are shown, while the steps of encryption, decryption and evaluation part is introduced and explained. In Chapter 3, a detailed introduction to Homomorphic encryption and its related research works are provided. Also, the contemporary trend of widely used cloud computing applications is analyzed. From what is studied, the model of cloud computing and the application of intergration with Homomorphic Encryption will be depicted. In Chapter 4 of the thesis, an overview

of several existing related works on LTV Fully Homomorphic Encryption scheme is scrutinized. Chapter 5 presentsf our proposed work for LTV Fully Homomorphic Encryption scheme. In this chapter, an LFSR based multiplier architecture is proposed to perform polynomial multiplication in set $R_q[x]$. Besides, proposed modular adder, modular subtractor and modular multiplier will be explicitly described. In Chapter 6, the complexities of proposed work is analyzed and summarized. The FPGA implementation results are presented along with comparision of proposed work and existing work. In Chapter 7, the conclusion and possible future work are discussed.

2 MATHEMATICAL PRELIMINARIES

This chapter first introduces truncated polynomial ring, as the mathematical background of LTV Fully Homomorphic Encryption scheme. Then, an overview of LTV Fully Homomorphic Encryption system is given, which includes key generation, encryption, decryption and evaluation. The parameter sets selection for LTV Fully Homomorphic Encryption is also discussed.

2.1 Truncated Polynomial Ring

2.1.1 Ring

Definition A *Ring* is a non-empty set R together with binary operations $+$ and $\cdot$ defined on R with two mappings:

$$+ : R \times R \to R, \quad (a, b) \mapsto a + b$$
$$\cdot : R \times R \to R, \quad (a, b) \mapsto a \cdot b$$

such that addition operation satisfies:

Additive associativity: $a + (b + c) = (a + b) + c$, for a, b, c, $\in$ R.

Additive communitivity: $a + b = b + a$, for a, b $\in$ R.

Additive identity: There is an identity element e in R such that for all $a \in R$, $a + e = e + a = a$.

Additive inverse: For $a \in R$, there exists an inverse element $a^{-1} \in R$ such that $a + a^{-1} = a^{-1} + a = e$.

Also multiplication operation satisfies:

Multiplicative associativity: For $a, b, c \in R$, we have $a \cdot (b \cdot c) = (a \cdot b) \cdot c$.

Distributivity: $a \cdot (b + c) = a \cdot b + a \cdot c$, and $(a + b) \cdot c = a \cdot b + b \cdot c$.

Examples of ring are populous and here is a few of them:

All of $\mathbb{Z}, \mathbb{Q}, \mathbb{R}$ and $\mathbb{C}$ under the operations of addition and multiplication are commutative rings with identity (with the number 1 as the identity); Under the usual

matrix addition and matrix multiplication for collection of matrices over some ring R form a matrix ring; Modulo 2 arithmetic on $0, 1$ is a two element commutative ring with identity. More generally, for addition and multiplication modulo n on $0, 1, ..., n-1$ considered, we also get a commutative ring with identity; Polynoimials with real coefficients form a commutative ring with identity under the usual addition and multiplication.

2.1.2 Truncated polynomial ring

Consider the polynomial ring that contains all the polynomials with degree up to a certain value $N-1$ taken modulo $\phi(x) = x^N + 1$ and with integer coefficients. This is called truncated polynomial ring (with integer coefficients) and denoted by $R[x]/(x^N+1)$. Let $a(x), b(x) \in R[x]/\phi(x)$:

$$a(x) = a_0 + a_1 x + ... + a_{N-1} x^{N-1} \tag{1}$$

$$b(x) = b_0 + b_1 x + ... + b_{N-1} x^{N-1} \tag{2}$$

We must able to add and multiply elements of $R[x]/\phi(x)$ so that the axioms defining a ring hold true.

Here are some basic properties of truncated polynomial ring:

- Equality

  If $a(x) a_0 + a_1 x + ... + a_{N-1} x^{N-1}$ and $b(x) = b_0 + b_1 x + ... + b_{N-1} x^{N-1}$ are in ring R , then $a(x) = b(x)$ stands if and only if $a_i = b_i$, for every integer $i \geq 0$.

  In other words, two polynomials are declared to be equal if and only if their corresponding coefficients are equal.

- Addition

  If $a(x) = a_0 + a_1 x + ... + a_{N-1} x^{N-1}$ and $b(x) = b_0 + b_1 x + ... + b_{N-1} x^{N-1}$ are

both in ring R, then:

$$a(x) + b(x) = c_0 + c_1 + ... + c_{N-1}x^{N-1} \tag{3}$$

Where

$$c_i = a_i + b_i, \text{ for each } 0 \leq i \leq t.$$

To elaborate, definition of addition of two polynomila in polynomial ring is to add two polynomial by adding their coefficients and collecting terms. To add $1 + x$ and $3 - 2x + x^2$ we consider $1 + x$ as $1 + x + 0x^2$ and add, according to the given definition, to obtain as their sum $4 - x + x^2$.

The multiplication defined for R should be more complicated.

- Multiplication

  if $a(x) = a_0 + a_1x + ... + a_{N-1}x^{N-1}$ and $b(x) = b_0 + b_1x + ... + b_{N-1}x^{N-1}$, then

$$a(x)b(x) = c_0 + c_1 + ... + c_{N-1}x^{N-1} \tag{4}$$

where

$$c_t = \pm \sum_{i=0}^{m} \sum_{j=0}^{t-i} a_i b_j = \pm a_t b_0 \pm a_{t-1} b_1 \pm a_{t-2} b_2 \pm ... \pm a_o b_t, 0 \leq t \leq N-1 \tag{5}$$

This definition explain the multiplication in polynomial ring is to multiply two polynomial by multipliying out the symbols formally, use the relation $x^\alpha x^\beta = x^{\alpha+\beta}$. The kth coefficient c_k is simply the dot product of the coefficients of a and the coefficients of b except that coefficients of b are listed in reverse order and are rotated around k positions.

Example: Let

$$a(x) = 1 + x - x^2,$$

$$b(x) = 2 + x^2 + x^3$$

Thus

$$\begin{aligned}
c_0 &= a_0 b_0 = 1 \cdot 2 = 2, \\
c_1 &= a_1 b_0 + a_0 b_1 = 1 \cdot 2 + 1 \cdot 0 = 2, \\
c_2 &= a_2 b_0 + a_1 b_1 + a_0 b_2 = (-1) \cdot 2 + 1 \cdot 0 + 1 \cdot 1 = -1, \\
c_3 &= a_3 b_0 + a_2 b_1 + a_1 b_2 + a_0 b_3 = 2, \\
c_4 &= a_4 b_0 + a_3 b_1 + a_2 b_2 + a_1 b_3 + a_0 b_4 = 0, \\
c_5 &= a_5 b_0 + a_4 b_1 + a_3 b_2 + a_2 b_3 + a_1 b_4 + a_0 b_5 = -1, \\
c_6 &= a_6 b_0 + a_5 b_1 + a_4 b_2 + a_3 b_3 + a_2 b_4 + a_1 b_5 + a_0 b_6 = 0, \\
c_7 &= c_8 = ... = 0.
\end{aligned}$$

According to definition of multiplication rules,

$$\begin{aligned}
(1 + x - x^2)(2 + x^2 + x^3) &= c_0 + c_1 x + c_2 x^2 + ...c_6 x^6 \\
&= 2 + 2x - x^2 + 2x^3 - x^5.
\end{aligned}$$

The multiplication of polynomial ring is commutative and it has a unit element.

- Inverse

  The inverse modulo q of a polynomial $a(x)$ is a polynomial $b(x)$ with the property that

$$a(x) \cdot b(x) = 1 \ mod \ q$$

  It can be also written as:

$$b(x) = a(x)^{-1} \ mod \ q$$

Not every polunomial has an inverse modulo q in the ring, but it is easy to determine if $a(x)$ has an inverse or not, or to compute the inverse if it exists.

For example:

Take $N = 7$, $q = 11$, $a(x) = 3 + 2x^2 - 3x^4 + x^6$.

The inverse of $a(x)$ *mod* q is

$$b(x) = -2 + 4x + 2x^2 + 4x^3 - 4x^4 + 2x^4 + 2x^5 - 2x^6$$

Since

$$\begin{aligned} a(x) \cdot b(x) &= (3 + 2x^2 - 3x^4 + x^6) \cdot (-2 + 4x + 2x^2 + 4x^3 - 4x^4 + 2x^4 + 2x^5 - 2x^6) \\ &= -10 + 22x + 22x^3 - 22x^6 \\ &= 1 \ \textit{modulo}\ 11 \end{aligned}$$

- Modular Operations

  $f(x)$ mod $P(x)$ means "the remainder of $(f(x) \div P(x))$" .

  - It can be denoted $f(x) = a(x)P(x)+b(x)$, where the degree of $b(x)$ is lower than that of $P(x)$, then $f(x)$ mod $P(x) = b(x)$.
  - Polynomial division: $(f(x) \div P(x))$ to obtain the remainder.

  Example:

  1. Let $f(x) = x^8 + 1$, $P(x) = x^3 + x^2 + 1$

  $$\begin{aligned} f(x)\ \textit{mod}\ P(x) &= x^8 + 1\ \textit{mod}\ x^3 + x^2 + 1 \\ &= 6x^2 - 3x + 5 \end{aligned}$$

  where the quotient is $x^5 - x^4 + x^3 - 2x^2 + 3x - 4$.

2. Let $f(x) = x^{10}$, and $P(x) = x^2 + x + 1$

$$f(x)\ mod\ P(x) = x^{10}\ mod\ x^2 + x + 1$$
$$= x$$

and the quotient is $x^8 - x^7 + x^5 - x^4 + x^2 - x$.

Truncated polynomial ring is the algebra primitive that LTV Fully Homomorphic Encyprion defined over with. The truncated polynomial ring, which is denoted as $R = \mathbb{Z}[x]/(x^n + 1)$ in the literature. N is a prime number. In this notation, $\mathbb{Z}[x]$ represent the set of polynomials of arbitrary degree with interger coefficients and $x^n + 1$ is the modulo that the polynomial is taken. It can be seen that the set of R contains all the polynomials with integer coefficients of degree up to $N - 1$.

In addition to the limitation for the degree on polynomial in the truncated polynomial ring, another particular restrain is also requested for LTV Fully Homomophic Encryption scheme. The elements in truncated polynomial ring R for LTV Fully Homomorphic Encryption scheme can be viewed as degree N polynomials with the coefficients taken modulo of prime number q, which demonstrate that all integer coefficients for the LTV truncated polynomial ring are supposed to be coming from range $[0, q]$.

So LTV Fully Homomorphic Encryption system works in the truncated polynomial ring R taken modulo $x^N + 1$ and q, which is expressed by $R_q = \mathbb{Z}_q[x]/(x^n + 1)$.

2.2 LTV Fully Homomorphic Encryption

Figure 2.1 shows the key generation, encryption, decryption and evaluation process of LTV Fully Homomorphic Encyption scheme.

2.2.1 Parameter sets

For LTV Fully Homomorphic Encryption to be both secure and correct, concrete sets of parameter need to be chosen carefully. An proven secure range of required parameters sets and values of selection for determined security level is shown below[11]:

- An integer N, fix N to be power of two.
- An integer p, fix $p = 2$. This choice provides a useful message space and cause the least possible expansion on the noise.
- Set χ to be the discrete Gaussian distribution $D_{\mathbb{Z}^N,r}$, for some standard deviation $r > 0$. Elements can be efficiently drawn from this distribution and moreover.
- With N a power of two and a degree N polynomial $\phi(x) = x^N + 1$, the noise distribution can be spherical and the worst case reduction still hold[12].
- Set $r = \sqrt{2N/\pi}$, so that ring-LWE is as hard as lattice problem in the worst case[12].
- Choose a prime $q \in [dN^6 ln(N), 2dN^6 ln(N)]$, such that $q \equiv mod\ 2N$. For $d = 25830$, correctness of the scheme can be guaranteed. Experimentally, less aggressive and lower value of d is obtained.

The parameters N, q, χ are public. The message space for the LTV Fully Homomorphic Encryption scheme is $M = 0, 1$ and all the operations are performed in the ring $R_q = \mathbb{Z}_q[x]/(\phi(x))$. We associate $\mathbb{Z}_q$ with the set $\{-\lfloor q/2 \rfloor, ..., \lfloor q/2 \rfloor\}$ throughout the thesis, to satisfy the absence of wrap round error.

2.2.2 Encrption

The LTV Fully Homomorphic Encryption scheme is parametrized by a prime number q and a bounded error distribution $R = \mathbb{Z}[x]/(x^N + 1)$ over polynomials whose coef-

ficients are all at most q in absolute value. All operations in this scheme take place in the ring $R_q = R/qR$.

- KeyGeneration

  Sample bounded polynomial f', $g \leftarrow \chi$

  Set $f := 2f' + 1$ so that $f \equiv 1(mod\ 2)$. If f is not invertible in R_q, re-sample f'.

  Compute the public key as:

$$K_{pub} := h = 2gf^{-1} \in R_q. \tag{6}$$

  Set secret key as:

$$K_{pri} := f \in R. \tag{7}$$

  For all $\tau \in \{0, ..., \lfloor logq \rfloor\}$, sample $s_\tau, e_\tau \leftarrow \chi$.

  Compute $\gamma_\tau = hs_\tau + 2e_\tau + 2^\tau f \in R_q$.

  Set evaluation key as:

$$K_{eva} = (\gamma_0, ..., \gamma_{\lfloor logq \rfloor}) \in R_q^{\lceil logq \rceil} \tag{8}$$

  The public key h, counterpart secret key f and evaluation key are acquired through above steps.

- Encryption(K_{pub}, m)

  $K_{pub} := h = 2gf^{-1} \in R_q$, $K_{pri} := f \in R$

  Sample polynomials $s, e \leftarrow \chi$. Output the ciphertext

$$c := hs + 2e + m \in R_q \tag{9}$$

  Where all operations are done with modulo q and $\phi(x)$.

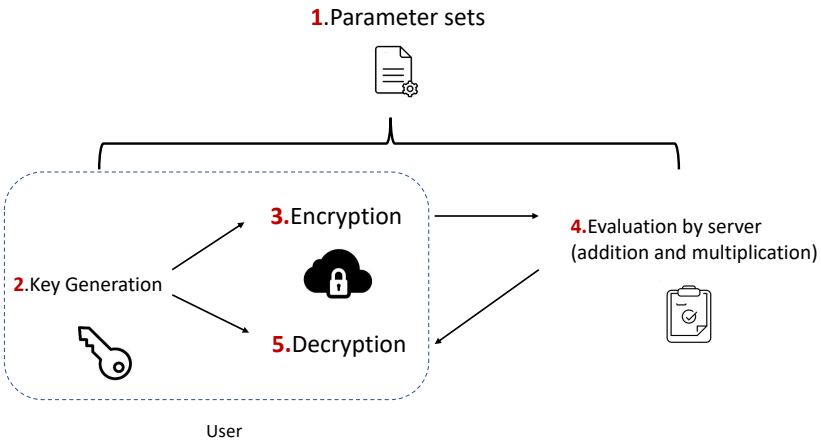

Fig. 2.1: Encryption process for LTV FHE

When above two steps are done, ciphertext is computed and output as polynomial c.

