Applying Fully Homomorphic Encryption: Practices and Problems

by

Yao Chen

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2021

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Radha Poovendran Professor, Dept. of Electrical and Computer Engineering, University of Washington
Supervisor:	Gong Guang Professor, Dept. of Electrical and Computer Engineering, University of Waterloo
Internal Member:	Oussama Damen Professor, Dept. of Electrical and Computer Engineering, University of Waterloo
Internal Member:	Mahesh Tripunitara Professor, Dept. of Electrical and Computer Engineering, University of Waterloo
Internal-External Member:	David Jao Professor, Dept. of Combinatorics and Optimization, University of Waterloo

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Fully homomorphic encryption (FHE) has been regarded as the "holy grail" of cryptography for its versatility as a cryptographic primitive and wide range of potential applications. Since Gentry published the first theoretically feasible FHE design in 2008, there has been a lot of new discoveries and inventions in this particular field. New schemes significantly reduce the computational cost of FHE and make practical deployment within reach. As a result, FHE schemes have come off the paper and been explored and tested extensively in practice. However, FHE is made possible with many new problems and assumptions that are not yet well studied. In this thesis we present a comprehensive and intuitive overview of the current applied FHE landscape, from design to implementation, and draw attention to potential vulnerabilities both in theory and in practice. In more detail, we show how to use currently available FHE libraries for aggregation and select parameters to avoid weak FHE instances.

Acknowledgements

My deepest gratitude goes to my supervisor, Prof. Guang Gong, who provided me with tremendous academic guidance and emotional support during the writing of this thesis. The amount of care and advocacy for her students is unmatched in any of my experience. I feel very lucky to have studied under her guidance and worked with her.

I would like to thank my committee members, Prof. Radha Poovendran, Prof. Oussama Damen, Prof. Mahesh Tripunitara, and Prof. David Jao, for their time and valuable feedback. Many of them have also taught me in various courses, and I deeply thank them for the knowledge.

I would like to thank Dr. Kalikinkar Mandal for helping me collect latest advancements in FHE appication and standardization.

I have had many collaborations and informative exchanges with other colleagues in the ComSec Lab. Specifically, I would like to thank Prof. Mark Aagaard for being a kind mentor with different expertise, Dr. Yin Tan for sharing his knowledge in mathematics, and Dr. Nuša Zidarič for her tea and friendly support.

Outside of the ComSec Lab, I have had the privilege to collaborate with Prof. Shuhong Gao and Dr. Benjamin Case on our RLWE attack.

I would also like to thank my wife, Lys, for her unconditional support and encouragement.

Dedication

To Lys and Amy with deepest love.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 A Brief History of (Fully) Homomorphic Encryption

"Privacy homomorphism", antecedent of the modern notion of fully homomorphic encryption (FHE), was proposed in 1978 by Rivest *et al.* [88], only one year after their publication of RSA, one of the first public-key cryptosystems. Rather than providing a solution, Rivest *et al.* posed a question on how to resolve the conflict between user privacy and data and computation delegation demands. They envisioned such an encryption scheme that makes computation possible directly on its ciphertexts. At the time of its proposal, they might not have foreseen that it took nearly three decades for this idea to manifest its feasibility and that its applications extend so far beyond cloud computing that it is often treated as a powerful cryptographic primitive.

Many existing public-key cryptosystems already have certain homomorphic properties. For example, the RSA cryptosystem [89] is multiplicatively homomorphic because $(x_1 \cdot x_2)^e = x_1^e \cdot x_2^e$; similarly, the ElGamal cryptosystem [47] is multiplicatively homomorphic; and the Goldwasser-Micali cryptosystem [57] is homomorphic with respect to exclusive-or. There are even schemes that support both modular addition and multiplication, as long as the number of operations is below a predetermined threshold. However, none of them is fully homomorphic, supporting the computation of arbitrary functions on ciphertexts.

The search for an FHE scheme persisted until Gentry's breakthrough in 2009 [51]. Gentry's design appears to have taken a circuitous route. He starts from a somewhat homomorphic encryption scheme – namely, one that can only support functions of limited

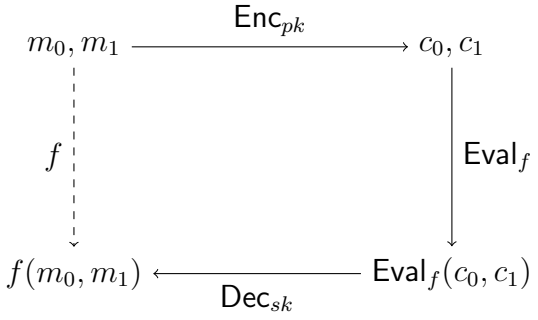

Figure 1.1: A paradigm of homomorphic encryption illustrated with a bivariate function f over plaintexts, and Eval_f is a function performed over ciphertexts, defined by $\mathsf{Eval}(pk, f, c_0, c_1)$.

complexity; he bases his scheme on ideal lattices, since they are essentially polynomial rings, and addition and multiplication – two operations sufficient for the construction of any functions – are inherently to polynomial rings; finally, he expresses the operations with circuit language – in fact, if implemented in binary, the operations are indeed XOR (addition) and AND (multiplication) gates.

Gentry's major innovation that makes FHE possible is the so-called "bootstrapping" process, which he describes as the homomorphic evaluation of the scheme's own decryption function. Further details of bootstrapping is discussed in Section 3.3.

Since Gentry's design, there have been many new and more efficient FHE schemes, particularly based on Learning with Errors (LWE) and Ring-LWE. However, all known FHE schemes follow the same approach of using noise to achieve security, letting noise accumulate through operations, and bootstrapping to remove excessive noise so that computation can continue indefinitely. Even designing FHE schemes with similar techniques may still be a tricky problem, as the failure of BL scheme [24], which tries to adapt Gentry's design to code-based schemes, shows.

Vaikuntanathan classfies the development of homomorphic encryption schemes into four generations [95], with Gentry's design as the first one.

The first notable second-generation FHE scheme is BGV [25]. Based on Ring-LWE, the BGV scheme is much easier to understand and implement. Its efficiency is also a significant improvement over Gentry's initial design, both theoretically and practically. It incorporates some novel techniques such as modulus switching, which slows down the noise built-up, and key switching, which allows the change of secret key without decryption and can be utilized to facilitate multi-party computation. Many later schemes were developed

with the same techniques. The BGV scheme is implemented in the open-source library, HElib [60]. Since its publication, it has been widely used, tested, and actively maintained.

Another second-generation FHE scheme with efficiency comparable to BGV is FHEW, designed by Ducas *et al.* [44]. Different from typical schemes, FHEW supports NAND gates and NAND gates only (NAND gate itself forms a universal set, meaning any combinatorial circuits can be redesigned into an equivalent one with only NAND gates). A single NAND operation is extremely efficient in FHEW, but a bootstrapping step has to follow each of them and thus putting its efficiency into the same level as BGV.

There is no clear cut between the second- and third-generation FHE schemes, but improvements are constantly made to their efficiency. For example, Fan and Vercauteren developed the FV scheme that is similar to but faster than BGV, and the test of its practical use is also in progress – an implementation, SEAL, of a slightly modified FV scheme was brought out by a Microsoft Research group [32]. Their earlier versions implemented YASHE (Yet Another Somewhat Homomorphic Encryption) scheme is another example with better efficiency, and it is designed based on NTRU.

There are also faster FHE schemes built upon the idea of FHEW. For example, Biasse and Ruiz relaxed the parameter and gate restrictions of original FHEW [19]; Chillotti *et al.* reduced the bootstrapping time by over 90% through the use of a different algebraic structure [38, 39]; Gao *et al.* further reduced its storage and communication overhead [50, 30].

Recently, with its optimization of homomorphic encryption on real numbers, the CKKS scheme [37], considered as the fourth generation of FHE, finds popularity in the applications of machine learning [66, 80].

1.2 Applications of Homomorphic Encryption

Applications of homomorphic encryption are plentiful. As Rivest pointed out, homomorphic encryption can lead to secure cloud computing—cloud servers are able to compute on ciphertexts without decrypting them, and therefore the computation can be done without leaking the knowledge about data or results. Apart from that, since such privacy-preserving computation is closely related to cryptographic notions such as zero-knowledge proof, many such primitives can be constructed from homomorphic encryption schemes.

Note that the original FHE with only one set of keys is insufficient for multi-party computation (MPC). Hence a new notion of *multi-key FHE* was created [78, 43]. A representative of such schemes, LTV, is described in Section 4.1.3.

Specific discussions of applying FHE to practical scenarios mainly involve MPC and privacy-preserving data mining/machine learning. While MPC protocols with FHE are mostly general, data-mining and machine-learning algorithms must be specially optimized according to the selected scheme. Examples of the former include [78, 43, 14], and the latter [66, 80, 42, 64]. Recent efforts pushing FHE into practice from the academic society and industrial practitioners have resulted in the process of homomorphic encryption standardization [9].

1.3 Organization

This thesis is organized as follows. Chapter 2 provides necessary mathematical foundations for the discussion of FHE througout the whole thesis. Chapter 3 lays out the general framework that every FHE design has followed. Chapter 4 explores some major schemes in depth. Chapter 5 provides latest development of FHE in practice. Our contribution are Chapters 6 and 7. Finally, Chapter 8 concludes the thesis and provide some outlook on the future.

Chapter 2

Preliminary

Lattice-based cryptography has gained much popularity in recent years. On one hand, it has proven to be useful in the construction of new cryptographic tools, in particular, quantum attack resistant cryptosystems and homomorphic encryption schemes. On the other hand, cryptanalysis based on lattice theory is very effective on some encryption schemes, even though they may not be originally described in lattice language (see, for example, lattice attacks against RSA [75]).

Lattice problems are also deeply connected with coding problems. They are similar in mathematical descriptions and properties (e.g., quantum resistance). The reduction of learning with errors problem further reveals their close relationship in hardness.

2.1 (Integer) Lattices

In cryptography, a lattice refers to a discrete additive subgroup of a Euclidean space.

Definition 1 (Lattice)**.** Let $B = \{ \mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{k-1} \mid \mathbf{v}_i \in \mathbb{R}^n, 0 \leq i \leq k-1 \}$ be a set of k linearly independent vectors, then

$$\Lambda(B) = \left\{ \sum_{i=0}^{k-1} a_i \mathbf{v}_i \;\middle|\; a_i \in \mathbb{Z}, 0 \leq i \leq k-1 \right\} \tag{2.1}$$

is called the *lattice generated by basis* B. The *dimension* of $\Lambda(B)$ is n, and the *rank* of $\Lambda(B)$ is k.

We usually consider full-rank lattices, where $k = n$ in the above definition.

Viewing lattice $\Lambda(B)$ as an additive subgroup of $\mathbb{R}^n$, we can consider the quotient group $\mathbb{R}^n/\Lambda(B)$. It is called a fundamental parallelepiped of the lattice; its volume is an invariant of the lattice regardless of the basis and will play an important role in the hardness of lattice problems of our interest.

Definition 2 (Fundamental parallelepiped)**.** Given a basis $B = \{ \mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{k-1} \}$ of a lattice Λ, then

$$\mathcal{F}(\Lambda) = \left\{ \sum_{i=0}^{k-1} b_i \mathbf{v}_i \;\middle|\; b_i \in [0,1), 0 \le i \le k-1 \right\} \tag{2.2}$$

is called the *fundamental parallelepiped of* Λ *with respect to* B. The volume of $\mathcal{F}(\Lambda)$ is called the *determinant of* Λ and denoted $\det(\Lambda)$.

2.1.1 Successive Minima

Associated with a lot of lattice problems and defining their difficulties are the *successive minima* of lattices, in particular, the first one, $\lambda_1(\Lambda)$ – it is so frequently used that the subscript is often dropped and its notation written just $\lambda(\Lambda)$.

Definition 3 (Successive Minima)**.** Given a lattice Λ with basis $B = \{ \mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{k-1} \}$, $\lambda_i(\Lambda)$ is defined as the length of the shortest vector(s) in Λ such that Λ-vectors no longer than $\lambda_i(\Lambda)$ span an i-dimensional sublattice:

$$\lambda_i(\Lambda) = \inf \{ r \mid \dim(\mathrm{span}(\Lambda \cap \mathbf{B}(0, r))) \ge i \},$$

where $\mathbf{B}(\mathbf{0}, r)$ is the closed ball of radius r centered at $\mathbf{0}$.

In particular, $\lambda_1(\Lambda)$ equals to the length of the shortest nonzero vector(s) in Λ:

$$\lambda_1(\Lambda) = \min \{ \|\mathbf{v}\| \mid \mathbf{v} \in \Lambda, \mathbf{v} \neq \mathbf{0} \}.$$

Apparently, $0 < \lambda_1(\Lambda) \le \lambda_2(\Lambda) \le \cdots \le \lambda_n(\Lambda)$ in any lattice Λ of rank n.

Bounds of successive minima are given in two Minkowski theorems.

Theorem 2.1.1 (Minkowski's First Theorem)**.** *For a full-rank lattice* Λ *with rank* n*,*

$$\lambda_1(\Lambda) \le \sqrt{n} \left(\det(\Lambda)\right)^{1/n}$$

Theorem 2.1.2 (Minkowski's Second Theorem)**.** *For a full-rank lattice* Λ *with rank* n*,*

$$\left(\prod_{i=1}^{n} \lambda_i(\Lambda) \right)^{1/n} \le \sqrt{n} \left(\det(\Lambda)\right)^{1/n}$$

2.2 Ideal Lattices

Ideal lattices are a special type of lattices with extra algebraic structures. The additional structure usually results in a much more efficient scheme with smaller expansions, but it can also introduce new vulnerabilities.

An ideal lattice is obtained by mapping the elements of an ideal to $\mathbb{R}^n$ by an additive homomorphism. Naturally, not every ring can facilitate such a mapping. Since we are also mostly concerned with integers and polynomials, the structure of our interest is the *ring of integers* in an *algebraic number field*. Details of the theory about algebraic number fields and ring of integers can be found on any standard algebraic number theory textbook, for example, [45]. Here we briefly review definitions and results directly pertaining to ideal lattices.

A complex number $\omega \in \mathbb{C}$ is called *algerbraic* if it is a root of some $p(x) \in \mathbb{Q}[x]$. The *minimal polynomial* of ω is the *monic* rational polynomial $m(x) \in \mathbb{Q}[x]$ with the lowest possible degree.

Definition 4 ((Algebraic) number field)**.** Let $\omega \in \mathbb{C}$ be algebraic, then

$$K = \mathbb{Q}[\omega] = \{\, a + b\omega \mid a, b \in \mathbb{Q} \,\}$$

is an *(algebraic) number field*.

K is a field, $\mathbb{Q} \subset K \subsetneq \mathbb{C}$. If $m(x) \in \mathbb{Q}[x]$ is the minimal polynomial of ω, then

$$K \cong \mathbb{Q}[x]/(m(x)).$$

All the elements in a number field K are algebraic numbers, which have minimal polynomials with rational coefficients; if an algebraic number has integer-coefficient minimal polynomial, then it is called an *algebraic integer*. All the algebraic integers of K form its *ring of integers*.

Definition 5 (Ring of integers)**.** The ring of integers $\mathcal{O}_K$ of a number field K is the set

$$\mathcal{O}_K = \left\{ \alpha \in K \;\middle|\; \begin{array}{c} \text{there exist } c_0, c_1, \ldots, c_{n-1} \in \mathbb{Z} \text{ such that} \\ \alpha^n + \sum_{i=0}^{n-1} c_i \alpha^i = 0 \end{array} \right\}.$$

2.2.1 Number Fields and Rings of Integers

Due to the precision limit of computers, lattice problems used in cryptography are usually defined over a finite ring consisting of either integers or integer-coefficient polynomials.

Polynomial rings are of particular interest in the design of LWE-type encryption schemes because they are associated with *ideal lattices*, which is often used to shorten the lengths of the key and ciphertext and improve the efficiency of LWE-based schemes [91]. Ideal lattices will be discussed in Section 2.2. Such improvement usually involves the use of *Chinese Remainder Theorem* (CRT).

Basic Concepts of Ring Theory

Rings are a type of algebraic structures modelled after integers.

Definition 6 (Ring)**.** A ring is a set R with two binary operations $+$ and $\cdot$ that satisfies the following properties:

1. Additive subgroup: $(R, +)$ is an abelian group; the additive identity is usually denoted 0;
2. Multiplication: $(R, \cdot)$ satisfies all group axioms except invertibility; the multiplicative identity is usually denoted 1;
3. Distributive property: for any $a, b, c \in R$,

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c). \tag{2.3}$$

The operations $+$ and $\cdot$ are generalized from *addition* and *multiplication* of integers, and we will call them so for simplicity. Note that since $(R, \cdot)$ is not required to satisfy invertibility property, elements in R, except 1, may not have inverses. A ring where all elements except 0 are multiplicatively invertible is called a field. Henceforth when we call a structure a ring, it always contains at least one element other than 0 that is not invertible. We also restrict our scope to commutative rings, i.e., in addition to the above properties, multiplication is also commutative, since there has not been non-commutative rings in such applications.

Although $(R, \cdot)$ is not a group, there are still interesting substructures pertaining to multiplication. One of such structures is *ideals*, which are derived from products and factors in integers.

Definition 7 (Ideal)**.** An ideal of ring R is a subset I such that

1. Closure on addition: $(I, +)$ is a group;
2. Absorption on multiplication: for any element $a \in R$ and $binI$, $ab \in I$.

Some ideals are generated by a single element. They are called *principal ideals*. For example, $(a) = aR = \{\, ab \mid b \in R \,\}$ is a principal ideal of R generated by a.

We can add or multiply ideals by adding or multiplying their elements. For any two ideals $I, J \subset R$,

$$I + J = \{\, a + b \mid a \in I, b \in J \,\},$$
$$IJ = \{\, ab \mid a \in I, b \in J \,\}.$$

The results of ideal addition and multiplication are still ideals in the same ring. If an ideal $I \subset R$ cannot be written as the product of two ideals not involving itself, then it is a *prime ideal*.

Definition 8 (Prime ideal)**.** A proper ideal $P \subsetneq R$ is *prime* if for any elements $a, b \in R$, their product ab falling in P implies that at least one of a and b is in P.

We can rephrase some properties of the multiplication of integers with the language of ideals. Each integer a can be seen as a principal ideal (a) generated by itself; the multiplication between two integers a, b are then equivalent to the ideal multiplication of (a) and (b); therefore, a number $p > 0$ is prime if and only if (p) is a prime ideal in $\mathbb{Z}$. Note that (0) is a prime ideal but 0 is not a prime; $(1) = R$ cannot be decomposed, but it is not a proper ideal in R and hence not a prime ideal. There is a one-to-one correspondence between prime numbers and *maximal ideals* in Z. A maximal ideal $M \subset R$ is defined as having no other ideals between R and itself.

Definition 9 (Maximal ideal)**.** If $M \subset I \subset R$ and I is an ideal in R, then either $I = M$ or $I = R$.

An ideal $I \subset R$ can induce a *quotient ring* R/I, which consists of the cosets of I. Operations in the quotient ring R/I are then operations on R modulo I. Quotient rings of the form $R[x]/(f(x), n)$ are often used as plaintext and ciphertext spaces in encryption schemes involving rings because of multiple benefits: the ideal $(f(x), n)$ can induce a finite set, which makes implementation more convenient; the operations are still easy to define and compute; and performance can usually be improved with the decomposition of such rings.

Chinese Remainder Theorem on Integers Modulo n

Theorem 2.2.1 (CRT on integers modulo n)**.** *If $n = pq$, where $1 < p, q$, and p is coprime with q, then $\mathbb{Z}_n$ can be decomposed as*

$$\mathbb{Z}_n \cong \mathbb{Z}_p \times \mathbb{Z}_q.$$

This decomposition is an isomorphism $\psi : \mathbb{Z}_n \to \mathbb{Z}_p \times \mathbb{Z}_q$, and there are efficient algorithms (Euclidean algorithms) to compute the mappings ψ and ψ^{-1}. The implication is that, for $a, b \in \mathbb{Z}_n$, we can find $(a_p, a_q), (b_p, b_q) \in \mathbb{Z}_p \times \mathbb{Z}_q$ and have

$$\begin{aligned} \psi(a + b) &= \psi(a) + \psi(b), \\ \psi(a \cdot b) &= \psi(a) \cdot \psi(b), \end{aligned}$$

and vice versa. Such decomposition has a lot of applications in cryptography, among which are the RSA and Rabin cryptosystems. Since a typical n is large, operations on $\mathbb{Z}_n$ are usually more costly to perform than those on $\mathbb{Z}_p$, $\mathbb{Z}_q$, and the conversion overhead combined. Therefore, we can speed up the computation on $\mathbb{Z}_n$ by taking the decomposition and perform the operations on $\mathbb{Z}_p$ and $\mathbb{Z}_q$ instead.

Note that it is not required that either p or q is a prime in CRT, and the decomposition can certainly be generalized to the product of more than two factors by using the theorem repeatedly.

Chinese Remainder Theorem on Polynomial Rings

For any ring R, we can construct a polynomial ring $R[x]$. $R[x]$ consists all the formal sums

$$a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n,$$

where $n \geq 0$ and $a_i \in R$. Operations on $R[x]$ are polynomial addition and multiplication.

We have CRT on polynomial rings similar to that on integers.

Theorem 2.2.2 (CRT on polynomial rings)**.** *Let $f(x), g(x)$, and $h(x)$ be polynomials over R. If $f(x) = g(x)h(x)$, where both $g(x)$ and $h(x)$ have degree > 0, and $g(x)$ and $h(x)$ are coprime with each other, then*

$$R/(f(x)) \cong R/(g(x)) \times R/(h(x)).$$

Likewise, there is an isomorphism $\psi : R/(f(x)) \to R/(g(x)) \times R/(h(x))$ that can be efficiently computed.

