Rochester Institute of Technology

RIT Scholar Works

Theses

7-2018

Exploring the Application of Homomorphic Encryption for a Cross Domain Solution

Cody W. Tinker
cwt9976@rit.edu

Follow this and additional works at: https://scholarworks.rit.edu/theses

Recommended Citation

Tinker, Cody W., "Exploring the Application of Homomorphic Encryption for a Cross Domain Solution" (2018). Thesis. Rochester Institute of Technology. Accessed from

This Thesis is brought to you for free and open access by RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.

Exploring the Application of Homomorphic Encryption for a Cross Domain Solution

Cody W. Tinker

July 2018

A Thesis Submitted in Partial Fulfillment of the Requirements for the
Degree of Master of Science
in Computer Engineering

R·I·T | Kate Gleason
College of ENGINEERING

Department of Computer Engineering

Exploring the Application of Homomorphic Encryption for a Cross Domain Solution

Cody W. Tinker

Committee Approval:

Dr. Marcin Łukowiak *Advisor* Date
RIT, Department of Computer Engineering

Dr. Michael Kurdziel Date
Harris Corperation

Dr. Andres Kwasinski Date
RIT, Department of Computer Engineering

Dr. Stanisław Radziszowski Date
RIT, Department of Computer Science

Acknowledgments

There are a number of people I would like to acknowledge and thank for their influence and support in my pursuit to obtain my Master's degree.

I would like to thank my family. My Aunt Pegge and late Uncle Bill, who have always supported my passions and pushing me to succeed. My siblings, Sean, Travis, Conner, and Kaleigh, who I will always share a special sibling bond with.

I would like to thank all my friends. Saaim Valiani, for being someone who I can always place my trust in and feel comfortable to share anything. Luke Boudreau, for being someone who shares similar passions and for introducing me to new hobbies. And to all my friends, who always supported me when school or thesis work became stressful and daunting.

I would like to thank all my colleagues I worked with in the Applied Cryptography and Information Security (ACIS) lab: Michael Foster, Daniel Stafford, Prathibha Rama, Kevin Millar, Stephanie Soldavani, and Andrew Ramsey.

I would like to thank my advisors and committee members. Dr. Kurdziel, for always brightening the mood with your contagious positivity. Dr. Radziszowski, for sharing your passion for mathematics and cryptography with me. Dr. Kwasinski, for providing your knowledge of network systems. And thank you, Dr. Łukowiak, for taking under your guidance and introducing me to cryptography and igniting my passion for the field.

Abstract

A cross domain solution is a means of information assurance that provides the ability to access or transfer digital data between varying security domains. Most acceptable cross domain solutions focus mainly on risk management policies that rely on using protected or trusted parties to handle the information in order to solve this problem; thus, a cross domain solution that is able to function in the presence of untrusted parties is an open problem.

Homomorphic encryption is a type of encryption that allows its party members to operate and evaluate encrypted data without the need to decrypt it. Practical homomorphic encryption is an emerging technology that may propose a solution to the unsolved problem of cross domain routing without leaking information as well as many other unique scenarios. However, despite much advancement in research, current homomorphic schemes still challenge to achieve high performance. Thus, the plausibility of its implementation relies on the requirements of the tailored application.

We apply the concepts of homomorphic encryption to explore a new solution in the context of a cross domain problem. We built a practical software case study application using the YASHE fully homomorphic scheme around the specific challenge of evaluating the gateway bypass condition on encrypted data. Next, we assess the plausibility of such an application through memory and performance profiling in order to find an optimal parameter selection that ensures proper homomorphic evaluation. The correctness of the application was assured for a 64-bit security parameter selection of YASHE resulting in high latency performance. However, literature has shown that the high latency performance can be heavily mitigated through use of hardware accelerators. Other configurations that include reducing number of SIMON rounds or avoiding the homomorphic SIMON evaluation completely were explored that show more promising performance results but either at the cost of security or network bandwidth.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem and Motivation

The purpose of cryptography is to protect data in the presence of an adversary with concerns regarding the following: authentication, confidentiality, integrity, and non-repudiation. A growing aspect in industry is to offload large data storage and expensive computation to third party services. Although traditional cryptographic systems are good at keeping data confidential and intact during transfer, the security then falls onto the third party service provider once the data is decrypted for operational use. As soon as the data is decrypted, the confidentiality aspect of the data is lost as the data is now available in plain sight to the servicer.

The cross domain problem is concerned with transferring information from different security domains while keeping the information secure. An application is said to be a cross domain solution (CDS) if it solves this problem. Most current solutions are based on a system of risk management that usually relies on protected party members. However, there still exists an unsolved problem of how to create an autonomous system that routes highly classified data through a lower or untrusted classified network without revealing any information about the data to the intermediate parties. The reason this problem is still open is because of a lack of solutions that allow an untrusted or semi-trusted party to operate on the encrypted data to determine its

routing endpoint without decrypting the data first.

Homomorphic Encryption (HE) schemes pose a potential solution to the cross domain problem. HE is a type of encryption that allows a party to perform computations on encrypted data without the need to decrypt it. These computations result in ciphertext that matches a ciphertext as if the evaluation result with the original plaintexts was encrypted. The concept of a homomorphic encryption scheme has been cited for the first time by Rivest, Adleman, and Dertouzous in 1978 [3]; however, the first construction of a plausible fully homomorphic encryption system was not proposed until 2009 in Craig Gentry's PhD dissertation [4]. Although implications of such a scheme that is able to perform arbitrary operations on data is huge, the current schemes have glaring limitations, primarily performance, that remain an open problem.

1.2 Homomorphic Encryption

Homomorphic Encryption is a type of encryption that allows a party to compute operations on encrypted data without the need to know the underlying plaintext data. Homomorphism is an algebraic term that comes from the greek words *homos*, meaning "same", and *morphe*, meaning "form" [5]. It is a term that is used to define a map between two algebraic structures that preserves the operations of the structures. In other words, given two algebraic structures with their own respective operations, both denoted by $(A, *)$ and $(B, \circ)$, and a map f, denoted $f : A \mapsto B$, the map f is said to be homomorphic if the following holds:

$$f(x * y) = f(x) \circ f(y); x, y \in A$$

It follows to say that an encryption scheme that exhibits homomorphism is one whose mapping function, namely the encryption function, preserves the operations between the plaintext space and ciphertext space. In other words, operations on ciphertexts

will result in a similar operation being performed on the underlying plaintext. The homomorphism property, as it applies to encryption, is a powerful feature as it allows untrusted parties to operate on data without the need to decrypt it and compromise the privacy of the users. This feature is highly sought after for applications where most of the computation is handed off to third parties for large scale operations but it is desired to keep the information secret from such party members. For example, the applications of handling large medical records or sensitive data handling in the cloud computing environment could benefit from the privacy feature of homomorphic encryption.

1.3 This Work

The objective of this thesis is to explore and assess the viability of using homomorphic encryption to create a solution to the cross domain problem. The results shed light on the practicality of using a fully homomorphic encryption scheme as well as a potential solution to the cross domain problem. The Yet Another Somewhat Homomorphic Encryption Scheme (YASHE) [6] is used as an encryption scheme basis for building a proof-of-concept application that determines routing endpoints for several pieces of encrypted data based on a hierarchy of data attributes. The application was profiled and analyzed to evaluate any potential problems that come with using homomorphic encryption as a solution basis, as well as revealed methods to improve upon these issues.

Despite the advancements in research related to current Homomorphic Encryption schemes, there still exists several challenges that prevent them from being practically implemented in a useful application. The largest one that is still being addressed by research is the computational cost of all the homomorphic schemes. Due to the complex mathematical structures utilized by current Homomorphic Encryption schemes and the selection of parameters that guarantee evaluation correctness and security,

the computational cost required to satisfy a system that holds both confidentiality and integrity of the data handled is expensive. In particular, the ciphertext expansion of the evaluation data is quite large and also exhibits a large execution time to completely evaluate.

Work first began with investigating the Brakerski-Gentry-Vaikuntanathan (BGV), Brakerski/Fan-Verauteren (B/FV), and YASHE FHE schemes and related mathematical structures to ensure familiarity with the schemes. In particular, the YASHE scheme was chosen for implementation after being heavily investigated. YASHE was chosen over the other two schemes for its simplicity and its performance benefits for small scale applications. Familiarity was reinforced with toy examples to demonstrate proof-of-concept and correctness of the scheme. A design of the application was constructed that meets the needs of a cross domain solution. A proof-of-concept application of the design was implemented using the YASHE scheme using the open source work from [7] as a base. The application was then subjected to memory and performance profiling using different scheme parameters to produce an assessment of the practicality and correctness of the application.

Chapter 2

Background

2.1 Cross Domain Solutions

A cross domain solution (CDS) is a form of controlled interface that provides the ability to access, either manually or automatically, or transfer information between varying security domains [8]. It is a means of information assurance that aims to achieve data confidentiality, integrity, and availability through a network. The scenario, as described in Fig. 2.1, generally follows that a user would like to provide access or transfer varying degrees of classified data to endpoint networks. This may be in the form of an encrypted payload with an encrypted metadata tag. The endpoint networks should not obtain data whose classification does not match their own. The difficult problem is that no party on the network except those whose classification domain match the data should be able to recognize the type of data or how it is would be routed. In other words, the data must be kept completely secure until it reaches the proper endpoint. The National Institute of Standards and Technology (NIST) provides a large catalog of security frameworks which includes a number of non-mutually exclusive scenarios for cross domain solutions, primarily under the information flow enforcement section, from which an organization can select to fulfill their requirements [9]; however, the methods and policies used are primarily determined by the implementing organization. Most CDS-related definitions have a focus

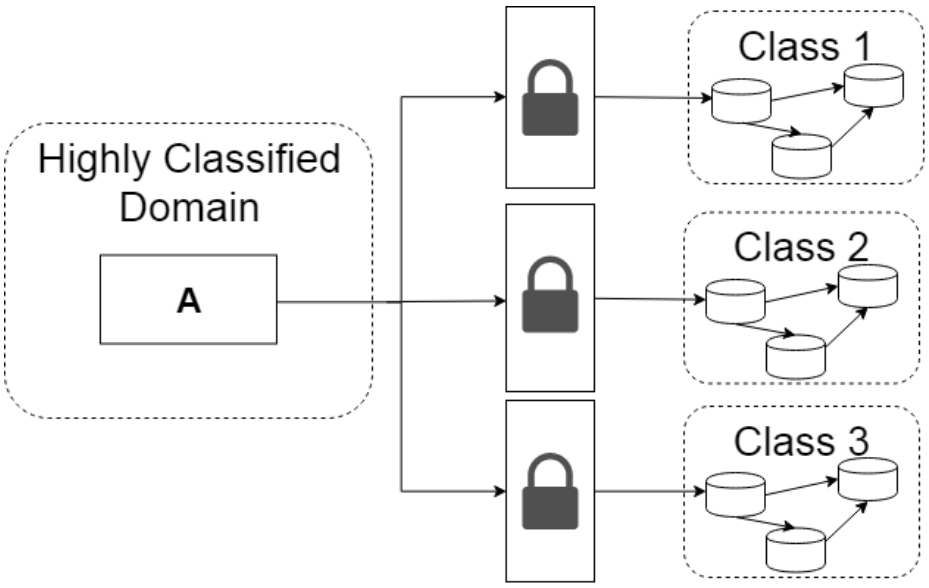

Figure 2.1: The Cross Domain Network Problem

on security policy and risk management using protected domains with authorized human personal to manually evaluate and control the flow of information. Therefore, a system that is able to deploy a scheme to automatically use information while maintaining security inside an untrusted domain is an open problem. The use of homomorphic encryption as a means to achieve this remains a mostly unexplored territory and proposes a potentially interesting solution.

2.2 A Basic Scheme

A basic scheme is presented to demonstrate the homomorphism properties as it applies to an encryption scheme. The scheme is heavily borrowed from Van Dijk's FHE scheme over integers [10]. The encryption methods demonstrate how a message is transformed into a ciphertext and the decryption method demonstrates how the message can be retrieved. The addition and multiplication operations demonstrate the homomorphism of the scheme.

2.2.1 Parameters

The scheme only requires a prime value p. This value is the secret key. The scheme has a message space $m \in \{0, 1\}$.

2.2.2 Encryption

Generate two values, e and q, randomly from some distribution such as a uniform or normal distribution. The message m is encrypted to obtain ciphertext c by:

$$c = E(m) = m + 2e + pq \tag{2.1}$$

Many homomorphic encryption schemes follow a similar format for encryption. The idea is that a message can be encrypted by masking it with a product of two element, in this case p and q, plus a small "noise" term, in this case $2e$.

2.2.3 Decryption

The ciphertext c is decrypted to obtain the message m by:

$$m = D(m) = (c \bmod p) \bmod 2 \tag{2.2}$$

The proof of correctness is as follows:

$$m = (c \bmod p) \bmod 2$$
$$m = (m + 2e + pq \bmod p) \bmod 2$$
$$m = (m + 2e) \bmod 2$$
$$m = m \ \square$$

It should be noted that the ciphertext only decrypts correctly if $(m + 2e) < p$. This can be seen by considering both the cases when $m + 2e = p$ and when $m + 2e > p$.

When $m + 2e = p$, it is easy to deduce that the message is lost after the modulo p operation as

$$(m + 2e) \bmod p \equiv p \bmod p \equiv 0$$

and is only a true results if m is zero which does not make for a complete case. When $m+2e > p$, then the parity of the message can be altered after the modulo p operation which will result in the incorrect message after the modulo 2 operation.

2.2.4 Addition

Two ciphertexts, c_1 and c_2, that encrypt two messages, m_1 and m_2, can be added to obtain a new ciphertext, c_3. Decrypting c_3 results in a plaintext whose value equals the evaluation of $m_1 + m_2$. The proof of homomorphism of the scheme over addition is as follows:

$$c_3 = c_1 + c_2$$
$$c_3 = m_1 + 2e_1 + pq_1 + m_2 + 2e_2 + pq_2$$
$$c_3 = (m_1 + m_2) + 2(e_1 + e_2) + p(q_1 + q_2)$$
$$c_3 = (m_1 + m_2) + 2e_3 + pq_3$$

It can be seen that the resulting ciphertext retains an encryption structure of the sum of messages m_1 and m_2.

2.2.5 Multiplication

Two ciphertexts, c_1 and c_2, that encrypt two messages, m_1 and m_2, can be multiplied to obtain a new ciphertext, c_3. Decrypting c_3 results in a plaintext whose value equals the evaluation of $m_1 m_2$. The proof of homomorphism of the scheme over multiplication is as follows:

$$c_3 = c_1 c_2$$

$$c_3 = (m_1 + 2e_1 + pq_1)(m_2 + 2e_2 + pq_2)$$
$$c_3 = (m_1 m_2) + 2(m_1 r_2 + m_2 r_1 + 2e_1 e_2) + kp$$

Note that the value k is simply the sum of product terms with the value p from the multiplication of the two ciphertexts. It can be seen that the resulting ciphertext retains an encryption structure of the product of messages m_1 and m_2.

2.2.6 Summary

It can be seen that the scheme presented is indeed homomorphic. It should be noted that in order for the scheme is defined to only operate on binary data. This ensures that a user can evaluate any abstract operation by constructing a binary circuit using the homomorphic operations. A binary addition operation results in an operation that matches a binary XOR-gate and a binary multiplication operation results in an operation that matches a binary AND-gate. A construction using these gates can perform any operation as these two binary operations form a functionally complete set. Other FHE schemes follow similar constructions.

2.3 Types of Homomorphic Encryption

Homomorphic encryption refers to the type of encryption that allows a user to operate on encrypted data; however, there are actually different HE schemes that fall within the definition of three different types: Partially Homomorphic, Somewhat Homomorphic, and Fully Homomorphic. The main distinction lies with what operations, primarily among addition and multiplication, a user can perform with the data and how the operations are restricted by the scheme.

2.3.1 Partially Homomorphic Encryption

A Partially Homomorphic Encryption (PHE) scheme is one that allows an unbounded operation on encrypted data but the type of operation allowed is limited to that

single operation. Although PHE schemes can be useful, the applications of which they are applicable is limited due to their limited operation set. There exists a couple of applications where such schemes are used including e-voting [11] and private information retrieval (PIR) [12].

2.3.2 Somewhat Homomorphic Encryption

A Somewhat Homomorphic Encryption (SWHE) scheme is one that allows a set of operations, normally both addition and multiplication, but with a bound on either one or both operations. Normally, these schemes allow one operation, usually addition, to be performed an unlimited amount of times while the other operation, usually multiplication, can be performed a limited number of times. SWHE schemes are applicable to applications where the complexity and depth of the functions involved are known beforehand but fall short for applications that desire arbitrary computation on data. In addition, data can only be operated on to a certain extent before the decrypt function will stop evaluating the ciphertext correctly, limiting the types of applications that can be implemented using such a scheme.

2.3.3 Fully Homomorphic Encryption

A Fully Homomorphic Encryption (FHE) scheme is one that allows an arbitrary set of operations to be evaluated on encrypted data an unlimited number of times. Despite the usefulness of the prior two types of schemes, this type of scheme is the most promising as it offers nearly no limitations on the type of computation one can execute with encrypted data. However, despite much research effort, it has proved difficult to construct a scheme that achieves practical full and unbounded homomorphism. No such schemes were devised until Craig Gentry's breakthrough in 2009 [4]. Since then, several FHE schemes have been published and much improvement has been seen in their development, but there are still several challenges that need to be addressed

before these schemes will realistically see use in industry. It should also be noted that a *leveled* fully homomorphic encryption scheme is one that can evaluate any circuit of multiplicative depth L as an auxiliary input. In this sense, it is somewhat analogous to say that such a scheme sits somewhere between SWHE and FHE, but for all intents and purposes, it is grouped under FHE as all circuits can be fully evaluated if the multiplicative depth L of the evaluation circuits are known beforehand and parameters are chosen to accommodate for such evaluation.