- Decryption(sk, c)

  Output

$$\mu = fc \in R_q \tag{10}$$

$$m' := \mu \ (mod\ 2) \tag{11}$$

Decrypted message m is obtained after the success completion of above computations.

Decryption works because:

$$\begin{aligned} fc \ (mod\ q) &= f(hs + 2e + m) \ (mod\ q) \\ &= 2(gs + ef) + fm \ (mod\ q) \\ &= 2(gs + ef) + fm \end{aligned}$$

Where the last quality is true since $|2(gs + ef) + fm| < q/2$. Taking this quantity mod 2 then gives the message m since $f \equiv 1 \ (mod\ 2)$.

2.2.3 Evaluation

In this section we show how to homomorphically add and multiply two elements. Given two cihpertexts c_1, c_2, there are a series of different public keys connected with each ciphertext. Public keys will be represented as K_1, K_2 respectively.

- *Evaluation addition*
  - Two ciphertexts c_1 and c_2 are encrypted by the corresponding public key sets K_1,and K_2.
  - Output ciphertext:

$$c_{add} = c_1 + c_2 \in \ R_q \tag{12}$$

  - Output the set $K_{add} = K_1 \cup K_2$ as its corresponding public key set.

- *Evaluation multiplication*
  - Given two ciphertexts c_1 and c_2 with corresponding public key set K_1, K_2, computing ciphertext $\tilde{c}_0 = c_1 \cdot \ c_2 \in \ R_q$,and let $K_1 \cap \ K_2 = \{K_{pub_{i1}}, ..., K_{pub_{ir}}\}$
  - If $K_1 \cap K_2 = \varnothing$, let $c_{mult} = \tilde{c}_0$.
  - Otherwise for $j \in= [r]$ and $\tau = \{0, ..., \lfloor log\ q \rfloor\ \}$, define $\tilde{c}_{j-1,\tau}$ so that

$$\tilde{c}_{j-1} = \sum_{\tau}^{\lfloor log\ q \rfloor} \tilde{c}_{j-1,\tau} 2^{\tau} \tag{13}$$

    is binary representation for $\tilde{c}_{j-1}$
  - Parse

$$K_{eva_{i_j}} = (r_{i_j,0}, ..., r_{i_j,\lfloor log\ q \rfloor}) \tag{14}$$

    let

$$\tilde{c}_j = \sum_{\tau=0}^{\lfloor log\ q \rfloor} \tilde{c}_{j-1,\tau} r_{i_j,\lfloor log\ q \rfloor} \tag{15}$$

– Output $c_{mult} = \tilde{c}$ as encryption of product of messages, output the set $K_{mult} = K_1 \cap K_2$ as it corresponding public key set.

3 HOMOMORPHIC ENCRYPTION AND CLOUD SECURITY

This chapter introduce the concept of homomorphic encryption. In fact, all different homomophic attempts can be generally categorized under three types of schemes with respect to the allowed times of operations and supported type(s) of operations on the encrypted data as Partially Homomorphic Encryption, Somewhat Homomophic Encryption and Fully Homomorphic Encyption. The research timeline and history of homomorphic encryption methods are given, as well as the important corresponding homomorphic properties.

Cloud computing security in modern society is analyzed. Solutions to cloud security based on applications of homomorphic encryption are discussed. Illustrations of specific encyption process scheme that are applied in cloud computing scenario are also presented.

3.1 Homomorphic Encryption

Homomorphic encryption is divided into three categories: Partially Homomorphic Encryption(PHE), Somewhat homomorphic encryption (SWHE) and Fully Homomorphic Encryption(FHE) scheme. Partially Homomorphic Encryption support only one type of operation with a unlimited time of usages. While Somewhat Homomorphic Encryption allow some type of computations such as addition or multiplication operations carried on cipher text with a limited number of time, Fully Homomorphic Encryption supports securely unlimited time of arbitrary computations over encrypted data without decrypting the data[9].

Due to its unique security properties, Homomorphic Encryption(HE) gained much attention from various industries. A series of applications and potential areas of use are illustrate in 3.1.

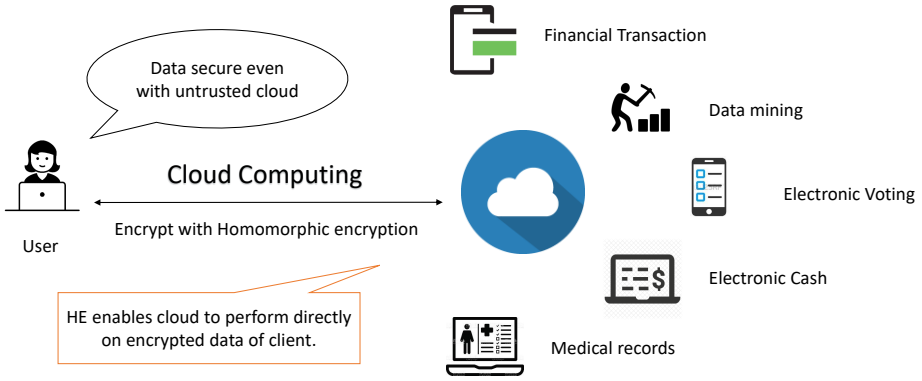

Fig. 3.1: An overview of HE applications

The idea of Homomorphic Encryption problem was first mentioned by Rivest, Adleman[8] in 1978 with an alternative term of privacy homomorphism. The open problem maintained a mirage for over 30 years since then and have been known to be the"holy grail" of cryptography[13]. The construction and solution of such scheme seemed to be even more impractical until Stanford PhD thesis proposed by Gentry[9] was theoretically achieved in 2009.

There was an appreciable amount of homomorphic encryption schemes introduced during this period: unpadded RSA[14] or ElGamal encryption scheme [15] and cryptography of Goldwasser-Micali[16], Benaloh [17], Paillier[18], Damgard[19] and Regev [20] [21]. The examples given above allow computation either one type of operation or limited numner of operations such as addition or multiplication on cipher texts.

Huge breakthrough was made when first plausible homomorphic encryption scheme proposed by Gentry[9] supports both addition and multiplication operations based on ideal lattices. The long last question about whether it is possible to realize a fully homomorphic encryption system is finally solved due to availability of arbitrary computation on cipher text in polynomial time. The paper initiate the research enthusiasm in cryptography and computer science field carried out on fully homomorphic encryption. A list of extension of Gentry's work can be found in [21], [12],[22],[23],[24],[25]. The security of these efficient schemes are mostly based on hardness of learning-with-

error (LWE) or ring-learning-with-error (RLWE) problems.

3.1.1 Patially Homomorphic Encryption

PHE schemes are often seen in particular application of electronic-voting or private information retrieval. But the application of PHE method is restricted in terms of the type of homomorphic evaluation opertions. In orther words, it can only be used in some certain scenarios with only addtion or multiplication operation are required.

RSA is one of the representative public key cryptography systems introduced by Rivest, Shamir, and Adleman[14] in 1978. Its security is based on hard problem of large prime factorization. Using the term 'privacy homomorphism' for the first time in history, the paper showed the homomorphic property over only multiplication, where $E(m_1 * m_2)$ can be directly evaluated by using $E(m_1)$ and $E(m_2)$ without decrypting them first. Apparently RSA does not support homomorphism of addition, thus it is classified as partially homomorphic encrytion.

Given RSA public key (N, h) and two plaintext m_1, m_2, the multiplication of two ciphertext returns a computed multiplication in the encrypted domain as follows:

$$\begin{aligned} Enc(m_1) \times Enc(m_2) &= (m_1^h \ mod \ N) \times (m_2^h \ mod \ N) \\ &= (m_1 \times \ m_2)^h \ mod \ N \\ &= Enc(m_1 \times m_2) \end{aligned} \tag{16}$$

In 1985, based on original Diffie-Hellman key exchange algorithm, a new public key encryption scheme called Elgamal were proposed by Taher Elgaml[15], which is based on the hard mathmetic problem of discrete logarithm. The Elgamal cryptosystem is multiplicatively homomorphic and not satisfy addtitive homomorphism.

Pailleir[18] proposed a novel probabilistic encryption scheme based on composite residuosity problem which is homomorphic over addition, while later work by Damgard-jurik [19] presented another PHE scheme as a generalization and improve-

ment of Paillier system.

The Pailleir scheme [18] is limited to only additive homomorphism. Given the public key (N, h), a random number (r_1, r_2) and two plaintext m_1 and m_2, the multiplication of two ciphertexts returns a computed addition in the encrypted domain as follows:

$$\begin{aligned} Enc(m_1) \times Enc(m_2) &= [(g^{m_1} \cdot r_1^N) \times (g_{m_2} \times r_2^N)] mod\ N^2 \\ &= [(g^{m_1+m_2})(r_1 r_2)^N] mod\ N \\ &= Enc(m_1 + m_2) \end{aligned} \tag{17}$$

3.1.2 Somewhat Homomorphic Encryption

There were SWHE researched before and after first FHE scheme came out in 2009. Some major SWHE methods were used as stepping stone towards practical FHE, while some were proposed with performance associated with FHE scheme.

BGN, one of the most significant SWHE, was introduced by Boneh-Goh-Nissim[26] in 2005. The hardness of the scheme is based on subgroup decision problem. Homomorphism over addition and multiplication are satisfied but further operations are not supported.

Sander[27] described a SWHE scheme supported a limited depth circuit evaluation on a semi group, and Kawachi's proposal[28] allow additive homomorphism.

3.1.3 Fully Homomorphic Encryption

If a encryption scheme allows unlimited number of arbitrary evaluation operation on encrypted data, it is called Fully Homomorphic Encryption(FHE). 30 years after the presentation of homomorphism concept by Rivest,etc[8] in 1978, Standford University graduate C.Gentry introduced the first plausible structure to construct a feasible Fully Homomorphic Encryption scheme in his PhD thesis[9]. Plenty of research works

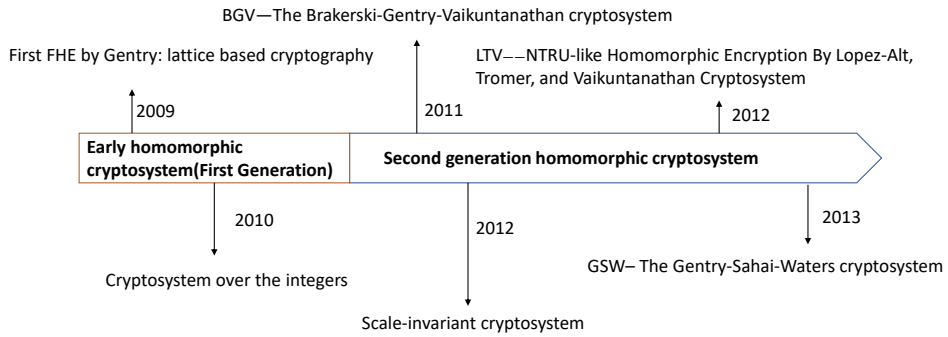

Fig. 3.2: An overview of FHE timeline

afterwards indulged in creating new FHE system and improve Gentry's innovation from 2009. Most of the later efforts are generally based on the blueprint provided by Gentry himself, that is, Gentry proposed a conventional framework for other researchers to follow his basic design. Diagram 3.2 shows the history of development of FHE researches.

Several new techniques that were developed starting 2011 led to the development of much more efficient Fully Homomorphic Cryptosystems. These include:

- Lattice based FHE

  Gentry's first Fully Homomorphic Encryption, also known as ideal lattice based FHE scheme, uses ideal lattice to make huge breakthrough from SWHE structure. An SWHE can only evaluate ciphertext homomorphically for limited number of times. When dealing with a certain amount of calculations on ciphertext, decryption process is not available to retrieve the original message accurately.

  The increasing chunk of noise must be reduced to convert noisy ciphertext to corresponding plaintext. Homomorphic operations can be applied to ciphertext with only small amount of noise. When the noise parameter become close to lattice point, or threshold, it is not feasible to decrypt cipher properly. Gentry proposed a novel method named bootstrapping to support a particular amount

of homomorphic operations to be performed on ciphertext, which iterate works for unlimited time. In this way, the scheme become fully homomorphic encryption.

The bootstrapping process basically works as follow: first the homomorphic decryption of the noisy ciphertext removes the noise, then the new homomorphic encryption introduces new small noise to the ciphertext. The ciphertext is now like just encrypted. Further homomorphic operations can be computed on this fresh ciphertext until reaching to a threshold again. This is where FHE become impractical for its noticeably computation cost and result in a major drawback for Fully Homomorphic Encryption.

A noticeable number of following FHE methods were presented to improve Gentry's original work. Gentry himself introduced a advanced key generation algorithm in [29] which improved the initial proposal's security level based on a quantum worst case reduction problem since the former one's key generation algorithm is used for a particular purpose only and the generation of an ideal lattice with a good basis is left without a solution[30]. Another probabilistic decryption algorithm with lower multiplicative degree, which is the aquare root of previous decryption circuit degree, was gived by Stehle and Steinfeld[31]. In addition, a new FHE scheme, which is a variant of Gentry's scheme was introduced in[32]. This scheme uses smaller ciphertext and key sizes than Gentry's scheme without sacrificing the security. Alternative works [33] [34] [35]focused on the optimizations in the key generation algorithm in order to implement the FHE efficiently.

- FHE over integers

  After Gentry's ideal lattice based scheme, a novel FHE scheme is presented in [36] which used bootstrapoing method to construct Fully Homomorphic En-

cryption. The new proposed approach is over the integer and the hardness of the method is based on Approximate Greatest Common Divisor(AGCD), with the primary motivation behind the scheme having conceptual simplicity. The paper[36] provide a way for transforming existing symmetric version of HE to asymmetric HE scheme, which states that the scheme is a public key encryption system.

While the relatively complex components of structure based on ideal lattice is replaced by a uncomplicated architecture over integers, the simplicity attained trade off for computation cost. A few following optimization works[37] [38][39]concentrate on decreasing size of public keys.

Some subsequent methods for FHE schemes over integers are proposed too. Scale invariant FHE over the integers [40], a new scheme with integer plaintexts[39], a new SWHE scheme for computing arithmetic operations on large integer numbers without converting them into bits[41], a new symmetric FHE without bootstrapping[42], a new FHE for non binary message space[43]. All Schemes mentioned above improved FHE over the integers scheme in their own way.

- LWE based FHE

  Learing With Error(LWE) problem which is conjectured hard to be solved, is as hard to solve as several worst case lattice problem even for post quantum algorithm. The hardness of worst case lattice problem like SVP is reduced to LWE, that is, the problem of SVP would be solved in polynomial time with a particular algorithm if this algorithm could be used to work out LWE in efficient time. The idea stimulate the growing researches over post quantum cryptography with relatively small ciphertext size. A significant variant of LWE problem was introduced by Lyubashvsky[12] regarding algebra structure of ring-Learing with error. The ring-LWE is able to be reduced to worst case on ideal

lattice problem which is not efficiently breakable in polynomial time by quantum algorithm and considered more efficient for practical applications and stronger security level than original LWE problem.

Brakerski and Vaikuntanathan[44] constituted a SWHE scheme using efficient feature of ring-LWE make it one step closer to practical FHE with Gentry's basic bootstrapping and squashing techniques, because ring-LWE is regarded as better performance than LWE scheme. A leveled FHE scheme without using bootstrapping method was provided in [22], in respond to the fact that bootstrapping process cost too much computation resource to finish. In Brakerski's new scale invariant Fully Homomorphic Ecryption scheme, noise grows linearly with evaluation of homomorphic operation instead of exponentially and the scheme is based on hard problem of GapSVP. Some notable modifications include [45] reducing overhead of key switching and faster evaluation of homomorphic operations and [46] using relinearization to improve efficiency.