Both the integer and polynomial ring CRTs above involve only principal ideals. This is sufficient for integers, since all ideals in $\mathbb{Z}$ are principal; but polynomial rings can contain non-principal ideals, for example, $(2, x) \subset \mathbb{Z}[x]$ cannot be generated by any single element. In that case, we need to generalize the theorem with ideals.

Coprime integers or polynomials correspond with *comaximal* ideals. Two ideals $I, J \subset R$ are comaximal if $I + J = R$. If I and J are comaximal, then $IJ = I \cap J$.

The theorem can then be written as follows.

Theorem 2.2.3 (CRT on polynomial rings)**.** *Let I, J be comaximal ideals in R, then*

$$R/(IJ) \cong R/I \times R/J.$$

Chinese Remainder Theorem on Dedekind Domains

The most generalized form of CRT used throughout this thesis is over *Dedekind domains*. We briefly introduce some definitions here in order to properly define a Dedekind domain, but in practice we always use rings associated with number fields, and the ring of integers of any number field is a Dedekind domain, guaranteed by Theorem 2.2.4, therefore the properties always apply. Further expositions on these subjects can be found on many algebra textbooks, such as [45].

A module can be seen as a generalization of vector spaces. A vector space is based on a field, while a module is based on a ring.

Definition 10 (R-module)**.** Let R be a commutative ring with identity 1. An *R-module* is an abelian group $(M, +)$ with an action $R \times M \to M$ denoted by rm, for all $r \in R$ (scalar multiplication), which satisfies

1. $(r + s)m = rm + sm$, for all $r, s \in R, m \in M$;
2. $(rs)m = r(sm)$, for all $r, s \in R, m \in M$;
3. $r(m + n) = rm + rn$, for all $r \in R, m, n \in M$; and
4. $1m = m$, for all $m \in M$.

Definition 11 (Noetherian ring)**.** A *Noetherian ring* R is a ring whose every ascending chain of ideals is finite, i.e., for any chain of ideals

$$I_1 \subset I_2 \subset \cdots \subset R,$$

there is a positive integer N such that for every $k \geq N$, $I_k = I_N$.

Definition 12 (Noetherian module)**.** Let R be a Noetherian ring. A finitely generated R-module M is a *Noetherian module*, i.e.,

$$M = R[\alpha_1, \ldots, \alpha_n]$$

for a finite integer n.

Noetherian modules are analogous to vector spaces of finite dimensions.

Definition 13 (Integral domain)**.** An *integral domain* D is a commutative ring with identity 1 and no zero divisors, i.e., if $ab = 0 \in R$, then either $a = 0$ or $b = 0$.

Definition 14 (Integral closure)**.** An integral domain D is *integrally closed* if

$$\left\{ \alpha = \frac{a}{b} : a, b \in D \;\middle|\; \begin{array}{c} \text{there exist } c_0, c_1, \ldots, c_{n-1} \in D \text{ such that} \\ \alpha^n + \sum_{i=0}^{n-1} c_i \alpha^i = 0. \end{array} \right\} = D.$$

Definition 15 (Dedekind domain)**.** A *Dedekind domain* D is an integral domain that satisfies all of the following conditions:

1. D is Noetherian;
2. D is integrally closed; and
3. Every prime ideal in D is maximal.

Theorem 2.2.4. *The ring of integers, $\mathcal{O}_K$, of a number field K is a Dedekind domain. [82, Chapter 1, Theorem 3.1].*

Theorem 2.2.5 (CRT on Dedekind domains)**.** *Let I, J be comaximal ideals of Dedekind domain D, then*

$$D/(IJ) \cong D/(I) \times D/(J)$$

2.2.2 Ideal Lattices

Definition 16 (Ideal lattice)**.** Let K be an number field and R its ring of integers. Let $I \subset R$ be an ideal. Let σ be an additive homomorphism

$$\sigma : K \to \mathbb{R}^n,$$

then $\sigma(I) \subset \mathbb{R}^n$ is an *ideal lattice*.

Note that Definition 16 is general. A ring of integers is not necessarily monogenic – namely, it may not be of the form $\mathbb{Z}[x]/(f(x))$. In that case, we usually resort to using a subring that is monogenic to simplify the handling of elements; however, the properties of the corresponding number field and mappings still hold. Many newer scheme designs use cyclotomic fields for enhanced performance; their ring of integers are always monogenic [82, Chapter 1, Proposition 10.2].

For every problem on a general lattice, we can define its counterpart on an ideal lattice. For example,

Definition 17 (Ideal-SVP)**.** Given I, an ideal of the ring of integers of a number field K, and σ, an additive homomophism from K to $\mathbb{R}^n$, the ideal-SVP problem is to find an $a \neq 0 \in I$ with the smallest $\|\sigma(a)\|$.

Embeddings of Ideal Lattices

In Definition 16 of ideal lattices, the only restriction on the mapping σ is that it must be an additive homomorphism. Such a mapping does not necessarily capture the multiplicative structures of number fields. Therefore, some mappings better preserve the structures and properties in an ideal lattice. This turns out to be an important factor in noise management in the context of homomorphic encryption, since noise is measured by the norm defined over the lattice. Notably, two types of embeddings have been widely used and discussed: *coefficient embeddings* and *canonical embeddings* (the latter are called Minkowski embeddings in some literature).

The ring R of consideration in homomorphic encryption is of the form $\mathbb{Z}[x]/(f(x))$, where $\deg(f) = n$. Each element a in the ring can be written as a polynomial

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

A coefficient embedding simply maps the polynomial a to a vector, $[\, a_0, a_1, \ldots, a_{n-1} \,]$, of its coefficients. An ideal lattice constructed with coefficient embeddings, therefore, has points

whose coordinates are determined by the coefficients. In contrast, a canonical embedding will use the evaluation of $a(x)$ on each root of $f(x)$. Assuming $f(x)$ has r real roots, $\alpha_1, \ldots, \alpha_r$, and s complex roots, $\alpha_{r+1}, \ldots, \alpha_{r+s}$, then we define the embedding σ as

$$\sigma(a) = [\, \sigma_1(a), \ldots, \sigma_r(a), \mathrm{Re}(\sigma_{r+1}(a)), \mathrm{Im}(\sigma_{r+1}(a)), \ldots, \mathrm{Re}(\sigma_{r+s}(a)), \mathrm{Im}(\sigma_{r+s}(a)) \,],$$

where $\sigma_i(a) = a(\alpha_i)$. One way to understand canonical embeddings is to view them as a result of Chinese Remainder Theorem; alternatively, when $f(x)$ is a cyclotomic polynomial, the mapping σ is equivalent to a discrete Fourier transform.

Whenever convenient, canonical embeddings are preferred over coefficient embeddings for two reasons. Firstly, polynomial multiplication is faster since there are $O(n)$ number multiplications, as opposed to $O(n^2)$ in naive implementation of a coefficient-embedding multiplication, and without the need of reducing against $f(x)$. Secondly, the original hardness result for RLWE is based on sampling noise from a joint, n-dimensional Gaussian distribution with diagonal covariate matrix on canonical embedding. Sampling from such a distribution is equivalent to sampling from n independent univariate Gaussian distributions; changing from a canonical embedding to a coefficient embedding necessarily introduces distortion of through a linear transformation [72]. However, canonical embeddings of general polynomial rings can contain real numbers, which are not favoured in implementations due to precision and efficiency problems. As a result, many schemes restrict their ciphertext and plaintext spaces to quotient rings generated by cyclotomic polynomials to take advantage of their neat decompositions.

Prime Decomposition of Cyclotomic Rings

When the schemes are implemented on a computer, we often prefer embeddings that can be represented by integers to those relying on real numbers (floating point) for the ease of implementation and efficiency. Since the ring of integers $\mathcal{O}_K$ of a number field K is a Dedekind domain, by Chinese Remainder Theorem, there is an isomorphism between the decomposition of the ideal $(q) \subset \mathcal{O}_K$, generated by a prime integer $q \in \mathbb{Z}$, and the decomposition of $\mathcal{O}_K/q\mathcal{O}_K$:

$$\mathcal{O}_K/q\mathcal{O}_K = \mathcal{O}_K/(q) \cong \mathcal{O}_K/I_1^{e_1} \times \cdots \times \mathcal{O}_K/I_n^{e_n},$$

where $I_1, \ldots, I_n$ are prime ideals and

$$(q) = I_1^{e_1} \cdots I_n^{e_n}.$$

Recall that prime ideals are maximal in $\mathcal{O}_K$, I_i's are pairwise comaximal, and so are their powers; therefore the application of Theorem 2.2.5 here is appropriate. Furthermore,

the number and size of those quotient rings can be related to the degree of irreducible factors of $\Phi_n(x)$ modulo q. As a result, Theorem 2.2.6 [85, Result 1.2.7] gives the prime decompositions in cyclotomic fields. Similar results can be found in standard algebraic number theory books (for example [82]). We shall see an application of result in the parameter selection process of the open-source FHE package "HElib" for an optimized performance.

Theorem 2.2.6. *Let p be a prime in $\mathbb{Z}$ and $\Phi_n(x)$ be the n-th cyclotomic polynomial. ζ_n is a primitive n-th root of unity in $\mathbb{C}$. Write $n = p^e n'$ with $\gcd(n', p) = 1$, then the prime ideal decomposition of (p) over $\mathbb{Q}(\zeta_n)$ is*

$$(p) = (I_1 \cdots I_k)^{\varphi(p^e)},$$

where $I_1, \ldots, I_k$ are distinct prime ideals and $k = \varphi(n')/ord_{n'}(p)$. If t is an integer not divisible by p and $t \equiv p^s \pmod{n'}$ for some integer s, then the field automorphism $\zeta_n \mapsto \zeta_n^t$ fixes the ideals I_i.

2.3 Hard Problems from Lattices and Rings

2.3.1 Lattice Problems

Lattice problems in cryptography are usually variants of *Shortest Vector Problem* (SVP) and *Closest Vector Problem* (CVP). Both SVP and CVP over general lattices are NP-hard [16, 2], but for some variants, the exact computational complexity remains an open problem.

To describe the problems, we need a measurement of the length of vectors. By convention, we use Euclidean norm (l_2 norm) $\|\cdot\|_2$, unless otherwise specified. For a vector $\mathbf{a} = [\, a_0, a_1, \ldots, a_{n-1} \,]$, its Euclidean norm is

$$\|\mathbf{a}\|_2 = \sqrt{\sum_{i=0}^{n-1} a_i^2}.$$

Definition 18 (SVP)**.** Given a basis $B = \{\, \mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{k-1} \,\}$, the SVP problem is to find the shortest nonzero vector in the lattice $\Lambda(B)$, whose length is $\lambda(\Lambda)$, i.e.,

$$\arg\min_{\mathbf{0} \neq \mathbf{w} \in L(B)} \|\mathbf{w}\| \tag{2.4}$$

Definition 19 (CVP)**.** Given a basis $B = \{ \mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{k-1} \}$ and a vector $\mathbf{u} \in \mathbb{R}^n$, the CVP problem is to find a vector $\mathbf{w} \in \Lambda(B)$ with the shortest distance to $\mathbf{u}$, i.e.,

$$\arg\min_{\mathbf{w} \in L(B)} \|\mathbf{u} - \mathbf{w}\| \tag{2.5}$$

Both SVP and CVP have approximation versions ($\mathsf{apprSVP}_\delta$ and $\mathsf{apprCVP}_\delta$) and gap versions (GapSVP_β and GapCVP_β). Here $\delta = \delta(k)$ and $\beta = \beta(k)$ are fixed functions of k, the dimension of the lattice in a problem instance.

2.4 Learning with Errors

Regev described the learning with errors (LWE) problem and discovered its relation to random codes and to standard lattice problems such as GapSVP [86]. The LWE problem asks the following question: given N *approximate* linear equations of vector $\mathbf{s}$

$$\begin{aligned} \langle \mathbf{s}, \mathbf{a}_1 \rangle &\approx_\chi b_1 \pmod p, \\ \langle \mathbf{s}, \mathbf{a}_2 \rangle &\approx_\chi b_2 \pmod p, \\ &\vdots \\ \langle \mathbf{s}, \mathbf{a}_N \rangle &\approx_\chi b_N \pmod p, \end{aligned}$$

where $x \approx_\chi y$ means that $y = x + e$ with e a random noise drawn from the distribution χ, can we solve the equations for $\mathbf{s}$? It turns out that if LWE is efficiently solvable then hard lattice problems such as GapSVP can be solved efficiently, too.

Definition 20 ((Search) LWE)**.** Given integers $n = n(\lambda)$ and $q = q(\lambda) \geq 2$, and distribution $\chi = \chi(\lambda)$ over $\mathbb{Z}$. Fix secret $\mathbf{s} \leftarrow_\chi \mathbb{Z}_q^n$, obtain N samples $[\, \mathbf{a}_i \mid b_i \,], 0 \leq i \leq N-1$ by letting $\mathbf{a}_i \leftarrow_U \mathbb{Z}_q^n$, $e_i \leftarrow_\chi \mathbb{Z}_q$, and computing $b_i = \mathbf{a}_i \mathbf{s}^\top + e_i$. The samples form an $N \times (n+1)$ matrix. The (search) LWE problem is to find the secret $\mathbf{s}$.

An algorithm $\mathcal{A}$ is said to (N, t, ε)-solve the (search) LWE problem if it runs for at most time t, takes N samples and outputs the correct $\mathbf{s}$ with probability ε.

Note that the running time is at least $O(N)$ (input size), which means an upper bound of running time implies an upper bound of the number of samples. Specifically, an algorithm that efficiently solves LWE problem, which runs in polynomial time, must take only polynomially many samples. Thus the number of samples is often omitted in some descriptions of LWE. Here we use both interchangeably.

The belief that LWE is intractable can be established by two observations. First of all, the search LWE problem is equivalent to the decoding of a random code. Suppose there are N samples $[\,\mathbf{a}_0 \mid b_0\,], \ldots, [\,\mathbf{a}_{N-1} \mid b_{N-1}\,]$, we have the relationship

$$\begin{bmatrix} b_0 \\ \vdots \\ b_{N-1} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_0 \\ \vdots \\ \mathbf{a}_{N-1} \end{bmatrix} \mathbf{s}^\top + \begin{bmatrix} e_0 \\ \vdots \\ e_{N-1} \end{bmatrix} = A\mathbf{s}^\top + \mathbf{e}^\top.$$

If we view A as the generator matrix, $\mathbf{s}$ as the message, and $\mathbf{e}$ as the error vector, then clearly an algorithm that decodes a random linear code could obtain $\mathbf{s}$ and hence solve (search) LWE.

The second observation is the reduction of LWE to worst-case lattice problems, e.g., GapSVP and SIVP. The first such reduction was contributed by Regev but it involves quantum algorithm [86]; Peikert came up with a classical one that only works for large q (exponential of n) [83]; an average-case classical reduction was finally done by Brakerski *et al.* in 2013 [26].

Definition 21 (Decisional LWE)**.** Given integers $n = n(\lambda)$ and $q = q(\lambda) \geq 2$, and distribution $\chi = \chi(\lambda)$ over $\mathbb{Z}$, the decisional LWE problem is to distinguish the following two distributions of $[\,\mathbf{a}_i \mid b_i\,]$:

1. Uniform distribution: $[\,\mathbf{a}_i \mid b_i\,] \leftarrow_U \mathbb{Z}_q^{n+1}$
2. Fix secret $\mathbf{s} \leftarrow_U \mathbb{Z}_q^n$, compute $[\,\mathbf{a}_i \mid b_i\,]$ by
   $\mathbf{a}_i \leftarrow_U \mathbb{Z}_q^n$,
   $e_i \leftarrow_\chi \mathbb{Z}_q$,
   $b_i = \mathbf{a}_i\mathbf{s}^\top + e_i$

It has been shown that the search and decisional versions of LWE are equivalent for $q = p^e$, where p is a prime, in the sense that search LWE can be reduced to decisional LWE with only polynomial overhead [15]. We will not distinguish between them in later sections.

The Ring Learning with Errors (RLWE) problem is the extension of the LWE problem on ideal lattices [72]. Combining LWE and RLWE, Brakerski *et al.* also defined the General Learning with Errors (GLWE) for the general version of their BGV homomorphic encryption scheme [25]. GLWE problems with parameter $d = 1, n > 1$ are LWE problems, GLWE problems with parameter $d > 1, n = 1$ are RLWE problems, yet little research has been done for other cases.

There is another name, Polynomial Learning with Errors (PLWE), for specific cases of RLWE [28]. The biggest difference between the two notions is that PLWE uses *coefficient embedding*, while RLWE uses *canonical embedding*.

Definition 22 (RLWE)**.** Given polynomial $f(x)$ of degree $d = d(\lambda)$, integer $q = q(\lambda)$, ring $R = \mathbb{Z}[x]/(f(x))$, uniform distribution U, and discrete Gaussian distribution $\chi = \chi(\lambda)$ over R. Let $R_q = R/qR = \mathbb{Z}_q[x]/(f(x))$, the RLWE problem is to distinguish the following two distributions of $[\, a_i \mid b_i \,]$:

1. Uniform distribution: $[\, a_i \mid b_i \,] \leftarrow_U R_q^2$;
2. Fix secret $s \leftarrow_U R_q$, compute $[\, a_i \mid b_i \,]$ by
$$\begin{array}{rcl} a_i & \leftarrow_U & R_q \\ e_i & \leftarrow_\chi & R_q \\ b_i & = & a_i \cdot s + e_i \end{array}.$$

Definition 23 (GLWE)**.** Given integer dimension $n = n(\lambda)$, polynomial $f(x)$ of degree $d = d(\lambda)$, integer $q = q(\lambda)$, uniform distribution U, and distribution $\chi = \chi(\lambda)$ over $R = \mathbb{Z}[x]/f(x)$. Let $R_q = R/qR = \mathbb{Z}_q[x]/f(x)$, the GLWE problem is to distinguish the following two distributions of $[\, \mathbf{a}_i \mid b_i \,]$:

1. Uniform distribution: $[\, \mathbf{a}_i \mid b_i \,] \leftarrow_U R_q^{n+1}$;
2. Fix secret $\mathbf{s} \leftarrow_U R_q^n$, compute $[\, \mathbf{a}_i \mid b_i \,]$ by
$$\begin{array}{rcl} \mathbf{a}_i & \leftarrow_U & R_q^n, \\ e_i & \leftarrow_\chi & R_q, \\ b_i & = & \mathbf{a}_i \mathbf{s}^\top + e_i \end{array}.$$

In some newer articles, for example [7], GLWE is also referred to as the *module* LWE problem.

2.4.1 Noise Distributions

To guarantee a unique solution to LWE-type problems, which is important to many cryptographic applications of LWE, we need the noise e to be sufficiently small. Otherwise, the noise combined with reductions modulo q can produce arbitrary values and, as a result, incorrect decryptions.

Hardness proofs in [86] and [72] rely on discrete Gaussian distributions, but in the practice of FHE, subgaussian distributions, with increased likelihoods for smaller noises,

behave similarly and are often used to give more room for noise growth. Using these distributions may result in a weaker problem, other parameters unchanged; but with the enormous sizes of parameters needed to enable even moderately complicated homomorphic evaluations, they are unlikely to severely compromise security.

2.5 Miscellaneous Utilities and Notations

In most lattice-based encryption schemes, modulo-q residues are taken not in the range of $[0, q)$ but the range of $(-q/2, q/2]$. We will denote such a modular operation by $[\cdot]_q$ for simplicity. Formally,

$$[a]_q = (a \bmod q) - \left\lceil \frac{q}{2} \right\rceil + 1. \tag{2.6}$$

Decomposing polynomial coefficients is one of the effective noise management techniques and has been used across various schemes, and therefore we describe the notations here. Let q be the modulus of the polynomial ring $R_q = \mathbb{Z}_q[x]/f(x)$, where $\deg(f) = n$. A polynomial $a(x) = \sum_{i=0,n-1} a_i x^i \in R_q$ can be written as a vector of its coefficients, $\mathbf{a} = [\, a_0, a_1, \ldots, a_{n-1} \,]$. $\mathsf{Decomp}_{b,q}(\mathbf{a})$ then extracts each digit of a_i under base b.

$$\mathsf{Decomp}_{b,q}(\mathbf{a}) = [\, \mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_{\lfloor \log_b q \rfloor} \,], \tag{2.7}$$

where

$$\mathbf{w}_j = [\, a_0^{(j)}, a_1^{(j)}, \ldots, a_{n-1}^{(j)} \,]$$

such that

$$a_i = \sum_{j=0}^{\lfloor \log_b q \rfloor} a_i^{(j)} b^j.$$

When a decomposed vector, $\mathbf{a}_1$, need to be multiplied by another vector, $\mathbf{a}_2$, we can process the second vector $\mathbf{a}_2$ by multiplying powers of b to it, hence preserving the inner product operation. Namely,

$$\mathsf{PowersOf}_{b,q}(\mathbf{a}) = \left[\, [\mathbf{a}]_q, [b\mathbf{a}]_q, \ldots, \left[b^{\lfloor \log_b q \rfloor} \mathbf{a}\right]_q \right], \tag{2.8}$$

and

$$\left[\left\langle \mathsf{Decomp}_{b,q}(\mathbf{a}_1), \mathsf{PowersOf}_{b,q}(\mathbf{a}_2) \right\rangle\right]_q = \left[\langle \mathbf{a}_1, \mathbf{a}_2 \rangle\right]_q.$$

Chapter 3

General Approaches to FHE Schemes

3.1 Definitions

A fully homomorphic encryption scheme allows arbitrary functions to be evaluated over the ciphertext.

A basic encryption scheme is described by three algorithms:

1. The key generation algorithm $\mathsf{KeyGen}(1^\ell)$ returns a pair of keys (pk, sk) for the given security level ℓ. Note that this gives us a symmetric-key scheme when $pk = sk$ and a public-key scheme otherwise. Known symmetric-key FHE designs can usually be converted into a public-key version by setting the public key as encryptions of zero [8].
2. The encryption algorithm $\mathsf{Enc}(pk, m)$ encrypts m under the key pk.
3. The decryption algorithm $\mathsf{Dec}(sk, c)$ decrypts c under the key sk.

