博 士 論 文

A Study on Homomorphic Property of Ring–Based Lattice Cryptography

Sari HANDA
半田 沙里

情報セキュリティ大学院大学
情報セキュリティ研究科
情報セキュリティ専攻

2019 年 9 月

A Study on Homomorphic Property of Ring-Based Lattice Cryptography

Sari Handa

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in Informatics
in the Graduate School of Information Security

INSTITUTE of INFORMATION SECURITY, JAPAN

2019

ABSTRACT

Computing secret data keeping their secrecy is a significant technology problem of modern digital society. It enables to analyze sensitive data such as personal data protecting their privacy or enables distributed computation without leakage of secret against involving parties or engines. This secure computation is realized by technologies having homomorphic property, that is, performing arithmetic operations on data through their some form of privacy-preserving encodings. Those technologies are mainly multi-party computation (MPC) and homomorphic schemes. In this thesis, we study the homomorphic schemes: the homomorphic encryption as encryption scheme and multilinear maps as encoding scheme.

Homomorphic Encryption. We improve algebraic structure for homomorphic parallel computation. The slot structure of Ring-LWE encryption based on the cyclotomic ring is GF(p^d). To multiply integer plaintexts on this slot, only one dimension is available though computation needs full dimensions. We focus the fact that on the decomposition ring, which is a subring of the cyclotomic ring, the slot structure becomes $\mathbb{Z}_{p^l}$. We construct FV-type and BGV-type homomorphic encryption schemes on the decomposition ring and implement these schemes. In the experiment, we verify the effect of our slot structure on the decomposition ring. The timings of homomorphic arithmetic operations are approximately linear with respect to the number of slots, and it shows that our schemes perform parallel computing without waste of a plaintext space. Comparing to HElib, our schemes results in several times faster computation.

We also construct bootstrap functions for our schemes. On the cyclotomic ring, conversion between the ring and slots is complex and it causes bottleneck in the bootstrap procedure. In contrast on the decomposition ring, the conversion is just multiplying a matrix by the vector of the ring or slots, and we can compute this conversion homomorphically in a simple way. Our implementation of the bootstrap performs fast at small parameters for an initial experiment.

Multilinear Maps. Multilinear Maps was realized as leveled encoding schemes based on ideal lattice [18]. We study applications of multilinear maps using the leveled encoding scheme and propose two schemes: a group key exchange protocol and a witness encryption scheme based on the Hamilton Cycle Problem. These schemes are both based on the same method of generating a product of all parties' secrets homomorphically by pilling up their encodings.

Acknowledgements

First of all, I would like to express my appreciation to my supervisor Professor Seiko Arita. He adviced from his deep insights into cryptography and interests in mathematics. Cryptography and number theory attracted me and I had struggled with and enjoyed studying them during my master's and doctoral programs.

I am also grateful to Professor Atsuhiro Goto, Professor Hiroshi Doi and Professor Akira Otsuka for their constructive comments and suggestions.

Many thanks to members of Arita laboratory. I got widely perspective through discussion about their interesting researches.

I wish thank to members of algebraic number theory seminar. They helped me in understanding the textbook.

I acknowledge the cooperation and understanding of colleagues and managers of my company. Thanks to their recommend to study at the institute, I could know this interesting research world.

This work was supported by JST CREST Grant Number JPMJCR1503. This work is further supported by the JSPS KAKENHI Grant Number 17K05353.

Finally, I am deeply grateful to my family and friends. They always support and encourage me. I could continue studying long years due to their persistent help.

Contents

Chapter 1

Introduction

1.1 Secure Computation

Computing secret data keeping their secrecy is a significant technology problem of modern digital society. It enables to analyze sensitive data such as personal data protecting their privacy or enables distributed computation without leakage of secret against involving parties or engines. Main technologies to realize the secure computation are secret sharing, garbled circuit and homomorphic encryption. Here we outline these technologies.

In the secret sharing, shares of a secret are distributed to multiple parties. We can restore the secret only when we gather those shares more than threshold. A representative secret sharing method is a Shamir's secret sharing scheme [43]. We can compute secret data securely using this secret sharing method by gathering the calculated results of secret's shares from parties.