In addition, some later proposal [23],[24],[25] are introduced to extend the performance and achieved better optimization results.

3.2 LTV Fully Homomorphic Encryption

A more practical and applicable FHE scheme LTV was derived from an old and secure cryptosystem NTRUEncrypt, which is a public key cryptography system proposed by Hoffstein, J [47]. It was named after the initials of three authors of Lopez-Alt, Eran Tromer, and Vinod Vaikuntanathan[10], who obtained a multi-key FHE scheme based on NTRUEncrypt. Lopez-Alt discoverd the homomorphic properties hidden in NTRUEncrypt scheme and developed LTV method with a few differences. First, the variant add a set of noises from a discrete Gaussian distribution with bounded support in stead of a deterministic set in original scheme. Second, the computation environment is different, LTV method performs operations in polynomial ring set

$R_q = \mathbb{Z}[x]/(x^N + 1)$ where N is power of 2, while in NTRUEncrypt cryptosystem the base algebra structure is polynomial ring $R_q = \mathbb{Z}[x]/(x^N - 1)$ where N is prime. Third, more aggressive parameter sets provide higher security level and more complex computation to support fully homomorphism.

One of the unique features of LTV Homomophic Encryption scheme is the multikey homomorphism for two party computation. Let (h_1, f_1) and (h_2, f_2) be two different public and secret key sets. Let $c_1 = h_1 s_1 + 2e_1 + m_1$ and $c_2 = h_2 s_2 + 2e_2 + m_2$ be two ciphertexts encrypted under public key h_1 and h_2 respectively. The computation of ciphertexts that decrypted to sum and the product of m_1 and m_2. Using the secret key $f_1 \cdot f_2$ the ciphertext $c_{mult} = c_1 \cdot c_2$ and $c_{add} = c_1 + c_2$ can be decrypted to the product and the sum of m_1 and m_2 respectively.

The multikey homomorphism properties for two party computation is shown below using joint secret key set k_1, k_2.

- Multikey Homomorphism over addition

$$\begin{aligned} f_1 f_2(c_1 + c_2)(mod\ 2) &= 2(f_1 f_2(e_1 + e_2) + f_2 g_1 s_1 + f_1 g_2 s_2) + f_1 f_2(m_1 + m_2)(mod\ 2) \\ &= 2e_{add} + f_1 f_2(m_1 + m_2) \\ &= m_1 + m_2(mod\ 2) \end{aligned} \tag{18}$$

- Multikey Homomorphism over multiplication

$$\begin{aligned} f_1 f_2(c_1 c_2)(mod\ 2) &= 2(2g_1 g_2 s_1 s_2 + g_1 s_1 f_2(2e_2 + m_2) + g_2 s_2 f_1(2e_1 + m_1) \\ &\quad + f_1 f_2(e_1 m_2 + e_2 m_1 + 2e_1 e_2)) + f_1 f_2(m_1 m_2)(mod\ 2) \\ &= 2e_{mult} + f_1 f_2(m_1 m_2) \\ &= m_1 m_2(mod\ 2) \end{aligned} \tag{19}$$

Since $f_1 \equiv 1(mod\ 2)$ and $f_2 \equiv 1(mod\ 2)$.

In other words, joint secret key $f_1 f_2$ can be used to decrypt $c_{add} = c_1 + c_2$ and $c_{mult} = c_1 \cdot c_2$. We can extend this argument to any combination of operations, with ciphertexts encrypted under multiple public keys. It is also seen that multikey homomoprhic operation increase noise more than a single key homomorphic evaluation. However, $m_1 + m_2$ and $m_1 m_2$ can still be recovered correctly using the joint obtained secret key since f, g, s, e are all sampled from bounded ditribution χ. In other words, the decryption part works well if the noise parameters e_{add} and c_{mult} are smaller than $[-\lfloor q/2 \rfloor, \lfloor q/2 \rfloor]$.

3.3 Homomorphic Properties

Let c_1 and c_2 be two ciphertexts. When two messages m_1 and m_2 are both encrypted by the shared secret key f, the homomorphic properties could be expess as:

- Addition

  Define the sum of two ciphertexts as:

$$c_{add} = c_1 + c_2$$

  For the additive homomorphic property, we have:

$$\begin{aligned} f \cdot c_{add} \ mod \ 2 &= f(c_1 + c_2) \ mod \ 2 \\ &= f \cdot c_1 + f \cdot c_2 \ mod \ 2 \\ &= \mu_1 + \mu_2 \ mod \ 2 \\ &= m_1 + m_2 \ mod \ 2 \\ &= m_1 + m_2 \end{aligned} \tag{20}$$

  From above, the summation of two ciphertext can be decrypted as summation of two plaintexts or messages with additive homomorphic property.

- Multiplication

  Define the product of two ciphertext as:

$$c_{mult} = c_1 \ \cdot \ c_2$$

  For multiplicative homomorphic property, we have:

$$\begin{aligned} f^2 \cdot c_{mult} \ mod \ 2 &= f^2 \cdot (c_1 \cdot c_2) \ mod \ 2 \\ &= (f \cdot c_1) \cdot (f \cdot c_2) \ mod \ 2 \\ &= \mu_1 \cdot \mu_2 \ mod \ 2 \\ &= m_1 \cdot m_2 \ mod \ 2 \\ &= m_1 \cdot m_2 \ mod \ 2 \end{aligned} \tag{21}$$

  So that the product of two ciphertext can be decrypted as product of two plaintexts or messages with multiplicative homomorphic property.

- Addition and multiplication

  From

$$f = 2f' + 1$$

  we get

$$f \equiv 1 \ mod \ 2$$

  Then

$$f^k \cdot m \ mod \ 2 = m \tag{22}$$

  A combination of additions and multiplications can be reallized through multiplying the encrypted ciphertext by f^k, f is the secret key and k is the depth of the longest chain of multiplication. Noise in result ciphertext grows with the increaing number of additions and multiplications.

One of the main challenges for extension of Somewhat Homomorphic Encryption to Fully Homomoprhic Encryption is the management and control of noise. This is often the reason why fully homomorphic encryption schemes are considered less efficient. Noise grows linearly with addition operations and while exponentially with multiplication operations. As long as the noise remains below a certain threshold, the message will be recovered without failure. However, if the magnitude of growing noise after each operation is over a certain level, it is hard to obtain the correct result of homomorphic encryption operation, which also affects the security and homomorphism. The importance of this trade off between noise level and security becomes significant and apparent when considering the homomorpic properties of the scheme.

- Relinearization

  The product ciphertext is supposed to be multiplied with the secret key for k times in order to decrypt a product of k ciphertext. The depth of the circuit will have to be kept track of, which is somehow turned out to be complex if the circuit is complicated.

  Relinearization enable multiplying the result ciphertext by secret key only for one time, which make it possible to decrypt arbitrary combination of operations including additions and multiplications of the ciphertext. Evaluation key introduced in Section 2.2.3 is used in this process.

- Modulus reduction

  When an exponential gain on the depth of the computational circuit is generated in evaluation stage with the noise growth rapidly, modulus reduction offers a noise management technique to control the noise after every time of the evaluation operation through scaling the ciphertext value. The maximum depth of the computational circuit is therefore consequently increased by repeatedly

Fig. 3.3: Cloud computing applications

applying the process.

3.4 Cloud Computing and Homomorphic Encryption

Cloud computing is another evolution in information technology after the presence of computer and Internet. According to NIST[48], cloud computing is a model for enabling convenient, on demand network access to a shared pool of configurable computing resources(e.g, networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimum management effort or service provider interaction. Nowadays we see cloud as a set of solutions integrating multiple technology and corresponding application. It incorporate distributed computing, virtualization, grid computing, load balance and parallel computing into information ecosystem. The application of cloud technology enable every enterprise and personal user to make better and faster use of advanced information technology. The Graph 3.3 shows cloud computing technique connect differenct devices and terminal together.

The rise of cloud computing made it possible for public users to have access of computation service of low cost, high performance, large quantity, and fast configuration. While the introductory of cloud computing benefits various IT fields with

economic scale and high scalability, it also expose certain latent threat in security aspect due to its technical features. Based on the different service categorization that cloud computing provides, it is classified as three basic service delivery modes:

- IaaS(Infrastructure as a Service):

  Cloud provider deploy technique such as virtualization on fundamental platform to coordinate infrastructure construction and improve utilization. Users are able to get access to a virtual server (an IaaS provider), bypassing the time and effort they contribute on establishing their own data centers or deploying services. IaaS provide function of pay-as-you-go and mensurable resource pool in heterogeneous resources environment, while satisfy user demand and fully utilize hardware resources. Amazon web service's elastic computer cloud, also known as EC2, is a typical example for IaaS platform.

- PaaS(Platform as a Service):

  PaaS provides enterprise development, operation interface and environment for companies to realize self service and thus achieve unified service on platform level. Cloud provider give customers platform and every business service they need to build, dispatch and manage application of their own, omitting the process of installing platforms and supporting environment on client side. PaaS lies on top of IaaS hierarchy. Not only does it focus on integration of underlying hardware resource, it is also required to accommodate platform environment for tenants to implement developing and debugging applications. Therefore, the security issue of PaaS is definitely a major concern for providers. Google Apps and Microsoft Azure are most famous PaaS provider.

- SaaS (Software as a Service) :

  SaaS offers users with uniformed service interface, for example, provide the service using multi-user architecture with browser or other client application.

SaaS providers deliver applications hosted on the cloud infrastructure as Internet based service for end users. While it take full advantage of utilizing lower level resources, it is supposed to provide clients with customized application service through deploying one or several software environment as well. Despite the merits of operational efficiency and reduced costs, enterprises often ponder on applying it into practice in term of the obscurity between security level and data storage. SalesForce CRM is one of the prominent example of SaaS supplier.

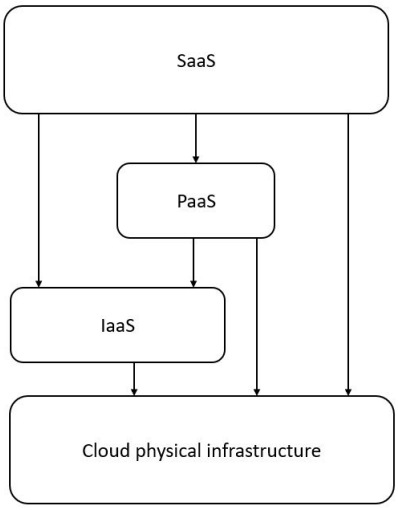

Fig. 3.4: Cloud service delivery model

Each service type is based on distinct implementation layer, as shown in Figure 3.4 [49]. It can be seen from the figure that service resources provided by upper level service provider, such as SaaS service, are able to set up service resource independently. Also SaaS application developer can accommodate them on other leased PaaS platforms.

While the innovative service mechanism make users obtain almost infinite computation ability and wide range of information service, it is the evolvement of combination of distributed computing, parallel computing and grid computing. The novel pattern use network to integrate and expand needed resources (hardware, platform, software) to build a reliable high performance computation platform.

3.4.1 Security threat

Multiple categories of security threat over cloud computing system are listed below:

1. Privileged user access: With user data stored in server from cloud service provider, processing data out of reign of user per se boost security risk to some extent since administrator of service operator might have permission to access and process of data.

2. Compliance: Cloud computing operator has obligation in supporting third party institute to offer censorship of accuracy and security level. The security of data will be threatened if some service provider turn down external audit and safe authentication.

3. Data location: Cloud computing clients have no knowledge of the location where data store at.

4. Data segregation: All users under cloud computing environment process their data in shared pool of system. Total data safety requires not only encryption, but also data segregation.

5. Data recovery: Providers are supposed to back up entire data chunk, in order to recover user data shortly after occurrence of physical catastrophe and artificial vandalism.

6. Investigative support: With multiple users data stored as a group, it will be difficult to investigate when delinquent behaviors happens.

7. Long term viability: Service providers are not supposed to go bankrupt in terms of avoiding influencing user data stability and sustainability.

8. Support in reducing risking: Instructions are provided to administrators and

managers for setting and monitoring policies to evaluate information and support customer staff to understand how to safely and reliably use their product.

3.4.2 Application of security system

With all users data stored on cloud, they lost absolute control over their private information on the fly. Therefore, cloud service provider must pledge secure and strong guarantee to ensure authenticity, integrity and confidentialitty of those contents. In order to do that, encryption over the data being transmitted and saved is inevitable essential.

The homomorphism of Fully Homomorphic Encryption can perform any procession to sensitive data after encrypting original message without leaking contained private information. Performing Fully Homomorphic Encryption algorithm on processing data with enciphering bring us reliable safety measure during cloud computation. To put it in another way, encrypting sensitive data before synchronizing local data to boost data security level. Even if these data were stolen, malicious users only retrieve cipher text after encryption. One can not work out the plain text without keys where processed by users instead of cloud side. Meanwhile, with the help of homomorphism of Fully Homomorphic Encryption, cloud will possess the ability to process cipher text data. The defects in wasting resources and low efficiency of traditional operations can thus be avoided. while normal encryption approaches need to sent cipher data to user terminal, eventually transmit data back to cloud side on decryption operation performed on them by users.

A data security scheme constructed by cloud computing based on Fully Homomorphic Encryption is consist of user terminal and cloud side as shown in Figure 3.5.

There are different role for user and cloud in the cloud computing sytem. For users, personal computers, client machines, cell phones and other devices provide hardware

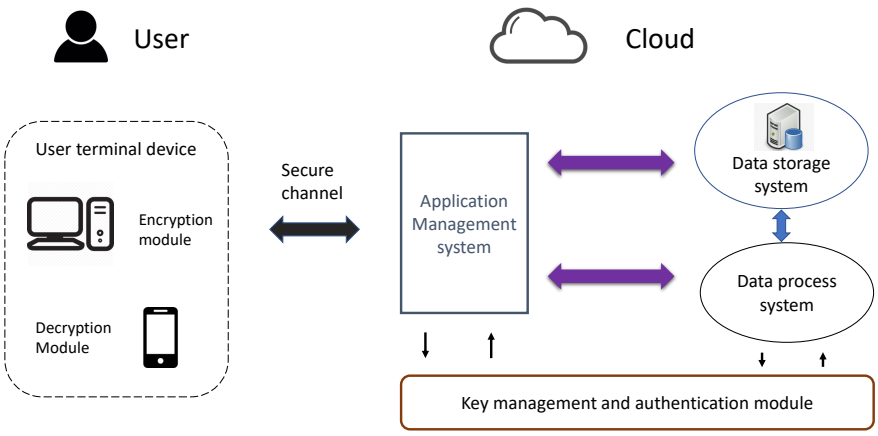

Fig. 3.5: Cloud security scheme model

conditions for users to implement information transactions through cloud computing system login program. Users are also responsible for submitting user request over the cloud, encrypting and decrypting user private data, uploading and downloading cipher text.

The works by cloud include application managing system, data processing system, data storage system, key management and authentication module. Most important one of all is application manage system, which is obliged to coordinate process user requests. For example, basic order over inquiry for encryption key or user authentication is delivered to key management and authentication module, assignment for computation upon raw data is handed over to data process system.