Regardless of security, we require the scheme to be correct in the sense that it always decrypts successfully:

$$\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m.$$

This condition implies that the encryption map is injective.

A homomorphic encryption scheme is a basic encryption scheme with an evaluation function $\mathsf{Eval}(pk, f, \mathbf{c})$. The parameters are the public key pk (sometimes a different notation, ek – the *evaluation key* – is used, but we do not distinguish them here because ek is also public), a description of the function f to be computed, and some ciphertexts $\mathbf{c}$ as

the arguments of f. For such an encryption scheme to qualify as a homomorphic one, we require the evaluation function to satisfy two conditions [51]:

1. (Correctness) if $\mathbf{c} = (c_1, \ldots, c_t) = (\mathsf{Enc}(pk, m_1), \ldots, \mathsf{Enc}(pk, m_t))$, then

$$\mathsf{Dec}(sk, \mathsf{Eval}(pk, f, \mathbf{c})) = f(m_1, \ldots, m_t),$$

   and

2. (Performance) given that f is a polynomial-time computable function, the output length of $\mathsf{Eval}(pk, f, \mathbf{c})$ must be $O(\text{poly}(\ell))$, where ℓ is the security parameter.

The performance condition is placed to rule out trivial schemes that do not actually compute the function f but instead append the input to Eval to the ciphertext, waiting for the computation to be processed at decryption.

Note that the description of f may vary depending on the model of computation adopted by a particular homomorphic encryption scheme. For example, the boolean circuit model appeared in Gentry's initial design and is still the most widely used one; later, the Turing machine model was also explored by Goldwasser *et al.* [56]; for the BGV scheme [25] introduced in Section 4.1.1, a polynomial model is more accurate.

Depending on the types of functions f that can be evaluated by Eval, we can categorize homomorphic encryption schemes by their evaluation capability:

- If f contains only addition or multiplication, the scheme is called *additively homomorphic* or *multiplicatively homomorphic*, respectively. RSA is an example of multiplicatively homomorphic encryption scheme. One special case is that when the scheme supports addition and scalar multiplication, the scheme is called *linearly homomorphic*.

- If f can be any function up to certain complexity, i.e., some restrictions apply to the input length of f, then the scheme is called *somewhat homomorphic*.

- If f can be an arbitrary function with no restriction on input length or complexity, then the scheme is called *fully homomorphic*.

3.2 General Framework of Homomorphic Encryption Schemes

Current constructions of FHE schemes all follow the same paradigm: starting from a somewhat homomorphic encryption scheme, a FHE scheme is obtained through the *bootstrapping* process, in which the decryption algorithm of the scheme is evaluated over the ciphertext [51]. This paradigm is developed by Gentry in his ground-breaking work of the first FHE scheme. However, Gentry's scheme is complicated, slow, and difficult to bootstrap; the principal ideals with a short generator, which it is based on, are later found to introduce vulnerability [20]. It was soon replaced by simpler and more efficient designs.

To successfully perform bootstrapping, the scheme must be able to evaluate circuit with depth larger than its own decryption function; the difference between these two will be the remaining evaluation capability after bootstrapping.

There is also an intermediate state between somewhat homomorphic and fully homomorphic schemes – *leveled* homomorphic encryption schemes. In leveled schemes, every bootstrapping switches the keys to a different set; when all pre-determined keys are exhausted, no more bootstrapping can be performed – hence the name "leveled." Details of leveled and fully homomorphic encryption schemes regarding bootstrapping are discussed further in Section 3.3.

3.2.1 Somewhat Homomorphic Encryption Schemes

The construction of somewhat homomorphic encryption scheme usually involves an underlying ring (lattice) and added noise. The ring inherently supports addition and multiplication, and the noise acts as a secure mask. This type of scheme is only somewhat homomorphic because the noise will accumulate along with evaluations and finally to a magnitude such that decryption will fail to produce a correct plaintext.

3.3 Bootstrapping

Bootstrapping is an evaluation of the decryption function $\mathsf{Dec}(sk, c)$. But one of its input, c, should not be encrypted once more, otherwise we will eventually obtain $c' = \mathsf{Enc}(pk, c)$, a ciphertext different from c, the one to be refreshed. Since c is known, we can rewrite the decryption function as a function of the secret key, selected by the ciphertext: $\mathsf{Dec}_c(sk)$, as shown by Figure 3.1. Note that the diagram depicts a change of the key set, as indicated

by the different subscripts. To facilitate an indefinite number of bootstrapping procedures, the key sets can be made the same, i.e., $pk_1 = pk_2$ and $sk_1 = sk_2$, but that introduces an encryption of the secret key under its corresponding public key, often requiring assumptions of *key-dependent message security* [28] as a result.

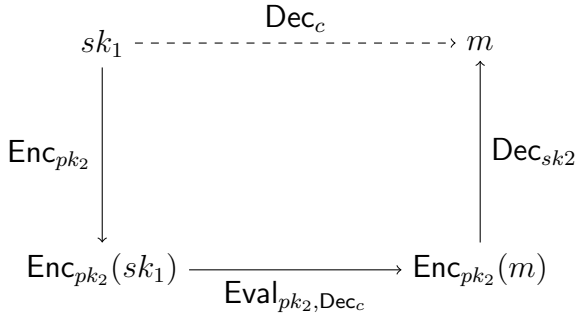

Figure 3.1: General paradigm of bootstrapping.

Bootstrapping is the key step in Gentry's framework. It is also the most difficult to design. Intuitively, there seems to be a never-ending race – the complexity (or depth of circuit) of decryption increases as parameters gets larger; and to evaluate a more complicated decryption circuit, parameters has to become larger in order to support the evaluation.

Gentry's success in bootstrapping his very first scheme depends on two factors:

1. Selecting a scheme with low decryption complexity;
2. Giving information of the secret key in the evaluation key, thus further lower the decryption complexity – this is typically referred to as "*squashing circuit*" in early FHE literature [51].

We will look at them separately.

3.3.1 Decryption Complexity

One of the reasons that successful FHE schemes are often based on LWE or similar lattice problems is that they often allow a simple decryption function of the form

$$m = \mathsf{Dec}(\mathbf{s}, \mathbf{c}) = f(\mathbf{c} \cdot \mathbf{s}),$$

where $\mathbf{c}$ is a ciphertext encrypting m under the secret key $\mathbf{s}$, "$\cdot$" is the inner product of vectors, and f is a *rounding function* that reduces a range of values to a point based

on some predefined metric of closeness. By the formula, this type of decryption involves several ring multiplications and additons (for the inner product), and a mapping for f.

3.3.2 Circuit Squashing

Circuit squashing is an idea that Gentry borrowed from server-aided secure computation protocols. Specifically, the ring multiplications, introduced by the inner product $\mathbf{c} \cdot \mathbf{s}$, were eliminated from decryption by pre-computing a list of products whose subset sum is exactly $\mathbf{c} \cdot \mathbf{s}$.

To make it possible, firstly a set $S' = \{ \mathbf{s}'_i \}_i$ is made public, where a subset of S sums to $\mathbf{s}$. If $A = \{ i_k \}_k$ are the indices of such a subset, i.e.,

$$\sum_k \mathbf{s}'_{i_k} = \mathbf{s}.$$

Then the decryption function can be rewritten as

$$\mathsf{Dec}(\mathbf{s}, \mathbf{c}) = f\left(\sum_k \mathbf{c} \cdot \mathbf{s}'_{i_k}\right) = f\left(\sum_i \mathbb{1}_A(i)\mathbf{c} \cdot \mathbf{s}'_i\right),$$

where $\mathbb{1}_A$ is the indicator function of set A, i.e.,

$$\mathbb{1}_A(x) = \begin{cases} 0 & x \notin A, \\ 1 & x \in A. \end{cases}$$

Since the set S is public, the inner products $\mathbf{c} \cdot \mathbf{s}'_i$ can be calculated in the clear and do not introduce any additional noise to the ciphertexts; meanwhile the indicator function $\mathbb{1}_A$ values are dependent only on $\mathbf{s}$ and set S' and can be encrypted fresh, with a minimal amount of noise.

3.3.3 Bootstrapping with GSW-Type Schemes

Another general method to bootstrap LWE-based FHE schemes is to use the GSW scheme or adaptations of it [29, 13]. Here we describe the adapted version used by Alperin-Sheriff and Peikert [13].

Gadget Vector and Matrix

To describe the GSW scheme, we first define the *gadget matrix*, a tool to implement randomized (binary) decomposition, in GSW-related literature.

Definition 24 (Gadget vector and gadget matrix)**.** For parameters q, n, let $l = \lceil \log_2 q \rceil$:

- The *gadget vector* over $\mathbb{Z}_q$ is
$$\mathbf{g}_q = \begin{bmatrix} 1 \\ 2 \\ \dots \\ 2^{l-1} \end{bmatrix}.$$
- The *gadget matrix* of size $n \times nl$ over $\mathbb{Z}_q$ is the tensor product $\mathbf{g}_q^\top \otimes I_n$, i.e.,
$$G = \begin{bmatrix} 1, \dots, 2^{l-1} & & & \\ & 1, \dots, 2^{l-1} & & \\ & & \ddots & \\ & & & 1, \dots, 2^{l-1} \end{bmatrix} \in \mathbb{Z}_q^{n \times nl}.$$

  By an inner product, the gadget vector induces a mapping from $\mathbb{Z}_q^l$ to $\mathbb{Z}_q$:
$$\begin{aligned} \phi_{\mathbf{g}} : \mathbb{Z}_q^l &\to \mathbb{Z}_q, \\ \mathbf{x} &\mapsto \langle \mathbf{g}, \mathbf{x} \rangle . \end{aligned}$$

There are exactly q^{l-1} preimages $x \in \mathbb{Z}_q^l$ such that $\phi_{\mathbf{g}}(x) = y \in \mathbb{Z}_q$.

Similarly, the gadget matrix G induces a mapping from $\mathbb{Z}_q^{nl}$ to $\mathbb{Z}_q^n$:
$$\begin{aligned} \phi_G : \mathbb{Z}_q^{nl} &\to \mathbb{Z}_q^n, \\ \mathbf{x} &\mapsto G\mathbf{x}. \end{aligned}$$

For any $\mathbf{y} \in \mathbb{Z}_q^n$, there are $q^{n(l-1)}$ preimages $\mathbf{x} \in \mathbb{Z}_q^{nl}$ such that $\mathbf{y} = G\mathbf{x}$, and they are easy to compute. In most literature, a procedure that generates a random *subgaussian* preimage for $\phi_{\mathbf{g}}$ and ϕ_G are denoted simply by $\mathbf{g}^{-1}(\cdot)$ and $G^{-1}(\cdot)$, respectively, but to avoid confusion, here we use the language of sampling from subgaussian distributions $\chi_{\phi_{\mathbf{g}}^{-1}(\cdot)}$ and $\chi_{\phi_G^{-1}(\cdot)}$ instead.

The basic scheme of GSW is as follows.

Setup The plaintext space is $\{0,1\} \subset \mathbb{Z}$; the ciphertext space is $\mathbb{Z}_q^{n \times nl}$, where n and q are instantiation parameters of the GSW scheme, and $l = \lceil \log_2 q \rceil$; noise is sampled from a subgaussian distribution χ over $\mathbb{Z}$.

KeyGen Let $\mathbf{s}' \leftarrow_{\chi^{n-1}} \mathbb{Z}^{n-1}$ and secret key $sk = \mathbf{s} = [\mathbf{s}' \mid 1] \in \mathbb{Z}^n$.

Enc Sample $C' \leftarrow_U \mathbb{Z}_q^{(n-1) \times nl}$ and $\mathbf{e} \leftarrow_\chi \mathbb{Z}^r$. Let $\mathbf{b}^\top = \mathbf{e}^\top - \mathbf{s}'^\top C' \pmod q$. The ciphertext is

$$C = \begin{bmatrix} C' \\ \mathbf{b}^\top \end{bmatrix} + mG.$$

Dec Let $\mathbf{c}$ be the second last column of C. The plaintext is obtained by

$$m = \lfloor \langle sk, \mathbf{c} \rangle \rceil_2 .$$

Like most HE schemes based on LWE, the GSW scheme supports the homomorphic evaluation of both additions and multiplications of its plaintexts, as are described below, with multiplication noise growth much higher than that of additions. But the GSW scheme also has an interesting property that its multiplication noise growth is asymmetric with respect to the noise magnitude by the two operands, as is demonstrated by Lemma 3.3.1. This property often allows us to rearrange operands in a circuit cleverly to minimize overall noise growth.

Given two GSW ciphertexts C_1 and C_2, its two basic homomorphic evaluation operations are

Addition $C_\boxplus = C_1 + C_2$; and

Multiplication $C_\boxdot = C_1 \cdot X$, where $X \leftarrow_{\chi_{\phi_G^{-1}(C_2)}} \mathbb{Z}_q^{nl \times nl}$.

Lemma 3.3.1. *If GSW ciphertexts C_1 and C_2 has noise vectors $\mathbf{e}_1$ and $\mathbf{e}_2$, respectively, then $C_\boxdot$ has noise vector*

$$\mathbf{e}_\boxdot = X^\top \mathbf{e}_1 + m_1 \mathbf{e}_2,$$

where X is the matrix used for evaluation and m_1 is the plaintext of C_1. The components of $X^\top \mathbf{e}_1$ are independent and subgaussian with parameters $O(\|\mathbf{e}_1\|)$ [13].

3.3.4 Additional Security Concerns: Circular Security

Besides the problem of reducing the complexity of decryption functions, bootstrapping also creates a security dilemma. In order to compute $\mathsf{Dec}_c(sk)$, an encryption of the secret key under itself could be passed over to an untrusted server, and the security implication of this encryption is little known. Gentry offers a mitigation of switching to a different set of keys in the process, which means the secret key will be encrypted under a new key. If client-server interactions during computation are allowed, then new keys can be generated and sent on the fly; if client is absent in the process, then the keys have to eventually form an encryption circle to support indefinitely many operations. In the latter case, *key-dependent message security*, or *circular security*, must be considered, and so far it usually exists as an assumption, potentially weakening the strength of encryption.

Circular encryptions of keys have been shown to potentially compromise the indistinguishability of LWE-type ciphertexts for almost any reasonable length key chain, even if the scheme has been shown to be IND-CPA secure without circular encryption [67, 5]. Since these results are derived on a different design, they do not necessarily mean it is impossible to achieve KDM security with LWE assumptions – rather, they separate KDM security from CPA security, meaning that they are independent assumptions, but they also show that schemes relying on KDM security need to be carefully designed and analyzed.

3.4 Avoiding Bootstrapping

Bootstrapping is considerably more costly than the other components of an FHE scheme following Gentry's blueprint. Many applications opt for only leveled homomorphic encryption for this reason. The high cost calls for a search for new design approaches to circumvent bootstrapping.

One such direction is secure multi-party computation (MPC). Assuming two-party interactions are allowed in the evaluation process, Couteau *et al.* created a protocol that does not need bootstrapping [40]. They convert ciphertexts between additively and multiplicatively homomorphic encryption schemes, with the help of communication, to match the operation to be executed. Without the requirement to support both operations, such schemes are readily available, very efficient, and noise-free. Therefore, the noise reduction problem, which prevails most FHE schemes and which bootstrapping was invented to solve, no longer exists. The tradeoff is communication overhead and security and availability concerns inherent to MPC protocols.

Chapter 4

Major FHE Schemes

This chapter describes three major categories of fully homomorphic encryption schemes. These selected schemes have relatively good performance and readily available implementations.

All of the following schemes have both symmetric and asymmetric forms. In fact, due to their homomorphic property and similar constructions, there is a generic transformation from the symmetric form of an encryption scheme to an asymmetric one [8, Section 5] (but the asymmetric form may not be unique or the most efficient). Let $E(k, m)$ and $D(k, c)$ be the symmetric encryption and decryption functions, respectively; then an asymmetric scheme can be generated as such:

KeyGen Generate t random ciphertexts $c_0, c_1, \ldots, c_{t-1}$ of zero ($c_i = E(k, 0)$ for all $0 \leq i < t$), then

$$\begin{aligned} sk &= k, \\ pk &= [c_0, c_1, \ldots, c_t]; \end{aligned}$$

Enc Select t random coefficients $r_0, r_1, \ldots, r_{t-1}$, and the encryption function is

$$\mathsf{Enc}(pk, m) = m + \sum_{i=0}^{t-1} r_i c_i;$$

Dec Decryption function remains the same:

$$\mathsf{Dec}(sk, c) = D(k, c).$$

4.1 Classification of Schemes

4.1.1 The BGV Scheme and Its Variants

Brakerski *et al.* described two similar schemes [28, 27] based on LWE-type problems. Then in [25] they proposed a generalized version of LWE (GLWE) to provide a unified framework for these schemes. However, we note that current research is still mostly on the special cases, LWE and RLWE, rather than GLWE.

The BGV scheme is of central interest here because many of its techniques – plaintext batching (also called SIMD), relinearization, and modulus switching and its related scale invariant perspective – are also applicable to other designs, as we will demonstrate in the following sections.

The Basic Scheme

Here we describe the BGV scheme with the language of GLWE and treat LWE and RLWE as two special cases. In the description, we assume that vectors are column vectors; sometimes the transpose of row vectors are omitted if the omission does not cause confusion; $\langle \mathbf{x}, \mathbf{y} \rangle$ denotes inner product; $[x]_q$ falls into the interval $(-q/2, q/2]$.

The setup of the scheme is as follows:

- Let $t \ll q$ be two coprime integers: t and q will be the parameters for the rings forming plaintext space and ciphertext space, respectively.
- Let $\Phi_u(x)$ be the u-th cyclotomic polynomial;
- Let $R_t = \mathbb{Z}_t[x]/(\Phi_u(x))$ be the plaintext space M, and $R_q = \mathbb{Z}_q[x]/(\Phi_(x))$ be the base of ciphertext space C;
- Let N be the number of samples (or, equivalently, the number of approximate linear equations in the LWE-type problem);
- Let $\chi = D_{\mathbb{Z}^n,\sigma}$ be a discrete Gaussian distribution with zero mean and standard deviation $\sigma = \alpha q$ for $0 < \alpha < 1/2t$: sampling from χ is equivalent to sampling from a continuous Gaussian distribution and rounding the sample to the nearest integer. In other words, χ can be seen as a discrete approximation to the Gaussian distribution with similar cumulative distribution functions;

- Let U be the uniform distribution.

KeyGen Sample a matrix $A' \leftarrow_U R_q^{N\times n}$ and a vector $\mathbf{s}' \leftarrow_\chi R_q^n$. Obtain $\mathbf{e}$ by sampling $e_i \leftarrow_\chi R_q, 0 \le i \le N-1$. Let $\mathbf{b} = A'\mathbf{s}' + t\mathbf{e}$ and $A = [\,\mathbf{b} \mid -A'\,]$;

Public key: $pk = A$;

Private key: $sk = \mathbf{s} = [\,1 \mid \mathbf{s}'\,]$;

Enc Expand plaintext $m \in R_t$ to an $(n+1)$-dimensional vector over R_t by adding zeros: $\mathbf{m}' = [\,m \mid \mathbf{0}\,]$. Sample $\mathbf{r} \leftarrow_\chi R_q^N$, then

$$\mathbf{c} = \mathsf{Enc}(pk, m) = \mathbf{r}^\top A + \mathbf{m}';$$

Dec Decryption is done by

$$m = \mathsf{Dec}(sk, \mathbf{c}) = \Big[\big[\langle \mathbf{c}, \mathbf{s}\rangle\big]_q\Big]_t.$$

Note that there is a slight difference of the RLWE-based scheme ($d > 1, n = 1$) – in encryption, small additional random errors must be added to both components of the ciphertext due to a lack of random errors in the public key.

The Evaluation Function

As is mentioned earlier in this chapter, computable functions can be represented as combinations of additions and multiplications. Here we describe how the BGV scheme supports these two operations. We also provide a summary of the *key switching* and *modulus switching* technique for the multiplication case. For the sake of simplicity, we will use the PLWE problem with coefficient embedding and L_∞ norm.

Before we proceed, we need a measurement of noise magnitude in order to assess the growth in each operation. Consider the decryption function

$$m = \Big[\big[\langle \mathbf{c}, \mathbf{s}\rangle\big]_q\Big]_t.$$

Since the noise is what eliminated by mod$t$, the noise term e can be expressed as

$$e = \langle \mathbf{c}, \mathbf{s}\rangle - m.$$

Apparently e is a polynomial

$$e(x) = \sum_{i=0}^{d-1} e_i x_i$$

with integer coefficients. As long as $|e| < q/2$, we can decrypt correctly. Hence we can measure the noise by the l_∞ norm $\|e\|_\infty$ (sometimes just $\|e\|$ for brevity), which is the largest absolute value among the coefficients of $e(x)$:

$$\|e\|_\infty = \max_{i=0}^{d-1} |e_i| .$$

To derive the Eval function, we first notice that the decryption function of the BGV scheme is a modular inner product, which can be viewed as a linear polynomial of the secret key $\mathbf{s}$ with coefficients from the ciphertext $\mathbf{c}$. Hence we have plaintext corresponding to the point-value representation and ciphertext corresponding to the coefficient representation of the same polynomial. Additions and multiplications on plaintexts are equivalent to polynomial additions and multiplications on ciphertexts.