Yao proposed the garbled circuit protocol [49] in which two parties, called garbler and evaluator, interactively evaluate a garbled boolean circuit as follows. First, the garbler labels input/output wires of the circuit with random encodings and represents the circuit using the encodings, which is called the garbled circuit. The garbler also generates the encodings of his input values to the circuit. Next, the evaluator receives the garbled circuit as well as the encodings of the input value via oblivious transfer protocol, and evaluates the circuit on the input encodings, and then sends the result encoding to the garbler. The garbler finally gains an output value of the circuit from the received encoding. The evaluator evaluates the circuit without knowing any information about input value and output value since the circuit and the wires are encoded.

The above two methods are executed on the multi-party protocol (MPC). The computation of

the each party is light in MPC; however, the communication size becomes large.

The homomorphic encryption is the encryption scheme that enables to calculate on the ciphertexts without being decrypted. In contrary to MPC, the homomorphic encryption performs in a stand-alone environment, but it requires huge computational cost and consumes large memory.

Recently, the hybrid systems of MPC and the homomorphic encryption are studied toward real-word usage of the secure computation. TASTY [47] combines Yao's garbled circuit with the homomorphic encryption. The study of [12] is a hybrid of the homomorphic encryption and Private Set Intersection (PSI), which is new type of MPC protocol allows two parties to compute the intersection of their sets without exposing anything except the intersection. They achieve a small communication overhead by the homomorphic encryption.

In this work, we study *homomorphic property*, that is, performing arithmetic operation on data through their some form of privacy-preserving encodings. There are two types in cryptography schemes having the homomorphic property, those are encryption scheme and encoding scheme. The encryption scheme recovers calculated data from input encodings. The encoding scheme does not recover the calculated data but extracts some deterministic value with respect to the calculated encoding. We study the homomorphic encryption for the encryption scheme and the multilinear maps for the encoding scheme.

1.2 Homomorphic Encryption

1.2.1 Background

Since the breakthrough construction given by Gentry [22], many efforts have been dedicated to make homomorphic encryption scheme more secure and more efficient. Especially, HE schemes based on the Ring-LWE problem [38, 6, 17, 39] have obtained theoretically-sound proof of security and well-established implementations such as HElib [27] and SEAL v2.0 [37]. Nowadays many researchers apply HE schemes to privacy-preserving tasks for mining of outsourced data such as genomic data, medical data, financial data and so on [26, 35, 11, 34, 36].

1.2.2 SIMD operation using Chinese Reminder Theorem

In 2014, Smart and Vercauteren introduced a packing technique for homomorphic encryption by decomposing the plaintext space using the Chinese Remainder Theorem (CRT). This technique allows us to encrypt multiple message into a single ciphertext and execute Single Instruction Multiple Data (SIMD) operations homomorphically on it.

CRT on integers is an isomorphism among $\mathbb{Z}_p$ and the product of residual fields $\mathbb{Z}_{p_i}$, where each p_i are co-prime factors of p. In the SIMD of homomorphic encryption, it is CRT on the cyclotomic ring R using the prime factorization, that is, the principal ideal (p) of a prime p decomposes into prime ideals $\mathfrak{P}_i$ in the cyclotomic ring. The isomorphism is among a residual ring R/pR and the product of residual fields $R/\mathfrak{P}_i$. We call each of residual fields $R/\mathfrak{P}_i$ a "slot".

We know the structure of the slot by observing how prime p decomposes in the cyclotomic ring. In the m-th cyclotomic ring $R = \mathbb{Z}(\zeta)$, let d be a multiplicative order of p in m and $g = \varphi(\mathsf{m})/d$, then the prime p decomposes into g prime ideals $\mathfrak{P}_0, \cdots, \mathfrak{P}_{g-1}$, and the structure of each plaintext slot is a finite field $\mathrm{GF}(p^d)$ which is represented as a set of polynomials in $\mathbb{Z}_p[X]$ with degree d.

For parallel calculation of plaintexts in $\mathbb{Z}_p$, we set each plaintext to $\mathbb{Z}_p$ coefficients of the polynomials in each slot. When we add two packed ciphertexts homomorphically, we add corresponding polynomials coefficient-wise for each slot. However, in multiplication, we get product only from the constant terms of corresponding polynomials for each slot, and the other terms, which are cross-terms, are not used. The slot structure of finite field $\mathrm{GF}(p^d)$ causes heavy computation. We study structure of plaintext space from the perspective of algebraic number theory and construct more simple plaintext slot space.