2.4 Gentry's Breakthrough

In 2009, Craig Gentry presented in his PhD dissertation [4] the first achievable FHE scheme. This was a huge milestone for the long term problem of finding a FHE scheme; arguably, the more valuable information that came from his dissertation was the general blueprint that he outlines for achieving FHE schemes that has paved the way for researchers. Before Gentry, research focused mainly on maximizing the circuit complexity that an HE scheme can evaluate. Gentry shows that one can obtain a FHE scheme using a SWHE scheme as a basis using two techniques he calls "squashing" and "bootstrapping". Gentry's ingenious, yet conceptually simple, idea is if a SWHE scheme is bootstrappable, a ciphertext can be "refreshed" or "re-encrypted" with an elaborate scheme of encrypting and decrypting a ciphertext *homomorphically* to produce a clean ciphertext (i.e. one with little noise). Therefore, Gentry's approach is to instead focus on minimizing the decryption circuit of an HE scheme in order to create a *bootstrappable* scheme.

2.4.1 Squashing

A SWHE scheme is said to be *bootstrappable* if it is able to evaluate its own decryption circuit. However, as it turns out, a decryption circuit usually involves a circuit of large depth. The squashing technique is a method that reduces the decryption circuit

depth. The squashing technique involves finding a set whose sum of elements equals the inverse of the secret key and then multiplying a ciphertext by each of the set elements. This reduces the decryption circuit to one with depth small enough for the SWHE scheme to evaluate.

2.4.2 Bootstrapping

The bootstrapping technique is a method to create a "fresh" ciphertext from a noisy ciphertext that will decrypt to the same plaintext. The bootstrapping method assumes there are two sets of public and secret key pairs and a ciphertext that is encrypted with the first key pair. First, squashing is applied to a ciphertext to make it bootstrappable. The technique basically involves decrypting the ciphertext homomorphically using an encryption of the first secret key and then applying encryption using the second public key. The resulting ciphertext is a ciphertext with "fresh" noise but encrypted under a new key pair. This technique allows a SWHE scheme to act as a FHE scheme by refreshing a ciphertext after any operations have been evaluated. However, it should be noted that this particular method is computationally expensive which is a major drawback to the implementation of a FHE scheme.

Chapter 3

Mathematical Constructs and Hard Problems

3.1 Ring Theory

A ring is one of the fundamental algebraic structures of abstract algebra. It consists of a set of elements, often denoted R, and is equipped with two binary operations, often denoted $+$ and $\cdot$, that satisfy the ring axioms [13]. The ring axioms are as follows:

1. R is an abelian group under addition:
   - $(a + b) + c = a + (b + c)$ for all $a, b, c \in R$.
   - $a + b = b + a$ for all $a, b \in R$.
   - There exists an additive identity element $0 \in R$ such that $a + 0 = a$ for for all $a \in R$.
   - There exists an additive inverse element $-a \in R$ such that $a + -a = 0$ for all $a \in R$.
2. R is a monoid under multiplication:
   - $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ for all $a, b, c \in R$.
   - There exists a multiplicative identity element $1 \in R$ such that $a \cdot 1 = 1 \cdot a = a$ for for all $a \in R$.

3. R is distributive under multiplication:

   - $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ for all $a, b, c \in R$.
   - $(b + c) \cdot a = (b \cdot a) + (c \cdot a)$ for all $a, b, c \in R$.

The primary type of rings of interest are generalized quotient ring of polynomial rings. A polynomial ring is that is formed from the set of polynomials with one or more indeterminates with coefficients of elements from another ring or field. A polynomial ring, represented as $K[x]$, over the indeterminate x with coefficient in the ring or field K, has the form of

$$\sum_{i=0}^{\infty} a_i x^i$$

where the coefficients a_i are elements of K. The quotient ring, $K[x]/(f(x))$, where $f(x) \in K[x]$, is set of elements in $K[x]$ mod $f(x)$. A d-th cyclotomic polynomial is the unique irreducible polynomial with integer coefficients which divides $x^d - 1$ but does not divide $x^k - 1$ for all values $k < d$ [14, 13]. If $f(x)$ is the d-th cyclotomic polynomial, $\phi_d(x)$, then the ring can be uniquely represented by all the polynomials in $K[x]$ with degree less than that of $n = \varphi(d)$, where φ is Euler's totient function [14]. Any addition and multiplication operations involving two elements from the ring follow the same form of addition and multiplication of polynomials but the result is reduced modulo $\phi_d(x)$.

As such, polynomial ring elements as described before can be expressed as polynomials with normal addition and multiplication operations. For example, let $\phi_d(x) = x^4 - 1$ and elements $a = x^3 + x + 1$, and $b = x^2 + x + 1$ where $x, y \in \mathbb{Z}/\phi_d(x)$. The addition of x and y follow polynomial component wise addition:

$$a + b = (x^3 + x + 1) + (x^2 + x + 1) = x^3 + x^2 + 2x + 2.$$

The multiplication of x and y follow polynomial component wise distributive multiplication where the result is reduced modulo $\phi_d(x)$:

$$a \cdot b = ((x^3 + x + 1) \cdot (x^2 + x + 1)) \bmod x^4 - 1$$
$$a \cdot b = (x^5 + x^4 + 2x^3 + 2x^2 + 2x + 1) \bmod x^4 - 1$$
$$a \cdot b = 2x^3 + 2x^2 + 3x + 2.$$

Note that, traditionally, modular reduction can be efficiently performed using the Extended Euclidean method [15].

3.2 Lattices

The algebraic construct of the lattice has had an increasing presence in cryptography primarily for hardness problems that rely on its underlying structure that is shown to be resistant to currently known quantum computer algorithms [1]. Gentry uses lattices, albeit primarily to take advantage of the shallow depth of the decryption circuit of lattice-based cryptography schemes, as part of his strategy for formulating a FHE scheme.

Let $\mathbb{R}^m$ be an m-dimensional vector of real numbers. A *lattice* in $\mathbb{R}^m$ is the set

$$L = \mathcal{L}(b_1, ..., b_n) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i : x_i \in \mathbb{Z} \right\}$$

of all combinations of n linearly independent vectors $\mathbf{b}_1, ..., \mathbf{b}_n$, where $\mathbf{b}_i \in \mathbb{R}^n$ [1, 14]. The integer n is called the rank of the lattice and the integer m is called the dimension of the lattice. It is common to work with lattices that are full-rank, or in other words, where $m = n$. The sequence of $\mathbf{b}_i$ is called the basis of the lattice and is usually represented as matrix

$$\mathbf{B} = [\mathbf{b}_1, ..., \mathbf{b}_n] \in \mathbb{R}^{mxn}.$$

with the basis vectors as columns. Any given lattice has an infinite number of bases. For any lattice vector $\mathbf{z}$, the vector $\mathbf{x} = (x_1, ..., x_n)$ is called the coefficient vector where $\mathbf{z} = \mathbf{Bx}$. For a given lattice vector $\mathbf{z}$, the ℓ_p norm of the vector, denoted $||\mathbf{z}||_p$, is a strictly positive value that follows

$$||\mathbf{z}||_p = \Big(\sum_{i=1}^{n} |x_i|^p)^{1/p} \Big).$$

In other words, $||\mathbf{z}||_p$ defines the length or size of the vector $\mathbf{z}$.

Geometrically speaking, a lattice can be represented as a set of infinite points of an n-dimensional grid. For example, a 2-dimensional lattice can be represented as in Fig. 3.1 [1]. The the basis of lattice in Fig. 3.1 is

$$\mathbf{B} = \begin{bmatrix} 1 & 1 \\ 2 & -1 \end{bmatrix}$$

where any point is the result of a linear combination of integer coefficients of the basis vectors

$$\mathbf{b}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \mathbf{b}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

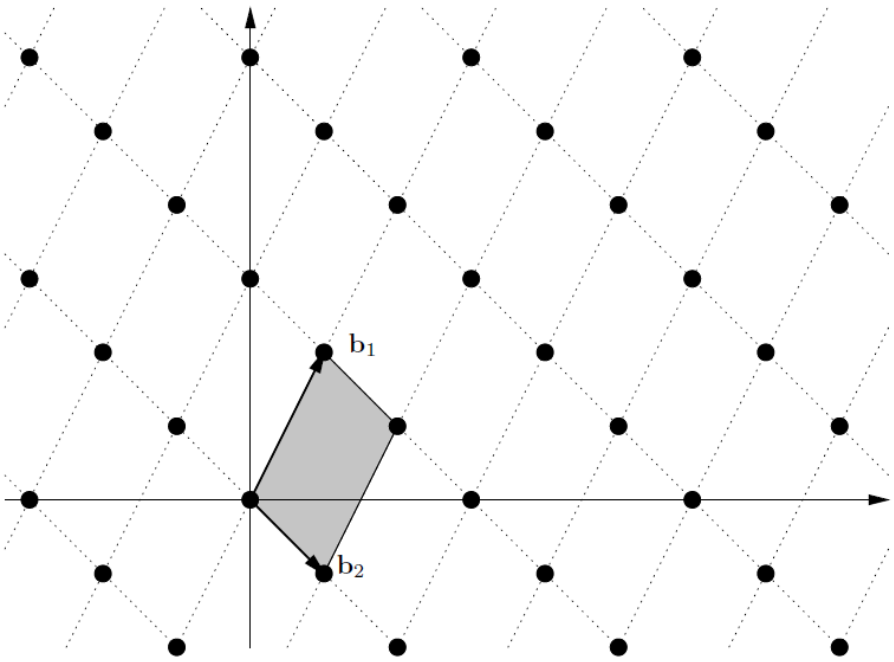

Figure 3.1: Example of a 2-dimensional Lattice [1]

3.3 Shortest Vector Problem

The Shortest Vector Problem (SVP) is one of the most well-known hard problems on lattices. Let $\lambda(L)$ be the length of the shortest vector in the lattice L with dimension n defined by basis matrix $\mathbf{B}$. The SVP is formally defined as

$$\text{minimize}_{\mathbf{x}\in\mathbb{Z}^n,\mathbf{x}\neq 0}||\mathbf{Bx}||_p$$

and return $\mathbf{x}$ [16]. The problem essentially asks to find the shortest vector in L with length $\lambda(L)$.

An approximate version of the problem, denoted $\text{SVP}_\gamma(n)$, asks to find a non-zero vector whose length is at most $\gamma(n) \cdot \lambda(L)$. The approximation version of the problem essentially asks to find a vector who is relatively short.

3.4 Closest Vector Problem

The Closest Vector Problem (CVP) is another hard problem on lattices. Let L be a lattice with dimension n defined by basis matrix $\mathbf{B}$ and $\mathbf{t} \in \mathbb{R}^m$ be vector that does not necessarily lie on the lattice. The CVP is formally defined as

$$\text{minimize}_{\mathbf{x}\in\mathbb{Z}^n}||\mathbf{Bx} - \mathbf{t}||_p$$

and return $\mathbf{x}$ [16]. The problem essentially asks to find a vector in L that lies closest to the given vector $\mathbf{t}$.

3.5 Learning with Errors

The Learning With Errors (LWE) problem is a hard problem that was first proposed by Oded Regev [17]. The problem is defined formally as follows. Fix a size parameter $n \geq 1$, a modulus $q \geq 2$, and a distribution χ on $\mathbb{Z}_q$. Set a secret vector $\mathbf{s} \in \mathbb{Z}_q^n$. Sample an arbitrary length sequence of vectors $\mathbf{a}_i \in \mathbb{Z}_q^n$ uniformly at random and

$e_i \in \mathbb{Z}_q$ according to χ. The LWE problem claims that it is difficult to retrieve $\mathbf{s}$ with high probability provided a sequence of tuples in the form of $(\mathbf{a}_i, \langle \mathbf{a}_i \mathbf{s} \rangle + e_i)$ [17, 18]. The problem has been shown to be reduced to worst-case lattice problems [17, 18, 19].

For example, say for $n = 3$ and $q = 17$, let

$$12s_1 + 3s_2 + 7s_3 \approx 11 \bmod 17$$
$$3s_1 + 1s_2 + 3s_3 \approx 1 \bmod 17$$
$$4s_1 + 8s_2 + 9s_3 \approx 6 \bmod 17$$

be the linear combinations obtained from the calculating the cross products of vectors $\mathbf{a}_i$ and $\mathbf{s}$ where

$$\mathbf{a}_1 = \begin{bmatrix} 12 \\ 3 \\ 4 \end{bmatrix}, \mathbf{a}_2 = \begin{bmatrix} 3 \\ 1 \\ 8 \end{bmatrix}, \mathbf{a}_3 = \begin{bmatrix} 7 \\ 3 \\ 9 \end{bmatrix} \mathbf{s} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix}$$

and adding a small noise term sampled uniformly from -1 and 1 to each result. LWE claims that it is difficult to find vector $\mathbf{s}$. The solution is

$$\mathbf{s} = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}.$$

3.6 Ring Learning with Errors

The Ring-Learning With Errors (R-LWE) problem is a special case of LWE that instead handles with algebraic ring structures [18]. The R-LWE problem is defined as follows. Fix a dimension parameter $n \geq 1$ where n is a power of 2 and a prime modulus $q \geq 2$ satisfying $q = 1 \bmod 2n$. R_q is a ring defined by $\mathbb{Z}_q[x]/\langle x^n + 1 \rangle$ and let χ be some distribution over R_q. Set a secret element $\mathbf{s} \in R_q$. Sample an arbitrary length sequence of elements $\mathbf{a}_i \in R_q$ uniformly at random and $\mathbf{e}_i \in R_q$ according to χ. The R-LWE problem claims that it is difficult to retrieve $\mathbf{s}$ with high probability

provided a sequence of tuples in the form of $(\mathbf{a}_i, \mathbf{a}_i \cdot \mathbf{s} + \mathbf{e}_i)$ [17, 18]. The R-LWE scheme offers increased efficiency over LWE in the context of cryptographic applications. For example, each $(a, b) \in R_q \times R_q$ sample from the R-LWE distribution can replace n samples $(\mathbf{a}, b) \in Z_q^n \times Z_q$ from the LWE distribution. This exhibits itself in reduced key sizes as well as faster computations. The RLWE problem is also reduced to hard lattice problems [19].

For example, say for $n = 4$ and $q = 17$, let

$$(3x^3 + 5x^2 + 7x + 10)\mathbf{s} \approx 13x^3 + 11x^2 + 7x + 11 \in R_q$$
$$(12x^3 + 14x^2 + 3x + 6)\mathbf{s} \approx 13x^3 + 12x^2 + 5x + 10 \in R_q$$
$$(2x^3 + 1x^2 + 6x + 6)\mathbf{s} \approx 12x^3 + 5x^2 + 14x + 13 \in R_q$$
$$(3x^3 + 9x^2 + 12x + 15)\mathbf{s} \approx 13x^3 + 11x^2 + 15x + 1 \in R_q$$

be the linear combinations obtained from the calculating the products of ring elements $\mathbf{a}_i$ and $\mathbf{s}$ where

$$\mathbf{a}_1 = 3x^3 + 5x^2 + 7x + 10 \in R_q$$
$$\mathbf{a}_2 = 12x^3 + 14x^2 + 3x + 6 \in R_q$$
$$\mathbf{a}_3 = 2x^3 + 1x^2 + 6x + 6 \in R_q$$
$$\mathbf{a}_4 = 3x^3 + 9x^2 + 12x + 15 \in R_q$$

and adding a small noise term where the coefficients are sampled uniformly from -1 and 1 to each result. R-LWE claims that it is difficult to find the ring element $\mathbf{s}$. The solution is

$$\mathbf{s} = 13x^3 + 2x^2 + 3x + 2 \in R_q.$$

Chapter 4

Fully Homomorphic Encryption Schemes

4.1 Preliminaries and Notation

Both the message space and ciphertext space of the schemes are defined with some relation to the ring R. The ring R is defined as $\mathbb{Z}[X]/(\phi_d(X))$ for an integer d. The set of elements in R can thus be uniquely represented as all polynomials of degree less than $n = \varphi(d)$ with integer coefficients. These elements may be represented as a vector with elements from the coefficients of the polynomial form. The coefficients of all ciphertext elements are often reduced by an integer q which is denoted by a map, $[\cdot]_q$, that reduces a given coefficient element by modulo q into the interval $(-q/2, q/2]$. This mapping extends to a ring element by applying the map to each individual coefficient of the polynomial. A second coefficient modulus $t < q$ determines the plaintext message space R/tR, or in other words, where messages are in R with coefficients modulo t. An integer w is used to represent integers in a radix-w system. Let $\ell_{w,q} = \lfloor \log_w(q) \rfloor + 1$. The notation R_q is used to denote the ring R whose coefficients are reduced by the modulus q. For a ring or field K, the notation K^n denotes an n-tuple list of elements from the the ring or field K. Elements that are a single element are denoted in lowercase notation such as z and vectors or tuples are denoted in lowercase boldface notation $\mathbf{z}$.