The data ready for direct saving is processed by data storage system waiting for collaboration of information sharing between users. Data process system is made up of data retrieval, data computation, data upgrading sections. Data storage system perform storing of cloud data, which utilizing user data location recorded to store and extract user data. Key manage and authentication system is responsible for key generation and storing.

Below are specific steps for the scheme:

1. User access to cloud: After receiving user requests, application manage system repost user information to key mange and authentication module, where a pair of symmetry key based on user information from certain algorithm is generated afterwards to bind user information and key together before finally encrypted and stored. Application manage system send key pairs to users via safe channel.

2. User encrypt data on client terminal with keys, transmit encrypted cipher text data towards cloud side. Application manage system store them in storage system on receiving those data.

3. When user need to use the data, application manage system extract data back to user from storage system upon receival of user requests before users obtain the data, decrypt it and apply it for practical usage.

4. With the feature of Fully Homomorphic Encryption algorithm, data process system perform direct retrieval in cipher text database for encrypted key words when operations of data searching is needed.

5. It then deliver cipher text data retrieved to users, who will have access to it after decryption without decrypting entire cipher database before retrieval. The method discussed saves time, boost benefits for users and reduce system cost.

3.4.3 FHE scheme over cloud: a scenario

This scenario use Fully Homomorphic Encryption which has wide applications in various fields:

(1) Privacy protection: User data is transmitted as cipher text to cloud and saved. In this way, security of data during transmission process is assured, as well as storage of data, for cloud service provider impossible to obtain plain text information.

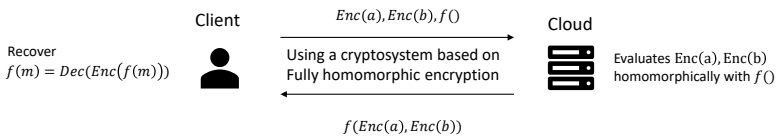

Fig. 3.6: FHE over cloud: a scenario

(2) Data process: Fully Homomorphic Encryption scheme let user and trusted third party perform process operations directly on cipher text data, with absence of original data. Having the computation results in hands, users start to decrypt and thus retrieve data processed.

For example, in medical or health care information system, electronic prescriptions are all stocked on cloud server in the format of cipher text. Sometimes hygiene department need to know distribution of geographic location and age from patients with some particular diseases in certain areas for emergency measure taken against public sanitation safety issue. Ciphered electronic prescription is handed over to professional data process service business partner for desired correct data when decrypted after dealing with results.

(3) Cipher text search: cipher text retrieving method can perform searching directly towards cipher data based on Fully Homomorphic Encryption technology. Not only search privacy and retrieval efficiency can be assured and boosted, addition and multiplication computation can be perform on data searched without amending corresponding plain text as well.

An example of homomorphic operation between user and cloud is illustrated in Figure 3.6.

In this scenario, firstly user encrypt his personal data a and b to $Enc(a)$ and $Enc(b)$ with public key, and send the encrypted results towards cloud server. Next step, when user wants to do computation on his data, he send a request and a function $f()$ to cloud provider. Now that the server got the encrypted data $Enc(a)$ and $Enc(b)$ and

function $f()$, it compute homomorphic operation $f(Enc(a), Enc(b))$ without knowing actual contents of a or b using evaluation addition or evaluation multiplication.

After the cloud computing process, server send back encrypted results to the client. Finally user use private key to decrypt the result $f(Enc(a), Enc(b))$ and recover $f()$, which is the same as it carried out operations on raw data. From this example, we can see that the homomorphic operation at server side does not require private key of user and support arbitrary computation such as addition and multiplication on encrypted client data.

4 AN OVERVIEW OF RECENT RELATED WORKS

In this chapter we briefly review the existing related works in the past few years.

In 2012, Lopez-Alt, Eran Tromer, and Vinod Vaikuntanathan[10] first proposed LTV Homomorphic Encryption scheme. They use the NTRUEncrypt, one of the earliest attempts towards encryption based on latiice problem, to obtain a practical Fully Homomorphic Encryption scheme with several differences from original NTRU-Encrypt.

In 2014, two years following proposal by Lopez-Alt, et al[10], LTV homomorphic encryption is implemented in software complied C environment by K.Rohloff [50]. This method relied on the manipulation of very large integers so that the system was both secure and capable of supporting the evaluation of sufficiently large circuits. Prior implementations designs, for the most part, reckon on single threaded execution on commoditary CPU due to difficulty of lack of native support for multi-thread execution with underlying software libraries. This case in addition to the inherent computational cost of secure computing using known SWHE and FHE schemes, prevented the practical use of SWHE and FHE.

Therefore, the paper reported a design, implementation and evaluation of a scalable Fully Homomorphic Encryption scheme which addresses the limitation for secure arbitrary computation. It used multi-core processor as advantage of parallelism with further support on key switching and modulus reduction. The basis of the design is a layered software services stack to provide high level Fully Homomorphic Encryption operations supported by lower level lattice based primitive implementations runnning on a computing substrate. The Fully Homomorphic Encryption is implemented and evaluated to run on a commodity CPU-based computing enviroment with Mathworks Matlab environment and Matlab coder toolkit to generate an ANSI C representation. This ANSI C using gcc to run as an executable in a Linux environment. Presented experiment results of their performance show that their Fully Homomorphic Encryp-

tion implementation provide at least an order of magnitude improvement in runtime as compared to recent publicly known evalution results of other software implementation.

Another work by Dai [51] in 2014 proposed a GPU optimized library to support Fully homomorphic Encryption. Despite the active advances witnessed 5 years after Gentry's first proposed plausible Fully Homomorphic Encryption scheme and 2 years after the introduction of LTV Fully Homomorphic Encryption scheme, FHE still has not sufficiently prepared to directly benefit public. Motivated by some main bottlenecks concentrated on practical FHE, this paper build a large polynomial arithmetic library optimized for NVIDIA GPU to support Fully Homomorphic Encryption scheme. In the experiment, a server equipped with Intel Xeon E5-2609 running @2.5GHz, 64 GBytes of memory, and a NVIDIA GeForce GTX 690 running @915MHz with 3072 stream processors and 4 GBytes of memory are used.

To realize the large polynomial arithmetic library, they convert polynomial with large coefficients using Chinese Remainder Theorem(CRT) into many polynomial with small coefficients, and then carry out modular multiplication in the residue space using custom developed Discrete Fourier Transform based library. The library is further extended to support the homomorphic evaluation operations and Number Theoretical Transform(NTT) is utilized to realize large polynomial arithmetic. This method demonstrated significant speedups over previous reported results. Additional conclusion is drawn that the efficiency is limited by the memory of the target GPU. Hence, even greater speedup may be achieved simpley by moving to GPU with more memory.

A new multiplier architecture introduced by Y.Doroz[52] in 2015. A custom hardware accelerator optimized for a class of reconfigurable logic to bring LTV based Homomophic encryption scheme on estep closer to deployment in real life application. The accelerator achieved time complexity of $O(NlogNloglogN)$ and space of $O(Nn^2)$

in circuit level. The accelerator introduced is connected via a fast PCIe interface to a CPU platform to provide homomorphic evaluation service to any application that needed to support computations. A number theoretical transform based multiplier architecture capable of effciently handling very large polynomials is introduced. The gap between what is currently available on a CPU and what is practical in real life is somehow too far to consider software only solutions. This trigger researchers to study the use of alternative platforms such as graphic processing units (GPU) and reconfigurable logic such as FPGAs. What is more, even further domain specific ASIC designs to accelerate homomorphic evaluations.

The main contributions of the paper are FPGA architecture presented to accelerate LTV FHE scheme and FPGA device connected to PC with high speed interface. This architecture may be considered as implementation of an external FHE accelerator that will speedup homomorphic evaluations taking place on a CPU. AES block cipher is used as an accelerator to homomorphic evaluation synthesized for the Xilinx Virtex 7 family. Efficiency of large polynomial computation is improved on different platforms as similar CPU and GPU implementations.

In 2016, an FPGA based computation accelerator as part of a homomorphic encryption processing unit co-processor was designed to implement key computation application [53]. In this paper, advanced designing and an FPGA based computation accelerator are implemented as part of a homomorphic encryption processing unit co-processor. This hardware accelerator technology improves the practicality of computing on encrypted data by reducing the computational bottleneck of lattice encryption primitives that support homomorphic encryption scheme.

The strategy of the paper is to accelerate key computational bottleneck common across Homomoephic Encryption schemes which can be much more rapidly executed on FPGA architectures. Chinese Remainder Transform and inverse Chinese Remainder Transform are used to accelerate power of 2 rings and other basic ring arith-

metic such as ring addition, ring subtraction and ring multiplication. The paper focus on experimental performance analysis on LTV Fully Homomorphic Encryption scheme, instantiating this capability in a Xilinx Virtex-7 FPGA that can attached to a host computer through either a PCI express port or Ethernet. They experimentally compare performance with reference software implementation of the CRT and iCRT bottleneck on a commodity CPU.

Anothe paper [62] provide a new potential of FPGAs for speeding up homomorphic encryption operations. In this work, a functional FPGA implementation of homomorphic encryption multiplication operation of large polynomials in $Z_q[x]/(x^N+1)$ whose ring dimension is determined by N which is a positive integer of form 2^k. While the paper use NTT algorithm to perform polynomial multiplication, which has time complexity of $O(NlogN)$, it demonstrate architecture space complexity of $O(Nn^2)$. It also provide results and performance measurement of implementation equipped with Xilinx Spartan 6 LX100 FPGA.

A paper published in 2017 [1] proposed a software/hardware co-designed accelerator implemented through a high level synthesis flow to accelerate homomorphic computation. They demonstrated a large modular polynomial multiplier configurable in both degree and coefficients size, as well as proposed a modular polynomial reducer based on polynomial with general form allowing optimization in the homomorphic context.The hardware part implements the product of large coefficients in RNS domain, and the schoolbook multiplication algorithm at polynomial level resulting in complexity of $O(N^2)$ in terms of time and $O(n^2)$ in terms of area. The designing and implementation of the modular multiplier which integrates a multiplier on polynomial with large coefficients and large degree with fast software computation are based on the FPGA underlying computation accelerator.

The proposed high speed accelerator works in the same primitive as LTV cryptosystem with polynomial ring of form $R_q[x] = Z_q[x]/(\phi(x))$ with respect to the

Table 4.1: Truncated polynomial ring multiplier performance in [1]

N	n	Platform	# Register	Latency(μs)
512	32	XC7V585T	102	66.41
1024	26	Spartan 6	114	69.1

$\phi(x) = x^N + 1$ where N is power of 2. The interface between hardware and software can be performed through a high performance AXI bus or a PCI express bus in the case of computer precessor. Results of their approach are demonstrated in Table 4.1.

5 PROPOSED ARCHITECTURE FOR FULLY HOMOMORPHIC ENCRYPTION

This chapter gives a comprehensive presentation and explanation of our proposed work. We put our minds into optimizing the process of encryption part where the operations are performed on user side in the cloud computing model since the generation of public keys and secret keys are precomputed. Thus the calculations performed by the server including evaluation or more, are beyond our focus.

A novel variant of LFSR arithmetic architecture is proposed to implement LTV Fully Homomorphic Encryption. Conventional LFSR structure is expanded and modified for performing the truncated polynomial ring multiplication in $R_q[x]$.

The specific part we are going to concentrate on is the encryption equation $c = hs + 2e + m$ where ciphertext c is derived from public key h, message m and sample polynomial s, e. The efficiency of LTV system is significantly impacted by the speed of multiplication of truncated polynomials with high degree and large coefficients. Taking the advantage of parallelism for computation of multiplication in LTV Fully Homomorphic Encryption scheme, our proposed work based on LFSR structure reduced the time complexity of quasi-linear time $O(NlogNloglogN)$ in [52] and $O(NlogN)$ in [62] to linear time $O(N)$ for polynomial multiplication, which is beneficial with large value of degree N.

As for the large integer multiplication, we then proposed to choose Mersenne number q to further speedup the modular multiplication. The special moduli sets, including $2^n + 1$,2^n, $2^n - 1$,etc, which have drawn much more attention than normal moduli when constructing modular adders and modular multipliers, offers a great time reduction on modular reduction when performing modular multiplications as well as modular additions.

Novel and efficient modular multiplier, modular adder and modular subtractor

are proposed for high speed processing of LFSR operations. We leverage to select parallel prefix structure which provide efficient modular addition over other popular adders. What is more, booth parallel multiplier are used to perform modular multiplication with reduced number of partial products during which the accumulation process taking place.

5.1 LFSR Based Structure

In this section, firstly basic knowledge is introduced about LFSR(Linear Feedback Shift Register) and its applications. Then the proposed LFSR based architecture is presented to realize truncated polynomial ring multiplication in encryption part of LTV Fully Homomorphic Encryption scheme. The details about the proposed architecture is explained with given algorithm and a architecture complexity analysis. A table of register contents is shown for better understanding of how the multiplication works in the LFSR-like multiplier system.

5.1.1 Linear Feedback Shift Register

LFSR (Linear Feedback Shift Register) is an elegant way to generate long pseudo-random sequences with most applications in pesudo-random numbers, fast digital counters, and cryptography which has cryptographically strong properties and uncomplicated hardware implementation advantages. An LFSR consisted of clocked storage elements(flip-flop) and a feedback path. The number of storage elements represent the degree of LFSR, while feedback network computes the input for the last flip-flop as XOR-sum of certain flip-flops in the shift register.

The stream value of outputs generated by registers and logic arithmetic is dependent on its previous state with maximum length of $2^N - 1$ when initial state as non-zero value, where N is highest degree of characteristic polynomial.

A example given in Figure 5.1 illustrate the basic structure of an LFSR initialized

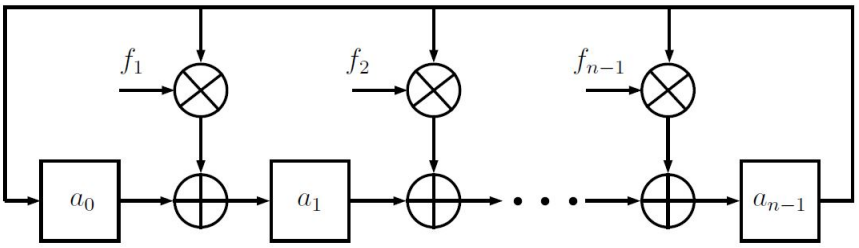

Fig. 5.1: Linear Feedback Shift Register

by characteristic polynomial $f(x)$. which is represented as:

$$f(x) = f_N x^N + f_{N-1} x^{N-1} + ... + f_1 x^1 + 1 \tag{23}$$

In figure 5.1, a $\bigoplus$ refers to an adder, a $\bigotimes$ refers to a multiplier and $\square$ represents a register. Assume that registers are initially loaded with coefficients of polynomial $A(x) = (a_0, a_1 ... a_{N-1})$, a shift to right operation of LSFR is equivalent to performing $A(x) \times x \ mod \ f(x)$, where x is root of characteristic polynomial $f(x)$.

5.1.2 Proposed truncated polynomial ring multiplier

The polynomial multiplication and addition in LTV Fully Homomorphic Encryption scheme is taken modulo $x^N + 1$, a modification of original LFSR is designed to accommodate the truncated polynomial ring computation.