Let $\mathbf{c} = [\, c_0, c_1, \ldots, c_n \,]$ and $\mathbf{c}' = [\, c'_0, c'_1, \ldots, c'_n \,]$ be two ciphertext on which we will perform our operations. Let e and e' be the noise terms associated with them. With coefficient representation, assuming that no decryption error occurs, we can write

$$m = \mathsf{Dec}_{\mathbf{c}}(\mathbf{s}) = \left(\sum_{i=0}^{n} c_i s_i \bmod q \right) \bmod t,$$
$$m' = \mathsf{Dec}_{\mathbf{c}'}(\mathbf{s}) = \left(\sum_{i=0}^{n} c'_i s_i \bmod q \right) \bmod t.$$

Then addition is trivial (component-wise addition modulo q)

$$\mathbf{c} + \mathbf{c}' = [c_0 + c'_0, c_1 + c'_1, \ldots, c_n + c'_n],$$

because

$$m + m' = \mathsf{Dec}_{\mathbf{c}+\mathbf{c}'}(\mathbf{s}) = \left(\sum_{i=0}^{n} (c_i + c'_i) s_i \bmod q \right) \bmod t.$$

It is obvious that the noise magnitude of the sum, $\|e_+\|$ is bounded by the sum of the noise magnitudes of two addends:

$$\|e_+\| \leq \|e\| + \|e'\| .$$

The formula for multiplication can be deduced in the same way, but it is slightly more complicated because the result is a tensor product $\mathbf{c} \otimes \mathbf{c}'$ encrypted under the tensor product $\mathbf{s} \otimes \mathbf{s}$ of the secret key $\mathbf{s}$:

$$\mathbf{c} \cdot \mathbf{c}' = \mathbf{c} \otimes \mathbf{c}' = [c_0 c'_0, c_0 c'_1, \ldots, c_0 c'_n, c_1 c'_0, c_1 c'_1, \ldots, c_1 c'_n,$$
$$c_2 c'_0, \ldots, c_{n-1} c'_n, c_n c'_0, c_n c'_1, \ldots, c_n c'_n]$$

The upperbound of the noise magnitude of the product, $\|e_\times\|$, is given by

$$\begin{aligned}\|e_\times\| &\leq \|e \cdot e' + m \cdot e' + m' \cdot e\| \\ &\leq \|e\|\,\|e'\| + t\,(\|e\| + \|e'\|)\end{aligned}$$

Relinearization and Key Switching

Multiplication brings another problem: after one multiplication, the length of ciphertext (and also the secret key) squared due to the tensor product. To solve this problem, key switching is invented to eliminate the growth of ciphertext size and key size by transforming the ciphertext $\mathbf{c}_1$ under $\mathbf{s}_1 \otimes \mathbf{s}_1$ to $\mathbf{c}_2$ under $\mathbf{s}_2$, preserving the inner product. During the transform, we have two sets of parameters. We will denote them by subscripts corresponding to $\mathbf{s}_1$ and $\mathbf{s}_2$. Note that we can let $\mathbf{s}_2$ be the same as $\mathbf{s}_1$, but nevertheless it involves issues regarding key-dependent message (KDM) security [74].

Here we demonstrate the idea of key switching with a straightforward construction first. This construction has some problem, but the problem and a fix (the construction in [25]) will be mentioned later.

Since we can sample as many times with the same secret key $\mathbf{s}$ as we need in the generation of public key, we can add the components of $\mathbf{s}_1 \otimes \mathbf{s}_1$ to the first column of such a public key. If we set $N_2 = (n_1 + 1)^2$, the length of $\mathbf{s}_1 \otimes \mathbf{s}_1$, sample $A_2' \leftarrow_U R_q^{N_2 \times n_2}$ and $\mathbf{e}_2 \leftarrow_\chi R_q^{N_2}$, computer $\mathbf{b}_2 = A_2'\mathbf{s}_2 + \mathbf{e}_2$, and let $A_2 = [\,\mathbf{b}_2 \mid -A_2'\,]$ to be the generated public key, then apparently

$$B = [\,\mathbf{b}_2 + (\mathbf{s}_1 \otimes \mathbf{s}_1) \mid -A_2'\,]$$

satisfies $\Big[\big[\mathbf{c}_1^\top B\mathbf{s}_2\big]_q\Big]_t = \Big[[\langle \mathbf{c}_1, \mathbf{s}_1 \otimes \mathbf{s}_1\rangle]_q\Big]_t$ if the noise is sufficiently small. Therefore we can let $\mathbf{c}_2$ be $\mathbf{c}_1^\top B$, and B is called a key switching matrix.

The above procedure has one problem: the noise growth is not predictable since ciphertext $\mathbf{c}_1$, which can have a large norm, gets multiplied by the added noise contained in $\mathbf{b}_2$. One way to fix this is to decompose each coefficient of each component in $\mathbf{c}_1$ into digits in base b, $\mathsf{Decomp}_{b,q}(\mathbf{c}_1)$ (see (2.7)), when key switching is conducted. It also requires us to include in the public key an encryption of $\mathsf{PowersOf}_{b,q}(\mathbf{s}_1 \otimes \mathbf{s}_1)$ (see (2.8)). After this change, each component of $\mathbf{c}_1$ is an element in R_q, which can be either $\mathbb{Z}_q$ or a polynomial ring over $\mathbb{Z}_q$; the decomposition yields at most $(\lfloor \log_b q \rfloor + 1)$ terms, and they amplify the noise by a factor of at most $(b-1)(\lfloor \log_b q \rfloor + 1)$, i.e.,

$$\|e_2\| \leq (b-1)(\lfloor \log_b q \rfloor + 1)\,\|e_1\|\,.$$

Therefore, the relinearization key switching matrix can be generated with $N_2 = (n_1 + 1)^2(\lfloor \log_b q \rfloor + 1)$ and

$$B' = [\, \mathbf{b}_2 + \mathsf{PowersOf}_{b,q}(\mathbf{s}_1 \otimes \mathbf{s}_1) \mid -A'_2 \,].$$

And the corresponding relinearization process is

$$\mathsf{Relinear}(\mathbf{c}_\times) = \mathsf{Decomp}_{b,q}(\mathbf{c}_\times)^\top B'.$$

Note that b can be arbitrary but has a slight effect on additional noise brought by key switching. The base $b = 2$ is chosen in [25]. The above procedure can also be used to switch keys when no relinearization is required; in this case, simply generate the key switching matrix with $\mathsf{PowersOf}_{b,q}(\mathbf{s}_1)$ instead of $\mathsf{PowersOf}_{b,q}(\mathbf{s}_1 \otimes \mathbf{s}_1)$ under new secret key $\mathbf{s}_2$ (the two secret keys may be the same).

Noise Management – Modulus Switching

The core technique of noise management in the BGV scheme is called "modulus switching". Lemmas 1 and 4 of [25] show that if we scale a ciphertext vector $\mathbf{c}$ by the ratio p/q, where $p < q$ are two moduli, and round it to the closest vector $\mathbf{c}'$ such that $c \equiv c' \pmod{r}$, then the magnitude of noise will be roughly p/q of the original. Though the noise-modulus ratio does not improve, the noise growth will be much slower for multiplication. For example, if $p \approx q/2$ and E_q is the upperbound of the noise magnitudes of two ciphertexts $\mathbf{c}$ and $\mathbf{c}'$ that we will multiply, then with modulus q we have

$$\|e_{\times,q}\| \leq E_q^2 + 2tE_q \approx E_q^2,$$

but with modulus p, $E_p \approx p/qE_q \approx E_q/2$ and

$$\|e_{\times,q}\| \leq E_p^2 + 2tE_p \approx E_q^2/4 + tE_q \approx E_q^2/4. \tag{4.1}$$

This gives us an noise magnitude upperbound with respect to p approximately a quarter of that with respect to q, and the noise-to-modulus ratio is reduced to roughly a half.

Modulus-switching is usually only performed after a multiplication evaluation, following a key switching. Figure 4.1 is an illustration of this process.

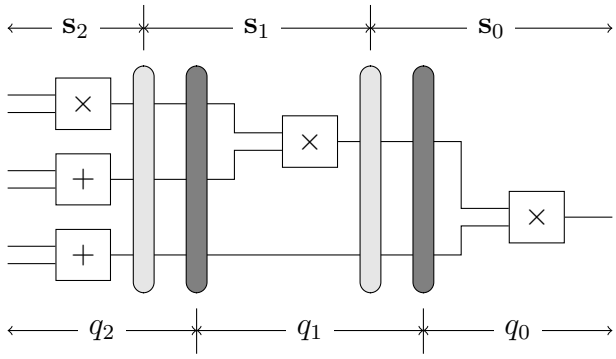

Figure 4.1: An illustration of the execution of BGV scheme. Integers q_i are moduli and $\mathbf{s}_i$ are corresponding keys. Light gray and dark gray bars are key switching and modulus switching procedures, respectively.

The Scale-Invariant Technique and the FV Scheme

The design of the FV (or BFV in some literature) scheme is essentially the same to that of the BGV scheme based on RLWE but with an application of the *scale-invariant* technique introduced in [23] specifically designed to mimic the effect of modulus-switching after a homomorphic multiplication without one. The technique involves inverting the parity of plaintext and noise in an encryption and homomorphic multiplication with an additional coefficient t/q. Despite the equivalence in normal LWE-based encryption schemes, such an inversion turns out to be an optimization for homomorphic evaluations. In some occasions it also allows truncation of the least significant bits of keys and ciphertexts [50, 30], leading to smaller storage and communication overhead. These desirable properties makes it ubiquitous in the more recent schemes designed for practicality.

In Table 4.1 we describe the FV scheme in contrast to the BGV scheme to demonstrate this technique. For the ease of description, a symmetric version is adapted from FV.

		FV		BGV
Parameters		- Select $f(x) = \Phi_u(x)$ with degree $\varphi(u)$, q, and t based on the security parameter; - $R = \mathbb{Z}[x]/(f(x))$, $R_q = R/qR$ and $R_t = R/tR$, respectively; - Select a subgaussian distribution χ over R based on the security parameter; - U is uniform distribution		
KeyGen		$\mathbf{k} = [\,1 \mid s\,]$, where $s \leftarrow_\chi R_q$		
Enc		- $a \leftarrow_U R_q$; - $e_1, e_2 \leftarrow_\chi R$; - $\mathbf{c} = \left[\, as + e_1 + \left\lfloor \frac{q}{t} \right\rfloor m, -a + e_2 \,\right]$		- $a \leftarrow_U R_q$; - $e_1, e_2 \leftarrow_\chi R$; - $\mathbf{c} = [\, as + te_1 + m, -a + e_2 \,]$
Dec		$m = \left[\left\lfloor \frac{t}{q} \left[\langle \mathbf{c}, \mathbf{k} \rangle \right]_q \right\rceil \right]_t$		$m = \left[\left[\langle \mathbf{c}, \mathbf{k} \rangle \right]_q \right]_t$
Eval	Parameters	$\mathbf{c}_0 = [\, c_{00}, c_{01} \,], \mathbf{c}_1 = [\, c_{10}, c_{11} \,]$, secret key $\mathbf{k}_2 = [\,1 \mid s_2\,]$, decomposing base b		
	Addition	$\mathbf{c}_+ = \mathbf{c}_0 + \mathbf{c}_1$		
	Multiplication	$\mathbf{c}_\times = \frac{t}{q} [\, c_{00}c_{10}, c_{01}c_{10} + c_{00}c_{11}, c_{01}c_{11} \,]$ with secret key $[\,1, s_1, s_1^2\,]$		$\mathbf{c}_\times = [\, c_{00}c_{10}, c_{01}c_{10} + c_{00}c_{11}, c_{01}c_{11} \,]$ with secret key $[\,1, s_1, s_1^2\,]$
	RLKeyGen	Method 1: same as BGV	Method 2: - Select p, a scaling factor; - $a_2 \leftarrow_U R_{pq}$; - $e_{21}, e_{22} \leftarrow_{\chi'} R_{pq}$; - $B' = [\, a_2 s_2 + e_{21} + p s_1^2 \mid -a_2 + e_{22} \,]$; - $B = \left[\, I \;\middle|\; B'^\top \,\right]$	- $\mathbf{a}_2 \leftarrow_U R_q^{\lfloor \log_b q \rfloor + 1}$; - $\mathbf{e}_{21}, \mathbf{e}_{22} \leftarrow_\chi R_q^{\lfloor \log_b q \rfloor + 1}$; - $\mathbf{b}_2 = s_2 \mathbf{a}_2 + \mathbf{e}_{21}$; - $B = [\, \mathbf{b}_2 + \mathsf{PowersOf}_{b,q}(s_1^2) \mid -\mathbf{a}_2 + \mathbf{e}_{22} \,]$
	Relinearization/ key switching		$\mathbf{c}'_\times = B^\top \mathbf{c}_\times$	$\mathbf{c}'_\times = B^\top \mathsf{Decomp}_{b,q}(\mathbf{c}_\times)$
	Modulus switching	N/A		Return $\mathbf{c}_p$, the closest vector to $\frac{p}{q}\mathbf{c}_q$ s.t. $\mathbf{c}_p \equiv \mathbf{c}_q \pmod t$

Table 4.1: Comparison between the FV and BGV schemes.

4.1.2 The FHEW Scheme and Its Variants

Among the many FHE schemes developed so far, FHEW-type schemes are unique in two aspects:

1. They provide evaluation for NAND gates rather than addition and multiplication (XOR and AND gates, respectively, in binary cases);
2. The evaluation of NAND gates is achieved by computation in an extension ring of the ciphertext space; at some point the ciphertext must be brought back to its original space, and this is exactly the bootstrapping process of FHEW-type schemes.

Basic Scheme

The basic scheme is similar to that of the LWE-based variant of BGV, but instead of decryption by parity (reduction modulo t), its decryption is done by rounding to the nearest plaintext points, which are evenly distributed in the interval $(-t/2, t/2]$. Decryption is correct as long as the noise magnitude does not exceed $q/2t$.

The setup of FHEW basic scheme is as follows.

- Let $t = 2$, q an integer divisible by t (the authors of FHEW did not mention whether t must divide q, but the condition will make the distribution of plaintext points perfectly even, and it also holds for their proposed parameters; to support the evaluation function, t must be doubled to 4 and q need to be divisible by 16 to tolerate the noise;
- Let χ be a discrete Gaussian distribution with zero mean and standard deviation $\sigma = \alpha q$ for $0 < \alpha < 1/2t$;
- Let U be the uniform distribution.

We describe the symmetric version of FHEW, as the authors did in their original paper. An asymmetric version can be obtained through the conversion at the beginning of this chapter. The description is modified to be close to the other schemes presented here to maintain the consistency of the notations so that the reader can conveniently compare them. For example, the ciphertext vector components are switched to match the typical definition of LWE problems and other LWE-based schemes, and its *randomized rounding function* is also replaced with sampling from a distribution.

KeyGen Sample a vector $\mathbf{s} \leftarrow_{\chi} \mathbb{Z}_q^n$, and let the secret key be $\mathbf{k} = (1, \mathbf{s})$;

Enc Sample a vector $\mathbf{a} \leftarrow_U \mathbb{Z}_q^n$ and noise $e \leftarrow_\chi \mathbb{Z}_q$, and compute the ciphertext

$$\mathbf{c} = (c_0, \mathbf{c}_1) = \mathsf{Enc}(k, m) = (\langle \mathbf{a}, \mathbf{s} \rangle + mq/t + e, -\mathbf{a}).$$

Note that our description of the encryption is equivalent to that in the original paper only when t divides q;

Dec Decryption is done by rounding $\frac{t}{q}(\langle \mathbf{c}, \mathbf{k} \rangle)$ to the nearest integer modulo t:

$$m = \left[\left\lfloor \frac{t}{q} \langle \mathbf{c}, \mathbf{k} \rangle \right\rceil \right]_t.$$

As the FHEW scheme is still based on LWE, the same key switching and modulus switching techniques (see Section 4.1.1) apply.

Evaluation Function

To support the evaluation, we start from encryptions for plaintexts embedded in a slightly larger ring $\mathbb{Z}_4$ (setting $t = 4$ in the basic scheme but keeping the plaintext binary). The evaluation function of FHEW supports only the NAND gate. From the plaintext's point of view, the evaluation computes the sum of two plaintexts. So the sums 0 and 1 correspond to NAND output 1, and the sum 2 corresponds to NAND output 0. By carefully shifting the values, a decryption with $t = 2$ can reveal the NAND output for the two inputs:

$$\mathsf{NAND}(\mathbf{c}, \mathbf{c}') = (5q/8 - c_0 - c_0', -\mathbf{c}_1 - \mathbf{c}_1').$$

Unlike BGV, where noise can grow quickly in multiplication, this summation will only increase the noise by a small amount since it involves only addition. But FHEW faces a different problem: the evaluation result must be lifed back into an encryption of plaintexts from $\mathbb{Z}_4$, otherwise the computation is not sustainable. The bootstrapping of FHEW is essentially solving this problem.

4.1.3 The LTV Scheme and Its Variants

López-Alt *et al.* present their design of a SWHE scheme based on the NTRU cryptosystem in [71]. Their scheme is capable of handling a multitude of keys, thus becoming an ideal candidate for encrypted multi-party computation, given that the number of keys needed is known at parameter selection time.

One drawback of their scheme is that, on top of RLWE assumption and KDM security assumptions, it also needs a rather unconventional assumption that the *Decisional Small Polynomial Ratio* (DSPR) problem is hard in their chosen parameter range. DSPR assumption states that the ciphertext of their scheme is indistinguishable from a uniformly random element.

The Basic Scheme

The setup of the scheme is as follows.

- Select integer u a power of 2 and prime q based on the security parameter;
- Let $\Phi_u(x) = x^{u/2} + 1$, the u-th cyclotomic polynomial; let $n = \deg(\Phi_u) = u/2$;
- Let $R = \mathbb{Z}[x]/(\Phi_u(x))$ and $R_q = R/qR$;
- Let $\chi = D_{\mathbb{Z}^n,r}$ be the truncated discrete Gaussian distribution with standard deviation $r > 0$, whose sample gives an element in R by representation of its coefficients;
- Let $M = \{0, 1\}$ be the plaintext space;
- Let $C = R_q$ be the ciphertext space;

KeyGen For the i-th participant, sample $g_i \leftarrow_\chi R$; sample $f_i' \leftarrow_\chi R$ until $f_i = 2f_i' + 1$ is invertible in R_q; let $h_i = 2g_i f_i^{-1} \in R_q$;

Public key: $pk_i = h_i$;

Private key: $sk_i = f_i$;

Enc Sample $s, e \leftarrow_\chi R$, then the ciphertext is obtained by

$$c_i = h_i s_i + 2e_i + m \in R_q;$$

Dec The plaintext is recovered by

$$m = \left[[f_1 \cdots f_N \cdot c]_q \right]_2,$$

where $f_1, \ldots, f_N$ are all of the N private keys.

The Evaluation Function

An NTRU ciphertext encrypted by the public key h with corresponding private key f can potentially be decrypte by αf, a multiple of f, given that $\alpha \in R_q$ is sufficiently small so that $[\alpha f \cdot c]_q$ does not cause the constant term to wrap around q. This observation naturally leads to the multikey property of the LTV scheme. Below we will demonstrate how addition and multiplication are evaluated by the LTV scheme, under two sets of keys.

Assuming we have ciphertexts

$$c_1 = h_1 s_1 + 2e_1 + m_1$$

and

$$c_2 = h_2 s_2 + 2e_2 + m_2,$$

then $c_1 + c_2$ and $c_1 c_2$ decrypt to $m_1 + m_2$ and $m_1 m_2$, respectively, under private key $f_1 f_2$, given that $f_1 f_2$ is sufficiently small to not cause a wrap around in the constant term. Since

$$f_1 f_2 (c_1 + c_2) = 2(f_2 g_1 s_1 + f_1 g_2 s_2) + 2 f_1 f_2 (e_1 + e_2) + f_1 f_2 (m_1 + m_2),$$

we have

$$[f_1 f_2 (c_1 + c_2)]_2 = m_1 + m_2.$$

And since

$$\begin{aligned} f_1 f_2 (c_1 c_2) = {} & 4 g_1 g_2 s_1 s_2 + 2 f_1 g_2 s_2 (2e_1 + m_1) + 2 f_2 g_1 s_1 (2e_2 + m_2) \\ & + 2 f_1 f_2 (2 e_1 e_2 + m_1 e_2 + m_2 e_1) \\ & + f_1 f_2 m_1 m_2, \end{aligned}$$

we have

$$[f_1 f_2 (c_1 c_2)]_2 = m_1 m_2.$$

Note that multiplying a ciphertext by another under the same private key f will result in one that has to be decrypted under f^2, but this can be mitigated by the relinearization technique provided by BGV, detailed in Section 4.1.1.

YASHE

Since the LTV scheme also involves modulus switching, the scale invariant technique (see Section 4.1.1) can also be applied to replace the overhead caused by the modulus ladder. One such result is YASHE [22].

YASHE also draws results from [93] to eliminate the DSPR assumption by using a wider Gaussian distribution. To achieve this, YASHE optimizes LTV multiplication with the decomposing technique in key-switching (Section 4.1.1) to reduce noise growth and, hence, allows the key generating polynomials to be sampled from a wider distribution. On the down side, the multi-key feature is no longer available due to increased noise.

Table 4.2 shows a side-by-side comparison between the two schemes.