The schemes define two functions $D_{w,q}$ and $P_{w,q}$ which are called the word decom-

position and powers of functions, respectively. The notion of the word decomposition function follows that since an integer z in the interval $(-q/2, q/2]$ can be written uniquely as $\sum_{i=0}^{\ell_{w,q}-1} z_i w^i$ where z_i are integers within the bound $[0, w]$, then a ring element $x \in R$ can be written as $\sum_{i=0}^{\ell_{w,q}-1} x_i w^i$ where $x_i \in R$ with coefficients in $(-w/2, w/2]$. The word decomposition function is defined as

$$D_{w,q} : R \to R^{\ell_{w,q}}, x \mapsto ([x_0]_w, [x_1]_w, ..., [x_{\ell_{w,q}-1}]_w).$$

In other words, the word decomposition function maps a ring element to a vector of $\ell_{w,q}$ elements where each vector element is a ring element whose coefficients are the individual word decompositions of the individual original coefficients. The powers of function is defined as

$$P_{w,q} : R \to R^{\ell_{w,q}}, x \mapsto ([x]_q, [xw]_q ..., [xw^{\ell_{w,q}-1}]_q).$$

In other words, the powers of function maps a ring element to a vector of $\ell_{w,q}$ elements where each vector element is the original ring element scaled with an iteratively increasing exponential of the radix integer. The Powersof2 function is a special denotation of the powers of function for a radix value of 2.

4.2 Brakerski-Gentry-Vaikuntanathan Scheme

The Brakerski-Gentry-Vaikuntanathan (BGV) [20] presented a novel leveled fully homomorphic scheme with large improvements in performance at the cost of weaker security assumptions. The scheme still primarily relies on R-LWE problem as a security basis. The objective of the BGV scheme focuses on improving the per-gate computation of the homomorphic operations. BGV builds off of the techniques introduced by the BV scheme such as their method of avoid the squashing method [21]. A core component of this construction, however, is its effective way for managing the noise of the ciphertext terms by improving the modulus switching technique from

BV [21]. BGV uses an iterative approach to the modulus switching technique so that the noise grows linearly instead of quadratically with homomorphic multiplication operations at the expense of gradually reducing the homomorphic capacity. With this technique, it is able to effectively avoid the bootstrapping method to create a leveled fully homomorphic scheme and evaluate an L-level circuit with $\tilde{O}(\lambda \cdot L^3)$ per-gate computation complexity. However, Brakerski et. al. do propose a follow-up scheme that uses bootstrapping, somewhat ironically, as an optimization to achieve $\tilde{O}(\lambda^2)$ per-gate computation complexity that is independent of the circuit depth. All ciphertexts are represented as a pair of ring elements.

The basic leveled BGV scheme based on the R-LWE assumption is stated [20]. Note that another version of the scheme that implements bootstrapping to construct a completely FHE scheme from BGV is provided by [20].

- ParamsGen(λ, L): Given the security parameter λ, fix a positive integer d that determines R and a distribution χ on R. For $j = L$ down to 0, generate a decreasing ladder of moduli q_i. Output (d, q_i, χ).

- KeyGen(d, q, χ): For $j = L$ down to 0, Sample $s'_i \leftarrow \chi$ and set $\mathbf{s}_i = (1, s'_i)$. Sample $a'_i \leftarrow R_{q_i}$ uniformly and an element $e_i \leftarrow \chi$ and set $b_i = a'_i s'_i + 2e_i$. Set $\mathbf{a}_i = (b_i, -a'_i)^T$. Set $s'_j = s_j \otimes s_j \in R_{q_j}^{\binom{2}{2}}$. Set $\mathbf{b}_i = \mathbf{a}_i +$ Powersof2(s_i) (Add Powersof2(s_1) $\in R_{q_i}^{\lceil log_2(q_i) \rceil}$ to the $\mathbf{a}$'s first column). Set $\tau_{s'_{j+1} \to s_j} = \mathbf{b}_i$ except for when $j = L$. Set the secret key sk to a vector of $\mathbf{s}_i$ and the public key pk a vector of $\mathbf{a}_i$ and a vector of $\tau_{s'_{j+1} \to s_j}$ is a public parameter.

- Encrypt(pk, m): To encrypt a message $m \in R_2$, set $\mathbf{m} = (m, 0) \in R_2^2$. Sample $r \leftarrow \chi$ and $\mathbf{e} \leftarrow \chi^2$ output the ciphertext

$$c = \mathbf{m} + 2 \cdot \mathbf{e} + \mathbf{a}_L^T \cdot r \in R_{q_L}^2.$$

- Decrypt(sk, $\mathbf{c}$): Suppose the ciphertext c is encrypted under s_j. To decrypt c, compute

$$m = [[\langle \mathbf{c}, \mathbf{s}_j \rangle]_{q_j}]_2.$$

- SwitchKey($\tau_{s'_j} \to s_{j-1}$, c, q_j): Output the new ciphertext

$$c_1 = BitDecomp(c)^T \cdot \mathbf{b}_j.$$

- Refresh(c, $\tau_{s'_j} \to s_{j-1}$, q_j, q_{j-1}): Suppose the ciphertext is encrypted under s'_j. Do the following:

  1. Switch Keys: Set $c_1 \leftarrow$ SwitchKey($\tau_{s'_j} \to s_{j-1}$, c, q_j), a ciphertext under the key s_{j-1} for modulus q_j.
  2. Switch Moduli: Set $c_2 \leftarrow$ Scale(c_1, q_j, q_{j-1}, 2), a ciphertext under s_{j-1} for modulus q_{j-1}.

- Add(pk, c_1, c_2): Takes two ciphertexts encrypted under the same s_j (Use Refresh to make it so if they are not). Set $c_3 = c_1 + c_2$, a ciphertext encrypted under s'_j. Output the ciphertext

$$c_{add} = \text{Refresh}(c_3, \tau_{s'_j} \to s_{j-1}, q_j, q_{j-1}).$$

- Mult(pk, c_1, c_2): Takes two ciphertexts encrypted under the same s_j (Use Refresh to make it so if they are not). Set c_3 to the coefficient vector of $L^{long}_{c_1,c_2}(x \otimes x)$. Output the ciphertext

$$c_{mult} = \text{Refresh}(c_3, \tau_{s'_j \to s_{j-1}}, q_j, q_{j-1}).$$

4.3 Brakerski/Fan-Verauteren Scheme

The Brakerski/Fan-Verauteren (BFV) [22], also commonly denoted FV, is a port of the scale-invariant FHE B12 scheme by Brakerski [23] based on the LWE assumption into one based on the R-LWE assumption. FV also exhibits two methods for relinearization that are more optimal than the technique used in B12. In addition, despite

the avoidance of the technique in [23], FV introduces modulus switching from [21] as an optimization just as the BGV scheme for bootstrapping proposes. Like BGV and a few others, FV avoids the use for the squashing technique. The FV scheme offers a fully constructed leveled FHE scheme whose noise grows linearly with the depth of the evaluation circuits due to the scale-invariance techniques adopted from B12. All ciphertexts are represented as a pair of ring elements.

The leveled FV scheme based on the first variation of the relinearization technique is stated [22]. Note that another version of the scheme that implements bootstrapping to construct a completely FHE scheme from FV is provided by [22].

- ParamsGen(λ): Given the security parameter λ, fix a positive integer d that determines R, moduli q and t with $1 < t < q$, and distributions χ_{key}, χ_{err} on R. Choose an integer radix w. Output (d, q, t, χ_{key}, χ_{err}, w).

- KeyGen(d, q, t, χ_{key}, χ_{err}, w): Sample s $\leftarrow \chi_{key}$. Sample $a \leftarrow R_q$ uniformly at random, $e \leftarrow \chi_{err}$, and compute $b = [-(as + e)]_q$. Sample $\mathbf{a} \leftarrow R_q^{\ell_{w,q}}$ uniformly at random, $\mathbf{e} \leftarrow \chi_{err}^{\ell_{w,q}}$, and compute $\gamma' = [P_{w,q}(s^2) - (\mathbf{e} + \mathbf{a}s)]_q \in R^{\ell_{w,q}}$ and set $\gamma = (\gamma', \mathbf{a})$. Output (pk, sk, **evk**) = $((b, a), s, \gamma)$.

- Encrypt((b, a), m): The message space is R/tR. Sample $u \leftarrow \chi_{key}$, $e_1, e_2 \leftarrow \chi_{err}$, and output the ciphertext

$$c = ([\lfloor \tfrac{q}{t} \rfloor [m]_t + e_1 + bu]_q, [e_2 + au]_q) \in R^2.$$

- Decrypt(s, c): To decrypt a ciphertext $c = (c_0, c_1)$, compute

$$m = [\lfloor \tfrac{t}{q} [c_0 + c_1 s]_q \rceil]_t \in R.$$

- Relin($\tilde{c}_{mult}$, **evk**): Let $(\mathbf{b}, \mathbf{a}) =$ **evk** and $\tilde{c}_{mult} = (c_0, c_1, c_2)$. Output the ciphertext

$$\tilde{c}_{mult_2} = ([c_0 + \langle D_{w,q}(c_2), \mathbf{b} \rangle]_q, [c_1 + \langle D_{w,q}(c_2), \mathbf{a} \rangle]_q).$$

- Add(c_1, c_2): Let $c_1 = (c_{1,0}, c_{1,1})$ and $c_2 = (c_{2,0}, c_{2,1})$. Output the ciphertext

$$c_{add} = ([c_{1,0} + c_{2,0}]_q, [c_{1,1} + c_{2,1}]_q).$$

- Mult(c_1, c_2, evk): Output the ciphertext

$$c_{mult} = \text{Relin}(\tilde{c}_{mult}, \mathbf{evk}), \text{ where}$$
$$\tilde{c}_{mult} = (c_0, c_1, c_2) = ([\lfloor \tfrac{t}{q} c_{1,0} c_{2,0} \rceil]_q, [\lfloor \tfrac{t}{q} (c_{1,0} c_{2,1} + c_{1,1} c_{2,0}) \rceil]_q, [\lfloor \tfrac{t}{q} c_{1,1} c_{2,1} \rceil]_q).$$

4.4 YASHE

The Yet Another Somewhat Homomorphic Encryption (YASHE) scheme is a leveled fully homomorphic encryption scheme developed by Bos et. al [6] based on the Stehlé and Steinfeld [24] and López-Alt et al. [25] encryption schemes. The scheme's security is based on the Ring-Learning With Error (R-LWE) hardness assumption [19] that has been gaining traction in the cryptographic community due to its promising performance and security properties. The scheme offers a regularization technique to perform a key switching operation after multiplication to transform the ciphertext to one encrypted under the original key and as well as prevent the need to expand the ciphertext size after each operation. Its ciphertext is representable with a single ring element and the scheme offers promising performance benefits.

The more practical variant of the YASHE scheme is stated [6].

- ParamsGen(λ): Given the security parameter λ, fix a positive integer d that determines R, moduli q and t with $1 < t < q$, and distributions χ_{key}, χ_{err} on R. Choose an integer radix w. Output (d, q, t, χ_{key}, χ_{err}, w).

- KeyGen(d, q, t, χ_{key}, χ_{err}, w): Sample f', $g \leftarrow \chi_{key}$ and let $f = [tf' + 1]_q$. If f is not invertible modulo q, choose a new f'. Compute the inverse $f^{-1} \in R$ of f modulo q and set $h = [tgf^{-1}]_q$. Sample $\mathbf{e}, \mathbf{s} \leftarrow \chi_{err}^{\ell_{w,q}}$, compute $\gamma = [P_{w,q}(f) + \mathbf{e} + h\mathbf{s}]_q \in R^{\ell_{w,q}}$. Output (pk, sk, evk) = (h, f, γ).

- Encrypt(h, m): The message space is R/tR. Sample $s, e \leftarrow \chi_{err}$ and output the ciphertext

$$c = [\lfloor \tfrac{q}{t} \rfloor [m]_t + e + hs]_q \in R.$$

- Decrypt(f, c): To decrypt a ciphertext c, compute

$$m = [\lfloor \tfrac{t}{q} [fc]_q \rceil]_t \in R.$$

- KeySwitch($\tilde{c}_{mult}$, **evk**): Output the ciphertext $[\langle D_{w,q}(\tilde{c}_{mult}), \mathbf{evk} \rangle]_q$.

- Add(c_1, c_2): Output the ciphertext

$$c_{add} = [c_1 + c_2]_q.$$

- Mult(c_1, c_2, evk): Output the ciphertext

$$c_{mult} = \text{KeySwitch}(\tilde{c}_{mult}, \text{evk}), \text{ where } \tilde{c}_{mult} = [\lfloor \tfrac{t}{q} c_1 c_2 \rceil]_q.$$

4.5 YASHE Toy Example

To better demonstrate the data structures and homomorphic properties of the YASHE scheme, a toy example is provided. The example demonstrates the encryption, decryption, binary addition, and binary multiplication operations, the intermediate operations involved, and how the resulting data looks.

4.5.1 Preliminaries

For the purposes of keeping the example small and comprehensible, the security of the scheme is ignored and instead the parameters are chosen to keep the sizes of the ring elements small. The integer d is chosen to be 8 to set the modulus polynomial of the ring R to the d-th cyclotomic polynomial, $\phi_d(x)$, of $x^4 + 1$. This limits the

elements of the ring R to polynomials of degree $\varphi(8) = 4$. The modulus parameter q is chosen to be 601 which determines the coefficient bounds of the polynomials. The modulus parameter t is chosen to be 2 since no batching will be utilized for the example and only bits of data are handled. The integer radix value w is chosen to be 2 for simplicity. Define $\ell_{w,q} = 10$.

The distribution χ_{key} is chosen to be a uniform distribution of the values of -1, 0, and 1. The distribution χ_{err} is chosen to be a bounded normal distribution with bounds of -3 and 3. When the scheme asks calls to sample a polynomial from the one of the distributions, it is sufficient to think of the sampling as setting every coefficient of the polynomial to a random sample from the distribution described.

The ring operations for addition and multiplication follow the same rules for polynomial addition and multiplication but all results are implicitly reduced by $\phi_8(x)$.

All examples were computed and evaluated using an implemented Matlab script that performed all operations utilizing Matlab's polynomial evaluation functions.

4.5.2 Key Generation

Ring elements f' and g are sampled from χ_{key}:

$$f' = -x^2 + x$$
$$g = -x^3 + x^2 - x + 1$$

The ring element f is computed using f':

$$f = [tf' + 1]_q$$
$$f = [(2)(-x^2 + x) + 1]_{601}$$
$$f = -2x^2 + 2x + 1$$

The ring element f^{-1} is computed using an inverse method such as the Extended Euclidean algorithm [15]:

$$f^{-1} = 255x^3 - 148x^2 + 280x + 206$$

The ring element h is computed using g and f^{-1}:

$$h = [tgf^{-1}]_q$$
$$h = [(2)(-x^3 + x^2 - x + 1)(255x^3 - 148x^2 + 280x + 206)]_{601}$$
$$h = -248x^3 + 66x^2 - 57x - 25$$

Values e and s are sampled $\ell_{w,q}$ times from χ_{err}. The $\ell_{w,q}$-element vector of ring elements that composes γ is calculated from e, s, and h by calculating $[P_{w,q}(f)+e+hs]_q$ to generate an $\ell_{w,q}$ number of ring elements. Each iteration of the calculation is represented in Table 4.1.

The public key $pk = h$, the secret key $sk = f$, and the evaluation key $evk = \gamma$.