Specifically, original LFSR can complete the operation without substantial multiplication because the polynomial coefficients always have value from $\{-1, 0, 1\}$ as ternary polynomial. In our case, however, the coefficients integer has much larger range of values in $\{0, q-1\}$.

The contents of registers are initialized as integer coefficients of polynomial $2e+m$, let values of register be $c^{(j)}$,(j is from 0 to N)in clock cycle N, contents of each register is denoted as $c^{(j)}_{N-1}, c^{(j)}_{N-2}, ..., c^{(j)}_1, c^{(j)}_0$. Specific steps to perform the truncated polynomial ring multiplication is shown in Algorithm 5.1.

In this algorithm, three inputs are polynomial $h(x)$ representing public key, $s(x)$

Algorithm 5.1 Multiplication in $R_q[x]$ for LTV FHE

Input: $h = h_{N-1}, ..., h_0$; $s = s_{N-1}, ..., s_0$; e ; m
Output: $c = c_{N-1}, ..., c_0 = hs + 2e + m \in R_q$
  $c^{(0)} := 2e + m \in R_q$
  for $j := 1$ to N **do**
    for $i := 0$ to $N - 1$ **do**
      if $i = N - 1$ **then**
        $c^{(j)}_{((i+1))} := -c^{(j-1)}_i + h_{((i+1))} \times s_{((N-j))}$ *mod* q
      else
        $c^{(j)}_{((i+1))} := c^{(j-1)}_i + h_{((i+1))} \times s_{((N-j))}$ *mod* q
      end if
    end for
  end for
  return $c^{(N)} := hs + 2e + m \in R_q$
  *$((A))$ denote A *mod* N.

as one of the sample polynomials, and $2e(x) + m(x)$ comprised of sample polynomial and message polynomial, each with N coefficients.

Because truncated polynomial rings in LTV Fully Homomorphic Encryption scheme are taken modulo $x^N + 1$, there are arithmetic scenarios of both addition and subtraction in this computation, in other words, either positive or negative value of multiplication result could be added to next round of operation.

Based on above algorithm, an LFSR originated architecture to perform LTV Fully Homomorphic Encryption scheme polynomial multiplication in $R_q[x]$ is proposed as Figure 5.2.

In this structure, it is required to lay out $N - 1$ modular q adders, 1 modular q subtractor, N modular q multipliers and N registers. Each register can store value of $[\log_2 q]$ bits length. The arithmetic of modular multiplication and modular addition are performed in modular q multiplier and modular q adder, which also cause noticeable time delay and will be introduced in next section. Registers $c = (c_{N-1}, c_{N-2}, ..., c_1, c_0)$ are initially loaded with coefficients of polynomial $2e(x) + m(x)$, the coefficients of polynomial $h(x)$ are input to each modular multiplier in parallel manner while the coefficients of polynomial $s(x)$ are input to every

modular multiplier in a serial fashion.

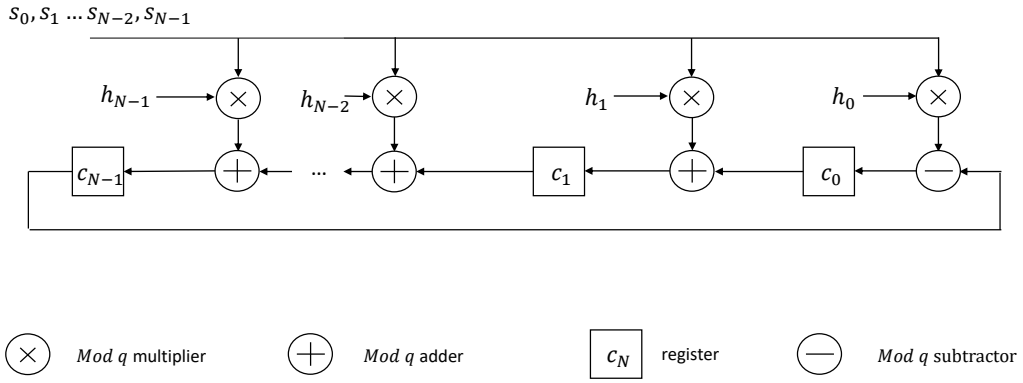

Fig. 5.2: LFSR Based LTV FHE Architecture

The Registers $c(x) = (c_N, c_{N-1}, ..., c_1, c_0)$ will store the result of multiplication $c^{(N)} = h(x)s(x) + 2e(x) + m(x)$ at clock cycle N. Table 5.1 shows the register contents in clock cycle $0, 1, ..., N-1, N$.

5.2 Mersenne Number for Moduli

The major cumbrance of architecture for performing modular reduction multiplication of large coefficient truncated polynomial described above is the costly large integer multiplication with additional long integer modulo computation which cause significant hardware usage and may limit the clock frequency in terms of long critical path.

We proposed a series of methods and optimization approaches to speedup the process of large integer multiplication and addition, which easily takes up most of the computation resourses in the proposed LFSR based structure for truncated polynomial ring multiplication in LTV Fully Homomorphic Encryption scheme encryption process.

Two basic operations are generalized in multiplication arithmetic, the generation of partial product and their accumulation. In order to speed up the most time consuming process, two available approaches can be considered. Reducing the number of

Table 5.1: Regsiter Contents

CLK (j)	$c_{N-1}^{(j)}$	$c_{N-2}^{(j)}$	...	$c_1^{(j)}$	$c_0^{(j)}$
0	$2e_{N-1} + m_{N-1}$	$2e_{N-2}m_{N-2}$		$2e_1 + m_1$	$2e_0 + m_0$
1	$2e_{N-2} + m_{N-2} + h_{N-1}s_{N-1}$	$2e_0 + m_0 + h_1 s_{N-1}$		$2e_0 + m_0 + h_1 s_{N-1}$	$-2e_{N-1} - m_{N-1} + h_0 s_{N-1}$
2	$2e_{N-3} + m_{N-3} +$ $h_{N-2}s_{N-1} + h_{N-1}s_{N-2}$	$2e_{N-4} + m_{N-4} -$ $h_{N-3}s_{N-1} + h_1 s_{N-2}$		$-2e_{N-1} - m_{N-1} + h_0 s_{N-1} +$ $h_1 s_{N-2}$	$-2e_{N-2} - m_{N-2} -$ $h_{N-1}s_{N-1} + h_0 s_{N-2}$
...			...		
$N-2$	$2e_1 + m_1 + h_2 s_{N-1} +$ $h_3 s_{N-2} ... + h_{N-2}s_3 + h_{N-1}s_2$	$2e_0 + m_0 + h_1 s_{N-1} +$ $h_2 s_{N-2} ... + h_{N-3}s_3 + h_{N-2}s_2$		$-2e_2 - m_3 - h_4 s_{N-1} -$ $h_5 s_{N-2} ... + h_0 s_3 + h_1 s_2$	$-2e_2 - m_2 - h_3 s_{N-1} -$ $h_4 s_{N-2} ... - h_{N-1}s_3 + h_0 s_2$
$N-1$	$2e_0 + m_0 + h_1 s_{N-1} +$ $h_2 s_{N-2} ... + h_{N-2}s_2 + h_{N-2}s_1$	$2e_{N-1} + m_{N-1} - h_0 s_{N-1} +$ $h_1 s_{N-2} ... + h_{N-3}s_2 + h_{N-2}s_1$		$-2e_2 - m_2 - h_3 s_{N-1} -$ $h_4 s_{N-2} ... + h_0 s_2 + h_1 s_1$	$-2e_1 - m_1 - h_2 s_{N-1} -$ $h_3 s_{N-2} ... - h_{N-2}s_2 + h_0 s_1$
N	$c^{(N)} = hs + 2e + m$				

partial product is one of them. Several classical algorithms such as booth algorithm is used to cover this part. Moreover, the goal of faster multiplication can be achieved by accelerating the process of accumulation of partial products. Apparently, when smaller number of partial products are generated, the complexity and time acquired to accumulate partial results are both reduced eventually.

The special and well established moduli sets of $2^n - 1, 2^n$ and $2^n + 1$ are widely

Table 5.2: A list of some known Mersenne prime

p	M_p	M_p digits	M_p bit length
2	3	1	2
3	7	1	3
5	31	2	5
7	127	3	7
13	8191	4	13
17	131071	6	17
19	524287	6	17
31	2147483647	10	31
61	2305843009213693951	19	61
89	618970019642...137449562111	27	89
107	162259276829...578010288127	33	107
127	170141183460...715884105727	39	127
...			

used in digital and signal processing, Residue Number System(RNS), cryptography areas due to its unique number theorectic properties. For example, a modulo $2^n - 1$ addition is equivalent to one's complement. However, only modulo number in form of $2^n - 1$ could be adopted in our design. Below are the reasons why.

In mathematics, a integer number with form of $2^n + 1$ is named as Fermat number, although there exist infinite amount of Fermat numbers, only a few of them are prime numbers. As for now, the solely 5 Fermat prime are discovered and listed: F_0, F_1, F_2, F_3, F_4, where F_n has form of $2^{2^n} + 1$. Obviously, special moduli set of $2^n + 1, 2^n$ are not available for LTV scenario because it require modulo number q to be a large prime number.

That left us with only choice of modulo $2^n - 1$. The q needs to be a prime number

and also satisfy the formation of $2^n - 1$. In mathematics, a Mersenne prime is a prime number that is one less than a power of two, that is a prime number of the form $M_n = 2^n - 1$ for some integer n. A list of some known Mersenne primes are show in Table 5.2.

Modular $2^n - 1$ arithmetic is congruent to zero modulo $2^n - 1$, while zero can be represented by an n-bit binary string of all zeros or all ones in modulo $2^n - 1$ arithmetic. Also the carry output at the most significant bit position of each stage has a weight of 2^n, which, in modulo $2^n - 1$ arithmetic, is equal to 1.

In LTV Fully Homomorphic Encryption scheme, one of the most crucial parts is deciding the value of q. Mersenne prime number q can be utilized to optimize modulo reduction operation on multiplier design level by selecting certain parameter sets from existing Mersenne prime numbers shown in Table 5.3.

Table 5.3: Selected parameter sets

q	bit length
M_{31}	31
M_{61}	61
M_{89}	89
M_{107}	107
M_{127}	127

5.3 Modular Adder

An adder is an indispensable component for a processing system and integrated circuits. In our proposed work, a particular kind of adder is needed for modular $2^n - 1$ addition with good implementation performances.

In general cases, area and time delay has always been two most important factors that should be taken in consideration when choosing the right adder. Theoretically, prefix adder is much faster than carry look ahead adder, carry skip adder, etc. Compared to other adder topologies, prefix adder has highly regular structure, it consists

of only a few sub-modules.

Integrated circuits have their own unique form of wiring to connect together the micro miniature components they contain. Interconnection in a given layer can not cross over each other without sorting out. So that the interconnection of one arithmetic module topology plays important role in operation unit performance. Prefix adder has simple layout and regular interconnections.

A comprehensive analysis of different adder topologies of different word size is provided through existing works. The analysis and results are helpful for choosing a specific adder for the application of our proposed architecture.

The performance comparison of different popular adders in the literature is shown as Table 5.4[54][55][56][57].

Table 5.4: An adder performance comparison

Adder size(bit)	16	32	64	128	16	32	64	128
Adder type	Delay(ns)[54][55][54]				Area(μm^2)[54]		Area[57]	
Kogge-Stone adder	0.42	0.49	0.87	1.21	1005.7	2254.98	481	1272
Ripple-carry-adder	1.14	2.23	-	1.61	285.33	570.67	418	9312
Carry lookahead adder	0.59	0.83	1.08	1.19	626.04	1360.50	-	410
Carry select adder	0.60	0.85	-	-	610.55	1387.25	-	-
Carry skip adder	0.65	0.96	1.59	2.50	445.7	780.28	-	-

From the table, it is appropriate to use parallel prefix adder as modular adder in the LFSR based multiplier. There are extra benefits of utilizing prefix adder. It is discovered carry propagation in the modular addition of two n-bit number is basically a prefix problem. The prefix structure is defined as the outcome of operation depends on the initial inputs. Therefore parallel prefix adder could be used to perform the efficient algorithm with execution of operations in parallel, which is done by segmentation into smaller pieces computed in parallel.

Since there are a dozen of different types of prefix adders in the literature. Some of the key architectures for carry calculations are Kogge-Stone adder, J.Sklansky adder, Ladner-Fisher adder, Brent Kung adder, Han Carlson adder and S.Knowles adder.

According to the implementation analysis[2] of Table 5.5, it is obvious that Kogge-Stone adder has the best results of time cost. So that Kogge-Stone style is used for the build up of the parallel prefix adder.

Table 5.5: A comparison of different prefix adders[2]

Adder type	Area(μm^2)	Total power (μW)	Delay(ps)
Brent kung	1354.99	85.08	825.9
Han Carlson	1832.34	102.51	593.9
S.Knowles	2345.57	127.64	565.7
Lander Fischer	1512.49	90.21	676.8
J.Sklansky	1681.52	99.58	831.8
Kogge Stone	2669.91	137.22	561.5

The structure of an parallel prefix adder is illustrate as Figure 5.3. In order to

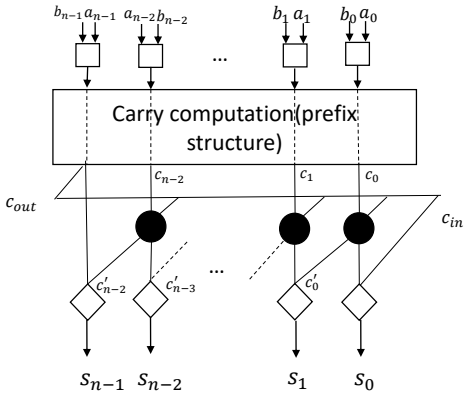

Fig. 5.3: General structure of prefix adder

describe the property of prefix problem, we define a operator $\bullet$, for $i = 0, 1, ..n$, suppose $A = a_{n-1}a_{n-2}...a_0$ and $B = b_{n-1}b_{n-2}...b_0$ represent two numbers to be added and $S = s_{n-1}s_{n-2}...s_0$ denote their output sum. Parallel prefix addition is staged into three steps. The pre-processing stage, prefix computation and post processing stage. In pre-processing stage, carry generate g, carry propagate p and half sum h_i

is computed as:

$$g_i = a_i b_i$$

$$p_i = a_i + b_i$$

$$h_i = a_i \oplus b_i$$

Carry computation can be transformed into a prefix problem using the operator $\bullet$ defined as follows:

$$(g_m, p_m) \bullet (g_k, p_k) = (g_m + p_m \cdot g_k, p_m \cdot p_k). \tag{24}$$

input is a vector of pairs of carry generate and carry propagate bits:

$$(g_n, p_n), (g_{n-1}, p_{n-1}), ..., (g_1, p_1), (g_0, p_0),$$

while the output is new pairs of vectors:

$$(G_n, P_n), (G_{n-1}, P_{n-1}), ..., (G_1, P_1), (G_0, P_0).$$

The carry bits c_i for each bit position $c_i = G_i$, where

$$\begin{cases} (g_0, p_0), & if\ i = 0, \\ (g_i, p_i) \bullet (G_{i-1}, P_{i-1}), & if\ 1 \leq i \leq n-1. \end{cases}$$

For $i = 0$ to $n - 1$,

$$c_i = \ the\ first\ component\ of\ (g_i, p_i) \bullet ... \bullet (g_0, p_0) \bullet (g_{n-1}, p_{n-1}) \bullet ... \bullet (g_{n+1}, p_{n+1}) \tag{25}$$

In Figure 5.3, consider $G_i^{'}, P_i^{'}$ be the group generate and group propagate in the next level, then

$$(G_i^{'}, P_i^{'}) = (G_i + P_i \cdot c_{-1}, P_i) \tag{26}$$

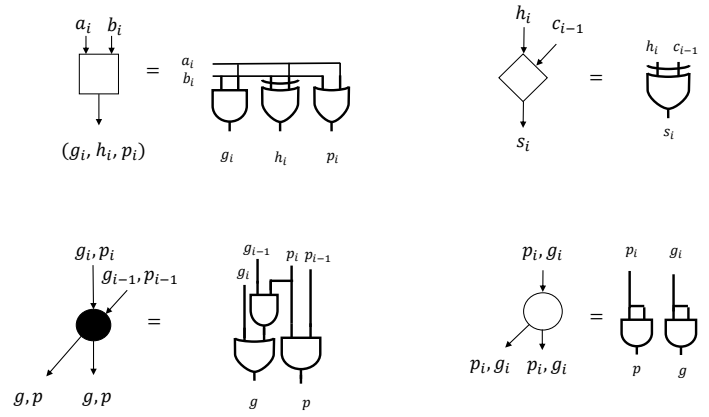

Fig. 5.4: Prefix adder logic operators and implementation

We get $c_i^{'} = \ the\ first\ component\ of\ (G_i^{'}, P_i^{'}) = (G_i + P_i \cdot G_{n-1}, P_i)$, which means that the parallel prefix adder is transformed into a modulo $2^n - 1$ adder using G_{n-1} as c_{in}.