		YASHE	LTV
Setup		- Select $\Phi_u(x)$ with degree $\varphi(u)$, $1 < t < q$ based on the security parameter; - $R = \mathbb{Z}[x]/(\Phi_u(x))$, $R_q = R/qR$ and $R_t = R/tR$, respectively; - Select two subgaussian distributions χ_{key} and χ_{err} over R based on the security parameter;	- Select $\Phi_u(x)$, where u is a power of 2, with degree $\varphi(u) = 2^{u-1}$, and prime q based on the security parameter; - Plaintext modulus $t = 2$; - $R = \mathbb{Z}[x]/(\Phi_u(x))$, $R_q = R/qR$; - Select a truncated discrete Gaussian distribution χ over R based on the security parameter
KeyGen		- Sample $f' \leftarrow_{\chi_{\text{key}}} R$ until $tf' + 1$ is invertible modulo q; sample $g \leftarrow_{\chi_{\text{key}}} R$; - $\mathbf{sk} = f$; - $\mathbf{pk} = h = [tgf^{-1}]_q$;	- Sample $f' \leftarrow_{\chi} R$ until $2f' + 1$ is invertible modulo q; sample $g \leftarrow_{\chi} R$; - $\mathbf{sk} = f$; - $\mathbf{pk} = h = [2gf^{-1}]_q$;
Enc		$c = \left[\left\lfloor \frac{q}{t} \right\rfloor m + e + hs\right]_q$	$c = [m + 2e + hs]_q$
Dec		$m = \left[\left\lfloor \frac{t}{q} [fc]_q \right\rceil\right]_t$	$m = \left[[fc]_q\right]_2$
Eval	Setup	c_1 with secret key $f_1 = \prod_i f_{1i}$, c_2 with secret key $f_2 = \prod_j f_{2j}$; decomposing base b	
	Addition	$c_+ = [c_1 + c_2]_q$ with secret key $f = \text{lcm}(f_1, f_2)$	
	RLKeyGen	For each key f, - Sample $\mathbf{e}, \mathbf{s} \leftarrow_{\chi_{\text{err}}} R^{\lfloor \log_b q \rfloor + 1}$ - $B_f = \mathsf{PowersOf}_{b,q}(f) + \mathbf{e} + h\mathbf{s}$	
	Multiplication	$c'_\times = \left[\left\lfloor \frac{t}{q} c_1 c_2 \right\rceil\right]_q$	$c'_\times = [c_1 c_2]_q$
	Relinearization/ Key switching	For each $f_k \in \{ f_{1i} \}_i \cap \{ f_{2j} \}_j$, where $\left\lvert \{ f_{1i} \}_i \cap \{ f_{2j} \}_j \right\rvert = L$, $c_\times = \left\langle \mathsf{Decomp}_{b,q}\left(\left\langle \cdots \left(\left\langle \mathsf{Decomp}_{b,q}(c'_\times), B_{f_{k_1}} \right\rangle\right), \cdots \right\rangle\right), B_{f_{k_L}} \right\rangle$ with secret key $f = \text{lcm}(f_1, f_2)$	

Table 4.2: Comparison of YASHE and LTV schemes. Note: when the two schemes share one cell, the parameter of LTV is always set at $t = b = 2$ and $\chi_{\text{err}} = \chi_{\text{key}} = \chi$

4.2 Known Attacks

Since the security of lattice-based cryptosystems are directly related to hard lattice problems, if we can solve the problem or approximate the solution, the same algorithm can be used to launch general attacks on such cryptosystems. Solving hard lattice problems exactly requires exponential time unless $\mathbf{P} = \mathbf{NP}$ [16, 2]. These attacks produce a spectrum between polynomial-time approximations (with poor approximation factor) and exponential-time exact solutions.

Algorithms solving lattice problems usually consist of two distinct steps: lattice basis reduction and a search for solutions. Both steps affect their speeds and approximation factors.

For lattice reduction, the LLL algorithm [68] is very efficient and produces good approximations. It is still widely in use today, often in the form of its BKZ variant [90], which uses exact SVP solvers for "blocks" of small sub-lattice. By controlling the block size, BKZ-LLL achieves an adjustable tradeoff between running time and approximation factors.

Although SVP and CVP are both **NP**-hard, CVP seems to be harder to approximate [55]. All CVP algorithms can be directly applied to SVP, but SVP can be solved faster or with smaller approximation factors [62]. The output of lattice reduction algorithms can be sorted and the shortest picked as an approximate solution to SVP; Babai's nearest plane algorithm [17] is a well-known and easy to implement approximation algorithm for CVP; while exact solutions of SVP and CVP can be found with a number of enumeration and sieving algorithms [77].

Another approach exploits the similarity between lattices and codes. BKW algorithm [21] is essentially a partial decoding by reducing the dimension of the generator matrix A until the noise reaches a predetermined threshold, and then the algorithm iterates over all possible values of the secret $\mathbf{s}'$ (with a reduced length) and examine the noise distribution. Lindner and Peikert also suggested a combination of lattice reduction and decoding to solve LWE problems [70]. A recent algorithm based on BKW performs asymptotically better than Linder and Peikert's attack [59] and breaks their parameters, but the authors did not suggest new parameters.

Although general lattice problems are **NP**-hard with appropriate parameters, ideal lattices may provide weaker in security. No efficient quantum algorithm is known to solve generic CVP and SVP problems with good approximation factor, but such quantum algorithm has been found for ideal lattices with greatly reduced approximation factor [20, 41]. Some ideal lattice choices are particularly vulnerable and can be exploited in weak-instance attacks [46, 48, 35].

Chapter 5

FHE in Practice

This chapter provides a summary on the practical aspect of FHE including library implementations and optimization techniques, applications, and standardization efforts.

5.1 Open Source FHE Implementations

5.1.1 HElib

HElib is an object-oriented implementation of the BGV scheme in C++ with number theory computation support from NTL. In addition to all the basic functions that are specified in the BGV scheme, HElib also implements the SIMD optimization described in [92] and [52]. By decomposing the ring R_t with Chinese Remainder Theorem into many "slots", they are able to pack multiple plaintexts into a single element of R_t; they also implemented the permutation of slots via an extension of key switching. Due to the parallelization, SIMD operations bring much better amortized performance.

HElib is developed based on the RLWE variant of BGV scheme. In the parameters, the plaintext space is the ring $R_t = \mathbb{Z}_t[x]/(\Phi_u(x))$, where t is prime or prime power and $\Phi_u(x)$ is the u-th cyclotomic polynomial; similarly, the base ring of the ciphertext space is $R_q = \mathbb{Z}_q[x]/(\Phi_u(x))$, where q is some odd integer coprime with t, and the ciphertext space is R_q^2. In particular, integers reduced modulo q take values in $(-q/2, q/2]$.

5.1.2 FHEW

The FHEW scheme was implemented, also in C++, by its inventors, Ducas and Micciancio, and made available under the same name. Most of the implementation retained the design in their paper [44], and an update was made to improve the efficiency and add support for gates other than NAND.

The FHEW project is much smaller and simpler compared with HElib. Such simplicity is gained in part by sacrificing the flexibility in parameter selection: the FHEW interface accepts no argument and the scheme is initialized with hard-coded parameters when used as is.

5.2 Optimization Methods

5.2.1 Chinese Remainder Theorem and Fast Fourier Transform

Chinese Remainder Theorem (CRT) may be the most cited number theory theorem in cryptography. Even before the FHE problem was solved by Gentry, CRT had played an important role in improving the efficiency of cryptographic applications. The isomorphic decomposition of a ring element (or ideal) under CRT leads to several smaller parts, and hence the computation on each of them becomes simpler and faster.

For example, in an RSA decryption of ciphertext c with secret key $(n = pq, d)$, instead of computing $c^d \bmod n$, the ciphertext can be decomposed into (c_p, c_q), and we obtain the decomposed plaintext by

$$(m_p, m_q) = (c_p^{d \bmod (p-1)}, c_q^{d \bmod (q-1)}).$$

When d is large compared with p and q, the number of multiplications, an expensive operation, can be significantly reduced, especially given the current recommended key size of 2048 bits.

Application in HElib

For an efficient implementation, HElib has further requirements for their moduli q's. More specifically, HElib picks $L + 1$ distinct primes p_i $(0 \le i \le L)$ such that $p_i \equiv 1 \pmod{u}$ and makes $q = p_0 \cdots p_L$ [60]. This decision allows them to represent an element of R_q by "double CRT" – the usage of CRT in two levels.

With q constructed in such a way, $\Phi_u(x)$ has $\phi(u)$ distinct integer roots ζ_i modulo q, and the canonical embedding of an element in R_q can be computed efficiently by evaluating the element, seen as a polynomial, at each root ζ_i. This is the first level of CRT.

The second level of CRT is in the representation of the integers resulted from the polynomial evaluation. Since q is the product of $L+1$ distinct primes, by CRT, an integer modulo q can be decomposed into $L+1$ components modulo each of q's prime factors.

With double CRT, both addition and multiplication can be done component-wise. Fast Fourier transform (FFT) is also applied to speed up the evaluation.

5.2.2 Single Instruction Multiple Data (SIMD) Operations

Another CRT application frequently referred to in FHE research borrowed the term *Single Instruction Multiple Data* (SIMD) from a classification of the architecture of parallel computers. It is also called *batching* in some articles.

Similar to the decomposition above of the ciphertext spce R_q, the plaintext space R_t can be decomposed as well. Since we do not operate directly on the plaintext space, it is less relevant to the running time, but the decomposition allows us to pack multiple messages into a single plaintext through the CRT mapping, which gives rise to the SIMD or batching operations:

$$\begin{aligned} R_t &= \mathbb{Z}_t[x]/(\Phi_m(x)) \\ &\cong \mathbb{Z}_t[x]/(F_1(x)) \oplus \mathbb{Z}_t[x]/(F_2(x)) \oplus \cdots \oplus \mathbb{Z}_t[x]/(F_s(x)), \end{aligned}$$

where $\Phi_m(x) \equiv \prod_{i=1}^{s} F_i(x) \bmod t$. Each $\mathbb{Z}_t[x]/(F_i(x))$ is called a "plaintext slot". The decomposition of R_t usually does not result in $\mathbb{Z}_t$ slots as t is small.

After packing, an operation (addition or multiplication) on the ciphertext can be seen as simutaneously applied to every plaintext slot, as long as the decryption is correct.

5.3 Current Development on FHE

This section contains recent progress on fast implementations of existing FHE schemes for privacy-preserving computations and the development of the FHE standard. Current research on FHE can be broadly categorized as

1. applications or developing protocols relying on FHE and

2. accelerate the performance of the known FHE schemes.

In the recent years, several HE schemes, namely, BFV, BGV, TFHE, and CKKS have received attentions where the security of the schemes are based on the hardness of LWE or RLWE. Microsoft SEAL supports the BFV and CKKS schemes. From a development perspective, SEAL recently released a .NET standard wrappers. Lattigo also implemented the BFV and CKKS and supports secure multiparty computation based on the threshold or distributed key. PALISADE supports the BGV, BFV, CKKS, and FHEW schemes.

5.3.1 Privacy-preserving Machine Learning using FHE

This subsection summarizes some recent schemes developed for performing machine learning on confidential data using fully homomorphic encryption. In the recent years, there has been a number of protocols and frameworks developed to perform machine learning algorithms on encrypted data based on FHE. These work demonstrate the potential and practicality of current FHE schemes and their implementations. Graepel *et al.* in [58] first proposed techniques to use an FHE scheme to train machine learning classification algorithms outsourced to a cloud. CryptoNets is proposed to train neural networks over encrypted data where the training task is outsourced to a cloud [54]. In [42], Crawford *et al.* proposed a system for training logistic-regression models on genomic data using the BGV scheme. The work of [12] proposes a protocol for a collaborative evaluation of random forests over a dataset contributed by multiple data owners based on a multi-key somewhat homomorphic encryption scheme. Nandakumar *et al.* [81] proposed a scheme for training deep neural networks on data encrypted in a non-interactive way using an FHE scheme. Badawi *et al.* developed a training protocol based on FHE where they accelerate the performance of CNNs on encrypted data by exploiting the GPU computation power [18]. Takabi *et al.* [94] also used GPUs to accelerate the computation of FHE schemes to provide efficient and scalable MLaaS. Hesamifard *et al.* [65] proposed CryptoDL, a client-server privacy-preserving deep neural network training protocol where the FHE scheme is instantiated with HELib. In a follow-up work, Hesamifard *et al.* in [63] considered training a convolutional neural network over an encrypted data. Badawi *et al.* [3] proposed PrivFT, a system for text classification using FHE, which serves the tasks of encrypted inference services and training an model on an encrypted dataset.

5.3.2 Accelerating Performance of FHE Schemes

There is another line of work that put effort to accelerate the computation of FHE schemes by leveraging hardware. Riazi *et al.* recently proposed a hardware architecture for accel-

erating the computation of underlying functions such as fast modular arithmetics and number-theoretic transform (NTT) in the CKKS homomorphic encryption scheme [87]. This results in improved performance of the SEAL library. Microsoft SEAL team is working on improve the performance of its library by implementing them in FPGAs at the server. Mert *et al.* proposed an NTT-based polynomial multiplier architecture for accelerating computations in SEAL [76]. Inspired by cuHE, Badawi *et al.* [4] developed a high-speed implementation of the FV somewhat homomorphic encryption scheme in CUDA.

5.3.3 Standardization Efforts

Due to the significant advancement of proposing new efficient schemes, open-source implementations and applications and growing demands from industry, there is a standard for FHE initiated to standardize FHE scheme(s) to have an unified and simplified API, and clear and understandable security properties for use by non-experts as well as experts. The homomorphic encryption standards meetings are held once a year at different locations. The participants to the standardization meetings are from industry, academia and government [6, 10, 11].

List of Candidates Below are a list of candidates (not an exhaustive list) submitted to the HE standard.

- HELib: An early and widely used library from IBM that supports the BGV scheme and bootstrapping.
- Microsoft SEAL: A widely used open source library from Microsoft that supports the BFV and the CKKS schemes.
- PALISADE: A widely-used open source library from a consortium of DARPA-funded defense contractors that provides lattice cryptography building blocks and supports leading homomorphic encryption schemes.
- FHEW / TFHE: (Torus-FHE) GSW-based libraries with fast bootstrapped operations. TFHE is designed from FHEW, which is no longer actively being developed.
- HeaAn: This library implements the CKKS scheme with native support for fixed point approximate arithmetic.
- $\Lambda \circ \Lambda$ (pronounced "L O L"): This is a Haskell library for ring-based lattice cryptography that supports FHE.

- NFLlib: This library is an outgrowth of the European HEAT project to explore high-performance homomorphic encryption using low-level processor primitives.
- HEAT: This library focuses on an API that bridges FV-NFLib and HeLIB.
- HEAT: A HW accelerator implementation for FV-NFLlib.
- cuHE: This library explores the use of GPGPUs to accelerate homomorphic encryption.
- Lattigo: This is a lattice-based cryptographic library written in Go.

Chapter 6

Arithmetic and Data Aggregation with HElib

We use HElib to implement integer arithmetics, study the performance and possible optimization when applied to data aggregations on the cloud. To our best knowledge, this is the first implementation involving operations on integers, and thus it shows the usage of FHE to some practical problems such as data aggregation.

Organization This chapter is organized as follows. Section 6.1 summarizes some related work and provides related background about HElib and integer full adder circuit; Section 6.2 describes the implementation, discusses alternative implementation approaches and optimizations, and estimates the levels of multiplications, an important parameter measuring the complexity, of arithmetic functions; Section 6.3 discusses parameter selection and measures the performance with experiments; finally, Section 6.4 summarizes the work, compares it with other privacy enhancing techniques on data aggregation, and proposes some future work.

6.1 Preliminary

6.1.1 Related Work

AES has been implemented a few times, for example in [53] and [36], to demonstrate the capability of homomorphic encryption schemes. However, the successful homomorphic

evaluation of AES does not fully represent the potential applications of FHE since the AES operations are defined over the finite field $GF(2^8)$, which are almost natively supported by most schemes, and the complexity of AES functions are also limited by its block size and key size.

Polynomial evaluation over a base ring (or field) is included in the HElib source, but it also falls short in demonstrating FHE in practice for similar reasons since they only involve operations reduced modulo p and/or some polynomial.

The possibility of using FHE on data aggregation is also discussed in [79]. They make their scheme support (a limited number of) integer additions and multiplications by enlarging the base of plaintext. The advantage of their technique is the simplicity, but it also has shortcomings: one is that it does not support division, and hence they are forced to return the numerator and denominator separately wherever division is involved; another is that once encrypted, it is very difficult to extract individual bits from the ciphertext.

We also note that a popular choice of masking sensitive data in data mining is data obfuscation such as that considered in [1]. This approach hides sensitive data fields with noise drawn from a designed distribution so that the impact on the aggregation result can be calculated and controlled. Data obfuscation introduces much less overhead compared with our approach, but the latter still has a few advantages—the general framework can be easily adapted to any specific computation, and, moreover, it also provides protection to the aggregation result and does not impact the accuracy.

6.1.2 HElib Features

Table 6.1 lists HElib interfaces that are used most frequently in implementations; the table also include shorthands that we will use to describe our algorithms. We note that the public key contains information for key switching and therefore must be passed to FHE.Mult; the public key is also passed to other evaluation functions for the uniformity of interfaces. The exact usage of HElib can be found in its document [60] coming with the source code.

6.1.3 Full adder circuit

Integer addition can be implemented as a full adder circuit (Figure 6.1. In this circuit, the i-th 1-bit full adder takes summands A_i and B_i and carry from the last bit C_{i-1}, and computes a sum for the current bit S_i and the carry C_i by

$$\begin{aligned} S_i &= A_i + B_i + C_{i-1}, \\ C_i &= A_i \cdot B_i + C_{i-1} \cdot (A_i + B_i). \end{aligned}$$

Category	HElib Interface	Abbr.
Basic scheme	`FHESecKey::GenSecKey` `add1DMatrices`	FHE.KeyGen
	`EncryptedArray::encrypt`	FHE.Enc
	`EncryptedArray::decrypt`	FHE.Dec
Evaluation	`Ctxt::addCtxt`	FHE.Add
	`Ctxt::addConstant`	FHE.addConstant
	`Ctxt::multiplyBy`	FHE.Mult
SIMD	`EncryptedArray::encode`	
	`EncryptedArray::decode`	
	`EncryptedArray::rotate`	
	`EncryptedArray::shift`	

Table 6.1: HElib interfaces in categories. "Abbr" is the shorthand used in this chapter for simplicity.

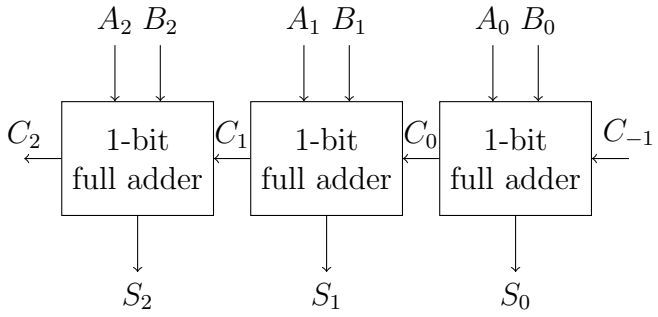

Figure 6.1: A full adder circuit

6.2 Implementation of Integer Arithmetic

In this section, we assume that the numbers are written in a little-endian base-b representation and encrypted digit by digit with the same public key pk.

6.2.1 Binary integer arithmetic algorithms

A binary ($t = 2$) plaintext space will naturally support binary arithmetic. Although $t = 2k$ ($k > 1$) can also be used, an additional reduction modulo 2 is necessary to produce the bits of the final output, and more importantly, the input to C_{out} during computation.

Furthermore, larger t does not seem to provide any advantage for our approach. Therefore, we avoid this complication by restricting the scope of our experiment to $t = 2$. This is usually not a problem since this parameter can be set manually and most demos in HElib and applications of FHE have binary plaintext space. Other bases will be discussed in Section 6.2.3.

In the following algorithms, the numbers are in two's complement representation for the ease of subtraction and division. The number of digits in A is denoted by $|A|$.

Addition, Subtraction and Multiplication

We implement addition as Algorithm 1 so that it can be reused in subtraction with two's complement representation. To subtract, we just need to negate the subtrahend and pass the minuend as A, the negation of subtrahend as B, and $C_{-1} = \mathsf{Enc}(pk, 1)$ to the algorithm.

Algorithm 1: Homomorphic full adder

Input: Public key pk, FHE encrypted summands A, B, initial carry C_{-1}
Output: Sum S, highest carry C

$len \leftarrow \max\{|A|, |B|\}$;
Sign extend the shorter one of A and B by copying the most significant digit;
$C \leftarrow C_{-1}$;
for $i \leftarrow 0$ *to* $len - 1$ **do**
    $temp \leftarrow \mathsf{FHE.Add}(pk, A_i, B_i)$;
    $S_i \leftarrow \mathsf{FHE.Add}(pk, C, temp)$;
    $C \leftarrow \mathsf{FHE.Add}(pk, \mathsf{FHE.Mult}(pk, A_i, B_i),$
                $\mathsf{FHE.Mult}(pk, C, temp))$;
end
return S, C;

As for multiplication, we need to be careful with two's complement representation because in order to ensure correctness we must sign extend both multiplicand and multiplier.

Division

For division, we adopt the non-restoring division algorithm. The algorithm starts from the most significant bit of the dividend, tries to subtract the divisor from each digit and compute the quotient and remainder accordingly. To deal with the conditionals, we introduce

two auxiliary functions NOT and Cond. NOT can be implemented as adding 1 to each digit by calling FHE.addConstant; Cond is described in Algorithm 3. The division algorithm is shown in Algorithm 4.

Algorithm 2: NOT gate

Input: Public key pk, FHE encrypted value V
Output: The bit-wise negation $\bar{V}$

```
for i ← 0 to |V| − 1 do
    V̄_i ← FHE.addConstant(pk, V_i, 1);
end
return V̄;
```

Algorithm 3: Computing the conditionals

Input: Public key pk, FHE encrypted condition bit c, FHE encrypted values V_T and V_F for c being true and false, respectively, with $|V_T| = |V_F|$
Output: A value V, $V = V_T$ if c is an encryption of 1 and $V = V_F$ otherwise

```
c̄ ← NOT(pk, c);
for i ← 0 to |V_T| − 1 do
    V_i ← FHE.Add(pk, FHE.Mult(pk, c, V_{T,i}),
                      FHE.Mult(pk, c̄, V_{F,i}));
end
return V;
```

6.2.2 Complexity of the implementation

Without bootstrapping, the parameters of a homomorphic encryption scheme is directly related to the depth of the circuit, in particular the maximum number of multiplications along an execution path, which we usually call the "level" of the HE scheme with specific parameters. HElib, for example, will compute a suitable u (for the cyclotomic polynomial) given level L and other parameters.