4.5.3 Encryption

Let plaintext messages, m_1 and m_2, be two binary values 0 and 1, respectively. Two ciphertexts, c_1 and c_2, can be computed for each bit message using the encryption function. The values for e and s are sampled from χ_{err} for each encryption. This function requires the knowledge of the public key:

$$c_1 = [\lfloor \tfrac{q}{t} \rfloor [m_1]_t + e_1 + hs_1]_q$$
$$c_1 = [(300)(0) + (-x^3 + 2x + 1) + (-248x^3 + 66x^2 - 57x - 25)(x^2 + x + 1)]_{601}$$
$$c_1 = -240x^3 - 16x^2 + 168x + 158$$
$$c_2 = [\lfloor \tfrac{q}{t} \rfloor [m_2]_t + e_2 + hs_2]_q$$
$$c_2 = [(300)(1) + (3x^3 - 1) + (-248x^3 + 66x^2 - 57x - 25)(-x^3 - 2x^2 + x)]_{601}$$
$$c_2 = 208x^3 - 255x^2 + 146x + 21$$

4.5.4 Decryption

The two ciphertexts, c_1 and c_2, can be decrypted using the decryption function. This function requires the knowledge of the secret key:

$$m_1 = [\lfloor \tfrac{t}{q} [fc_1]_q \rceil]_t$$

Table 4.1: Iterative Evaluation for a YASHE Evaluation Key

i	$P_{w,q}(f)$	e_i	s_i	γ_i
0	$-2x^2+2x+1$	$1x^3+0x^2+0x+1$	$1x^3+0x^2+2x+1$	$-140x^3+198x^2-118x-71$
1	$-4x^2+4x+2$	$-1x^3-1x^2-1x+0$	$0x^3-1x^2+1x+0$	$122x^3-37x^2-270x-285$
2	$-8x^2+8x+4$	$0x^3+0x^2+0x-1$	$2x^3+0x^2+1x-2$	$-89x^3+299x^2-35x-186$
3	$-16x^2+16x+8$	$1x^3-1x^2-3x+0$	$-1x^3+0x^2+2x-1$	$-195x^3+156x^2+86x-129$
4	$-32x^2+32x+16$	$2x^3+1x^2+2x-1$	$3x^3+0x^2+0x-1$	$175x^3+46x^2-107x+211$
5	$-64x^2+64x+32$	$0x^3-1x^2+2x+3$	$2x^3+1x^2+2x-3$	$168x^3+94x^2-298x+53$
6	$-128x^2+128x+64$	$1x^3+1x^2+2x-1$	$-1x^3+0x^2-3x+1$	$181x^3+138x^2+241x-162$
7	$-256x^2+256x+128$	$1x^3+0x^2+2x+1$	$0x^3-1x^2+1x+0$	$124x^3-288x^2-15x-158$
8	$89x^2-89x+256$	$0x^3-1x^2-1x+1$	$1x^3+2x^2+2x+0$	$-7x^3+172x^2+290x+77$
9	$178x^2-178x-89$	$-3x^3+1x^2-3x+0$	$0x^3-1x^2-1x+1$	$-260x^3-274x^2+140x-296$

$$m_1 = [\lfloor \tfrac{2}{601}[(-2x^2 + 2x + 1)(-240x^3 - 16x^2 + 168x + 158)]_{601} \rceil]_2$$

$$m_1 = [\lfloor 1.977x^3 + 0.013x^2 + 0.013x + 0.017 \rceil]_2$$

$$m_1 = [2x^3 + 0x^2 + 0x + 0 \rceil]_2$$

$$m_1 = 0$$

$$m_2 = [\lfloor \tfrac{t}{q}[fc_2]_q \rceil]_t$$

$$m_2 = [\lfloor \tfrac{2}{601}[(-2x^2 + 2x + 1)(208x^3 - 255x^2 + 146x + 21)]_{601} \rceil]_2$$

$$m_2 = [\lfloor .023x^3 + 1.983x^2 + 0.010x + 0.988 \rceil]_2$$

$$m_2 = [0x^3 + 2x^2 + 0x + 1 \rceil]_2$$

$$m_2 = 1$$

4.5.5 Homomorphic Addition

The two ciphertexts, c_1 and c_2, can be homomorphically added by performing a ring element addition with both ciphertexts:

$$c_3 = [c_1 + c_2]_q$$

$$c_3 = [(-240x^3 - 16x^2 + 168x + 158) + (208x^3 - 255x^2 + 146x + 21)]_{601}$$

$$c_3 = -32x^3 - 271x^2 - 287x + 179$$

Decrypting the result shows that the two underlying plaintext values were evaluated under binary addition:

$$m_3 = [\lfloor \tfrac{t}{q}[fc_3]_q \rceil]_t$$

$$m_3 = [\lfloor \tfrac{2}{601}[(-2x^2 + 2x + 1)(-32x^3 - 271x^2 - 287x + 179)]_{601} \rceil]_2$$

$$m_3 = [\lfloor 0x^3 + 1.997x^2 + 2.329x + 1.005 \rceil]_2$$

$$m_3 = [0x^3 + 2x^2 + 2x + 1 \rceil]_2$$

$$m_3 = 1 = m_1 + m_2$$

4.5.6 Homomorphic Multiplication

The two ciphertexts, c_1 and c_2, can be homomorphically multiplied by performing a multi-step process. First, an intermediate multiplication ciphertext, $\widetilde{c}_{mult}$, is com-

puted:

$$\widetilde{c}_{mult} = [\lfloor \tfrac{t}{q} c_1 c_2 \rceil]_q$$
$$\widetilde{c}_{mult} = [\lfloor \tfrac{2}{601}(-240x^3 - 16x^2 + 168x + 158)(208x^3 - 255x^2 + 146x + 21) \rceil]_{601}$$
$$\widetilde{c}_{mult} = -58x^3 + 113x^2 - 104x - 2$$

The ciphertext results in a homomorphic evaluation of the plaintext data under multiplication, however, due to the quadratic overhead of polynomial multiplication, it is technically encrypted under the square of the secret key. In fact, this data could be decrypted with such a key. However, since the data is intended to be reused for further evaluation, a key switching mechanism is used to align the ciphertext to the original secret key. First, the word decomposition of $\widetilde{c}_{mult}$ is computed using the binary bit decomposition of all the coefficients. The binary representation of each of the coefficients in $\widetilde{c}_{mult}$ is represented in Table 4.2. Note that the elements are expressed in their positive integer form by the modulus 601.

Table 4.2: Bit Decomposition of $\widetilde{c}_{mult}$ coefficients

	Bit	9	8	7	6	5	4	3	2	1	0
Coefficient	543	1	0	0	0	0	1	1	1	1	1
	113	0	0	0	1	1	1	0	0	0	1
	497	0	1	1	1	1	1	0	0	0	1
	599	1	0	0	1	0	1	0	1	1	1

The routine $D_{w,q}(\widetilde{c}_{mult})$ is used to generate $\ell_{w,q}$-element vector of ring elements generated from the bit compositions of each of the coefficients that make up the entire ring element. Each iteration of the calculation and the resulting ring elements that compose of the entire word decomposition of $\widetilde{c}_{mult}$ is represented in Table 4.3.

Table 4.3: Iterative Evaluation for $\widetilde{c}_{mult}$ Word Decomposition

i	$D_{w,q}(\widetilde{c}_{mult})$
0	$1x^3 + 1x^2 + 1x + 1$
1	$1x^3 + 0x^2 + 0x + 1$
2	$1x^3 + 0x^2 + 0x + 1$
3	$1x^3 + 1x^2 + 1x + 0$
4	$1x^3 + 1x^2 + 1x + 1$
5	$0x^3 + 1x^2 + 1x + 0$
6	$0x^3 + 1x^2 + 1x + 1$
7	$0x^3 + 0x^2 + 1x + 0$
8	$0x^3 + 0x^2 + 1x + 0$
9	$1x^3 + 1x^2 + 1x + 1$

The final ciphertext, c_3, is computed using the key switch routine. The key switch routine calls to compute the cross product between word decomposition of $\widetilde{c}_{mult}$ vector and the evaluation key vector. The results of each individual multiplication element and then the summation of the components is represented in Table 4.4. All operations are performed implicitly using the ring multiplication operation with a modulus of 601.

Table 4.4: Cross Product Evaluation for $\langle D_{w,q}(\tilde{c}_{mult}), \mathbf{evk} \rangle$

i	$D_{w,q}(\tilde{c}_{mult})_i \cdot \mathbf{evk}_i$
0	$-184x^3 + 96x^2 - 300x + 42$
1	$-163x^3 - 179x^2 - 233x - 15$
2	$-275x^3 - 213x^2 + 267x - 151$
3	$-129x^3 + 195x^2 - 156x - 86$
4	$-276x^3 - 25x^2 - 117x + 97$
5	$-204x^3 - 245x^2 - 115x - 262$
6	$257x^3 - 86x^2 - 129x - 205$
7	$288x^3 - 15x^2 - 158x - 124$
8	$172x^3 + 290x^2 + 77x + 7$
9	$45x^3 - 14x^2 - 187x + 165$
Sum	$157x^3 - 176x^2 + 151x + 69$

The resultant ciphertext, c_3, is therefore computed to be $157x^3+425x^2+151x+69$. Decrypting the result shows that the two underlying plaintext values were evaluated under binary multiplication:

$$m_3 = [\lfloor \tfrac{t}{q}[fc_3]_q \rceil]_t$$

$$m_3 = [\lfloor \tfrac{2}{601}[(-2x^2 + 2x + 1)(157x^3 + 425x^2 + 151x + 69)]_{601} \rceil]_2$$

$$m_3 = [\lfloor 0.346x^3 + 1.960x^2 + 0.007x + 0.0131 \rceil]_2$$

$$m_3 = [0x^3 + 2x^2 + 0x + 0 \rceil]_2$$

$$m_3 = 0 = m_1 * m_2$$

Chapter 5

Parameter Selection

There are a few concerns regarding the parameter selection of the FHE schemes. The parameters chosen for the implementation of a scheme have an effect on the several components of the scheme: security, ciphertext size, performance, circuit evaluation depth, and the plaintext space. The main difficulty with choosing the "best" parameters is that each of the components seem to be heavily intertwined such that one particular selection of parameters may have a positive effect on one component but an adverse effect on another. The main parameters that influence each of these components are the polynomial ring degree, denoted n, and the ciphertext coefficient modulus, denoted q. Other parameters that have an influence, but are not examined as heavily as the other two, are the secret and error distribution parameters, χ_{key} and χ_{err}.

5.1 Security

The analysis of the security R-LWE based schemes is not very straight forward due to the influence on the security from the mix of parameters. The security lies primarily in the masking of the plaintext data with the noise terms that are randomly sampled. In addition, the number of coefficients of each ring and the size of the coefficients, ie. the parameters n and q, have an effect on the analysis of finding the secret element. There has been a recent effort for standardization of FHE schemes and the

analysis of their security [26] in order to provide a recommendation of parameters. The authors made an security analysis for varying parameters by evaluating the bit security strength against three known attack algorithms: the unique shortest vector attack (uSVP) [27], the decoding attack [28, 29], and the dual attack [30]. These three attacks are based heavily off the usage of the BKZ-2.0 lattice reduction algorithm [31]. The authors based their analysis off of the analysis of Albrecht, Player, and Scott [29] who provide an open source and online estimator tool. In order to simplify their analysis, the error distribution is set constant to a Discrete Gaussian distribution with width $\gamma = 8/\sqrt{2\pi}$ and focusing on the effects of choosing a secret distribution from the uniform, error, and ternary (ie. uniformly from -1, 0, 1) distributions. For a desired bit security parameter, their analysis describes the required upper bound on coefficient size (in bits) of the coefficient modulus q for varying ring degrees n. We first begin focus on a security level of 128-bits for the interest of this study and then approach other security levels as needed. Although the authors provide a collection of results for the three separate secret distributions, it was found that the upper bound on the coefficient modulus q rarely varied more than a couple bits for a given degree n value. In the interest of this study, the ternary distribution will be the focus for the secret key distribution. The authors of YASHE provide a summary of similar parameter that ensure 80-bit security [6]. We utilized the same estimator tool by Albrecht et. al. [29] to obtain similar parameters that ensure 64-bit security. A summary of the results of the varying security analysis of the study is represented in Table 5.1. The upper bound of the coefficient size increases as the degree of the ring increases, so if any reason asks for it, a larger coefficient can be used as long as the ring degree is chosen appropriately. It should be noted, however, that the upper bound of the coefficient size decreases for higher levels of security [26].

Table 5.1: R-LWE Based FHE Parameters for Varying Security Levels

n	$\log_2(q)$	security
1024	56	64
	-	80
	29	128
	21	192
	16	256
2048	110	64
	79	80
	56	128
	39	192
	31	256
4096	218	64
	157	80
	111	128
	77	192
	60	256
8192	438	64
	312	80
	220	128
	154	192
	120	256
16384	885	64
	622	80
	440	128
	307	192
	239	256
32768	-	64
	1243	80
	880	128
	612	192
	478	256

5.2 Ciphertext Size

Ciphertext expansion, or the ratio between a plaintext and ciphertext space, is an important concern with FHE schemes as it tends to be large. The ciphertexts are represented with ring elements in the BGV, FV, and YASHE schemes. Therefore, it is easy to see that the size of the ciphertext would be directly related to the ring degree and coefficient sizes and no relation to the secret and error distributions chosen. For each ring element involved in representing a ciphertext, the minimal amount of space required to represent the ring, not accounting for overhead, would be the ring degree, n, times the coefficient size, $\log_2(q)$. Furthermore, for evaluation purposes, data is encrypted bit-wise, so the ciphertext expansion for an l-bit value would be $l \cdot n \cdot \log_2(q)$. A table of ring element sizes for a single bit encryption, ignoring the the technique of *batching* [32], with similar parameters listed in Table 5.1, is represented in Table 5.2. It is obvious to see that maintaining a small selection of parameters is important for minimizing the ciphertext expansion because it is large even for relatively small parameters.

5.3 Plaintext Space

There are two components of the plaintext space as it relates to the parameter n. For a plaintext modulus t, which is normally set to 2, the BGV, FV, and YASHE schemes define a plaintext space R_t as it relates to the ring R defined by ring degree parameter. Normally, a single bit defined by the modulus 2 would be encrypted at a time as this allows a boolean circuit to be homomorphically evaluated with the data. However, a user is not technically restricted to do such. A user can indeed use all n slots of the polynomial to encode a given message; however, this restricts their homomorphic operations to the finite field of $GF(2^t)$ under the irreducible of the ring, $\phi(X)$.

Table 5.2: Ciphertext Ring Element Sizes of Varying Parameters

n	$\log_2(q)$	Ring Element Size (kB)
1024	56	7
	29	3.625
	21	2.625
	16	2
2048	110	27.5
	79	19.75
	56	14
	39	9.75
	31	7.75
4096	218	109
	157	78.5
	111	55.5
	77	38.5
	60	30
8192	438	438
	312	312
	220	220
	154	154
	120	120
16384	885	1770
	622	1244
	440	880
	307	614
	239	478
32768	1243	4972
	880	3520
	612	2448
	478	1912

The more important aspect of the influence of n applies to the *batching* technique optimization. Batching effectively encrypts multiple plaintext blocks into a single encryption where follow up homomorphic circuits apply separately to each block but only requires the computation of a single execution [32]. The technique works through a crafty construction of changing the plaintext modulus t to a prime and then performing homomorphic operations on t-mod circuits. Through use of the Chinese Remainder Theorem [15], evaluating a circuit over the plaintext can be thought of operating on the individual factors of the modulus. The only restriction is that t

must be selected such that t is a prime and $t = 1 \bmod 2n$. A larger degree value n increases the number of batch slots that the plaintext can support. It should be noted, however, that such a technique induces slightly more noise into ciphertext [20], however, this is usually outweighed by increased throughput of the system.

5.4 Performance

The main concern with large parameters for n and q is their effect on the homomorphic operation execution time, primarily that of the multiplication operation. Since the ciphertext elements are synonymous with a polynomial representation, the execution time of performing such an operation will scale dramatically with n. The operation overhead of the naive approach to polynomial multiplication is at least quadratic $O(n^2)$ [33]. However, there do exist algorithms such as the Karatsuba [34] and Fast Fourier Transform (FFT) [33] with reduced complexities of $O(n^{1.584})$ and $O(n \log(n))$, respectively; however, despite the reduction in computation cost, it can be seen how large values of n can still have a significant impact on the expected complexity of these algorithms. It should also be noted that although these algorithms do not normally account for coefficient sizes whereas large coefficients are speculated to have similar adverse effects on performance. Considering that these operations will likely be executed many times, it is vital that these parameters be kept as small as possible.

5.5 Circuit Depth

The trend for parameter selection is to keep the parameters n and q as small as possible. The main culprit that prevents such small parameters from being used is the desired evaluation circuit depth. It should be noted that for YASHE, and for other schemes in general, that a ciphertext decrypts incorrectly if the norm of the noise term of the ciphertext exceeds half of q [6]. Due to the noise growth involved

in the homomorphic operations, the circuit depth L that a scheme can evaluate is limited by how fast the noise will grow with respect to q. Intuitively, if the error that is embedded in a ciphertext is sampled from a small distribution, the noise should grow slower than that of a ciphertext whose noise is sampled from a wide distribution. In addition, given an error distribution, the depth of a circuit that YASHE can evaluate increases with an increasing value for q which conflicts with the notions parameter selection for security and performance. The YASHE scheme defines that the bounds for L must satisfy

$$2(1+\epsilon_1)^{L-1}\delta^{2L}t^{2L-1}B_{key}^{L}((1+\epsilon_1)tV + L(tB_{key} + t^2 + \ell_{w,q}wB_{err})) < (\lfloor \tfrac{q}{t} \rfloor + [q]_t),$$

where $\epsilon_1 = 4(\delta t B_{key})^{-1}$, $\delta = n$, B_{err} is the bound of the error distribution, B_{key} is the bound of the key distribution, $l_{w,q}$ is the number of w-bits in q, and $V = ntB_{key}(2B_{err} + [t]_q/2)$ is the amount of inherent noise in a ciphertext [6]. Thus, theoretically, the expected guaranteed depth L that YASHE can evaluate can be calculated using the elaborate equation with assumptions based on the bounds of the error growth in a ciphertext and others parameters. A python script was written to determine the maximum value of the evaluation depth L that passes the inequality equation when $w = 2^{32}$, $B_{key} = 1$, $B_{err} = 48$, and $t = 2$. The guaranteed evaluation depth L that the YASHE scheme can evaluate with the parameters chosen for the proposed security levels is represented in Table 5.3. It should be noted, however, that the evaluation depth is slightly deeper as the authors of YASHE note that their bounds equation is very conservative with regards to the parameters [6] and thus should be regarded more as a guide than a rule. It should also be noted by the values presented that YASHE offers a relatively shallow multiplicative depth, especially for smaller parameters, which is a serious concern.

Table 5.3: Expected Multiplication Evaluation Depth of Varying Parameters

n	$\log_2(q)$	L
1024	56	1
	29	0
	21	0
	16	0
2048	110	3
	79	2
	56	1
	39	0
	31	0
4096	218	7
	157	5
	111	3
	77	2
	60	1
8192	438	14
	312	10
	220	7
	154	4
	120	3
16384	885	28
	622	19
	440	13
	307	9
	239	7
32768	1243	38
	880	26
	612	18
	478	14

5.6 Anticipated Issues

The primary concerns and focus of this thesis is on the performance and efficiency of FHE schemes and how it will affect the practicality of applying such schemes to a cross domain solution. These problems stem from the complexity of the mathematical structures required to utilize the hardness problems that the FHE schemes rely on. The main culprit for this is that the data needs to be encrypted bitwise in order to take advantage of the homomorphic evaluation properties. Each bit of a plaintext

value needs to be encrypted under the FHE scheme and results in a ciphertext for each individual bit. This also means that all operations on encrypted data must be evaluated using a circuit that operates bitwise with a combination of gates from a functionally complete set of logic gates. Generally, this set happens to include bitwise addition, which is analogous to a logic XOR-gate, and bitwise multiplication, which is analogous to a logic AND-gate, where each of these operations are performed under the homomorphism property. This requirement complicates the evaluation of even some of the most seemingly easy operations such as addition of two integers as such circuits are complicated compared to the scale of the individual gate operations.