After the computation of the carries c_i, in the last step of post processing stage, the sum bits s_i can be computed as follows where $i = 0$ to $n - 1$,

$$s_i = h_i \oplus c_{i-1}$$

The Modular adder with prefix structure are represented in four modules implemented by logic gates. The prefix adder logic operators are visualized in Figure 5.4.

The carry computation use Kogge and Stone style, which is the fastest among other existing structure, shown in Figure 5.5.

5.4 Modular Subtractor

The design of modulo $2^n - 1$ subtractors is based on the modulo $2^n - 1$ adder. Figure 5.6 shows the structure of the proposed modular subtractor adapted from modular adder.

Let $A = a_{n-1}...a_0$ and $B = b_{n-1}...b_0$ denote two n-bit modulo $2^n - 1$ operands, such that $0 \leq A, B \geq 2^n - 1$. The difference, D, of A and B modulo $2^n - 1$ is equal

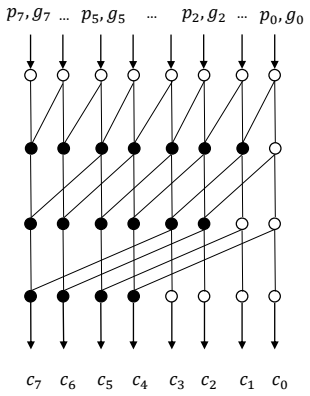

Fig. 5.5: Parallel prefix structure with n=8

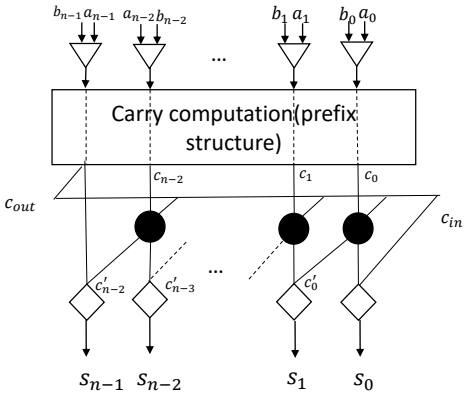

Fig. 5.6: Modular $2^n - 1$ subtractor

to

$$D = |A - B|_{2^n-1} = |A + (2^n - 1) - B|_{2^n-1} = \left|A + \overline{B}\right|_{2^n-1} \tag{27}$$

Where $\overline{B}$ denotes the one's complement of operand B. It is obvious that the difference D is actually an addition of A and $\overline{B}$ modulo $2^n - 1$, resulting that a straightforward hardware implementation based on modulo $2^n - 1$ adder.

A combined modulo $2^n - 1$ adder or subtractor can also be easily derived by replacing the □ module in modular adder structure with a new module ■ where the input XOR gates is amended to NXOR gate, and a NOT gate is added before the inputs to AND and OR gates in order to invert the bits of operand B in case of subtraction as shown in Figure 5.7.

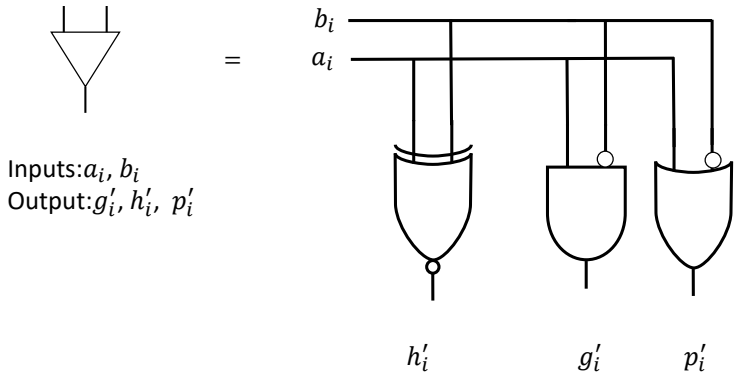

Fig. 5.7: Modular $2^n - 1$ subtractor submodule

The truth table of the subtractor submodule is given by Table 5.6.

Table 5.6: Truth table for Modular subtractor module

a_i	b_i	h'_i	g'_i	p'_i
0	0	1	0	1
0	1	0	0	0
1	0	0	1	1
1	1	1	0	1

5.5 Modular Multiplier

We adapted a radix-4 booth encoded modulo 2^n-1 multiplier for our design. Numbers of partial products are reduced through booth algorithm. Further reduction of the partial product can be implemented by CSA arrays The two operands produced are added in the parallel prefix modulo $2^n - 1$ adder which we covered previously. The final product is the addition result of the prefix adder.

For modulo multiplication:

$$P = X \cdot Y \ mod \ (2^n - 1) \tag{28}$$

Various ROM-based solutions using lookup table have been proposed befre sophisticated methods are introduced to reduce the table sizes by combining smaller lookup table with simple arithmetic operations, such as additions. For word lengths larger than particular bits, however, these solutions still require large ROM or too many numbers of clock cycles for evaluation.

Efficient modulo $2^n - 1$ multiplier are widely researched in selected works of [58] [59] [60] [61].

The major difference between conventional multiplier and modulo 2^n-1 multiplier is that the partial product array of normal binary multiplier is shaped as triangle, while the partial product array of modulo $2^n - 1$ multiplier is shaped as rectangle. All rows of partial product for the modulo $2^n - 1$ multiplier are located on the same bit position.

The carry output at the most significant bit position of each stage has a weight of 2^n, which is equal to 1 in modulo $2^n - 1$ arithmetic. In other words, carry out of the most significant bit should be shifted to the least significant bit position.

In traditional multiplier, the column size for different column is different, while for modulo $2^n - 1$ multiplier, the column size of any column is the same, which provide

better regularity and easier implementation.

In this section, a proposed modified radix-4 booth algorithm modulo $2^n - 1$ multiplier. Supposed that $A = a_{n-1}a_{n-2}a_{n-3}...a_2a_1a_0$ and $B = b_{n-1}b_{n-2}b_{n-3}...b_2b_1b_0$ are two n-bit input of the modulo $2^n - 1$ multiplier. A is the multiplicand and B is multiplier. Let $|A|_x$ denote the modulo x residue of A.

Using booth algorithm to reduce the number of partial products, the multiplicand B can be represented by:

$$B = \sum_{i=0}^{n-1} b_i 2^i = (b_0 - 2b_1) + \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} (b_{2i-1} + b_{2i} - 2b_{2i+1}) 2^{2i} \tag{29}$$

Where $b_j = 0$ for $j \geq n$ and $\lfloor k \rfloor$ denote the largest integer smaller than or equal to k.

The value of the product $A \times B$ modulo $2^n - 1$ can consequently be expressed by:

$$\begin{aligned} |AB|_{2^n-1} &= \left| A \sum_{i=0}^{\lfloor \frac{n+1}{2} \rfloor - 1} (b_{2i-1} + b_{2i} - 2b_{2i+1}) 2^{2i} \right|_{2^n-1} \\ &= \left| \sum_{i=0}^{\lfloor \frac{n+1}{2} \rfloor - 1} \left| A(b_{2i-1} + b_{2i} - 2b_{2i+1} 2^{2i}) \right|_{2^n-1} \right|_{2^n-1} \\ &= \left| \sum_{i=0}^{\lfloor \frac{n+1}{2} \rfloor - 1} P^{(i)} \right|_{2^n-1} \end{aligned} \tag{30}$$

Where $P^{(i)} = |A(b_{2i-1} + b_{2i} - 2b_{2i+1})2^{2i}|_{2^n-1}$

Since the term $b_{e,i} = (b_{2i-1} + b_2 i - 2b_{2i+1})$ can take values in the set $(-2, -1, 0, +1, +2)$, $b_{e,i} 2^{2i} = s2^j$, where

$$s = \begin{cases} +1, & if\ b_{e,i} = +1, +2 \\ 0, & if\ b_{e,i} = 0 \\ -1, & if\ b_{e,i} = -1, -2 \end{cases}$$

$$j = \begin{cases} 0, & if\ b_{e,i} = 0 \\ 2i, & if\ b_{e,i} = +1, -1 \\ 2i+1, & if\ b_{e,i} = +2, -2 \end{cases}$$

From this, we get $P^{(i)} = |sA^j|_{2^n-1}$.

Using the following expression, the partial products can derived as $P^{(i)} = |sA2^j|_{2^n-1}$, in n bits, directly from multiplicand A.

For the three circumstances:

- $s = +1$. we get :

$$\begin{aligned} \left|sA2^j\right|_{2^n-1} &= |a_{n-1-j}a_{n-2-j}...a_0a_{n-1}a_{n-2}...a_{n-j}|_{2^n-1} \\ &= a_{n-1-j}a_{n-2-j}...a_0a_{n-1}a_{n-2}...a_{n-j} \end{aligned}$$

- $s = -1$. Taking into account that $|-Y|_x = |x - Y|_x$, so that:

$$\begin{aligned} \left|sA2^j\right|_{2^n-1} &= |(2^n - 1) - (a_{n-1-j}a_{n-2-j}...a_0a_{n-1}a_{n-2}...a_{n-j})|_{2^n-1} \\ &= |\overline{a_{n-1-j}a_{n-2-j}}...\overline{a_0a_{n-1}a_{n-2}}...\overline{a_{n-j}}|_{2^n-1} \\ &= \overline{a_{n-1-j}a_{n-2-j}}...\overline{a_0a_{n-1}a_{n-2}}...\overline{a_{n-j}} \end{aligned}$$

- $s = 0$. In this case, $|sA2^j|_{2^n-1} = 0 = \underbrace{000...000}_{nbits}$

To give a more detailed expression of the all cases, Table 5.7 is provided with the formation of the partial products.

According to the first and the last row of Table 5.7, all 1s input and all 0s input are supposed to be treated as a zero input.

A Booth Encoder (BE) and Booth Selector (BS) are used for input transform of B to partial product bit b_i with multiplicand A. The Booth Encoder(BE) produces a signed digit represented by a sign bit s and two encoded magnitude bits c_k and c_{k+1}

Table 5.7: Formation of the partial products

b_{2i+1}	b_{2i}	b_{2i-1}	Meaning	$P^{(i)}$
0	0	0	0	000...000 or 111...111
0	0	1	$\|A2^{2i}\|_{2^n-1}$	$a_{n-1-2i}a_{n-2-2i}...a_0a_{n-1}a_{n-2}...a_{n-2i}$
0	1	0	$\|A2^{2i}\|_{2^n-1}$	$a_{n-1-2i}a_{n-2-2i}...a_0a_{n-1}a_{n-2}...a_{n-2i}$
0	1	1	$\|A2^{2i+1}\|_{2^n-1}$	$a_{n-1-(2i+1)}a_{n-2-(2i+1)}...a_0a_{n-1}a_{n-2}...a_{n-(2i+1)}$
1	0	0	$\|-A2^{2i+1}\|_{2^n-1}$	$\overline{a_{n-1-(2i+1)}a_{n-2-(2i+1)}...a_0a_{n-1}a_{n-2}...a_{n-(2i+1)}}$
1	0	1	$\|-A2^{2i}\|_{2^n-1}$	$\overline{a_{n-1-2i}a_{n-2-2i}...a_0a_{n-1}a_{n-2}...a_{n-2i}}$
1	1	0	$\|-A2^{2i}\|_{2^n-1}$	$\overline{a_{n-1-2i}a_{n-2-2i}...a_0a_{n-1}a_{n-2}...a_{n-2i}}$
1	1	1	0	000...000 or 111...111

from 3 consecutive multiplier bits b_{2i-1},b_{2i},b_{2i+1}. The leftmost bit of a BE bit slice b_{2i-1} is shared with the rightmost bit of the preceding BE bit slice b_{2i+1}. To generate the $P^{(i)}$ vector given by Table 5.7, the encoded bits, c_k,c_{k+1} are used to select either the multiplicand or one bit shift multiplicand. The sign bit s determine if the selected word need ot be complemented.

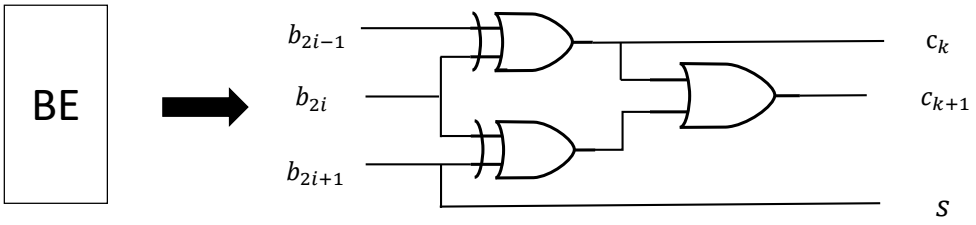

Fig. 5.8: Implementation of Booth Encoder(BE)

Table 5.8: Truth table for Booth Encoder(BE)

b_{2i+1}	b_{2i}	b_{2i-1}	s	c_{k+1}	c_k
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	0	1
1	0	0	1	1	0
1	0	1	1	0	1
1	1	0	1	0	1
1	1	1	1	0	0

A bit slice of the radix 4 Booth Encoder (BE) is given in Figure 5.8 with its truth Table 5.8. A bit slice of the radix 4 Booth Selector (BS) is given in Figure 5.9 with

its truth Table 5.9.