Since the circuit is a directed acyclic graph, given the algorithms, we can derive the maximum number $L(\cdot)$ of multiplications (AND gates) by a simple iteration. Let the inputs to the gate be X and Y, and let the output of that gate be Z, then for any XOR gate (addition), $L(Z) = \max\{L(X), L(Y)\}$; for any AND gate (multiplication), $L(Z) = \max\{L(X), L(Y)\} + 1$.

Algorithm 4: Homomorphic division algorithm

Input: Public key pk, FHE encrypted dividend A, FHE encrypted divisor B
Output: Quotient Q, remainder R

$R \leftarrow A_{|A|-1}$;
Zero extend R to the same length of B;
$R \leftarrow \mathsf{Subtract}(pk, R, B)$;
$Q_{|A|-1} \leftarrow \mathsf{NOT}(pk, R_{|B|-1})$;
for $i \leftarrow |A| - 2$ *to* 0 **do**
    $B' \leftarrow \mathsf{Cond}(pk, R_{|B|-1}, B,$
        $\mathsf{FullAdder}(pk, \mathsf{NOT}(\mathsf{B}), \mathsf{FHE.Enc}(\mathsf{pk}, 1)))$;
    Truncate R to the last $|B| - 1$ digits, and concatenate A_i to R as the least significant digit;
    $R \leftarrow \mathsf{FullAdder}(pk, R, B')$;
    $Q_i \leftarrow \mathsf{NOT}(pk, R_{|B|-1})$;
end
$R \leftarrow \mathsf{FullAdder}(pk, R,$
        $\mathsf{Cond}(pk, R_{|B|-1}, B, \mathsf{FHE.Enc}(pk, 0)))$;
return Q, R;

Below we analyze the integer addition and division algorithms. Analysis for other algorithms are similar.

Addition

For convenience of analysis, we label C in the i-th iteration C_i. According to the above formulae, in each iteration,

$$\begin{aligned} L(S_i) &= \max\{L(A_i), L(B_i), L(C_{i-1})\}, \\ L(C_i) &= \max\{L(A_i), L(B_i), L(C_{i-1})\} + 1 \\ &= L(S_i) + 1. \end{aligned} \tag{6.1}$$

Recurrences (6.1) can be further simplified to

$$\begin{aligned} S_i &= \max_{0 \le j \le i}\{\max\{L(A_j), L(B_j)\} + (i-j)\}, \\ C_i &= \max_{0 \le j \le i}\{\max\{L(A_j), L(B_j)\} + (i-j)\} + 1. \end{aligned} \tag{6.2}$$

It is not difficult to see that C_i dominates the level. If both A and B are fresh encryptions ($L(A_i) = L(B_i) = 0$ for all i), then $L(C_i) = i + 1$; furthermore, if the longer of A and B has n digits, then the carry from the most significant digits will have level n.

Division

Division is slightly more complicated, but the same computation with iteration still applies. Note that our NOT implementation does not increase the level.

We label B' and R computed the i-th iteration B'_i and R_i, respectively. Note that division starts from the most significant digit. Then we can apply the conclusions for integer addition above and obtain

$$\begin{aligned} L(R_{|A|-1,k}) &= \max\left\{\begin{array}{l} L(A_{|A|-1}) + k, \\ \max\limits_{0\le j\le k}\{L(B_j) + (k-j)\}\}\end{array}\right\}, \\ L(Q_{|A|-1}) &= L(R_{|A|-1,|B|-1}) \end{aligned} \tag{6.3}$$

in the initialization. In each subsequent iteration, due to the conditional computation of B', we have

$$L(B'_{i,k}) = \max\left\{\begin{array}{l} L(B_k), L(R_{i+1,|B|-1}), \\ \max\limits_{0\le j\le k}\{L(B_j)\} + (k-j)\end{array}\right\} + 1. \tag{6.4}$$

Therefore, the level of Q_i and R_i can be determined by

$$\begin{aligned} L(R_{i,k}) &= \max\left\{\begin{array}{l} \max\limits_{1\le j\le k}\{L(R_{i+1,j-1}), L(B'_{i,j})\} \\ \qquad\qquad\qquad\qquad + (k-j), \\ \max\{L(A_i), L(B_0)\} + k \end{array}\right\}, \\ L(Q_i) &= L(R_{i,|B|-1}). \end{aligned} \tag{6.5}$$

The final remainder R has level

$$L(R_k) = \max_{0\le j\le k}\left\{\max\left\{\begin{array}{l} L(R_{0,j}), \\ L(R_{0,|B|-1}) + 1, \\ L(B_j) + 1 \end{array}\right\} + (k-j)\right\}. \tag{6.6}$$

In (6.4) and (6.5), $R_{i,|B|-1}$ dominates the level. If both A and B are fresh encryptions, and

let $\hat{R}_i$ denote $R_{i,|B|-1}$, then

$$\begin{aligned} L(Q_{|A|-1}) &= L(\hat{R}_{|A|-1}) = |B| - 1, \\ L(B'_{i,k}) &= L(\hat{R}_{i+1}) + 1, \\ L(Q_i) &= L(\hat{R}_i) = L(\hat{R}_{i+1}) + |B|. \end{aligned}$$

For the final remainder R:

$$L(\hat{R}) = L(\hat{R}_0) + |B|.$$

Further simplifying the above formulae and taking the highest level only, we have

$$\begin{aligned} L(Q_0) &= L(\hat{R}_0) = |A||B| - 1, \\ L(\hat{R}) &= (|A| + 1)|B| - 1. \end{aligned}$$

This also indicates that we can save some levels if we do not need the remainder.

6.2.3 Nonbinary integer arithmetic

When our arithmetic is not binary (namely, the base $b > 2$), we can still build our algorithms, but we have to be careful since we are operating on a ring modulo b and the usual binary circuit no longer applies in this case. For example, to implement a ternary full adder, we can first determine the formulae for each digit in the sum and carry through polynomial interpolation.

However, there is an important difference between binary and nonbinary implementations: the complexity estimation will change with the base b. For example, if we use the same full adder design, then for $b = 3$, we have

$$\begin{aligned} C_{out} = {} & A \cdot B \cdot C_{in} \\ & - A^2 \cdot (B + C_{in}) - B^2 \cdot (A + C_{in}) - C_{in}^2 \cdot (A + B) \\ & - (A \cdot B + B \cdot C_{in} + A \cdot C_{in}), \end{aligned}$$

whose degree is 3, which means every carry has two more multiplication levels than the input. Similarly, for $b = 5$, we have a degree 5 polynomial for carry, which increases the level of multiplications of each carry by 4.

With a multiplication level of about 4, we can compute 4-bit addition for $b = 2$, 2-trit addition for $b = 3$, and only 1-digit addition for $b = 5$. From this small empirical example we can observe that representing numbers in a nonbinary form does not necessarily slow down the noise growth relative to the size of operands, nor does it reduce the parameter size, since the noise tolerance has a significant influence in parameter selection. A formula to predict the degree of C_{out} with respect to the base b is left as future work.

6.2.4 Bootstrapping

As we can see from the above computation, in order to perform division of integers with a moderate number of digits, we must select HE parameters such that the scheme can evaluate functions with a high level of multiplications. Such parameters will in turn lead to large key size and slow evaluation. One way to overcome this problem is bootstrapping.

In [61], Halevi and Shoup implement and test bootstrapping of HElib. Although their method pose special requirements to the parameter u, they comment that such u's are not difficult to find. They also include a table which shows that for various settings, where the scheme evaluates up to 20–24 levels, bootstrapping requires only 10–13 levels. Hence if bootstrapping is used, the minimum overhead it introduces is about 50% of the computation. The problem of optimizing the placement of bootstrappings has also attracted some research, for example [69]. It is also possible to find optimal solutions for the evaluation of specific algorithms such as the integer arithmetic in this chapter, but we leave this as a future work.

6.2.5 Application to data aggregation

With integer arithmetic, we can perform some of the basic statistical analyses on encrypted data, e.g., average, standard deviation, and correlation. The extra work is just computing the level. Minimizing the number of multiplication levels through alternative implementations, such as those exploiting bitwise operation, is another interesting problem and is somehow related to hardware design.

Taking average as an example: if we have two 3-bit integer and want to calculate the average, one way is to directly apply addition and division algorithms, where the division requires roughly level-10 evaluation capability; a shortcut would be finding the sum with the addition algorithm and then truncating the last bit (equivalent to a right shift in terms of bitwise operation). However, the shortcut will not work if we want to find the average of three integers, but the division algorithm applies for general cases: to compute the average of three 3-bit integers, we calculate the sum with addition algorithm, and then we encrypt 3 bit by bit and feed the sum and the encryption of 3 into the division algorithm to obtain the average. This approach requires about 7 levels for addition and 5 levels for division.

6.3 Performance

6.3.1 Parameter selection

HElib provides many parameters to fine-tune the system and make compromise between security and performance: security level ℓ; level of ciphertexts L; prime p and exponent r for the base ring $\mathbb{Z}/p^r\mathbb{Z}$ of plaintext space; SIMD related parameters, including the number of plaintext slots s and the degree of each slot d. Given these parameters, HElib can automatically pick a suitable u (u-th cyclotomic polynomial is used to define the plaintext and ciphertext space) by computing a bound of $\phi(u)$ with the aforementioned parameters.

The slight difference between the estimated L in Section 6.3.2 and that passed to HElib is caused by the definition of L: what we compute in Section 6.2.2 is the number of multiplications, while in HElib L refers to the "level of ciphertext". Each multiplication switches the level of the ciphertext once, e.g., to accommodate 4 multiplications, we need 5 levels of ciphertexts available.

In our experiment below, there is no restriction on d or s; we choose parameters $\ell = 128$, L according to our estimation in Section 6.2.2, $p = 2$, and $r = 1$ and let HElib decide u for us. The prime factors of the moduli chain of each parameter setting is shown in Table 6.3. The construction of moduli, by default of HElib, uses a small prime p_0 and a few large primes $p_1, \ldots, p_k$. The large primes are of roughly the same size and p_0 has about half the number of bits of the large primes. The chain of moduli altenates between moduli with p_0 as a factor and those without:

$$\begin{aligned} q_L &= p_0 p_1 \cdots p_k, \\ q_{L-1} &= p_1 \cdots p_k, \\ q_{L-2} &= p_0 p_1 \cdots p_{k-1}, \\ q_{L-3} &= p_1 \cdots p_{k-1}, \\ &\vdots \end{aligned}$$

6.3.2 Timing of integer arithmetic with HElib

Table 6.2 shows the actual timing for integer arithmetic operations with HElib. The experiments are run on a machine that has 8 Intel Xeon E7-L8867 2.13 GHz processors, each with 10 cores, and 512 GB RAM. However, each experiment can use only one core

# Bits	L	u	Timings (s)			
			Add.	Sub.	Mult.*	Div.
2	3	4051	0.042425	0.087679	-	-
	5	5461	0.141112	0.266649	0.572857	-
	7	7781	0.233375	0.429262	0.893516	1.6482
3	4	4051	0.07215	0.138559	-	-
	7	7781	0.390965	0.665168	2.20332	-
	13	13981	1.33441	2.37028	7.93776	12.4933
4	5	5461	0.334383	0.563787	-	-
	9	8191	0.720608	1.21162	5.7184	-
	21	18631	5.80991	10.0834	47.327	67.9477

Table 6.2: Timings for integer arithmetic at security level 128, binary plaintext. *Multiplication is implemented as unsigned

due to the lack of thread safety of NTL. In all the operations, both operands are fresh encryptions and have the same number of digits, shown in the first column; L is taken as the minimum level to ensure correct evaluation for each function in HElib. We also note that these experiments do not exploit SIMD: they use only one slot in the plaintext. If we pack multiple plaintexts into one, then we can perform the same operation simultaneously on all the slots and attain a much shorter amortized running time.

From the "Add" column in Table 6.2 we can see that the execution time for the same operation increases quickly with the parameters L and u. We have also shown in the analysis in Section 6.2.2 that the level of multiplications is proportional to the number of digits in addition, but for division it shows a quadratic growth, which in turn requires much larger parameters with just a slight increase in the input size. The parameters that HElib can handle are bounded (in its parameter-finding routine, it immediately gives up when $\phi(u)$ is beyond a bound posed by NTL). These facts combined shows that the division of large numbers cannot be supported without bootstrapping, even with reduced amortized running time brought by SIMD operations.

6.4 Conclusion and Future Work

Our work shows that the homomorphic evaluation of precise binary integer arithmetic and, furthermore, data aggregation functions can be practical only with small numbers. SIMD operations can be used to parallelize the computation for an better amortized performance.

# Bits	L	u	Prime Factors of Moduli			
2	3	4051	2333377	4893444145153	32079244951553	
	5	5461	2099089	14673688657921	11412868956161	127824132308993
			119998165024769			
	7	7781	3050153	17231744335873	11748916592641	11226742521857
			131587865837569	119055688138753		
3	4	4051	2333377	4893444145153	32079244951553	19573776580609
	7	7781	3050153	17231744335873	11748916592641	11226742521857
			131587865837569	119055688138753		
	13	13981	2348809	14073735413761	11962675101697	9851614789633
			8678803505153	7740554477569	7036867706881	454464372737
			432474161153	366503526401		
4	5	5461	2446529	16491667783681	10994445189121	105546673815553
			52773336907777			
	9	8191	2293481	16490661150721	13742217625601	9619552337921
			7970486222849	2013020161	1945919489	1543315457
	21	18631	2459293	15003662942209	14378510319617	13440781385729
			8439560404993	14847374786561	11721611673601	4532356513793
			1406593400833	16644688576513	14612942553089	2070818062337
			1992673984513	1758241751041	1484737478657	1133089128449

Table 6.3: Prime factors of the moduli chain

When we need to deal with large numbers, bootstrapping must be included to fix the problem of growing parameters. The effect of using nonbinary bases is complicated and needs more study. One extension of this work could be the computation over real numbers, e.g., floating-point numbers with FHE.

Compared with other techniques such as data obfuscation and the one in [79], this approach has the advantage of flexibility in that it can be easily adapted to support any function, in particular those with conditionals. The disadvantage is that the computational cost is relatively high. How to choose among them and how to select parameters for them in order to achieve the optimal trade-off remain interesting directions to explore.

Chapter 7

Homomorphic Attacks against RLWE

7.1 Introduction

Since Regev proposed Learning with Errors (LWE) [86], it has found many applications in cryptography. It is conceptually simple but also enjoys worst-case hardness like some other lattice problems [86, 26]. Unfortunately, with the benefits of LWE usually come large keys and ciphertexts that add to communication and storage overhead. In the search of a better alternative that is more competitive in terms of key size, Ring-LWE (RLWE) emerged [72] as a promising candidate. RLWE accomplishes its higher efficiency by exploiting additional algebraic structure of polynomial rings, but this approach is followed by attacks that can find weak instances, for the same reason. Similar to LWE, RLWE enjoys worst-case hardness, but only on ideal lattices.

Due to its efficiency, RLWE has been applied to many cryptographic constructions [27, 49, 22]; therefore, it has become more urgent to understand its strengths and weaknesses.

7.1.1 Related Work

Since RLWE can be reduced to lattice and LWE problems, all lattice reduction attacks such as LLL and general decoding attacks such as BKW apply to RLWE. However, people generally believe and the weak-instance attacks described below show that the RLWE problem is potentially easier to tackle than the LWE problem because of the additional algebraic structure.

Weak instances of RLWE are analogous to weak primes in factorization: their special properties significantly reduce the difficulty of the problem so that specialized algorithms

can be designed to launch an attack. Weak RLWE instances usually involve some ring homomorphism, under which the image of the error distribution can be distinguished from a uniform distribution.

Generally, weak instance attacks involve three steps:

1. Exploiting the algebraic property to reduce the search space for Step 2;
2. Exhausting the secret s;
3. Testing if the samples agree with a certain distribution generated with the guessed secret.

Considering different rings with special properties, leads to various weak instance attacks.

Algebraic Structures

The first such attack was developed by Eisenträger et al. [46] on Polynomial-LWE (Poly-LWE or PLWE), which is similar to RLWE but uses coefficient embedding instead of canonical embedding (see Section 2.2.2 for more details). They consider polynomial rings of the form $\mathbb{Z}_p[x]/(f(x))$, where p is prime and $f(x)$ is irreducible over $\mathbb{Q}$ but has a low-order root α modulo p. Their attack exploits a ring homomorphism induced by α into the finite field $\mathbb{F}_p$. When p is small enough, it becomes feasible to search for the image of secret s in $\mathbb{F}_p$.

Similar attacks were soon carried out on RLWE cases [48, 31]. The conditions for their attacks are similar, but the attacker is faced with bigger distortions. The distortion is introduced by the conversion of ring element representations from canonical embedding to coefficient embedding. This conversion is necessary to prepare for the evaluations of polynomials at α, in order for the original attack to work.

A more delicate attack was delivered by Chen et al. [33, 34] on RLWE instances based on families of Galois number fields whose ring of integers can be decomposed into orthogonal subspaces, where their homomorphism will likely nullify a component of the error drawn from a discrete Gaussian distribution.

Distinguishing Distributions

In Step 3 of the attack, we need to decide which distribution fits the one computed from the guessed secret and samples better: uniform or the error distribution (under the homomorphism). All previous works achieve this by comparing the sampled distribution with

uniform under the assumption that the mapped error distribution will be far enough apart from uniform.

For this particular task, there are general purpose distinguishers available such as the Chi-square test. Another distinguisher with dual lattices is considered by Peikert [84] and can be used in conjunction with all existing weak instance attacks.

7.1.2 Our Contribution

In this chapter we consider how the homomorphism attacks proposed in [46] and [48] can be modified to work against a Poly-LWE variant where the error polynomials are defined by drawing the coefficients from independent discretized Gaussian distributions rather than using a discrete Gaussian distribution over a lattice. We show that while these attacks can be modified for this variant, several changes must be considered. In particular, the mapped error distribution to be considered may be very different from what that the algorithms in [48] are designed to handle. We show how the mapped error distribution can be computed directly and used to create effective distinguishers.

This chapter is organized as follows: Section 7.2 summarizes the RLWE problem and other related background. Section 7.3 shows the image of error distribution under the homomorphism and introduces our method to compute it. Section 7.4 discusses different methods to distinguish distributions with samples and demonstrates our simulation results.

7.2 Background

Let $f(x)$ be a monic irreducible polynomial in $\mathbb{Z}[x]$ of degree n (not necessarily cyclotomic). Let $q \in \mathbb{Z}$ (not necessarily prime). If we let $p \in \mathbb{Z}$ it will always denote a prime. For the integers modulo q we use the notation $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$. We will be working in the following polynomial ring $\mathbb{Z}_q[x]/(f(x))$ which we denote as

$$R_q := \mathbb{Z}_q[x]/(f(x)).$$

Observe that there are q^n elements in R_q that are of the form

$$d_0 + d_1x + d_2x^2 + \cdots + d_{n-1}x^{n-1} + (f(x)),$$

where $d_i \in \mathbb{Z}_q$ $0 \leq i \leq n-1$. Roots of $f(x) \pmod q$ will be denoted as $\alpha_1, \ldots, \alpha_i$ for as many roots as there are. If only one root is being considered, it may be denoted as just α.

Since a root α is in $\mathbb{Z}_q$, we can talk about its order with respect to the multiplication in $\mathbb{Z}_q$,

$$\text{ord}(\alpha) := \min \left\{ m \in \mathbb{Z}^+ \;\middle|\; \alpha^m = 1 \right\}.$$

In general we will not assume that $f(x)$ factors completely mod q; however, we will assume that $f(x)$ has at least one root α mod q.

On the ring R_q, we consider the following discrete probability distributions. Let ${}_{\mathcal{U}}R_q$ denote the discrete uniform distribution on R_q, i.e. coefficients of the polynomials coming from the discrete uniform distribution on $\mathbb{Z}_q$. Let ${}_{\chi}R_q$ denote the discretized Gaussian distribution on R_q, i.e. coefficients of the polynomials coming from a discretized Gaussian distribution on $\mathbb{Z}_q$. The precise formulation of a discretized Gaussian on $\mathbb{Z}_q$ is as follows where we try to keep the definitions consistent with [86].

Definition 25. For $\beta \in \mathbb{R}^+$ the continuous distribution Ψ_β on $[0,1)$ is obtained by sampling from a normal distribution with mean 0 and standard deviation $\frac{\beta}{\sqrt{2\pi}}$ and reducing the result mod 1. This probability distribution is given as

$$\forall r \in [0,1),\ \Psi_\beta(r) := \sum_{k=-\infty}^{\infty} \frac{1}{\beta} \cdot \exp\left(-\pi \left(\frac{r-k}{\beta}\right)^2\right).$$

Now using Ψ_β, the discretized Gaussian distribution on $\mathbb{Z}_q$ is defined as follows.

Definition 26. The discretization of a Gaussian distribution on $\mathbb{Z}_q$ (we denote as $G_{q,\beta}$) is obtained by sampling from Ψ_β and multiplying by q. This probability distribution is give by

$$G_{q,\beta}(i) = \int_{(i-\frac{1}{2})/q}^{(i+\frac{1}{2})/q} \Psi_\beta(x)dx.$$

Note that one can easily generate random values from $G_{q,\beta}$, and one can numerically approximate the probability distribution of $G_{q,\beta}$ by using approximations for the infinite sum and the integral. We now introduce the two main problems of interest.

Problem 1 (Search Poly-LWE Problem)**.** Let $s(x) \in {}_{\mathcal{U}}R_q$ be secret. The *Search Poly*-LWE *Problem* is that of finding $s(x)$ given a poly(n) number of samples of the form

$$(a_j(x), b_j(x) := a_j(x) \cdot s(x) + e_j(x)) \in R_q \times R_q$$

where $a_j(x) \in {}_{\mathcal{U}}R_q$ and $e_j(x) \in {}_{\chi}R_q$. We call $e_j(x)$ the *error*.