As the parameter selection investigation shows, there exists a complicated tangling between the scheme parameters and their effects on security, performance, and circuit evaluation correctness. With this and the consideration described prior, it can be seen why fully homomorphic evaluation do result in large ciphertexts and poor performance. Performance optimization desires that parameters q and n be kept small to ensure homomorphic operations operate on simple structures. Circuit evaluation correctness requires that the ciphertext coefficient q parameter be large to ensure the scheme will allow enough homomorphic operations to ensure the utilized circuits can be evaluated. And security of the homomorphic schemes place upper bounds on parameter q for given parameter n which directly conflicts with the previous two considerations. So it was important that all parameters configurations were explored during experimentation in order to determine the best parameter selection for a given application.

The notion of the significance of memory and performance results produced by homomorphic encryption is supported by literature. For example, an AES-128 circuit was homomorphically evaluated with an implementation of the BGV scheme [20] in 2015 in order to gain an empirical assessment of these qualities. The AES circuit was chosen because of its natural benchmark features: AES is widely used, AES is

nontrivial, and AES has a regular structure. Both a non-bootstrapped variant and a bootstrap variant were implemented and compared for purposes of evaluating the affect that bootstrapping has on the performance. For the non-bootstrapped variant, 120 AES blocks were packed into a single ciphertext and were encrypted and decrypted homomorphically. For the bootstrapping variant, 180 AES blocks were packed into a single ciphertext and were encrypted and decrypted homomorphically. The program was executed on an Intel Core i5-3320M running at 2.6GHz with 4GB of RAM. The execution results of the program is shown in Table 5.4 [35].

Table 5.4: Homomorphic AES Performance

Test	m	$\phi(m)$	lvls	$\bar{Q}$	security	params/key-gen (s)	Encrypt (s)	Decrypt (s)	memory
no bootstrap	53261	46080	40	886	150-bit	26.45/73.03	245.1	394.3	3GB
boostrap	28679	23040	23	493	123-bit	148.2/37.2	1049.9	1630.5	3.7GB

The authors found that it took about four minutes to process an entire AES-128 operation and required roughly 3GB of RAM when implemented without bootstrapping. When they introduced the bootstrapping method, the operation time increased to eighteen minutes and required roughly 3.7GB of RAM. Many optimization techniques were utilized, such as SIMD batching techniques, that ultimately yielded an amortized process rate of two seconds per AES block without bootstrapping and six seconds per AES block with bootstrapping [35]. It is observable from such a proof-of-concept implementation that performance is a serious concern as it takes a lot of time and memory to process such a small amount of data. This improvement was only achieved through the optimization techniques devised between the time of implementations. It is noted by the authors that better results could probably be obtained using other methods such as utilizing parallelism techniques of which they did not take advantage [35].

Chapter 6

Case Study

6.1 Implemented Scheme

The introduction of using the R-LWE hardness assumption as a basis for the second generation schemes has severely cut down on the required memory resources to store keys and ciphertexts as well as sped up the homomorphic operations. The FHE scheme which was chosen for implementation was the YASHE scheme. The scheme is one of the most recently of the proposed "second" generation FHE schemes based on the R-LWE assumption and has some promising performance optimizations over the first generation schemes. It offers several features such as scale-invariance via a key switching mechanism and single ring element ciphertexts.

The scheme was chosen over the other two examined schemes. The reason for choosing YASHE over the BGV scheme is its relative simplicity over the BGV scheme. The BGV scheme offers powerful features, however, it is hardly fit for a lightweight application such as a network secure routing solution. It features large circuit depth evaluation and optimized operations to compensate but it requires large scale implementations due to such requirements like its large keys. In addition, BGV implementation libaries such as HElib [36] are complicated and have limited documentation. As such, it was decided it would be best to use a relatively smaller and simpler scheme.

The reason for choosing YASHE over the FV scheme is based primarily off of

the comparison results of Lepoint et al. [7]. In the authors' comparison study, they found that FV's operations with the exception of key generation take about twice as long as to compute as YASHE's. The decryption operation take roughly the same amount of computation time. However, it should be noted that this might simply be because of the ciphertext structure of FV since FV requires two ring elements to represent a given ciphertext whereas YASHE only requires one. The comparison may not be completely fair, therefore, as the operations of FV might be parallelizable in regards to both its ring elements. However, this would only make the implementation more complicated and it is speculated that both would have similar operation speeds afterwards as the primitive operations involved are very similar so depending on such an optimization is somewhat trivial.

6.2 SIMON

The SIMON block cipher is a lightweight block cipher that was publicly released by the National Security Agency (NSA) in 2013 [2]. The cipher exhibits a balanced Feistel network [37] with a round operation that consists of a mix of three simple bitwise operations: shift, XOR, and AND. The cipher consists of a round function and a key schedule function that are both executed repeatedly in series based on the number of rounds defined by a selection of parameters. A block diagram of the SIMON round function is represented in Fig. 6.1, where the operation S^i denotes a binary left rotation shift of i bits, the $\oplus$ operation denotes a binary XOR operation, and the & operation denotes a binary AND operation. The SIMON cipher offers a small collection of different groups of parameters to select based on block and key size requirements. In particular, the key schedule round changes based on the parameter configuration used. A block diagram of the SIMON key schedule function configured for a block size of 32 bits and a 64-bit key is represented in Fig. 6.2. The variable c is a constant set by the standard where $c = 2^{32} - 4$ and z_j is a periodic bit stream

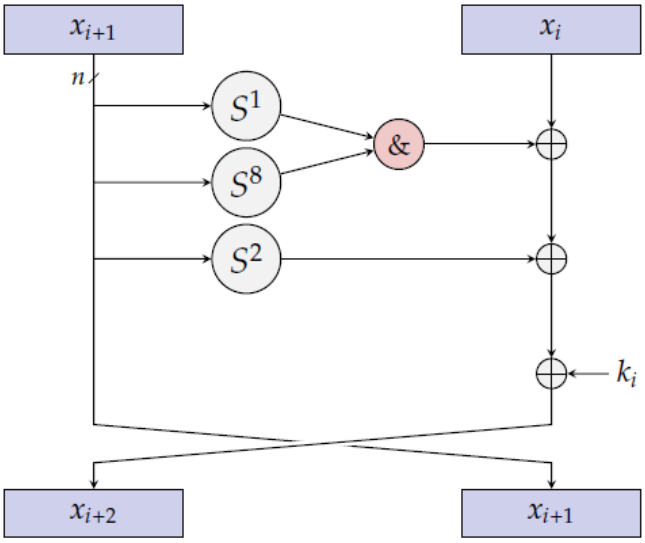

Figure 6.1: SIMON Block Cipher Round Function Block Diagram [2]

sequence $z_j = 11111010001001010110000111001 10...$ where the term $(z_j)_i$ refers to the i-th bit of z_j.

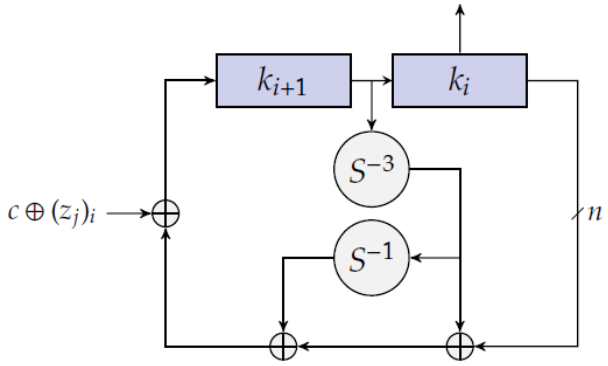

Figure 6.2: SIMON Block Cipher Key Schedule Function Block Diagram [2]

6.3 Case Study Scenario

The main focus of this work will be the cross domain solution application using HE. The first step in analyzing the practicality of homomorphic encryption as a tool for solving the cross domain problem is to establish a case study which contextualizes the problem. This will reveal glaring issues and open up the problem for analysis

of where HE can be applied. Following the analysis of the case study, a proof-of-concept application can then be implemented using the YASHE and SIMON schemes in order gain a practical profiling and gain some insight into the limitations of such an application. As stated, a CDS is a form of controlled interface that provides the ability to access, either manually or automatically, or transfer information between varying security domains [8]. Therefore, the case study will focus on the ability to route data from a source to a proper endpoint without revealing any information about the data over an untrusted network. The definition of the case study follows.

There exists multiple source users who want to relay information of varying classifications to specific endpoints. The data they relay is broadcasted on an untrusted network where the it will be received by all parties who act as network gateways. The purpose of the gateways is to only relay information whose classification matches that of the classification of the network at its endpoint. These gateways can also be treated as untrusted parties who must learn nothing about the type of data they relay nor the actual classification of its endpoint. Once a gateway has processed the data and determined the access level of the data it has observed, it will then relay the data to its endpoint at which the network can further handle the information. The system includes multiple types of classifications: Top Secret, Secret, Confidential, Restricted, Official, Unclassified. An endpoint network may have one or more users associated with it. A figure that represents the concept of the case study is represented in Fig. 6.3. Multiple networks may also share a classification, however, it should be noted that classified data does not necessarily need to be shared amongst all of them and there may be a further division of permission associated with the data according the the actual end users. In other words, although two networks may be classified as Top Secret, such as the networks defined with users A_{TS}, B_{TS}, and D_{TS}, the data may also have a destination address specified for a single member such as only for users B_{TS} or even D_{TS}. This responsibility may fall to the head of the network instead of

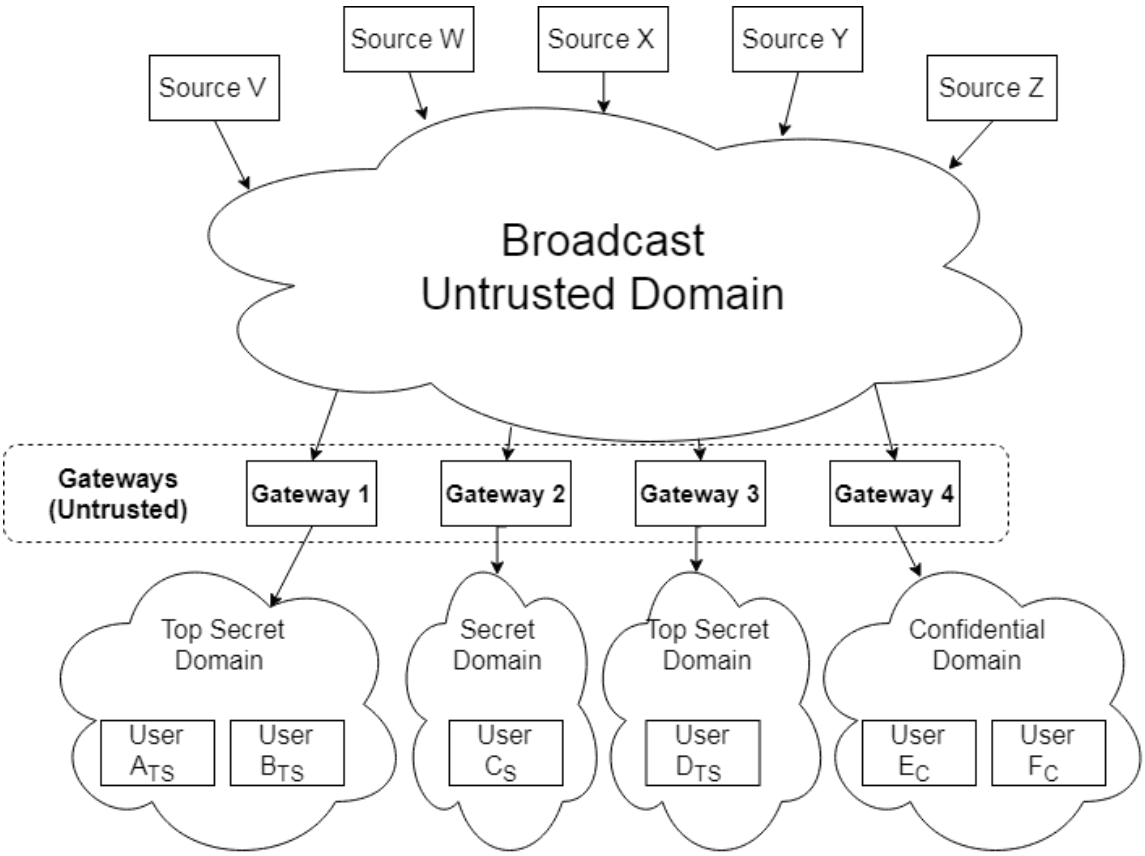

Figure 6.3: The Cross Domain Network Case Study

the gateway or some collaboration between the two.

Since the gateway is to learn nothing of the classification of the data and only whether the information should pass, the gateway is required to process a boolean result that is either true or false, regarding whether to route the result. As such, the gateway learns nothing about the data, or its intended destination, if the result does not end up passing the check. This prevents a rogue gateway and network from searching for specific information and gathering data.

6.4 The Application

As mentioned, the objective of the application is to route varying levels of classified data through an untrusted network without revealing any information regarding the

data. The approach taken is to build an application from the perspective of a single user, a gateway, and a network router. The user will be responsible for setting up the data to evaluate, the gateway will be responsible for evaluating the data, and the router will be responsible for observing the result data in order to properly determine the route end point. To achieve this, the idea is to use the YASHE scheme to securely evaluate the data to determine a proper router endpoint. This way no information regarding the attributes of the data is revealed but a result is still obtainable. The design is constructed with the following assumptions and requirements in mind:

- Payload data must remain encrypted at all times.
- Payload must have associated attribute data.
- Required network bandwidth should be minimized.
- Gateway must be treated as an untrusted party member.
- Gateway must not be able to decrypt homomorphic encryption data.
- Router can be semi-trusted at most.
- Router should only have access to result data.

The final concern to keep in mind with the construction of the design is the constraints and limitations of YASHE. Since performance and memory usage are a serious concern with YASHE and other FHE systems, the best approach to handling the data is to work with small but descriptive portions of data that is sufficient enough to determine what type of data is encrypted and its destination. To achieve this, a short piece of metadata will be computed and encrypted alongside the payload data. The metadata will be the primary data to evaluate homomorphically in order to determine a destination endpoint. This way, the amount of data that needs to be encrypted and evaluated homomorphically is minimized which should put the performance and

memory problems within bounds that are acceptable. The design requirements are addressed as follows.

6.4.1 Encrypted Payload

The payload can remain encrypted at all times by encrypting the data through traditional means. To ensure an extra layer of security for sensitive information that the payload conveys, the payload should even be encrypted under a scheme unrelated to the homomorphic portion of the protocol with a key that is only shared between the front end and back end users. This ensures that under the circumstance that the routing protocol is compromised, the data itself remains secure.

For all intents and purposes, the encryption and handling of the payload is ignored in the implementation of the application. This is because the focus of the application is about addressing and analyzing the feasibility of HE as a solution. Since the application does not actually operate on a network and instead speculates the effects on the network, the payload is effectively ignored. However, it should be noted that it might be important to analyze in future work as the size of the payload and its transfer may have an effect on the processing requirements of the solution.

6.4.2 Payload Attribute Data

The payload is accompanied by metadata that describes the attributes of the payload data. For the purpose of the application, the metadata is a small piece of information that contains a tag of what type of classification that identifies the payload. The metadata was chosen to be a 32-bit value whose assignment is equal to that of one of the masking values for each of the security classification values: Top Secret, Secret, Confidential, Restricted, Official, Unclassified. The choice to represent the metadata as a 32 bit value was made to avoid potential security concerns. The values representing each attribute need to be distinct enough from one another to ensure that an

Table 6.1: Assigned Metadata Classificaion Values

Description	Hex Value
Top Secret	`0xE7191C86`
Secret	`0x72CCDDC8`
Confidential	`0xC989E663`
Restricted	`0x636C6E0C`
Official	`0x99BB94C7`
Unclassified	`0xCCCCE185`

adversary can not spoof the system by guessing what a particular attribute's metadata value is and performing a homomorphic comparison between the actual data and their guess. A value size of 32-bits is non-trivial enough to defend against an adversary learning about the descriptions of metadata in this way. In addition, since this is the data that will need to be evaluated homomorphically in order to ultimately determine access, the data is chosen to be kept small to minimize the complexity of the homomorphic operations involved. Each classification attribute is randomly assigned a different and distinct value. The assigned values of each classification in the context of the application are represented in Table 6.1.