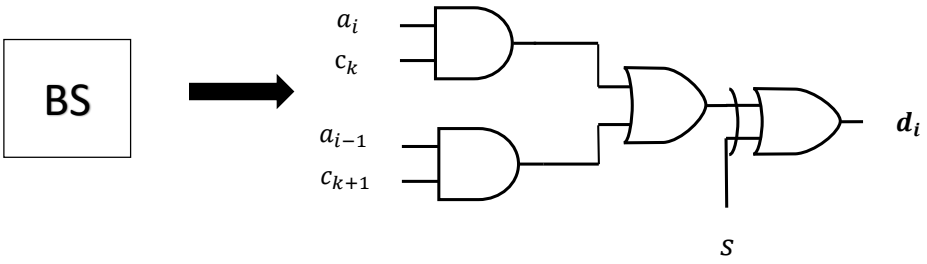

Fig. 5.9: Implementation of Booth Selector(BS)

Table 5.9: Truth table for Booth Selector(BS)

s	c_{k+1}	c_k	d_i
0	0	0	0
0	0	1	a_i
0	1	0	a_{i-1}
1	0	0	$\overline{a_{i-1}}$
1	0	1	$\overline{a_i}$
1	1	0	1

For radix-4 Booth encoding scheme, the number of partial products will be reduced to $\lfloor n/2 \rfloor + 1$ and $\lfloor n/2 \rfloor$. where $\lfloor a \rfloor$ represent the smallest integer greater that or equal to a.

To understand the essence of this algorithm, consider a multiplication operation in which the multiplier is positive and has a single block of 1s, for example, 0011110. To derive the product, we could adder four appropriately shifted versions of the multiplicand, as in the standard procedure. However, we can reduce the numbe of required operations by regarding this multiplier as the difference between two numbers:

$$\begin{array}{r} 0100000\ (32) \\ -0000010\ (2) \\ \hline 0011110\ (30) \end{array}$$

This suggest that the product can be generated by adding 2^5 times the multiplicand to the 2's complement of 2^1 times the multiplicand. For convenience, we can describe the sequence of required operations by recording the preceding multiplier as $0\ +1000\ -10$.

For high performance modulo multiplication, dedicated multipliers are required which can be implemented as combinational or pipeline circuits. A modulo $2^n - 1$ multiplier architecture with modulo reduced, Booth recorded partial products and with concurrent modulo reduction during carry save addition is improved in this paper. In following paragraph, the proposed design for a modulo $2^n - 1$ multiplier will

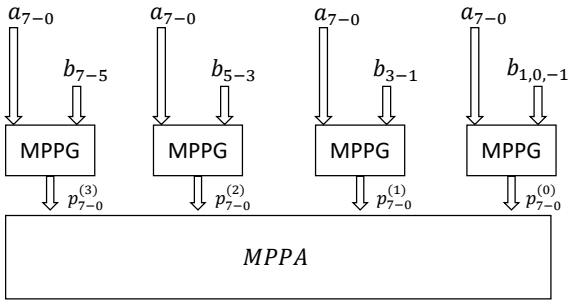

Fig. 5.10: Modular Multiplier

be presented. The Figure 5.10 shows the general structure of modular $2^n - 1$ multiplier with $n = 8$. The structure is consisted of modules of MPPG (Multiplication Partial Product Generation) and MPPA(Multiplication Partial Product Accumulation). The multiplicand B and multiplier A input to MPPG with output of partial product bits fed into MPPA module.

In MPPG module, there are 4 Booth Encoders and 32 Booth Selectors. After the multiplicand B and multiplier A inputs to Booth Encoder and Booth Selector and undergo radix 4 Booth algorithm, the partial product bits are generated at each level as $p_i^{(j)}$, where $p_i^{(j)}$ means the i bit of the partial product at level j. Figure 5.11 shows the structure of MPPG.

Note that all n-bit partial products $p_i^{(j)}$ have the same magnitude, as opposed to

ordinary multiplication, where the partial products have increasing magnitude. The number of products bits to be added is the same for all bit positions. This allows their addition by a highly regular modulo carry save adder layout composed of (3,2) counters.

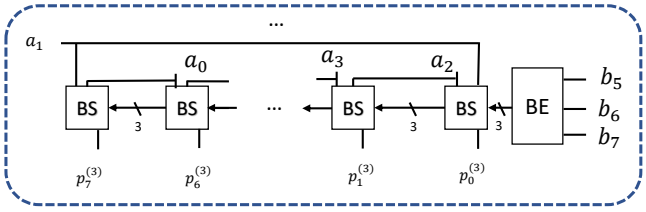

Fig. 5.11: Multiplication Partial Product Generation(MPPG)

In the MPPA module, a carry save adder tree is used to accelerate the addition of the partial products, which is a easily applicable technique to integrated with modulo multiplier. The proposed carry save adder tree structure is more regular than conventional multiplier, because the same number of bits is added for each bit position and the carry outs are fed back into the carry ins. In cell based designs, the lower regularity of tree structures compared to linear ones has a negligible impact on circuit area, while considerable speedup is achieved.

Figure 5.12 depicts the MPPA module of the modular multiplier architecture. Generated partial products bits are fed into n-operand carry save addition and go through carry propagate addition steps, note that the carry out is directly fed into carry in of next level. Final step is performing modulo $(2^n - 1)$ addition with modular adder proposed.

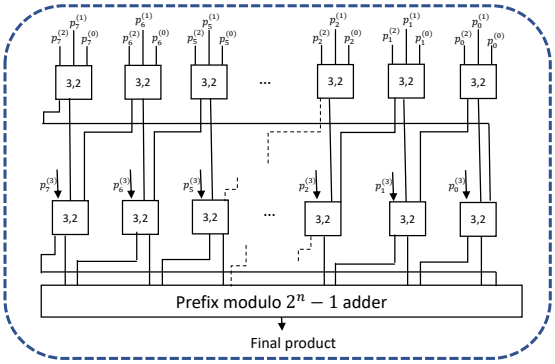

Fig. 5.12: Multiplication Partial Product Accumulation(MPPA)

6 COMPLEXITY ANALYSIS AND FPGA IMPLEMENTATION

This chapter gives a introduction for complexity in terms of area and time in proposed architecture of multiplication in $R_q[x]$ for LTV Fully Homomorphic Encryption scheme. The complexity of sub modules comprised for the general architecture is first given. For the different sub modules, we have a table shows the corresponding gate count and critical path delay. A list of area cost and critical path delay are presented in the next table on a general view for the LFSR based multiplier structure.

The results of implementations and simulations on FPGA paltform are also given in this chapter. After a general measurement of several main implementation technologies, we compare our results with some of the existing implementations outcome and analyzed the performance differences. We use device of Cyclone IV GX EP4CGX15BF14C6 for compilation on Quartus II 13.1 web edition.

6.1 Complexity Analysis

Table 6.1: Modules used in proposed architecture

Module	Modular multiplier	Modular adder	Register	Modular subtractor
# Modules	N	$N-1$	N	1

A brief estimation of complexity is presented in this section. This analysis assumes that each gate counts as one elementary gate for both area and delay. The model ignores fan in and fan out. The validation of the estimates that it produce will be carried out later by FPGA implementation.

Table 6.1 gives the module complexity of proposed polynomial multiplier in $R_q[x]$ architecture. The structure require $N-1$ modular q adders, 1 modular q subtractor, N modular q multipliers and N registers. Each register stores value of $[\log_2 q]$ bits

length.

Table 6.2: Space complexity for each module

Modular units	Space complexity				
	# XOR	# AND	# OR	# NXOR	# NOT
Adder	$2n-1$	$2n\lceil log_2 n\rceil + 3n$	$n + n\lceil log_2 n\rceil - 2^{\lceil log_2 n\rceil} + 1$	0	0
Subtractor	$n-1$	$2n\lceil log_2 n\rceil + 3n$	$n + n\lceil log_2 n\rceil - 2^{\lceil log_2 n\rceil} + 1$	n	n
Multiplier	$2n^2-1$	$3n^2-4n$	$\frac{3}{2}n^2 - n$	0	0

For combination of different modules. The gate count is show in Table 6.2. We use basic logic gate of XOR gate, AND gate, OR gate, NXOR gate and NOT gate count to measure the space complexity. Each full adder (3,2 counter) is of two XOR gates, 2 AND gates and 1 OR gate. From the table we can see that the number of gate count of modular subtractor is close to that of modular adder with same amount of XOR gate, AND gate, OR gate and more NXOR gate, NOT gate.

For time complexity, we show critical time delay in Table 6.3.

Table 6.3: Time complexity for each module

Modular units	Critical path delay
Adder	$2\lceil log_2 n\rceil + 3$
Subtractor	$2\lceil log_2 n\rceil + 4$
Multiplier	$\frac{3}{2}n + 2\cdot\lceil log_2 n\rceil + 4$

In the end, we give the comprehensive presentation of time and space complexity of proposed truncated polynomial ring multiplier in $R_q[x]$ with Table 6.4 and 6.5.

Table 6.4: Space complexity of proposed multiplier in $R_q[x]$

# XOR	$2N(n^2+n-1)-n$
# AND	$N(3n^2 + 2n\lceil log_2 n\rceil - n)$
# OR	$N(\frac{3}{2}n^2 + n\lceil log_2 n\rceil - 2^{\lceil log_2 n\rceil})$
# NXOR	n
# NOT	n

Table 6.4 shows that the total area cost is the gate count accumulation of all modular multiplier, modular adder and modular subtractor in the whole architecture

with AND gate, OR gate, XOR gate grow linearly with degree N and power of 2 with modulo length n.

Table 6.5 gives a general comparison of time and space complexity for proposed architecture and other related works, where the critical path go through modular mulitplier and modular adder/subtractor. The critical path delay for proposed architecture is estimated as $\frac{3}{2}n+4\lceil log_2 n\rceil-2^{\lceil log_2 n\rceil}+8$. It require N clock cycles to complete all multiplication of the polynomials. While the time complexity of polynomial multiplication operation is $O(N^2)$ in [1], $O(NlogNloglogN)$ in [52] and $O(NlogN)$ in [62], the architecuture level space complexity is $O(n^2)$ in [1], $O(Nn^2)$ in [52] and [62].

Table 6.5: Time complexity comparison

Work	Space complexity(gate count)	Time complexity (latency)
proposed	$O(Nn^2)$	$O(N)$
[1]	$O(n^2)$	$O(N^2)$
[52]	$O(Nn^2)$	$O(NlogNloglogN)$
[62]	$O(Nn^2)$	$O(NlogN)$

6.2 FPGA Implementation Results

6.2.1 Implementation results

We choose FPGA as the implementation platform with our research tool. The software is Altera Quartus II 13.1 web edition with target device of Cyclone IV GX EP4CGX15BF14C6. The details of implementation results are shown below in Table 6.6.

We collected simulation result data on FPGA implementation for LTV FHE encryption large polynomial multiplication with ring dimension of $N \in \{512, 1024, 2048, 4096\}$. A couple of representative integer word sizes n selected in range $\in \{31, 61, 89, 107, 127\}$ are provided. Although latency is the most significant experiment performance factor in our design, we manage to demonstrate a series of itmes in imple-

Table 6.6: FPGA results for the proposed system

N	n	Logic elements	Memory bits	Register	Fmax	Latency
512	31	2694	301568	129	41.05MHz	0.012ms
	61	8442	593408	189	26.46MHz	0.019ms
	107	23586	1040896	281	15.81MHz	0.032ms
1024	31	2712	349184	129	41.93MHz	0.024ms
	61	8450	687104	189	25.68MHz	0.039ms
	107	23591	1205248	281	16.20MHz	0.063ms
2048	61	8498	874496	189	25.98MHz	0.078ms
	89	16965	1275904	245	18.98MHz	0.107ms
	107	23023	1533952	281	16.37MHz	0.125ms
	127	27805	1533952	281	14.99MHz	0.136ms
4096	61	8507	1249280	189	26.62MHz	0.153ms
	89	16887	1822720	245	19.03MHz	0.215ms
	107	23680	2191360	281	16.62MHz	0.243ms
	127	27840	2191360	281	13.22MHz	0.309ms

mentation results in terms of logic elements, registers, memory bits and maximum frequency.

From the Table 6.6 we can see that the latency grows when value of N or n increases. We choose the sets of parameter according to the Mersenne number form $2^n - 1$ proposal.

6.2.2 Implementation comparison

The works on implementation of LTV methods can be roughly categorized into several types, each has their advantages and drawbacks explained below:

CPU: This may be a classic desktop or laptop CPU or the one of the embedded device. It usually has rather few computation cores (< 20), but it can use the ones it has very fast and can execute arbitrary instructions (in mostly arbitrary order) from its assembler language. Cryptography algorithms that run on CPUs are most software-implemented because the algorithms are merely information given to the CPU for execution. CPUs are best at running complex, linear algorithms.

GPU: This could be desktops or laptops graphics processor or it may be supercomputing computation accelerator. Its characteristics are the high number of cores, the

low speed of each core and the limited instruction set. GPUs are made to run simple algorithms massively parallel. Much assemble to CPUs, they accept their algorithms as pieces of information, so it's the software that implements the algorithm.

FPGA: The FPGA is typically found in small embedded devices and is running specialized algorithms. Its hardware can be configured post shipping, at the expense of lower speeds of operation than with ASICs. With FPGAs, changing the hardware layout of the integrated circuit to run the algorithm is possible. Hence algorithms run by FPGAs are considered to be hardware implemented, because in its current state, the hardware can run only this exact algorithm.

ASIC: This is an integrated circuit that is manufactured to run exactly one algorithm. ASICs provide high speed for this algorithm usually and are used when speed matters. An example application are Hardware Security Modules (HSMs) which commonly use ASICs to accelerate the execution of cryptographic operations (like AES encryption). Crypto processors commonly are simple processors with additional crypto-specific ASICs. We are confident to see the implementation of LTV FHE carried out on ASIC platform in the near future.

Next we generally discuss about the four types of implementation technologies in terms of security level, speed and cost.

Security: For the security of each platform, the tendency is that the security will increase if we go down the list from CPU to ASIC. CPUs are usually occupied by many different processes, allowing side-channel attacks, GPUs are sometimes used for cryptosystem, FPGAs should provide more security than CPUs and GPUs because there are less noise of other operations on the chip and ASICs have the same benefit, but with a bit more security assurance.

Speed: The speed increases similarly if we go down the list. CPUs must be capable of doing many different things and can not be too much optimized in one direction, some goes for GPUs although they have much more computation power if needed.

Table 6.7: Cryptosystem implementation types and comparison

Implementation types		Security Risk	Speed	Cost
Software	CPU	highest	slowest	lowest
	GPU	high	slow	low
Hardware	FPGA	low	fast	high
	ASIC	lowest	fastest	highest

FPGAs are faster because it can provide more optimizations and ASICs are fastest because it customized for a particular use.

Cost: The price has a similar order. CPUs are easy to obtain, cheap to program and it can make program running quickly. GPUs are also quite easy to obtain, it is a bit more expensive to effectively program and can run the code somewhat fast. FPGAs are more expensive and require to design the algorithm using the hardware language of FPGA, which cost a lot of time, expertise and money. ASICs need to be planned before they are built, they have long design cycles, but once the design is down. Manufacturing them is relatively easy, a low price is available.

Table 6.7 shows the differences between hardware implementation and software implementation platforms.

We evaluate the performance of our FPGA simulation of LTV encryption multiplication operation. The Table 6.8, and Table 6.9 below show the proposed work simulations compared with that of some existing works.

Table 6.8 collects accordingly some large combinations of parameters in the multiplication operation for polynomial degree $N \in \{512, 1024, 2048, 4096\}$. We provide the experimental data for the same parameter sets of LTV Fully Homomorphic Encryption scheme with other works.

The table illutrate the comparison with implementation of several runtime/latency results from existing experiments in[1],[50],[51],[62].