A related problem is to distinguish samples coming from a Search LWE Problem from uniform samples on $R_q \times R_q$, and our attack can be extended to work against this variant as well.

Problem 2 (Decision Poly-LWE Problem)**.** Given poly(n) samples from one of the following two distributions on $R_q \times R_q$ the *Decision Poly-*LWE *Problem* is to decide which distribution the samples are coming from

1. samples from a Search Poly-LWE Problem, i.e. of the form

$$(a_j(x), b_j(x) := a_j(x) \cdot s(x) + e_j(x)),$$

   where $a_j(x) \in {}_{\mathcal{U}}R_q$ and $e_j(x) \in {}_{\chi}R_q$, or

2. samples that are uniform, i.e. of the form

$$(a_j(x), b_j(x)),$$

   where $a_j(x), b_j(x) \in {}_{\mathcal{U}}R_q$.

Note that these problem definitions are a slight variant on the RLWE problem in [73, 48] as here the error is coming from a discretized Guassian distribution on $\mathbb{Z}_q$ rather than being taken from a discrete Gaussian over a lattice. Yet another variant is possible if we draw the coefficients from a bounded uniform distribution on $\mathbb{Z}_q$.

Attack

In [48] they develop the following attack against the Decision Poly-LWE Problem, which we describe here with a couple adjustments. For a polynomial $f(x)$ that is irreducible over $\mathbb{Z}$ let $p \in \mathbb{Z}$ be a prime such that $f(x)$ has at least one root α mod p. Note that in [48] the authors assume that $f(x)$ factors completely mod p which is often the case in practice so that fast multiplication can be done in the ring using the Chinese Remainder Theorem, but we will not need this assumption for our attack.

Given access to $L := \text{poly}(n)$ samples from a Decision Poly-LWE Problem

$$(a_j(x), b_j(x)) \in R_p \times R_p,$$

we want to transfer the problem to $\mathbb{Z}_p$. To do this, we will build a well defined ring homomorphism

$$\bar{\phi} : Z_p[x]/(f(x)) \to \mathbb{Z}_p.$$

Since we are assuming we have a root α of $f(x)$ mod p, we can consider the ring homomorphism

$$\begin{aligned} \phi : \mathbb{Z}_p[x] &\to Z_p, \\ y(x) &\mapsto y(\alpha). \end{aligned}$$

It is clearly well defined, and one can check it is a ring homomorphism. Moreover $(f(x)) \subseteq \mathrm{Ker}(\phi)$, which gives a well defined ring homomorphism

$$\begin{aligned} \bar{\phi} : Z_p[x]/(f(x)) &\to \mathbb{Z}_p, \\ z(x) + (f(x)) &\mapsto \phi(z(x)) = z(\alpha). \end{aligned}$$

Now we take the samples and map them according to $\bar{\phi}$:

$$(a_j(x), b_j(x)) \mapsto (a_j(\alpha), b_j(\alpha)) .$$

If the samples are coming from the Search Poly-LWE distribution, $\bar{\phi}(s) = s(\alpha)$ will be some element in $\mathbb{Z}_p$. For the attack we will guess $s(\alpha)$. For each $g \in \mathbb{Z}_p$ we assume g is the correct guess for $s(\alpha)$ and compute

$$b_j(\alpha) - a_j(\alpha)g.$$

Since both multiplication and addition are preserved by $\bar{\phi}$, this gives us

$$e_j(\alpha) = b_j(\alpha) - a_j(\alpha)g$$

for a correct guess $g = s(\alpha)$. If we can successfully distinguish the distribution of $e_j(\alpha)$ from uniform, then we can test for the correctness of our guesses.

We can now analyze the distribution of $e_j(\alpha)$ to decide which distribution the samples came from. The error polynomial $e_j(x)$ can be written as

$$e_j(x) = \sum_{i=0}^{n-1} e_{ij} x^i, \quad e_{ij} \in G_{p,\beta},$$

which when evaluated at α is

$$e_j(\alpha) = \sum_{i=0}^{n-1} e_{ij} \alpha^i, \quad e_{ij} \in G_{p,\beta}.$$

From now on we will be considering $e_j(\alpha)$ for a particular guess g and will drop the j subscript and just write

$$e(\alpha) = \sum_{i=0}^{n-1} e_i \alpha^i, \quad e_i \in G_{p,\beta}.$$

This can be simplified by considering the order of α, which we denote as r. For simplicity of notation assume r divides n. This gives

$$\begin{aligned} e(\alpha) =&(e_0 + e_r + \cdots + e_{\frac{n}{r}}) + (e_1 + e_{r+1} + \cdots + e_{\frac{n}{r}+1})\alpha + \cdots \\ &+ (e_{r-1} + e_{2r+(r-1)} + \cdots + e_{\frac{n}{r}+(r-1)})\alpha^{r-1}. \end{aligned}$$

It can be further simplified to

$$e(\alpha) = v_0 + v_1\alpha + \cdots + v_{r-1}\alpha^{r-1},$$

where

$$\begin{aligned} v_0 &= e_0 + e_r + \cdots + e_{\frac{n}{r}} \\ v_1 &= e_1 + e_{r+1} + \cdots + e_{\frac{n}{r}+1} \\ &\vdots \\ v_{r-1} &= e_{r-1} + e_{2r+(r-1)} + \cdots + e_{\frac{n}{r}+(r-1)}. \end{aligned}$$

The v_ℓ's are distributed as $G_{p,\sqrt{\frac{n}{r}}\beta}$. Now depending on the value of α and the order of α, this distribution $e(\alpha)$ may be either a discretized Gaussian or a periodic distribution, and depending on the parameters either one may be very close to a uniform.

In [48, Section 3.2 Case 2] the authors state that when α has small order ≥ 3, $e(\alpha)$ will be Gaussian given their starting choice of error for RLWE; this does not match with what we see when we start with error from a discretized Gaussian. Rather we will show in the next section that when α has small order ≥ 3, $e(\alpha)$ will be neither Gaussian nor uniform but more like a kind of periodic Gaussian (see the pictures below), hence the cosine test in [86] and [84] may not work well. We shall see that Chi-square test will suffice for our purposes.

7.3 Error Distribution

7.3.1 Computing probability distributions

In the above attacks, we need to understand the probability distribution of $e(\alpha)$. More generally, we consider

$$e = e_0 + e_1a_1 + \cdots + e_{n-1}a_{n-1} \in \mathbb{Z}_p,$$

where $a_i \in \mathbb{Z}_p$ are fixed and e_i's are chosen according to a given probability distribution on $\mathbb{Z}_p$.

What is the probability distribution of e? We shall see below that, in general, if n is large or a_i's are large, then e is approximately uniform random on $\mathbb{Z}_p$. We first show how to compute the exact distribution of e in time $O(np^2)$, and then demonstrate several possible distributions of e on $\mathbb{Z}_p$, including distributions that are neither Gaussian nor uniform.

Lemma 7.3.1. *Let u and v be independent random variables on $\mathbb{Z}_p$ with probability distributions $(a_0, a_1, \ldots, a_{p-1})$ and $(b_0, b_1, \ldots, b_{p-1})$, respectively. Let*

$$a(x) = \sum_{i \in \mathbb{Z}_p} a_i x^i, \qquad b(x) = \sum_{i \in \mathbb{Z}_p} b_i x^i.$$

Then the probability distribution of $u + v$ can be computed as the coefficients of the polynomial $a(x)b(x) \pmod{x^p - 1}$.

The proof is simple since, for any $k \in \mathbb{Z}_p$, the probability

$$P(u + v = k) = \sum_{i \in \mathbb{Z}_p} P(u = i)P(v = k - i \bmod p) = \sum_{i \in \mathbb{Z}_p} a_i b_{k-i},$$

where the subscript $k - i$ of b is computed modulo p.

Theorem 7.3.2. *Suppose $e_0, e_1, \ldots, e_{n-1}$ are independent random variables on $\mathbb{Z}_p$ with the same probability distribution $(c_0, c_1, \ldots, c_{p-1})$. Let $c(x) = \sum_{i \in \mathbb{Z}_p} c_i x^i$. Then, for any $a_1, \ldots, a_{n-1} \in \mathbb{Z}_p$, the probability distribution of $e = e_0 + e_1 a_1 + \cdots + e_{n-1} a_{n-1} \bmod p$ can be computed as the coefficients of the polynomial*

$$c(x)c(x^{a_1}) \cdots c(x^{a_{n-1}}) \pmod{x^p - 1}.$$

The theorem follows from the above lemma, since the random variables $e_0, e_1 a_1, \ldots, e_{n-1} a_{n-1}$ are independent and $c(x^{a_i})$ represents the probability distribution of $e_i a_i$. Also, the product can be computed by a simple loop: $t := c(x)$; and for i from 1 to $n - 1$ do $t := t \cdot c(x^{a_i}) \bmod x^p - 1$. This takes at most $O(np^2)$ operations in $\mathbb{Z}_p$.

7.3.2 Intuition for the error distribution

In this section we give several examples of the computed distribution

$$(a_0e_0 + a_1e_1 + \cdots + a_{n-1}e_{n-1}) \bmod p,$$

where e_i, $0 \leq i < n$, are independent identically distributed $G_{p,\beta}$ distributions on $\mathbb{Z}_p$ and $a_i \in \mathbb{Z}_p$, $0 \leq i < n$, are fixed constants. The number of terms n and the sizes of the a_i's greatly affects the shape of the above distribution. To get an idea of what one should expect, we look at three general cases.

Case 1: small coefficients

The first case we consider is when all the $a_0, ..., a_{n-1}$ coefficients are 1 and the standard deviation is fixed; in this case we consider how varying n affects the shape of the distribution. As n grows large, the distribution remains Gaussian but approaches uniform.

Example 1. Let $\beta = 0.01$, $p = 331$. The distribution $e_1 + e_2 + \cdots + e_n$ for $n = 1, 20, 40, 100$ with e_j iid $G_{p,\beta}$ is shown in Figure 7.1.

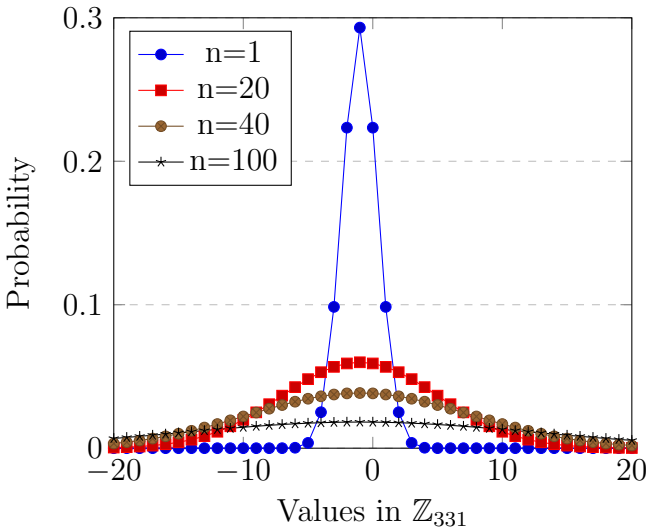

Figure 7.1: Sum of n iid discretized Gaussian distributions on $\mathbb{Z}_{331}$.

Case 2: large coefficients

The second case we consider is when all the $a_0, ..., a_{n-1}$ coefficients are large and the standard deviation is fixed; in this case we consider how varying n affects the shape of the

distribution. We want to answer the question of how large n needs to be for the distribution to be almost uniform. In this case n can be quite small and the distribution already be very close to uniform. However, note that the distributions in the following example are neither Gaussian nor uniform.

Example 2. We consider the distributions of $23e_0 + 45e_1$ and $23e_0 + 45e_1 + 43e_2$ and $23e_0 + 45e_1 + 43e_2 + 95e_3$ where $\beta = 0.01$ and $p = 331$ and e_j iid $G_{p,\beta}$. For each additional term in the sum the distributions gets considerably closer to uniform while remaining periodic. The three graphs are shown in Figure 7.2.

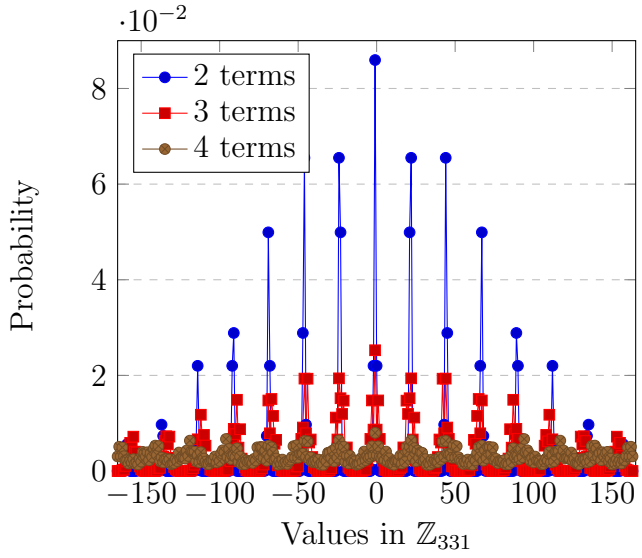

Figure 7.2: The distributions of $23e_0 + 45e_1$ and $23e_0 + 45e_1 + 43e_2$ and $23e_0 + 45e_1 + 43e_2 + 95e_3$ where e_j are iid discretized Gaussians on $\mathbb{Z}_{331}$

Case 3: root of small order

In the third case we consider a situation where the coefficients are all powers of a root α

$$e_0 + \alpha e_1 + \alpha^2 e_2 + \alpha^3 e_3 + \cdots + \alpha^{n-1} e_{n-1},$$

where e_j are sampled from i.i.d. discretized Gaussian distributions. We want to specifically consider what effect the order of α has on the distribution. We choose $f = x^n + ax + b$ to be an irreducible polynomial over $\mathbb{Z}$ that has a root α mod p.

We want to specifically consider the case when α has small order mod p and n is not too large. When α has low order, the distribution will be considerably farther from uniform compared to when α has large order. Consider the following example where f has one root of low order and another of high order.

Example 3. The polynomial $f = x^9 + 11x - 11$ is irreducible over $\mathbb{Z}$ but has a two roots $\alpha_1 = 31$, $\alpha_2 = 82 \bmod 331$ with α_1 having order 3 and α_2 having order 165. Consider the following distribution which arises if α_1 is used to define the homomorphism in the attack

$$e_0 + \alpha_1 e_1 + \alpha_1^2 e_2 + \alpha_1^3 e_3 + \cdots + \alpha_1^8 e_8.$$

Here we are assuming e_j are iid $G_{p,\beta}$ for $\beta = 0.01$. The graph of the distribution is show in Figure 7.3. It is neither Gaussian nor uniform.

When using the root α_2 with larger order, we see that the distribution $e_0 + \alpha_2 e_1 + \alpha_2^2 e_2 + \alpha_2^3 e_3 + \cdots + \alpha_2^8 e_8$ is much closer to uniform as seen in Figure 7.4.

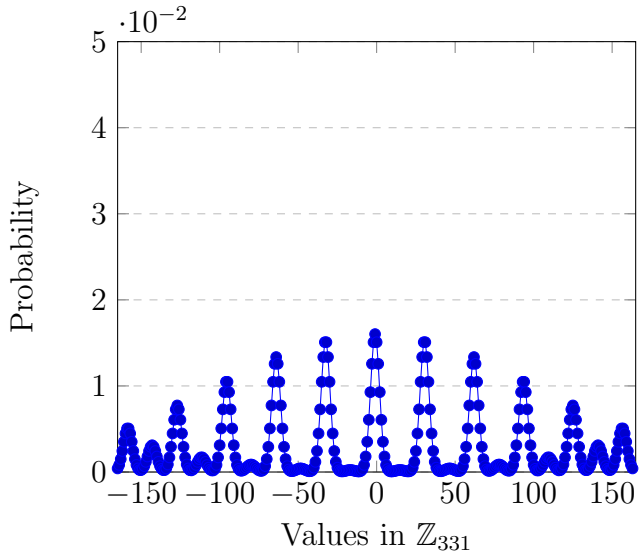

Figure 7.3: The distribution of $e_0 + \alpha_1 e_1 + \alpha_1^2 e_2 + \alpha_1^3 e_3 + \cdots + \alpha_1^8 e_8$ for $\alpha = 31$ of order 3 and e_j iid discretized Gaussians on $\mathbb{Z}_{331}$.

Summarize Cases

Considering these three cases, we note that if the distribution $e(\alpha)$ appears like Case 1 it can be fairly easily distinguished from the uniform for small n. For Case 2 distinguishing

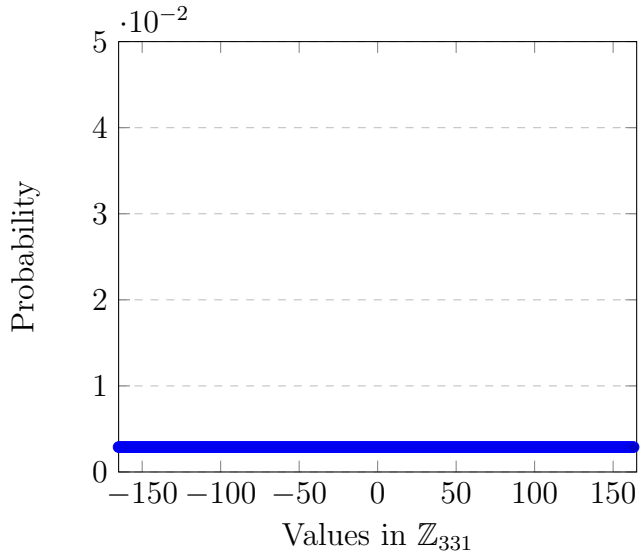

Figure 7.4: The distribution of $e_0 + \alpha_2 e_1 + \alpha_2^2 e_2 + \alpha_2^3 e_3 + \cdots + \alpha_2^8 e_8$ for $\alpha_2 = 84$ of order 165 and e_j iid discretized Gaussians on $\mathbb{Z}_{331}$.

from the uniform gets much harder because when adding a large number terms with large coefficients, the distribution rapidly approaches uniform. For Case 3 when α has small order this is similar to Case 2 with a small number of terms, but if the order of α is small enough and n not too large, we can hope to be able to distinguish this from uniform; we will consider this case further in the next section. In Case 3 when α has large order, this is similar to Case 2 when there are a large number of terms, and it is unlikely one would be able to distinguish this from uniform.

7.4 Statistical Tests and simulation

7.4.1 Distinguishing statistical tests

In this section we will discuss how one can decide if a guess g in the attack is correct. If the guess g is correct, we have shown in the previous section how we can compute what the distribution of the error will be; we denote this computed error distribution as $\mathcal{E}$. If the guess g is not correct then the samples will be uniform, which we denote as $\mathcal{U}$.

Assume that we have L samples of RLWE public keys:

$$(a_i(x), b_i(x)), \quad b_i(x) = a_i(x)s(x) + e_i(x), \quad i = 1, \ldots, L,$$

where $a_i(x) \in {}_{\mathcal{U}}R_p, e_i(x) \in {}_{\chi}R_p$. Let A and B be two random variables taking samples

$$A \in \{a_i(\alpha) \mid i = 1, \ldots, L\}$$
$$B \in \{b_i(\alpha) \mid i = 1, \ldots, L\}.$$

In general, (A, B) cannot be distinguished from (A, B'), where B' is uniform. But the homomorphism attack lets us the consider the distribution

$$\mathcal{S}(g) := B - Ag, g \in \mathbb{Z}_p.$$

Property 1. *$\mathcal{S}(g) \sim \mathcal{E}$ if $g = s(\alpha)$; otherwise $\mathcal{S}(g) \sim U$ if $g \neq s(\alpha)$.*

Now if $\mathcal{E}$ is not too close to $\mathcal{U}$, we will be able to decide which one $\mathcal{S}(g)$ matches by considering a reasonable number of samples. There are several methods one might use to decide which distribution $\mathcal{S}(g)$ fits.

Method 1: Chi-square Tests

First we will use a Chi-Square goodness-of-fit test to test the $\mathcal{S}(g)$ against a uniform distribution.

Let $\tilde{e}_k$ denote the number of $e_j(\alpha)$'s equal to k mod q. Let L denote the total number of samples. We set up our null hypothesis to be that $\mathcal{S}(g)$ is distributed according to a uniform distribution

$$H_0 : \mathcal{S}(g) \sim \mathcal{U},$$
$$H_1 : \mathcal{S}(g) \not\sim \mathcal{U}.$$

Then we compute the Chi-square statistic as

$$V = \sum_{k=1}^{p-1} \frac{(\tilde{e}_k - L/p)^2}{L/p},$$

where there are $q-1$ degrees of freedom. If V is too large or too small based on our choice of Type 1 error rate, we reject H_0. For this test to be considered reliable we need $L/p \geq 5$ samples, i.e. $L \geq 5p$.

Second, we test $S(g)$ against our computed distribution $\mathcal{E}$ using a Chi-squared test in a non-traditional way.

$$H_0 : S(g) \sim \mathcal{E},$$
$$H_1 : S(g) \not\sim \mathcal{E}.$$

Let $\mathcal{E}_k := Pr[\mathcal{E} = k \in \mathbb{Z}_p]$. Then we compute the Chi-square statistic as

$$V = \sum_{k=1}^{p-1} \frac{(\tilde{e}_k - L\mathcal{E}_k)^2}{L\mathcal{E}_k},$$

where there are $p-1$ degrees of freedom. We then reject or accept based on V and our desired Type 1 error rate. In this setup, we are inverting the usual setup for a Chi-square test. Here we are making the probability of rejecting when we should not equal to the Chi-squared test's Type 2 error. But this problem is also unusual in that it is a promise problem, giving us that the $S(g)$ must be one of two know distributions, which is not an assumption generally considered for most statistical tests like the Chi-square test.