6.4.3 Minimize Network Bandwidth

Even though the data that will be evaluated is small, the ciphertext expansion issue is still a glaring problem simply because the data will be transferred through a network and thus could have an adverse effect with the limited bandwidth of the network. A clever optimization to this problem that has been addressed in research [35, 7] is to instead transfer the data encrypted by means of a traditional block cipher and then decrypt the data with a homomorphically evaluated decryption circuit of the block cipher. The block cipher encrypted data simply needs to be encrypted with the FHE scheme first and then homomorphically evaluated with the block cipher decryption circuit alongside a homomorphic encryption of the block cipher key. The metadata can then be evaluated homomorphically just as if it had originally been

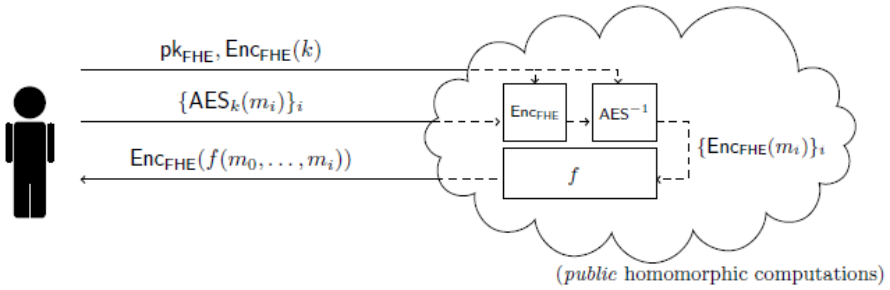

Figure 6.4: Server Data Flow for Homomorphically Evaluating Metadata

encrypted under the FHE scheme. An example data flow of this process as it would be processed by the gateway is represented in Fig. 6.4 [7]. Such a design policy requires an expensive one time transfer of the homomorphic encryption of the block cipher key and only the follow up transfers of the evaluation data encrypted under the block cipher. Note that this greatly reduces the required network bandwidth by a factor equal to that of the ciphertext expansion factor. However, it should also be noted that this will require more computation on the gateway side as the YASHE encryption operation has been off-loaded to the gateway in addition to the requirement of homomorphically decrypting the data. This computational cost will be based on how lightweight the block cipher is and how efficiently it can be evaluated homomorphically. Since the bandwidth is anticipated to be limited and the processing power of the users are anticipated to be weaker than the gateway elements, this is expected to be an improvement to the system.

It would thus be in the best interest of the proof-of-concept design to use a block cipher of simple complexity to simplify the effort required to evaluate the circuit in addition to minimize the noise growth of the resulting homomorphic ciphertext so that there is still enough leftover evaluation depth to perform the actual evaluation of the metadata. A block cipher that meets this need is the lightweight SIMON block cipher [2].

Compared to other traditional block ciphers such as AES, the operations of the

SIMON block cipher are relatively simple. The cipher operations consists of a mix of three simple bitwise operations: shift, XOR, and AND. These types of operations fit well into the context of FHE operations since YASHE and other FHE schemes utilize binary operations to achieve full homomorphism. In the context of homomorphic evaluation with a vector of ciphertexts that encrypt the individual bits of a block, a binary shift operation is implemented with a trivial index swapping of the ciphertexts, and XOR operation is implemented with a homomorphic addition operation, and an AND operation is implemented with a homomorphic multiplication operation. The only real concern involved with the homomorphic evaluation implemenation of this cipher is how many AND operations are performed in series which is strictly defined by the number of rounds of the round function. The key schedule routine does not exhibit any AND operations so it is not a concern. To minimize the number of rounds required, the application will use the smallest parameter selection with a 32-bit block size and a 64-bit key with a 32 round encryption routine [2].

6.4.4 Untrusted Gateway

The function of gateway is the heart of the cross domain solution. However, the gateway inherently needs to be untrusted as it pertains to the cross domain problem; if it were trusted on the same level as the data it handles, then there would not be a need for the cross domain solution. The gateway being untrusted means that it cannot obtain any information regarding the data it handles. This is where the application of HE comes into play. The YASHE scheme will allow the gateway to evaluate the encrypted data it receives and obtain an encrypted result. However, to prevent the gateway from observing the data encrypted by YASHE, it must not have access to the secret key. This will prevent it from decrypting any of the data that it has access. The gateway will still need access to the public key and evaluation key of the system in order to properly shape the encrypted metadata tag into the

homomorphic space and be able to evaluate on the data. However, since the gateway cannot decrypt the result of the evaluation, that task now has to be entrusted to the network router.

6.4.5 Semi-Trusted Router

The router is responsible for decrypting the result of the evaluation. For this reason, the router may need to be considered a semi-trusted party on the network in the sense that it needs to be entrusted with a sensitive piece of information, namely the secret key of the YASHE scheme. However, since the router now has access to any YASHE encrypted data, the router must not be allowed access to anything encrypted under the public key. In particular, the most important piece of information that the router must not have access to is the block cipher key that is encrypted under the YASHE public key. This can be avoided through secure communication of such data between the front end users and the gateways through an overhead layer of traditional cryptographic means. The overhead analysis of the secure communication will be ignored for the purpose of the case study application. Concretely, the only data that the router should have access to is the secret key and the encrypted evaluation results from the gateway.

6.4.6 Dataflow

The design of the case study application that fulfills the requirements is as follows. As a proof-of-concept for purposes of analysis, the application involves only a single user, gateway, and router. A top level design of data flow is represented in Fig. 6.5. The user generates a SIMON cipher key and encrypts it under YASHE encryption. This data is theoretically securely transferred to the gateway as a one-time costly operation. The same operation also requires a one-time homomorphically evaluated key expansion of the encrypted SIMON key. The user generates one random meta-

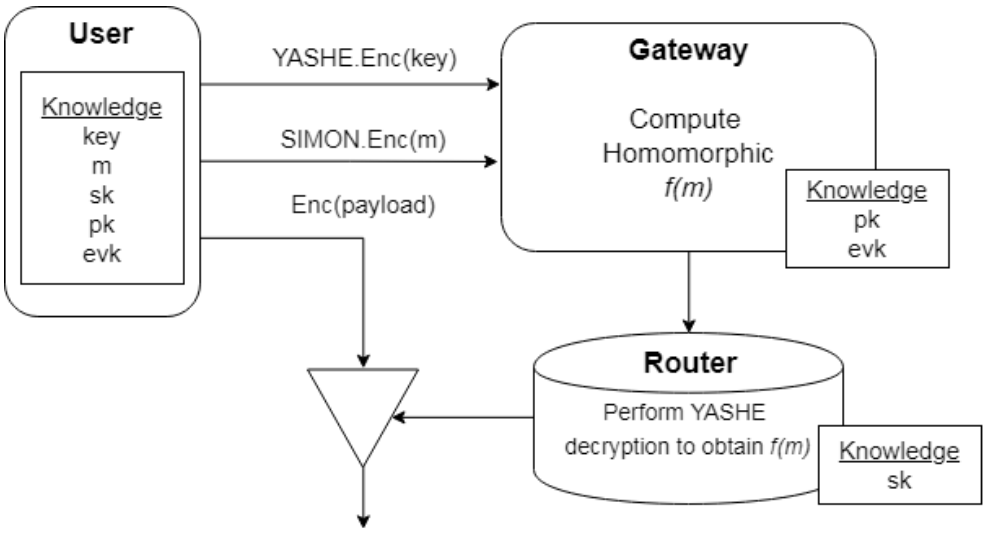

Figure 6.5: Top Level Application Design

data tag to describe a theoretical payload as described by the metadata scheme and encrypts it under SIMON encryption using the generated SIMON key. The gateway element theoretically receives this data and performs a homomorphic evaluation of the data to produce an encrypted result under YASHE encryption. The somewhat large resulting ciphertext is theoretically transferred to the router. The router performs a YASHE decryption on the data for final evaluation on whether to pass or drop the data through the network.

The main focus of the analysis lies with the operations performed by the gateway. A design of the data flow performed by the gateway is represented in Fig. 6.6. To actually evaluate the encrypted metadata under SIMON encryption, the gateway homomorphic decrypts the SIMON ciphertext using a homomorphically evaluated SIMON decryption circuit and the encrypted SIMON key under YASHE encryption. After words, the gateway now has the plain metadata but encrypted under YASHE encryption. A homomorphic bitwise compare circuit is used to correctly evaluate the metadata based on a simple string compare of the encrypted result against relevant encrypted metadata tags. An example of a 4-bit bitwise compare circuit is represented in Fig. 6.7. Note again that each XOR gate would be performed through a homo-

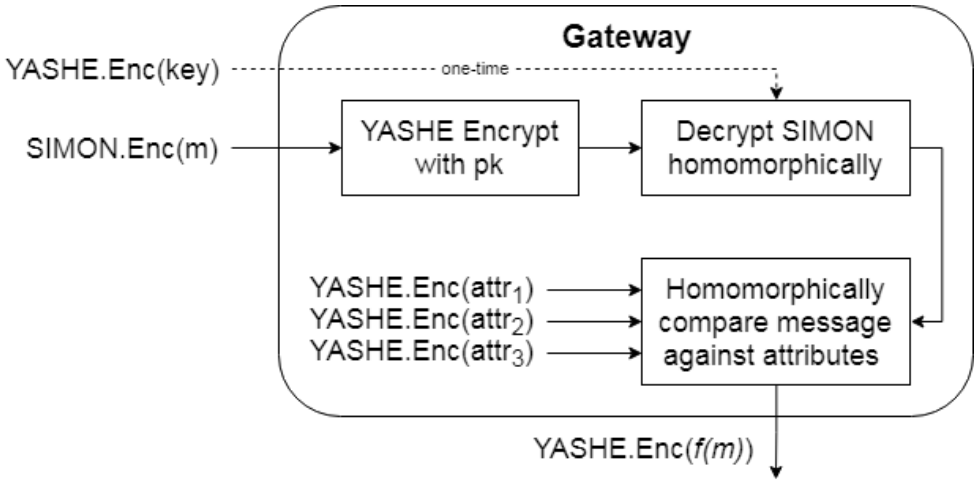

Figure 6.6: Gateway Process Flow

morphic addition operation between two encrypted bits and each AND gate would be performed through a homomorphic multiplication operation. The invert gate operation is executed through an XOR operation of the resulting bit with an encrypted value of '1'. The multiplicative depth of a full n-bit comparison circuit is equal to $\log_2(n)$; so, the 32-bit comparison circuit needed by the application has a multiplicative depth of 5. Theoretically, the encrypted metadata tags to compare the data can even be dynamically provided to the gateway through a subscribe system from the end-users to prevent the server from knowing the format of the metadata scheme; however, for this proof-of-concept application, the gateway will simply compare the string against all encrypted tags. After circuit evaluation, the gateway provides a complete encrypted summary result which is transferred to the router.

6.4.7 Software Implementation

The application was implemented in a C++ software application. Lepoint et al. [7] provide a preliminary open source implementation of the YASHE scheme. The implementation offers an easy to follow class architecture for YASHE which includes parameter set up, key generation, encryption, decryption, and ciphertext addition and multiplication operations. Their implementation also provide a homomorphic evalu-

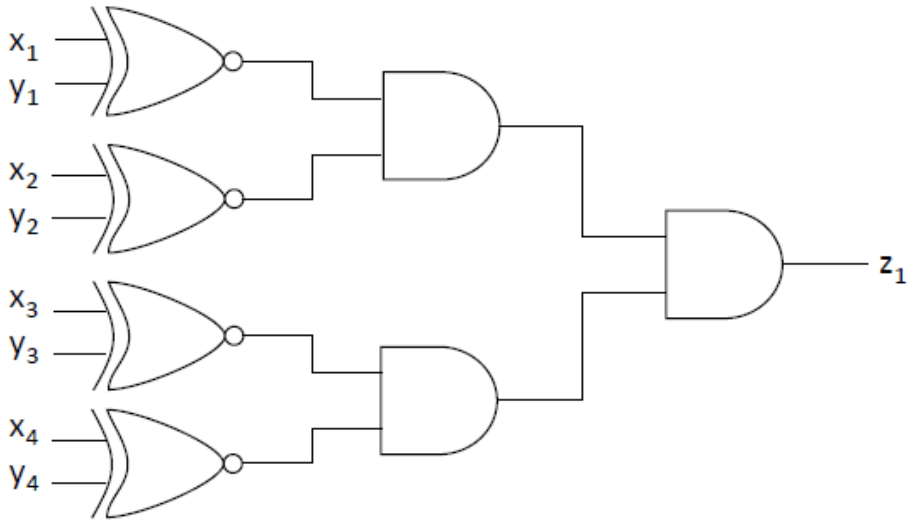

Figure 6.7: Bitwise Compare Circuit

ation implementation of the SIMON block cipher scheme. The application uses the Fast Library for Number Theory (FLINT) [38] and GNU Multiple Percision (GMP) [39] arithmetic libraries which are optimized libraries for working with large data structures such as the ones used in the YASHE scheme. The application utilizes multi-threading for YASHE and homomorphic SIMON operations. The application is compiled using gcc 4.4.7 and executed on an AMD A10-7850K running at 1.7GHz with 16GB of RAM for all experiments.

Chapter 7

Profiling Results

7.1 Results

In order to make a proper analysis of the application, all important portions of data were profiled for memory usage and execution time. The memory profiling results of the application, set with varying parameters to ensure 128-bit YASHE security, is represented in Table 7.1. The results of the memory profiling suggest exactly what was speculated. The ciphertext expansion for a small 32-bit piece of metadata is quite large with all portions of data representable within the kilobyte and even megabyte ranges. However, the data is not too large for a modern computing unit to handle. The more serious concern regarding memory is how much data needs to transmit over the network. For the most part, the public key, evaluation key, and encrypted SIMON key only need to be transmitted during a one-time expensive transaction so their sizes are not a primary concern. The metadata that is transferred with every payload is a small 32-bit of data encrypted under SIMON so there is no concern regarding how expensive it may be to transfer it the gateway. The follow-up encrypted results that need to be handed to the router do not exceed more than a few megabytes which is also acceptable.

The performance profiling results of the application with the same parameters is represented in Table 7.2. As opposed to the memory profiling, the performance

Table 7.1: Memory Profile Results of CDS Application with 128-bit Security Parameters

n	$\log_2(q)$	YASHE Secret Key (kB)	YASHE Public Key (kB)	YASHE Evaluation Key (kB)	YASHE Encrypted SIMON Key (kB)	Encrypted Metadata (Bytes)	Evaluation Data (kB)	Result (kB)
1024	29	8	8	8	512	4	256	8
2048	56	16	16	32	1,024	4	512	16
4096	111	64	64	256	4,096	4	2,048	64
8192	220	256	256	1,792	16,384	4	8,192	256
16384	440	896	896	12,544	57,344	4	28,672	896
32768	880	3,584	3,584	100,352	229,376	4	114,688	3,584

Table 7.2: Average Performance Profile Results of CDS Application with 128-bit Security Parameters

n	$\log_2(q)$	YASHE Encrypt SIMON Key (s)	YASHE Encrypt Tag (s)	Homomorphic SIMON Decryption (s)	Homomorphic Metadata Evaluation (s)	YASHE Result Decrypt (s)
1024	29	0.033	0.020	0.504	0.097	0.001
2048	56	0.070	0.038	1.492	0.290	0.003
4096	111	0.618	0.303	14.864	2.405	0.025
8192	220	2.515	1.190	111,186.000	17.705	0.199
16384	440	13.304	4.554	549,340.000	96,611.000	0.458
32768	880	62.657	2.055	2,399,270.000	436,046.000	1.550

profiling results suggest that there may exist some concerns. Again, the encryption of the SIMON key is a one time process and the amount of time needed to process the operation exceeded no more than over a minute so there is no need for concern there. The total execution time to encrypt the metadata and evaluate the data, however, is much more alerting. The results suggest that a large amount of processing time is required to completely evaluate the metadata. In fact, the amount of time required to process the data when $n = 32768$ almost exceeds an hour, which would make it difficult to use in a practical application even with optimizations. The amount of time required to process the data when $n \leq 16384$ does not exceed a little over 10 minutes which is more likely to fall within acceptable limits accounting for potential optimizations. The decryption of the result was the fastest process of all the operations barely requiring more than a second of processing time at most which should be acceptable within the processing bounds for the router in the setting. The param-

eters could always be increased to allow for more evaluation depth but it was already established that the $n = 32768$ parameters result in impractical performance so the idea of increasing the parameters further is disregarded.

7.2 Noise Growth

Despite the optimism provided by the results of the memory and performance profiling, it turns out the parameters to ensure 128-bit security are still a bit too strict for the homomorphic evaluation to complete successfully. For all parameters presented before, the resulting comparison evaluation failed to return the correct verified result. This is because the noise of the ciphertext grows too large to correctly decrypt the result after an evaluation of the entire circuit. The reason behind this is that the evaluation depth of the SIMON decryption depth is too deep for the provided parameters. To observe this effect and better conclude where the problem persists, the noise growth of the system was profiled for the varying levels 128-bit, 80-bit, and 64-bit security parameters.

7.2.1 Full Evaluation

The noise level of the entire evaluation circuit, that is both the SIMON decryption circuit followed by the comparison circuit, was measured and plotted for every multiplication level against all parameters with $n \leq 16384$. Due to the large size of the coefficients, the noise value is represented in bit width notation instead of actual values. The plots of the measured noise growths for the varying parameters of 128-bit, 80-bit, and 64-bit security are represented in Fig. 7.1, 7.2, and 7.3, respectively. Note that because the ciphertext becomes incorrect after a certain noise value, specifically when the noise value is greater than half the value of the coefficient q, the noise value could not be measured further than this point. This results in the noise saturating at the described value when the ciphertext is no longer correct and results in a sat-

uration appearance in the plot. The full evaluation of both the SIMON decryption circuit and the comparison circuit requires 37 layers of multiplicative depth.