Compared to CPU implementation in [50], our work has huge speed advantage

Table 6.8: Implementation speed comparison

Work	Parameter n	Platform	Latency(ms)	Speedup
N=512				
proposed	31	Cyclone IV	0.012	5.5×
FPGA[1]	32	Virtex 7	0.066	1×
proposed	61	Cyclone IV	0.019	3.5×
N=512				
proposed	61	Cyclone IV	0.019	155×
CPU[50]	64	Matlab	3.27	1×
N=1024				
FPGA [1]	26	Spartan 6	0.069	1×
proposed	31	Cyclone IV	0.024	2.9×
N=2048				
FPGA [62]	58	Spartan 6	0.282	1×
proposed	61	Cyclone IV	0.078	3.6 ×
N=2048				
CPU [50]	100	Matlab	7.94	1×
proposed	107	Cyclone IV	0.125	63.5×
N=4096				
proposed	61	Cyclone IV	0.153	7.8×
GPU[51]	64	GeForce 690	1.2	1×
N=4096				
proposed	107	Cyclone IV	0.243	71.5×
CPU [50]	109	Matlab	17.38	1×

of over 40 times for parameter $N = 512, 1024, 2048, 4096$. For the parameter of $N = 4096, n = 64$ for truncated polynomial ring multiplier in $R_q[x]$, work in [51] with GPU based environment achieved latency of 1.2ms while we provide result of 0.153ms for $N = 4096, n = 61$.

We measure the works in [1] and [62] which provide a certain experiment data based on implmentation with different target devices in FPGA platform. For value of $N = 512$ and $N = 1024$, [1] achieved latency of 0.066ms and 0.069ms respectively for LTV Fully Homomorphic Encryption scheme encryption multiplication part. Our work demonstrate 0.012ms for $N = 512, n = 31$, 0.019ms for $N = 512, n = 61$ and 0.024ms for $N = 1024, n = 31$.

The FPGA based experimental result in [62] shows a 0.282ms delay for $N =$

Table 6.9: FPGA implementations and comparison

Work	n	Platform	#Logic units used/#logic units total	Latency(ms)	Speedup
N=512					
proposed	31	Cyclone IV	2694/149k(1.8%)	0.012	5.5×
FPGA[1]	32	Virtex 7	171/91k(< 1%)	0.066	1×
proposed	61	Cyclone IV	8442/149k (5.6%)	0.019	3.5×
N=1024					
FPGA[1]	26	Spartan 6	182/0.6k(30%)	0.069	1×
proposed	31	Cyclone IV	2712/149k(1.8%)	0.024	2.9×
N=2048					
FPGA[62]	58	Spartan 6	3846/15k (24.3%)	0.282	1×
proposed	61	Cyclone IV	8498/149k (6%)	0.078	3.6×

$2048, n = 58$ in large truncated polynomial ring multiplication operation, while we achieve 0.078ms for $N = 2048, n = 61$.

From the Table 6.9 we get brief idea about the different family of FPGA device utilization.

For $N = 512, n = 32$ in [1], 171($< 1\%$) slice LUT are used, while in our work for $N = 512, n = 321$, 2765 (1.8%) LE are used. In addition, for parameter of N=1024, n=26, [1] used 182(30%) of the 0.6k slice LUT in total and we used 2712(1.8%) of total 149k LEs for $N = 1024, n = 31$.

For parameter $N = 2048, n = 58$, 3846 (24.3%) of total 15k slice LUT are used in [62] on Spartan 6. In our work for parameter$N = 2048, n = 61$, 8498(6%) of total 149k LEs are utilized on device Cyclone IV.

7 CONCLUSIONS

This chapter has folowing contents: an overview of research contributions, conclusions based on the implementation comparisons, and possible future work.

7.1 A Summary of Contributions and Significance

After the first chapter on introduction, the thesis provides the mathematical preliminaries of the Homomorphic Encryption, including truncated polynomial ring and specific steps to accomplish the LTV Fully Homomorphic Encryption process, which followed by an overview of existing works on LTV Fully Homomorphic Encryption. Then the architecture to perform encryption algorithm of LTV Fully Homomorphic Encryption are proposed. An elaborate comparison between the proposed works and existing works in complexity and performance is also presented.

The research contributions presented in this thesis include the followings,

- The efficiency of LTV system is significantly impacted by the speed of multiplication of truncated polynomials with high degree and large coefficients.
- An new architecture for computing encryption step in LTV Fully Homomorphic Encryption are presented. The architectural design is based on a new extension to LFSR, which provide compact and pipeline structure and is beneficial for multiplication of truncated polynomials with high degree and large coefficients.
- We also proposed to select a family of special moduli for modular arithmetic over modular reduction operations. Calculation speed is much improved due to the highly efficient modulo computation introduced by the proposed moduli. Within the architecture, novel and efficient modular multiplier, modular adder and modular subtractor are proposed for high speed computation.

- We analyzed the complexity of the proposed architecure in terms of space and time. Our proposed algorithm has great speed advantage obtaining linear time complexity of $O(N)$ over best result of $O(NlogN)$ for previous works. However, the proposed architecture takes space in gate count of $O(Nn^2)$ with little advantage in circuit area perspective.

- The simulation results of proposed architecrture implementation demonstrate great advantage of time and speed for different sets of parameter selections and various paltforms of implementation.

7.2 Possible Future Works

Based on the research works proposed in this thesis, the following research directions may be worthy of further investigation:

- Design of efficient architecture for evaluation process in LTV Fully Homomorphic Encryption scheme.

- Design of algorithm for implementation of LTV Fully Homomorphic Encryption system on ASIC.

- Applying the proposed structure and computation method for other Fully Homomorphic Encryption schemes.

REFERENCES

[1] A. Mkhinini, P. Maistri, R. Leveugle, and R. Tourki, "Hls design of a hardware accelerator for homomorphic encryption," in *Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2017 IEEE 20th International Symposium on.* IEEE, 2017, pp. 178–183.

[2] M. Talsania and E. John, "A comparative analysis of parallel prefix adders," in *Proc. Int. Conf. Comput. Design*, 2013, pp. 29–36.

[3] "September 2017 web server survey," https://news.netcraft.com/archives/2017/09/11/september-2017-web-server-survey.html, accessed: 2017-09-11.

[4] W. Yuan-Zhuo, J. XiaoLong, C. XueQ *et al.*, "Network big datapresent and future," Ph.D. dissertation, 2013.

[5] G. P. Agrawal, *Fiber-optic communication systems.* John Wiley & Sons, 2012, vol. 222.

[6] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[7] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM conference on Computer and communications security.* ACM, 2009, pp. 199–212.

[8] 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.

[9] C. Gentry, *A fully homomorphic encryption scheme.* Stanford University, 2009.

[10] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. ACM, 2012, pp. 1219–1234.

[11] D. Cabarcas, P. Weiden, and J. Buchmann, "On the efficiency of provably secure ntru," in *International Workshop on Post-Quantum Cryptography*. Springer, 2014, pp. 22–39.

[12] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2010, pp. 1–23.

[13] D. Micciancio, "A first glimpse of cryptography's holy grail," *Communications of the ACM*, vol. 53, no. 3, pp. 96–96, 2010.

[14] 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.

[15] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[16] 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*. ACM, 1982, pp. 365–377.

[17] J. Benaloh, "Dense probabilistic encryption," in *Proceedings of the workshop on selected areas of cryptography*, 1994, pp. 120–128.

[18] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1999, pp. 223–238.

[19] I. Damgård and M. Jurik, "A generalisation, a simpli. cation and some applications of paillier's probabilistic public-key system," in *International Workshop on Public Key Cryptography*. Springer, 2001, pp. 119–136.

[20] O. Regev, "New lattice-based cryptographic constructions," *Journal of the ACM (JACM)*, vol. 51, no. 6, pp. 899–942, 2004.

[21] ——, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, p. 34, 2009.

[22] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical gapsvp," in *Advances in cryptology–crypto 2012*. Springer, 2012, pp. 868–886.

[23] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Advances in Cryptology–CRYPTO 2013*. Springer, 2013, pp. 75–92.

[24] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) lwe," *SIAM Journal on Computing*, vol. 43, no. 2, pp. 831–871, 2014.

[25] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, p. 13, 2014.

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

[27] T. Sander, A. Young, and M. Yung, "Non-interactive cryptocomputing for nc/sup 1," in *Foundations of Computer Science, 1999. 40th Annual Symposium on.* IEEE, 1999, pp. 554–566.

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

[29] C. Gentry, "Toward basing fully homomorphic encryption on worst-case hardness," in *Annual Cryptology Conference.* Springer, 2010, pp. 116–137.

[30] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *arXiv preprint arXiv:1704.03578*, 2017.

[31] D. Stehlé and R. Steinfeld, "Faster fully homomorphic encryption," in *International Conference on the Theory and Application of Cryptology and Information Security.* Springer, 2010, pp. 377–394.

[32] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," in *International Workshop on Public Key Cryptography.* Springer, 2010, pp. 420–443.

[33] C. Gentry and S. Halevi, "Implementing gentrys fully-homomorphic encryption scheme," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques.* Springer, 2011, pp. 129–148.

[34] P. Scholl and N. P. Smart, "Improved key generation for gentrys fully homomorphic encryption scheme," in *IMA International Conference on Cryptography and Coding.* Springer, 2011, pp. 10–22.

[35] N. Ogura, G. Yamamoto, T. Kobayashi, and S. Uchiyama, "An improvement of key generation algorithm for gentrys homomorphic encryption scheme," in *International Workshop on Security*. Springer, 2010, pp. 70–83.

[36] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2010, pp. 24–43.

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

[38] H.-M. Yang, Q. Xia, X.-f. Wang, and D.-h. Tang, "A new somewhat homomorphic encryption scheme over integers," in *Computer Distributed Control and Intelligent Environmental Monitoring (CDCIEM), 2012 International Conference on*. IEEE, 2012, pp. 61–64.

[39] Y. G. Ramaiah and G. V. Kumari, "Towards practical homomorphic encryption with efficient public key generation," *International Journal on Network Security*, vol. 3, no. 4, p. 10, 2012.

[40] J.-S. Coron, T. Lepoint, and M. Tibouchi, "Scale-invariant fully homomorphic encryption over the integers," in *International Workshop on Public Key Cryptography*. Springer, 2014, pp. 311–328.

[41] P. S. Pisa, M. Abdalla, and O. C. M. B. Duarte, "Somewhat homomorphic encryption scheme for arithmetic operations on large integers," in *Global Information Infrastructure and Networking Symposium (GIIS), 2012*. IEEE, 2012, pp. 1–8.

[42] N. Aggarwal, C. Gupta, and I. Sharma, "Fully homomorphic symmetric scheme without bootstrapping," in *Cloud Computing and Internet of Things (CCIOT), 2014 International Conference on.* IEEE, 2014, pp. 14–17.

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

[44] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-lwe and security for key dependent messages," in *Annual cryptology conference.* Springer, 2011, pp. 505–524.

[45] T. Wu, H. Wang, and Y.-P. Liu, "Optimizations of brakerski's fully homomorphic encryption scheme," in *Computer Science and Network Technology (ICCSNT), 2012 2nd International Conference on.* IEEE, 2012, pp. 2000–2005.

[46] X. Zhang, C. Xu, C. Jin, R. Xie, and J. Zhao, "Efficient fully homomorphic encryption from rlwe with an extension to a threshold encryption scheme," *Future Generation Computer Systems*, vol. 36, pp. 180–186, 2014.

[47] J. Hoffstein, J. Pipher, and J. H. Silverman, "Ntru: A ring-based public key cryptosystem," in *International Algorithmic Number Theory Symposium.* Springer, 1998, pp. 267–288.

[48] P. Mell, T. Grance *et al.*, "The nist definition of cloud computing," 2011.

[49] M. Almorsy, J. Grundy, and I. Müller, "An analysis of the cloud computing security problem," *arXiv preprint arXiv:1609.01107*, 2016.

[50] K. Rohloff and D. B. Cousins, "A scalable implementation of fully homomorphic encryption built on ntru," in *International Conference on Financial Cryptography and Data Security*. Springer, 2014, pp. 221–234.

[51] W. Dai, Y. Doröz, and B. Sunar, "Accelerating ntru based homomorphic encryption using gpus," in *High Performance Extreme Computing Conference (HPEC), 2014 IEEE*. IEEE, 2014, pp. 1–6.

[52] Y. Doröz, E. Öztürk, E. Savaş, and B. Sunar, "Accelerating ltv based homomorphic encryption in reconfigurable hardware," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2015, pp. 185–204.

[53] D. B. Cousins, K. Rohloff, and D. Sumorok, "Designing an fpga-accelerated homomorphic encryption co-processor," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 2, pp. 193–206, 2017.

[54] R. Uma, V. Vijayan, M. Mohanapriya, and S. Paul, "Area, delay and power comparison of adder topologies," *International Journal of VLSI Design & Communication Systems*, vol. 3, no. 1, p. 153, 2012.

[55] D. H. Hoe, C. Martinez, and S. J. Vundavalli, "Design and characterization of parallel prefix adders using fpgas," in *System Theory (SSST), 2011 IEEE 43rd Southeastern Symposium on*. IEEE, 2011, pp. 168–172.

[56] D. H. Hoe, L. Bollepalli, and C. D. Martinez, "Fpga fault tolerant arithmetic logic: a case study using parallel-prefix adders," *VLSI Design*, vol. 2013, p. 17, 2013.

[57] T. K. Kumar and P. Srikanth, "Design of high speed 128 bit parallel prefix adders," *International Journal of Engineering Research and Applications*, vol. 4, no. 11, pp. 112–115, 2014.

[58] A. Skavantzos and P. B. Rao, "New multipliers modulo 2/sup n/-1," *IEEE Transactions on Computers*, vol. 41, no. 8, pp. 957–961, 1992.

[59] Z. Wang, G. A. Jullien, and W. C. Miller, "An algorithm for multiplication modulo (2/spl and/n-1)," in *Circuits and Systems, 1996., IEEE 39th Midwest symposium on*, vol. 3. IEEE, 1996, pp. 1301–1304.

[60] R. Zimmermann, "Efficient vlsi implementation of modulo (2/sup n//spl plusmn/1) addition and multiplication," in *Computer Arithmetic, 1999. Proceedings. 14th IEEE Symposium on*. IEEE, 1999, pp. 158–167.

[61] C. Efstathiou, H. T. Vergos, and D. Nikolos, "Modified booth modulo 2/sup n/-1 multipliers," *IEEE Transactions on Computers*, vol. 53, no. 3, pp. 370–374, 2004.

[62] D. D. Chen, N. Mentens, F. Vercauteren, S. S. Roy, R. C. Cheung, D. Pao, and I. Verbauwhede, "High-speed polynomial multiplication architecture for ring-lwe and she cryptosystems." *IEEE Trans. on Circuits and Systems*, vol. 62, no. 1, pp. 157–166, 2015.

VITA AUCTORIS

NAME:	Qiang Zeng
PLACE OF BIRTH:	Yueyang, China
YEAR OF BIRTH:	1993

EDUCATION:

2011 - 2015	Wuhan University of Technology, Wuhan, China Bachelor of Communication Engineering
2015 - 2018	University of Windsor, Windsor, Ontario, Canada Master of Applied Science, Electrical and Computer Engineering