Method 2: Expected Value of Cosine

A distinguisher, which is described by Regev in [86] and which Peikert in [84] uses to describe a general distinguisher working with the homomorphism attacks of [46, 48, 33, 34], can also be modified to work with this homomorphism attack, but in our case knowing the computed error distribution can let one choose the optimal threshold value for test. The test is to compute the following expected value: n

$$\bar{x} := \underset{e_j(\alpha) \in \mathcal{S}(g)}{E} \left[\cos(2\pi e_j(\alpha))\right].$$

If $S(g) \sim \mathcal{U}$, this should be approximately 0. If $S(g) \sim \mathcal{E}$ and $\mathcal{E}$ is not too close to $\mathcal{U}$, this may be non-zero and can be used to distinguish. Since we know the computed error distribution $\mathcal{E}$, we can compute

$$\mu := \underset{e \in \mathcal{E}}{E} \left[\cos(2\pi e)\right]$$

and use this to determine the best threshold for the test $T := \mu/2$. Then if $\bar{x} \leq T$ we decide that $S(g) \sim \mathcal{U}$, and if $\bar{x} > T$ we decide that $S(g) \sim \mathcal{E}$. However, if the error is periodic, this test will not necessarily perform well.

Method 3: Statistical Distance

For another test making use of the computed error distribution, we consider the statistical distance from $S(g)$ to $\mathcal{U}$ and $\mathcal{E}$ for each guess g and take the guess for which $S(g)$ is closest to $\mathcal{E}$. Let the distribution of $\mathcal{S}(g)$ be $\{t_i(g) \mid i \in \mathbb{F}_p\}$, i.e.,

$$P[\mathcal{S}(g) = i] = t_i(g), \quad i \in \mathbb{F}_p.$$

Let

$$\varepsilon(g) = \sum_i \left| t_i(g) - \frac{1}{p} \right|,$$
$$\delta(g) = \sum_i \left| t_i(g) - \mathcal{E}_k \right|.$$

Decode $s(\alpha) = g$, where

$$g = \arg\max_{g \in \mathbb{F}_p} \varepsilon(g) \tag{7.1}$$

or

$$g = \arg\min_{g \in \mathbb{F}_p} \delta(g). \tag{7.2}$$

This gives us a good idea of what the correct guess probably is if the samples are from Poly-LWE. If no one guess differed from the rest by more than some threshold, we would decide the samples were uniform. However, how this threshold should be chosen is unclear at this moment. We will leave that as a future work.

Type 1 Errors

One further thing that must be considered when using any tests like Methods 1 and 2 with a fixed Type 1 error probability is that using the test repeated for each of the p guesses will result in a much higher overall Type 1 error probability. To see this in detail, if γ is set to be the Type 1 error probability for a single test, then $1-\gamma$ is the probability of not having a Type 1 error on that test. If one runs p such tests, the probability of no Type 1 errors in all p tests is $(1-\gamma)^p$. Thus the probability of at least one Type 1 error is $1-(1-\gamma)^p$; to be clear, this is an upper bound on probability of a Type 1 error and in practice a test may have a true Type 1 error rate much lower, but we may no longer have a very good upper bound.

To see how much the Type 1 error bound can grow, consider that if $\gamma = 0.02$ and $p = 331$, then $1-(1-0.02)^{331} \approx 0.9987$.

It is possible to keep the overall Type 1 error bound to an desired rate. One way is to use a Bonferroni correction, which is a way of setting the Type 1 error rates on the individual tests to guarantee a particular overall Type 1 error. In particular if set the new Type 1 error for each test at $\beta_\gamma := \frac{\gamma}{p}$, the overall Type 1 error will still be bounded by γ. However, this may result in an impractically high Type 2 error rate, so in practice we would recommend using the tests at multiple levels and using a variety of tests as described above.

7.4.2 Simulations

We simulate these Chi-square methods and look at the there true Type 1 and Type 2 error rates, the expected value of cosine test does not perform well on either of these two examples.

Example 4. Continuing with the earlier Example 3, we show how we can use Method 1 of the Chi-square tests to tell for which guess of g the distribution $e(\alpha)$ follows the computed error distribution rather than a uniform distribution. With 2000 samples and the individual Chi-Square tests' Type 1 error rates set at 2%, the overall test is successful at rejecting the null hypothesis that the error are from the uniform every time out of ten independent simulations while only every giving two false rejections.

For those same ten simulations the Chi-square test against the uniform with Type 1 error rates set at 1×10^{-9} (corresponding to this tests Type 2 error rates because of the inverted setup) rejects all guesses correctly without giving any false rejections.

Example 5. Next we consider a new example. Let $f = x^{15}+125x-334$ which is irreducible over $\mathbb{Z}$ but has a root $\alpha = 396$ of order 3 mod 607. The distribution $\mathcal{E}$ is shown in Figure 7.5; clearly it is neither Gaussian nor uniform.

Using the Chi-square test against uniform with 5000 samples and having the individual tests set at a Type 1 error rate of 2%, we are able to reject the correct the guess correctly for every one of ten independent simulations while only every giving one false rejection.

For those same ten simulations the Chi-square test against the uniform with Type 1 error rate set at 1×10^{-9} (corresponding to this tests Type 2 error rate because of the inverted setup) is not helpful on this example as it never rejects anything.

Example 6. We used Method 3 (statistical distance) on the following instance: $f(x) = x^n + p - 1$, where $n = 8$, $p = 257$, and $\beta = 0.2$. In this setup, $f(x)$ has root $\alpha = 1 \bmod p$. With 1200 samples, the test is successful at finding the image of s in all of the ten independent simulations, when either of (7.1) and (7.2) is used.

This instance has been attacked successfully by [46], because of its simplicity, our method may be able to improve the efficiency of their attack by computing $s(\alpha)$ directly from the error distribution under the homomorphism. We leave that as a future work.

7.5 Conclusion

In summary, we have shown how the homomorphism attacks from [46, 48] can be extended to another variant of the Poly-LWE problem with several modifications. The main dif-

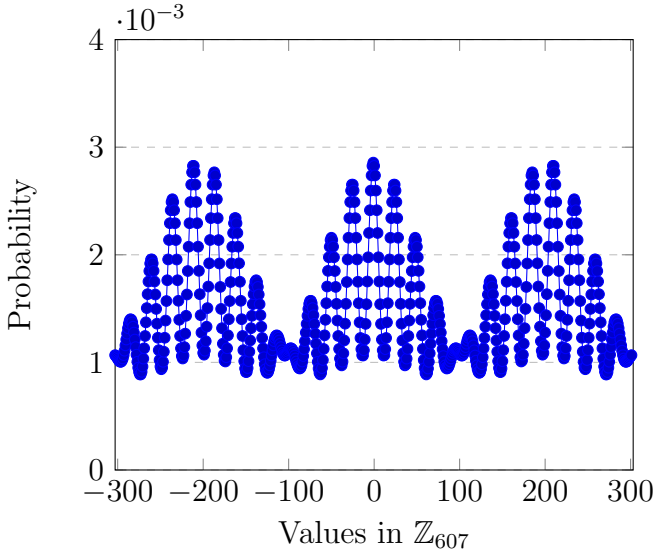

Figure 7.5: The distribution of $e_0 + \alpha e_1 + \alpha^2 e_2 + \alpha^3 e_3 + \cdots + \alpha^{12} e_{12}$ for $\alpha = 396$ of order 3 and e_j iid discretized Gaussians on $\mathbb{Z}_{607}$.

ference being that the mapped error distributions in the two variants behave differently requiring new approaches to distinguishing the distributions. By computing the probability distribution directly and using Chi-squared tests, we have shown how such distinguishing can be accomplished.

7.5.1 Future Work

Although we have discussed different statistical tests, their relative accuracy has not been thoroughly analyzed. We will leave that as future work. In another future work, we can consider a potential improvement to the weak-instance attack in [46] and [48], where we may multiply $f(x)$ by another polynomial $g(x)$, such that the number of terms in $f(x)g(x)$ is smaller than the order of α and $f(x)g(x)$ has small coefficients.

Chapter 8

Conclusions and Future Work

FHE has come a long way. With new, efficient schemes becoming more and more available, the future of FHE applications looks very promising. But we should not ignore the caveats: most of the foundations and techniques used in FHE had been developed in the past two decades, and they are not as well understood as some classical problems used in cryptography. The randomness ubiquitous in FHE designs makes the calculation of exact security level difficult. KDM security may be a weak link in many popular FHE schemes based on LWE and RLWE. Large gaps still exist between theory and reality.

Our work shows that in BGV, homomorphic evaluation of integer arithmetic and, furthermore, data aggregation functions can be practical only with small numbers. SIMD operations can be used to parallelize the computation for an better amortized performance. When we need to deal with large numbers, bootstrapping must be included to fix the problem of growing parameters. Schemes that support real numbers, such as CKKS [37], may improve the performance and can be investigated. Overall, choosing the right scheme to suit the need of the nature of secure computation in any project, instead of looking for a one-size-fits-all solution, is still the current strategy for applications.

We have also shown that even though lattice problems are generally **NP**-hard, a particular lattice instance may still have structural weaknesses, especially for those chosen for efficiency reasons. Further investigatoin is urgently needed to assess the security level of lattices. And users of FHE schemes should pay close attention to ensure proper parameters are selected.

References

[1] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *ACM Sigmod Record*, volume 29, pages 439–450. ACM, 2000.

[2] Miklós Ajtai. The shortest vector problem in l 2 is np-hard for randomized reductions. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 10–19. ACM, 1998.

[3] Ahmad Al Badawi, Louie Hoang, Chan Fook Mun, Kim Laine, and Khin Mi Mi Aung. Privft: Private and fast text classification with homomorphic encryption. *IEEE Access*, 8:226544–226556, 2020.

[4] Ahmad Al Badawi, Bharadwaj Veeravalli, Chan Fook Mun, and Khin Mi Mi Aung. High-performance fv somewhat homomorphic encryption on gpus: An implementation using cuda. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 70–95, 2018.

[5] Navid Alamati and Chris Peikert. Three's compromised too: Circular insecurity for any cycle length from (ring-) lwe. In *Annual International Cryptology Conference*, pages 659–680. Springer, 2016.

[6] Martin R Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin E Lauter, et al. Homomorphic encryption standard. *IACR Cryptol. ePrint Arch.*, 2019:939, 2019.

[7] Martin R Albrecht and Amit Deo. Large modulus ring-lwe $\geq$ module-lwe. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 267–296. Springer, 2017.

[8] Martin R Albrecht, Pooya Farshim, Jean-Charles Faugere, and Ludovic Perret. Polly cracker, revisited. In *Advances in Cryptology–ASIACRYPT 2011*, pages 179–196. Springer, 2011.

[9] Martin Albrecht et al., 2017-2020.

[10] Martin Albrecht et al., 2018.

[11] Martin Albrecht et al., 2019.

[12] Asma Aloufi, Peizhao Hu, Harry WH Wong, and Sherman SM Chow. Blindfolded evaluation of random forests with multi-key homomorphic encryption. *IEEE Transactions on Dependable and Secure Computing*, 2019.

[13] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *Annual Cryptology Conference*, pages 297–314. Springer, 2014.

[14] Prabhanjan Ananth, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Multi-key fully-homomorphic encryption in the plain model. In *Theory of Cryptography Conference*, pages 28–57. Springer, 2020.

[15] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *Advances in Cryptology-CRYPTO 2009*, pages 595–618. Springer, 2009.

[16] Sanjeev Arora, László Babai, Jacques Stern, and Z Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computer and System Sciences*, 54(2):317–331, 1997.

[17] László Babai. On lovász'lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[18] Ahmad Al Badawi, Jin Chao, Jie Lin, Chan Fook Mun, Jun Jie Sim, Benjamin Hong Meng Tan, Xiao Nan, Khin Mi Mi Aung, and Vijay Ramaseshan Chandrasekhar. The alexnet moment for homomorphic encryption: Hcnn, the first homomorphic cnn on encrypted data with gpus. *arXiv*, pages arXiv–1811, 2018.

[19] Jean-François Biasse and Luis Ruiz. Fhew with efficient multibit bootstrapping. In *Progress in Cryptology–LATINCRYPT 2015*, pages 119–135. Springer, 2015.

[20] Jean-François Biasse and Fang Song. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 893–902. SIAM, 2016.

[21] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):506–519, 2003.

[22] Joppe W Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In *IMA International Conference on Cryptography and Coding*, pages 45–64. Springer, 2013.

[23] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual Cryptology Conference*, pages 868–886. Springer, 2012.

[24] Zvika Brakerski. When homomorphism becomes a liability. In *Theory of Cryptography Conference*, pages 143–161. Springer, 2013.

[25] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 309–325. ACM, 2012.

[26] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 575–584. ACM, 2013.

[27] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 97–106. IEEE Computer Society, 2011.

[28] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Advances in Cryptology–CRYPTO 2011*, pages 505–524. Springer, 2011.

[29] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based fhe as secure as pke. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 1–12, 2014.

[30] Benjamin M Case, Shuhong Gao, Gengran Hu, and Qiuxia Xu. Fully homomorphic encryption with k-bit arithmetic operations. *IACR Cryptol. ePrint Arch.*, 2019:521, 2019.

[31] Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. Provably weak instances of ring-LWE revisited. In *Advances in cryptology—EUROCRYPT 2016. Part I*, volume 9665 of *Lecture Notes in Comput. Sci.*, pages 147–167. Springer, Berlin, 2016.

[32] Hao Chen, Kim Laine, and Rachel Player. Simple encrypted arithmetic library-seal v2. 1. In *International Conference on Financial Cryptography and Data Security*, pages 3–18. Springer, 2017.

[33] Hao Chen, Kristin Lauter, and Katherine E Stange. Attacks on search rlwe. 2015.

[34] Hao Chen, Kristin E Lauter, and Katherine E Stange. Vulnerable galois rlwe families and improved attacks. *IACR Cryptology ePrint Archive*, 2016:193, 2016.

[35] Yao Chen, Benjamin M Case, Shuhong Gao, and Guang Gong. Error analysis of weak poly-lwe instances. *Cryptography and Communications*, 11(3):411–426, 2019.

[36] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In *Advances in Cryptology–EUROCRYPT 2013*, pages 315–335. Springer, 2013.

[37] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 409–437. Springer, 2017.

[38] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachene. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *international conference on the theory and application of cryptology and information security*, pages 3–33. Springer, 2016.

[39] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Tfhe: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020.

[40] Geoffroy Couteau, Thomas Peters, and David Pointcheval. Encryption switching protocols. In *Annual International Cryptology Conference*, pages 308–338. Springer, 2016.

[41] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. Short stickelberger class relations and application to ideal-svp. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 324–348. Springer, 2017.

[42] Jack LH Crawford, Craig Gentry, Shai Halevi, Daniel Platt, and Victor Shoup. Doing real work with fhe: the case of logistic regression. In *Proceedings of the 6th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 1–12, 2018.

[43] Yevgeniy Dodis, Shai Halevi, Ron D Rothblum, and Daniel Wichs. Spooky encryption and its applications. In *Annual International Cryptology Conference*, pages 93–122. Springer, 2016.

[44] Léo Ducas and Daniele Micciancio. Fhew: Bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology–EUROCRYPT 2015*, pages 617–640. Springer, 2015.

[45] David Steven Dummit and Richard M Foote. *Abstract algebra*, volume 3. Wiley Hoboken, 2004.

[46] Kirsten Eisenträger, Sean Hallgren, and Kristin Lauter. Weak instances of plwe. In *Selected Areas in Cryptography–SAC 2014*, pages 183–194. Springer, 2014.

[47] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

[48] Yara Elias, Kristin E Lauter, Ekin Ozman, and Katherine E Stange. Provably weak instances of ring-lwe. In *Advances in Cryptology–CRYPTO 2015*, pages 63–92. Springer, 2015.

[49] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.

[50] Shuhong Gao. Efficient fully homomorphic encryption scheme. *IACR Cryptol. ePrint Arch.*, 2018:637, 2018.

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

[52] Craig Gentry, Shai Halevi, and Nigel P Smart. Fully homomorphic encryption with polylog overhead. In *Advances in Cryptology–EUROCRYPT 2012*, pages 465–482. Springer, 2012.

[53] Craig Gentry, Shai Halevi, and Nigel P Smart. Homomorphic evaluation of the aes circuit. In *Advances in Cryptology–CRYPTO 2012*, pages 850–867. Springer, 2012.

[54] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pages 201–210. PMLR, 2016.

[55] Oded Goldreich, Daniele Micciancio, Shmuel Safra, and J-P Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Information Processing Letters*, 71(2):55–61, 1999.

[56] Shafi Goldwasser, Yael Tauman Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In *Advances in Cryptology–CRYPTO 2013*, pages 536–553. Springer, 2013.

[57] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.

[58] Thore Graepel, Kristin Lauter, and Michael Naehrig. Ml confidential: Machine learning on encrypted data. In *International Conference on Information Security and Cryptology*, pages 1–21. Springer, 2012.

[59] Qian Guo, Thomas Johansson, and Paul Stankovski. Coded-bkw: Solving lwe using lattice codes. In *Advances in Cryptology–CRYPTO 2015*, pages 23–42. Springer, 2015.

[60] Shai Halevi and Victor Shoup. Design and implementation of a homomorphic-encryption library. *IBM Research (Manuscript)*, 2013.

[61] Shai Halevi and Victor Shoup. Bootstrapping for helib. In *Advances in Cryptology–EUROCRYPT 2015*, pages 641–670. Springer, 2015.

[62] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Algorithms for the shortest and closest lattice vector problems. In *International Conference on Coding and Cryptology*, pages 159–190. Springer, 2011.

[63] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. Deep neural networks classification over encrypted data. In *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*, CODASPY '19, page 97–108, New York, NY, USA, 2019. Association for Computing Machinery.

[64] Ehsan Hesamifard, Hassan Takabi, Mehdi Ghasemi, and Rebecca N Wright. Privacy-preserving machine learning as a service. *Proceedings on Privacy Enhancing Technologies*, 2018(3):123–142, 2018.

[65] Ehsan Hesamifard, Hassan Takabi, Mehdi Ghasemi, and Rebecca N. Wright. Privacy-preserving machine learning as a service. *Proceedings on Privacy Enhancing Technologies*, 2018(3):123–142, 2018.

[66] Xiaoqian Jiang, Miran Kim, Kristin Lauter, and Yongsoo Song. Secure outsourced matrix computation and application to neural networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1209–1222, 2018.

[67] Venkata Koppula and Brent Waters. Circular security separations for arbitrary length cycles from lwe. In *Annual International Cryptology Conference*, pages 681–700. Springer, 2016.

[68] Arjen Klaas Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[69] Tancrède Lepoint and Pascal Paillier. On the minimal number of bootstrappings in homomorphic circuits. In *Financial Cryptography and Data Security*, pages 189–200. Springer, 2013.

[70] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for lwe-based encryption. In *Topics in Cryptology–CT-RSA 2011*, pages 319–339. Springer, 2011.

[71] Adriana López-Alt, Eran Tromer, and Vinod 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*, pages 1219–1234, 2012.

[72] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)*, 60(6):43, 2013.

[73] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In *Advances in cryptology—EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Comput. Sci.*, pages 35–54. Springer, Heidelberg, 2013.

[74] Tal Malkin, Isamu Teranishi, and Moti Yung. Key dependent message security: recent results and applications. In *Proceedings of the first ACM conference on Data and application security and privacy*, pages 3–12. ACM, 2011.

[75] Alexander May. Using lll-reduction for solving rsa and factorization problems. In *The LLL algorithm*, pages 315–348. Springer, 2009.

[76] Ahmet Can Mert, Erdinç Öztürk, and Erkay Savaş. Design and implementation of a fast and scalable ntt-based polynomial multiplier architecture. In *2019 22nd Euromicro Conference on Digital System Design (DSD)*, pages 253–260, 2019.

[77] Daniele Micciancio and Panagiotis Voulgaris. Faster exponential time algorithms for the shortest vector problem. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1468–1480. SIAM, 2010.

[78] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key fhe. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 735–763. Springer, 2016.

[79] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 113–124. ACM, 2011.

[80] Karthik Nandakumar, Nalini Ratha, Sharath Pankanti, and Shai Halevi. Towards deep neural network training on encrypted data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[81] Karthik Nandakumar, Nalini Ratha, Sharath Pankanti, and Shai Halevi. Towards deep neural network training on encrypted data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[82] Jürgen Neukirch. *Algebraic number theory*, volume 322. Springer Science & Business Media, 2013.

[83] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 333–342. ACM, 2009.

[84] Chris Peikert. How (not) to instantiate ring-LWE. In *Security and cryptography for networks*, volume 9841 of *Lecture Notes in Comput. Sci.*, pages 411–430. Springer, [Cham], 2016.

[85] Alexander Pott. *Finite geometry and character theory*. Springer, 2006.

[86] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.

[87] M. Sadegh Riazi, Kim Laine, Blake Pelton, and Wei Dai. Heax: An architecture for computing on encrypted data. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20, page 1295–1309, New York, NY, USA, 2020. Association for Computing Machinery.

[88] Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

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

[90] Claus-Peter Schnorr. Block reduced lattice bases and successive minima. *Combinatorics, Probability and Computing*, 3(04):507–522, 1994.

[91] Nigel P Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *International Workshop on Public Key Cryptography*, pages 420–443. Springer, 2010.

[92] Nigel P Smart and Frederik Vercauteren. Fully homomorphic simd operations. *Designs, codes and cryptography*, 71(1):57–81, 2014.

[93] Damien Stehlé and Ron Steinfeld. Making ntru as secure as worst-case problems over ideal lattices. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 27–47. Springer, 2011.

[94] Daniel Takabi, Robert Podschwadt, Jeff Druce, Curt Wu, and Kevin Procopio. Privacy preserving neural network inference on encrypted data with gpus, 2019.

[95] Vinod Vaikuntanathan. Homomorphic encryption references, 2021.