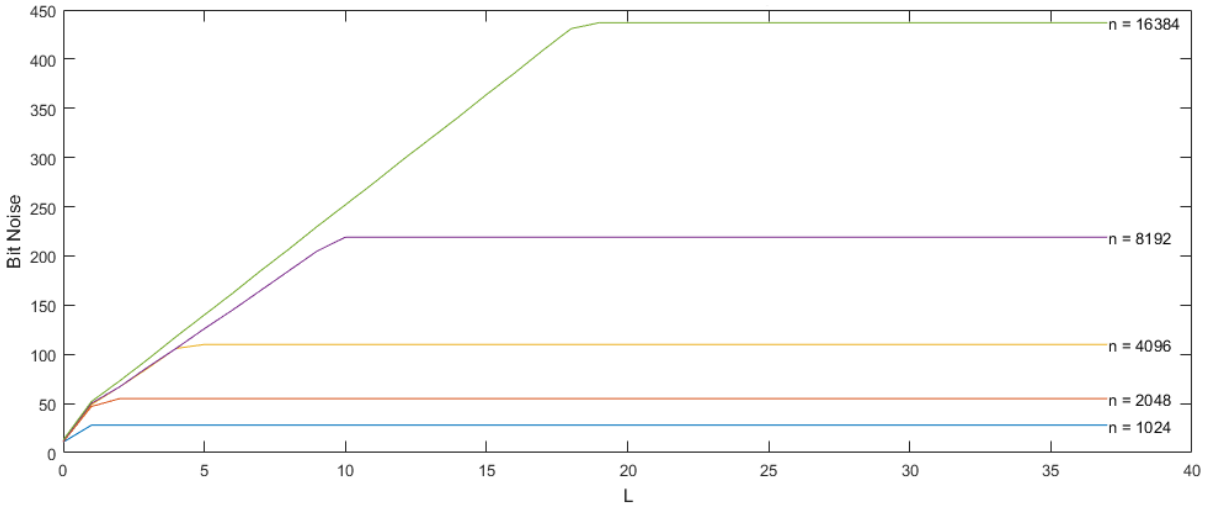

Figure 7.1: Ciphertext Noise Growth for SIMON and Compare Evaluation Circuits with 128-bit Security Parameters

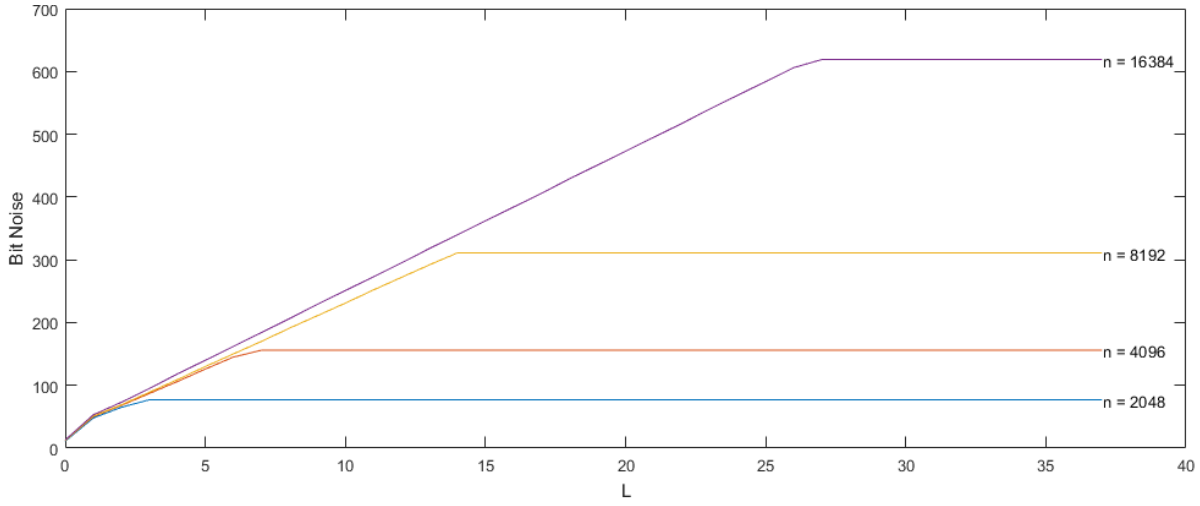

Figure 7.2: Ciphertext Noise Growth for SIMON and Compare Evaluation Circuits with with 80-bit Security Parameters

As such, the data suggests what was expected for the circuit evaluation with the 128-bit parameters. The data does show how the depth at which the circuit could be evaluated increases with the increasing parameter sizes suggesting that a complete evaluation could be achieved if the parameters were large enough. The noise growth, however, plateaued after a certain amount of multiplication depth for each

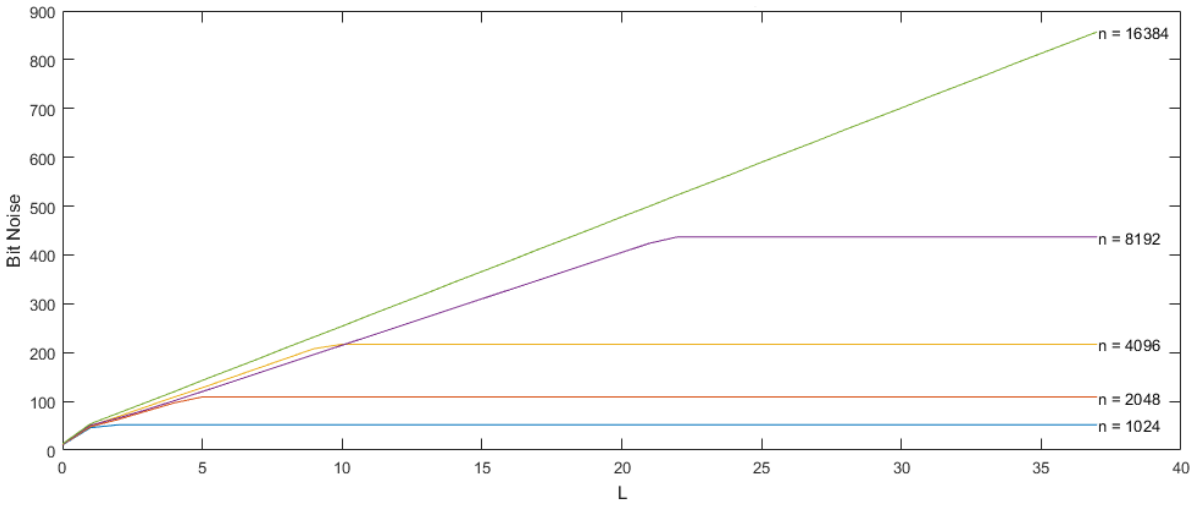

Figure 7.3: Ciphertext Noise Growth for SIMON and Compare Evaluation Circuits with 64-bit Security Parameters

parameter setup and all evaluations fail no later than halfway during the evaluation suggesting that achieving 128-bit security is probably not achievable within practical bounds. This means that the application is going to need to depend on a lower security parameter setup.

As the data suggests, 80-bit security is not achievable either under the current setup. All noise growth plots with the 80-bit security parameters plateaued after a certain depth as well. It is evident that the upper bound on the evaluation depth is larger than that of the more strict parameters given by the 128-bit setup. However, a complete and correct evaluation was achieved using a 64-bit security parameter setup when $n = 16384$, which shows that a configuration is possible. However, all smaller parameters failed, so this limits the application to using this set of parameters.

Fig. 7.4 compares the noise growth between the varying security levels when $n = 16384$. It is observable from this data that lower security parameters permit deeper evaluation depth. This is due to the larger ciphertext coefficient sizes that are permitted by lower security parameters and as such the noise saturates at later evaluation depths.

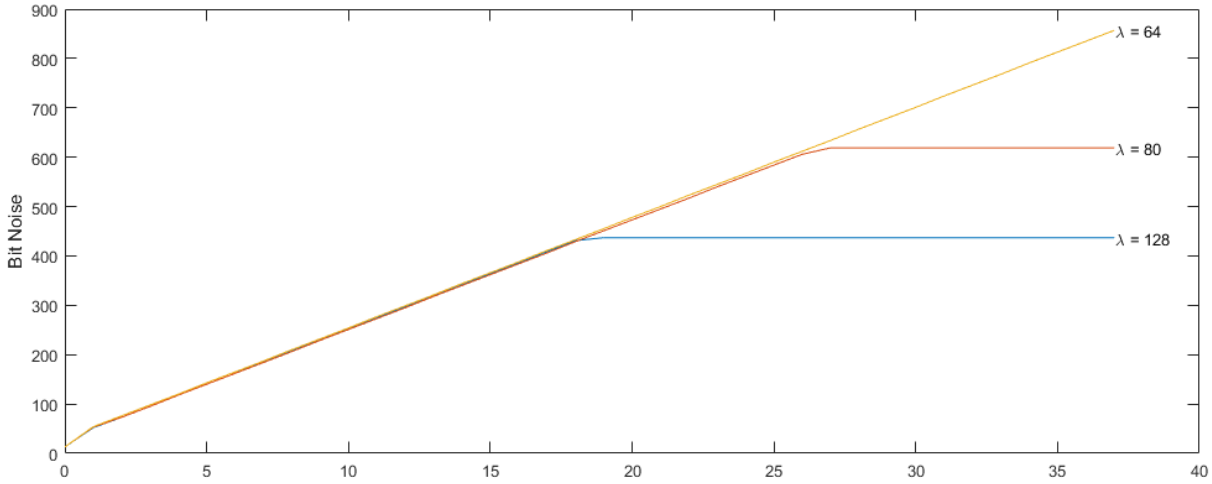

Figure 7.4: Noise Growth Comparison for Varying Bit Security Parameters with n=16384

7.2.2 Comparison Only Evaluation

A possible direction to move in order to fend off the noise growth problem is to avoid using the homomorphic evaluation circuit. To assess this option, the noise was again profiled after the SIMON encryption and homomorphic decryption procedures were removed thus leaving only the comparison evaluation circuit. Note that this results with memory and performance profiling similar to that of the results represented in Tables 7.1 and 7.2 but just ignoring the SIMON evaluation portions. The plots of the measured noise growths of only the comparison circuit for the varying parameters of 128-bit, 80-bit, and 64-bit security are represented in Fig. 7.5, 7.6, and 7.7, respectively. The full evaluation of the comparison circuit requires 5 layers of multiplicative depth.

Due to the significantly lower multiplicative depth of the comparison circuit, the data shows there are several parameter setups for each security option that achieved a complete and correct evaluation. The results show that a 128-bit security setup can be achieved using at the parameter setup for $n \geq 8192$, a 80-bit security setup can be achieved using the parameter setup for $n \geq 4096$, and a 64-bit security setup can be achieved using the parameter setup for $n \geq 4096$. It should also be noted that the

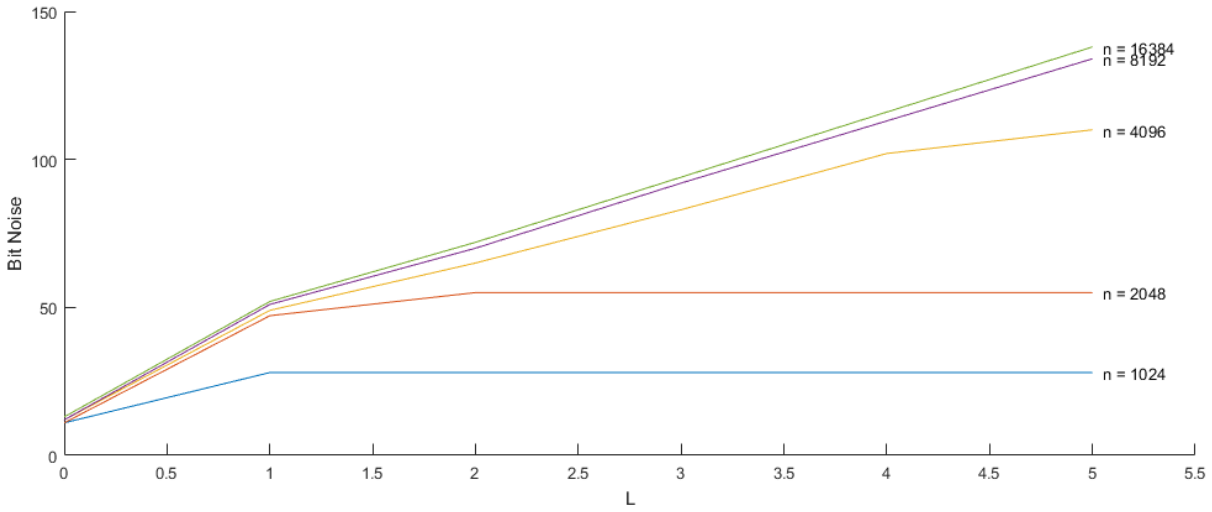

Figure 7.5: Ciphertext Noise Growth for Only the Comparison Evaluation with 128-bit Security Parameters

parameter setup for $n = 2048$ under the 64-bit security setup could potentially be achieved with slight modifications to some parameters as the noise growth just barely overreaches the level for correctness. These setups at least offers the higher security options as opposed to the setup with the SIMON evaluation. However, note that this sort of system flow would require the user to encrypt their data under YASHE and tranfer the large ciphertext over the broadcast network which may not be acceptable under certain network restrictions.

7.3 Varying Radix-w

The radix-w parameter is normally used as an optimization for reducing the dimension of the PowersOf and WordDecomp vectors which results in a smaller public key and a faster key switching routine. However, it comes at the expense of increasing the noise growth of the multiplication operation [6]. Therefore, reducing the value for w should increase the multiplicative depth that the scheme can evaluate for a given parameter selection at the expense of increasing the public key size and decreasing the performance of the multiplication operations. To assess this, the noise growth

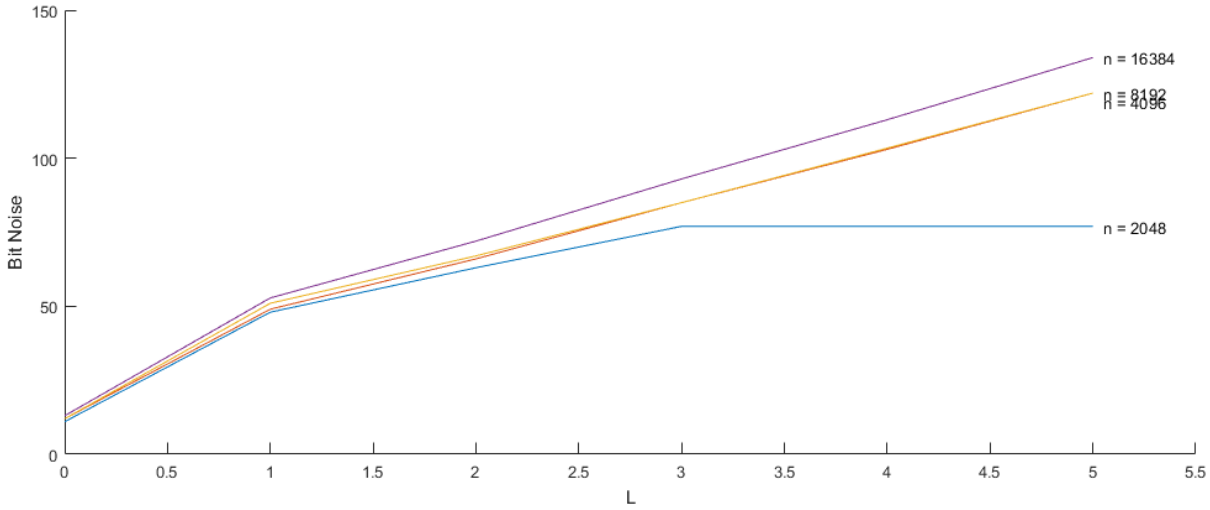

Figure 7.6: Ciphertext Noise Growth for Only the Comparison Evaluation with 80-bit Security Parameters

was profiled for varying values of w using the $n = 4096$ parameters for 64-bit security as a benchmark point. The noise growth plot for varying w is represented in Fig. 7.8. As the data suggests, decreasing values of w does in fact slow the noise growth. Unfortunately, the rate at which it slowed the growth only amounted to about a single extra multiplication depth. Accounting for memory and performance hits that decreasing the value of w results, it is recommended that this value be used primarily as a performance optimization instead of a multiplication depth enhancement optimization except under the circumstance where a single level of multiplication is absolutely needed. In fact, as long as the multiplication depth permits a few extra trade-offs, the value of w should be increased to benefit from the memory and performance benefits.

7.4 Varying Rounds for SIMON

Another natural option to attempt to optimize the parameter selection of the full application is to reduce the complexity of the decryption circuit. The only way to do so for the SIMON cipher is to reduce the number of rounds. To assess whether doing so will open other options for parameter selection, the noise growth was tracked

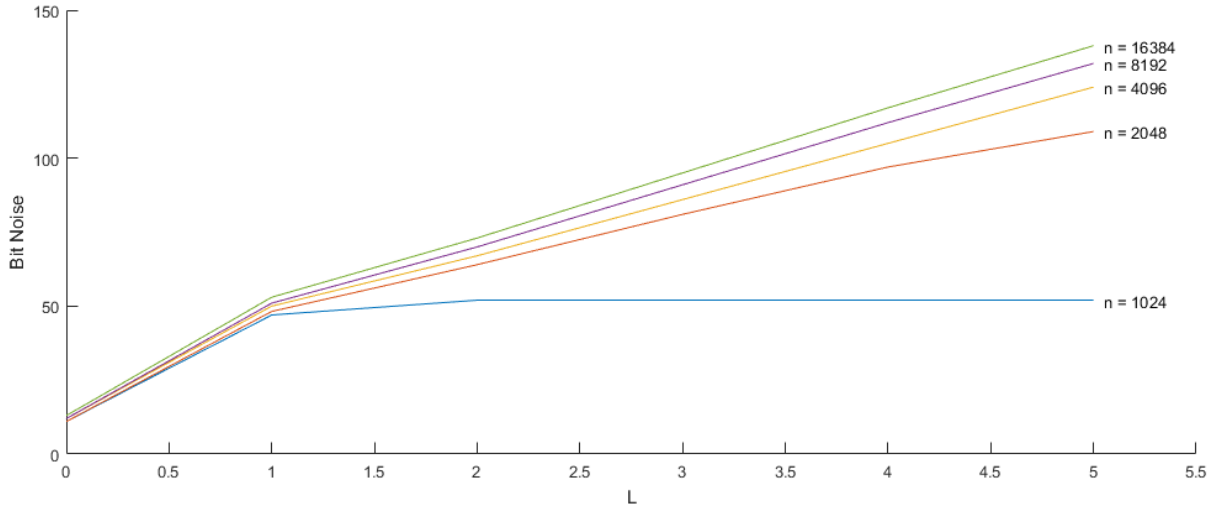

Figure 7.7: Ciphertext Noise Growth for Only the Comparison Evaluation with 64-bit Security Parameters

against the 64-bit security parameters of for a various number of SIMON rounds. The parameter selection for $n = 16384$ was ignored since it was already confirmed that this setup operates correctly. The final noise value after a labeled number of SIMON rounds against the varying parameter values is represented in Table 7.3. As it turns

Table 7.3: Bit Noise After Varying Rounds of SIMON

n	$\log_2(q)$	Rounds			
		32	16	8	4
1024	56	55	55	55	55
2048	110	109	109	109	109
4096	218	217	217	217	199
8192	438	437	437	296	198
16384	885	813	478	299	210

out, lowering the number of rounds does not offer many more options. The only parameter setups where correctness was ensured was for $n = 8192$ with at most 8 SIMON rounds and $n = 4096$ with at least 4 SIMON rounds. All other setups were shown to still fail. However, it should be noted that altering the SIMON cipher to use such a small number of rounds is synonymous with lowering the security of the cipher [40]. According the the authors of [40], the security analysis of SIMON starts to suggest heavy decay of security after reducing the number of rounds past 16.

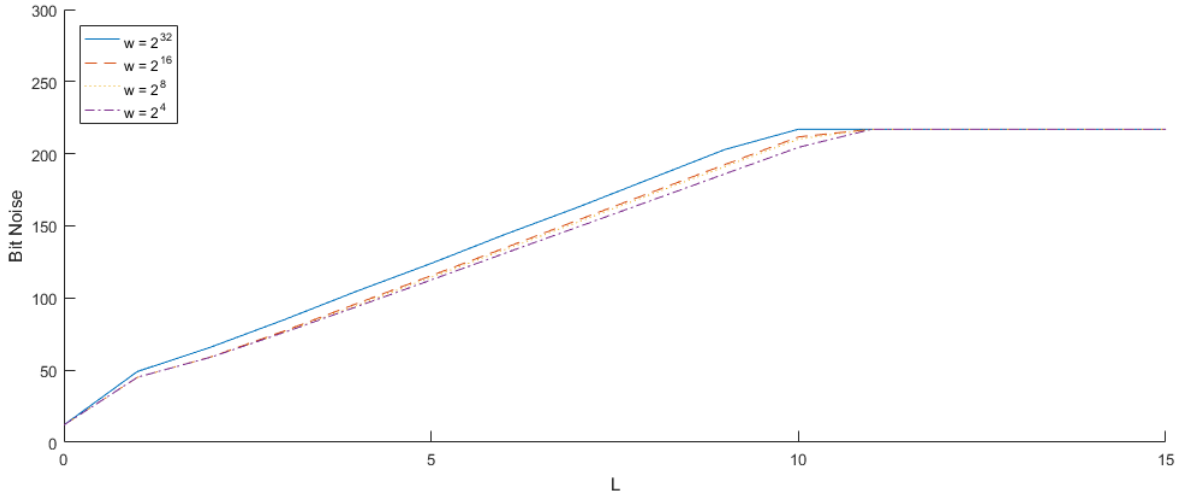

Figure 7.8: Ciphertext Noise Growth for Varying Radix-w

Although it would be appealing to take advantage of such a setup, the SIMON cipher is the first security layer in concealing the metadata and relying on too weak of an assumption would completely compromise the system. Therefore, this optimization is considered not very practical and should be avoided.

7.5 Final Parameter Selection Choices

The mentioned analysis techniques established allow for a small collection of parameter configurations to be established that caters to a variety of requirements. The configurations were established based on the smallest parameters that allowed for correct evaluation within the requirement restraints of the configuration. The first configuration, represented as configuration α, defines parameters for a configuration that correctly evaluates the SIMON decryption and comparison circuits while establishing 64-bit security under YASHE. The values of $n = 16384$, $\log_2(q) = 885$, and $w = 2^{32}$ were deemed the most optimal for such an establishment. The second configuration, represented as configuration β, defines parameters for a configuration that correctly evaluates evaluates an reduced 8-round SIMON decryption and comparison circuit while establishing 64-bit security under YASHE. The values of $n = 8192$,

Table 7.4: Memory Profile Results of CDS Application

Configuration	YASHE Secret Key (kB)	YASHE Public Key (kB)	YASHE Evaluation Key (kB)	YASHE Encrypted SIMON Key (kB)	Encrypted Metadata (Bytes)	Evaluation Data (kB)	Result (kB)
α	1,792	1,792	50,176	114,688	4	57,344	1,792
β	448	448	6,272	28,672	4	14,336	448
γ	64	64	448	N/A	2,097,152	2,048	64
δ	32	32	224	N/A	1,048,576	1,024	32

Table 7.5: Average Performance Profile Results of CDS Application

Configuration	YASHE Encrypt SIMON Key (s)	YASHE Encrypt Tag (s)	Homomorphic SIMON Decryption (s)	Homomorphic Metadata Evaluation (s)	YASHE Result Decrypt (s)
α	48.333	26.129	3,083.360	489.089	148.423
β	11.590	6.424	473.138	370.097	0.456
γ	N/A	N/A	N/A	3.206	0.024
δ	N/A	N/A	N/A	1.229	0.010

$\log_2(q) = 438$, and $w = 2^{32}$ were deemed the most optimal for such an establishment. The last two configurations, represented as configurations γ and δ, define parameters for configurations that correctly evaluate only the comparison circuit under 128-bit or 64-bit security. The values of $n = 4096$, $\log_2(q) = 111$, and $w = 2^{16}$ were deemed the most optimal for configuration γ to achieve 128-bit security under YASHE. The values of $n = 2048$, $\log_2(q) = 110$, and $w = 2^{16}$ were deemed the most optimal for configuration δ to achieve 64-bit security under YASHE. The memory profiling results for each configuration is represented in Table 7.4 and the performance profiling results for each configuration is represented in Table 7.5.

Each of the configurations offer a trade-off between strengths and weaknesses. Configuration α first and foremost offers a set of parameters that ensure correctness and security under YASHE for a fully established application data flow. However, the profiling results show that this configuration results in largest memory footprint and execution time of all the configurations. However, these were the only parameters established that could completely evaluate the application.

Configuration β offers less costly option against α at the trade-off of reducing the number of rounds in the SIMON cipher. This option is faster and requires less memory but offers weaker security under SIMON [40].

Configurations γ and δ offer alternative options to the application design. By cutting out the SIMON cipher from the design, the performance and memory footprints improve significantly. However, this change requires a crucial networking change. Instead of transferring a small 32 byte metadata under SIMON encryption, the users are now responsible for transferring the metadata encrypted under YASHE which is significantly larger. However, because the parameters are relatively small, the encrypted tags are not critically large and this trade-off may potentially outweigh benefits of the prior application if the network bandwidth allows it. However, it should be noted that this configuration can not effectively utilize the batching technique as the gateway has no way of packing multiple encrypted metadata tags into a single ciphertext anymore. However, any acceleration optimizations will still apply.

It should be noted, however, that performance optimization is still a possibility. For example, the throughput of the system could be increased significantly using the batching technique [32]. This would entail the gateway collecting several metadata samples and batching them into a single ciphertext for evaluation. The evaluation would then result in each metadata sample being evaluated separately at the cost of a single evaluation, resulting in a larger data throughput at the cost of nearly negligible pre- and post-processing procedures.

In addition, literature has shown that hardware acceleration is a promising option for optimizing homomorphic operations. Michael Foster devised a high level synthesis module for large polynomial multiplication using the Karatsuba algorithm in order to target the costly multiplication operation of homomorphic encryption schemes [41]. In his Master's thesis, he reports theoretical speedups up to 135 times over traditional optimized software methods [41]. Indeed, this work could naturally be implemented

using a cloud server environment with FPGA integration to achieve high performance optimizations. Wei Dai et. al. have built a discrete Fourier transform based library for Nvidia GPUs to support the homomorphic operations of addition and multiplication [42]. Their research has achieved speedups up to 2.8 times for homomorphic multiplication over a comparable CPU bound implementation [42]. Erdinç Öztürk et. al. have designed a custom FPGA hardware accelerator to optimize homomorphic multiplications [43]. They report speedups more than 102 times faster than software implementations [43]. Indeed, large speedups can be achieved through use of hardware accelerators through use of different mediums such as GPU, FPGA, and HLS designs. Other works include [44, 45, 46].

Chapter 8

Conclusions and Future Work

This work has shown that, provided the application process is kept lightweight and simple, a cross domain solution can be achieved using an application of homomorphic encryption. We demonstrated that a practical application that securely transfers cross domain information across an untrusted network can be achieved under a parameter selection that ensure 64-bit security under both the YASHE and SIMON schemes while guaranteeing correctness is achievable. Such a configuration requires an execution time of about one hour and no more than a 220 MB of total sum of processing memory, both of which are achievable by most modern computing systems. The system exhibits high latency due to the need to homomoprhically evaluate the SIMON decryption circuit to ensure that the required user to gateway network bandwidth remains low.

Alternative configurations were established that performance faster in the case that these results are considered too restrictive. One such configuration removes the SIMON decryption circuit to significantly decrease the amount of work required to process the encrypted data. Under this configuration, using parameters to ensure both correctness and 128-bit security results in a process that executes in less than a minute. Likewise, using parameters to both ensure correctness and 64-bit security executes in less than a couple seconds. The trade-off in both systems compared to the original is that the metadata must now be encrypted and transferred under YASHE

which moves the encryption workload to the user. This systems also requires a larger network bandwidth.

However, the conclusion on practicality of any of the configurations in a real world changes based on requirements and perspective. From the perspective of a government or military application, the performance results of the complete configuration system could likely fit within operable boundaries if security requirements outweigh the need for low processing latency. From the perspective of a commercial application, the performance results of the complete configuration system do not mesh well in the high speed competitive environment of today's industry where most applications are expected to execute in real time. These applications could, however, be able to afford the network bandwidth trade-off of the configurations described if the bounds of the system allow it.

Other fully homomorphic encryption schemes and techniques should be explored. The YASHE scheme is designed in a way to keep ciphertext and key elements small. However, other schemes, such as BGV, do not take this approach and instead focus on more robust designs that allow for the evaluation of deeper circuits. A route for further research is to explore such schemes and evaluate whether faster performance can be achieved under them and observe what trade-offs would be present.

The performance of current fully homomorphic encryption schemes, especially within the context of large parameters, can still be improved upon. Hardware acceleration is a natural route to take to approach this problem that has been explored previously in research [41, 42, 43]. The next step would be to explore and deploy such optimizations to expand upon this work.

Bibliography

[1] D. Micciancio and S. Goldwasser, *Complexity of Lattice Problems: A Cryptographic Perspective*, ser. The Kluwer International Series in Engineering and Computer Science. Boston, Massachusetts: Kluwer Academic Publishers, Mar. 2002, vol. 671.

[2] R. Beaulieu, D. Shors, J. Smith, and S. Treatman-clark, "The simon and speck families of lightweight block ciphers," Cryptology ePrint Archive, Report 2013/404, 2013, https://eprint.iacr.org/2013/404.

[3] R. L. Rivest and M. L. Dertouzos, "On Data Banks and Privacy Homomorphisms," 1978.

[4] C. Gentry and D. Boneh, *A Fully Homomorphic Encryption Scheme*. Stanford University Stanford, 2009, vol. 20, no. 09.

[5] H. G. Liddell and R. Scott, *An intermediate Greek-English lexicon: founded upon the seventh edition of Liddell and Scott's Greek-English lexicon*. Harper & Brothers, 1896.

[6] J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig, "Improved security for a ring-based fully homomorphic encryption scheme," Cryptology ePrint Archive, Report 2013/075, 2013, https://eprint.iacr.org/2013/075.

[7] T. Lepoint and M. Naehrig, "A comparison of the homomorphic encryption schemes fv and yashe," in *International Conference on Cryptology in Africa*. Springer, 2014, pp. 318–335.

[8] Committee on National Security Systems, "Committee on National Security Systems (CNSS) Glossary," no. 4009, p. 160, 2015. [Online]. Available: https://cryptosmith.files.wordpress.com/2015/08/glossary-2015-cnss.pdf

[9] NIST, "Security and Privacy Controls for Federal Information Systems and Organizations Security and Privacy Controls for Federal Information Systems and Organizations," *Sp-800-53Ar4*, pp. 400+, 2014.

[10] 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.

[11] J. D. C. Benaloh, "Verifiable secret-ballot elections," 1987.

[12] E. Kushilevitz and R. Ostrovsky, "Replication is not needed: Single database, computationally-private information retrieval," in *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*. IEEE, 1997, pp. 364–373.

[13] D. J. S. Robinson and P. (Firm), *An introduction to abstract algebra*, 1st ed. New York: Walter de Gruyter, 2003;2008;.

[14] H. Cohen, *Number Theory: Tools and Diophantine Equations*. New York: Springer, 2007, vol. 239.

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

[16] S. Khot, "Hardness of approximating the shortest vector problem in lattices," *J. ACM*, vol. 52, no. 5, pp. 789–808, Sep. 2005. [Online]. Available: http://doi.acm.org/10.1145/1089023.1089027

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

[18] ——, "The learning with errors problem," *Invited survey in CCC*, vol. 7, 2010.

[19] 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.

[20] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS '12. New York, NY, USA: ACM, 2012, pp. 309–325. [Online]. Available: http://doi.acm.org/10.1145/2090236.2090262

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

[22] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," Cryptology ePrint Archive, Report 2012/144, 2012, https://eprint.iacr.org/2012/144.

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

[24] D. Stehlé and R. Steinfeld, "Making NTRUEncrypt and NTRUSign as Secure as Standard Worst-Case Problems over Ideal Lattices," *Eurocrypt*, vol. 6632, pp. 27–47, 2011.

[25] 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*, ser. STOC '12. New York, NY, USA: ACM, 2012, pp. 1219–1234. [Online]. Available: http://doi.acm.org/10.1145/2213977.2214086

[26] M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, J. Hoffstein, K. Lauter, S. Lokam, D. Moody, and T. Morrison, "Security of Homomorphic Encryption," 2017.

[27] M. R. Albrecht, R. Fitzpatrick, and F. Göpfert, "On the efficacy of solving lwe by reduction to unique-svp," in *International Conference on Information Security and Cryptology*. Springer, 2013, pp. 293–310.

[28] R. Lindner and C. Peikert, "Better key sizes (and attacks) for lwe-based encryption," in *Cryptographers' Track at the RSA Conference*. Springer, 2011, pp. 319–339.

[29] M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of learning with errors," Cryptology ePrint Archive, Report 2015/046, 2015, https://eprint.iacr.org/2015/046.

[30] H. Chen, K. E. Lauter, and K. E. Stange, "Attacks on the search-rlwe problem with small errors," *CoRR*, vol. abs/1710.03739, 2017. [Online]. Available: http://arxiv.org/abs/1710.03739

[31] Y. Chen and P. Q. Nguyen, "Bkz 2.0: Better lattice security estimates," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2011, pp. 1–20.

[32] N. P. Smart and F. Vercauteren, "Fully homomorphic simd operations," *Designs, codes and cryptography*, vol. 71, no. 1, pp. 57–81, 2014.

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

[34] J. B. Lima, D. Panario, and Q. Wang, "A karatsuba-based algorithm for polynomial multiplication in chebyshev form," *IEEE Transactions on Computers*, vol. 59, no. 6, pp. 835–841, June 2010.

[35] C. Gentry and N. P. Smart, *Homomorphic Evaluation of the AES Circuit*, 2015. [Online]. Available: http://eprint.iacr.org/2012/099.pdf

[36] S. Halevi and V. Shoup, "Helib - an implementation of homomorphic encryption," https://github.com/shaih/HElib/, 2013.

[37] J. Katz, A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 1996.

[38] W. Hart, F. Johansson, and S. Pancratz, "FLINT: Fast Library for Number Theory," 2013, version 2.4.0, http://flintlib.org.

[39] "The GNU Multiple Precision Arithmetic Library," 2016, version 6.1.2, http://gmplib.org.

[40] H. A. Alkhzaimi and M. M. Lauridsen, "Cryptanalysis of the simon family of block ciphers," Cryptology ePrint Archive, Report 2013/543, 2013, https://eprint.iacr.org/2013/543.

[41] M. J. Foster, "Accelerating Homomorphic Encryption in the Cloud Environment through High-Level Synthesis and Reconfigurable Resources," 2017. [Online]. Available: http://scholarworks.rit.edu/theses

[42] 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.

[43] E. Öztürk, Y. Doröz, B. Sunar, and E. Savas, "Accelerating somewhat homomorphic evaluation using fpgas." *IACR Cryptology ePrint Archive*, vol. 2015, p. 294, 2015.

[44] W. Wang, Z. Chen, and X. Huang, "Accelerating leveled fully homomorphic encryption using gpu," in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 2800–2803.

[45] T. Pöppelmann, M. Naehrig, A. Putnam, and A. Macias, "Accelerating homomorphic evaluation on reconfigurable hardware," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2015, pp. 143–163.

[46] Y. Doröz, E. Öztürk, and B. Sunar, "Accelerating fully homomorphic encryption in hardware," *IEEE Transactions on Computers*, vol. 64, no. 6, pp. 1509–1521, 2015.