1.2.3 Our contributions

Decomposition ring. We focus on a special subring of cyclotomic ring named *decomposition ring*. This ring is the subring fixed by all maps which do not change prime ideals $\mathfrak{P}_i$ over the prime p. It is known that on the decomposition ring R_Z, the prime p factors into g prime ideals $\mathfrak{p}_0, \cdots, \mathfrak{p}_{g-1}$ and their norm is p, whereas the norm of $\mathfrak{P}_i$ is p^d in the cyclotomic ring. Since the slot structure is $\mathbb{Z}_p$ in the decomposition ring, we get products of plaintexts slot-wise with no waste for homomorphic multiplication. We show CRT of the cyclotomic ring R and the decomposition ring R_Z

in the figure blow.

$$\begin{array}{ccc} R = \mathbb{Z}[\zeta] & R/pR & \xrightarrow{\sim} \mathrm{GF}(p^d) \oplus \cdots \oplus \mathrm{GF}(p^d) \\ \big| & \big| & \\ R_Z & R_Z/pR_Z & \xrightarrow{\sim} \mathbb{Z}_p \oplus \cdots \oplus \mathbb{Z}_p \\ \big| & \big| & \\ \mathbb{Z} & \mathbb{Z}/p\mathbb{Z} & \end{array}$$

Construction. Based on the above investigation, we construct two types of homomorphic encryption schemes over the decomposition ring: DR-FV and DR-BGV. The DR-FV and DR-BGV schemes realize the FV [17] and the BGV scheme [6] over the decomposition ring, respectively. We also construct the bootstrapping function for our schemes. We show several bounds on the noise growth occurring among homomorphic computations and prove that both of DR-FV and DR-BGV schemes are fully homomorphic on modulus of magnitude $q = O(\lambda^{\log \lambda})$.

Security. For security we will need hardness of a variant of the decisional Ring-LWE problem over the decomposition ring. Recall the search version of Ring-LWE problem is already proved to have a quantum polynomial time reduction from the approximate shortest vector problem of ideal lattices in *any number field* by Lyubashevsky, Peikert, and Regev [38]. They proved equivalence between the search and decisional versions of the Ring-LWE problems only for cyclotomic rings. However, it is not difficult to see that the equivalence holds also over the decomposition rings, since those are subrings of cyclotomic rings and inherit good properties about prime ideal decomposition from them.

Benchmark result. Since the dimension of the decomposition ring is $g = \varphi(\mathsf{m})/d$, considering the same dimension to the cyclotomic ring, the decomposition ring has d times higher parallelism as compared with the cyclotomic ring.

We implemented our two decomposition ring homomorphic encryption schemes DR-FV and DR-BGV and compared efficiency with HElib, which is based on the BGV scheme over ordinal cyclotomic rings. In HElib, even if the number of slots becomes only two times larger, their timings becomes much larger then the two times. This indicates that HElib scheme cannot handle many slots with high parallelism. On the other hand in both of our DR-FV and DR-BGV, timings are approximately linear with respect to the number of slots. Here we describe timing results of

HElib and DR-BGV under the similar level of security parameters. The timing of Enc, Dec, Add, Mult, Exp-by-2^8 are respectively

HElib$(395.85, 2740.01, 10.92, 1397.89, 6664.32)$,

DR-BGV$(30.14, 29.35, 3.70, 282.11, 1678.52)$.

This benchmark indicates that DR-BGV is several times faster than HElib for integer plaintext, as expected.

We implemented the bootstrap procedure for our schemes and performed an initial experiment at small parameters. The bootstrap procedure needs many homomorphic transformations between the ring and the plaintext slots. We think our simple transformation, which is just multiply a matrix to the vector, makes our bootstrap efficient.

1.2.4 Related works

In 2009, Gentry [22] established the fully homomorphic encryption scheme for the first time. After this breakthrough, representative two schemes, BGV scheme [6] and FV [17] scheme, are proposed depending on the techniques such as key switching [7] and modulus switching [6]. Since the computational cost of homomorphic operations are very expensive, parallel computing is needed for higher throughput. The SIMD technique, proposed by [45], enables parallel homomorphic computation using the CRT over polynomials, and has been adopted in many HE schemes. We focus on the wasteful slot structure of such HE schemes based on cyclotomic ring, and improve the slot structure using the decomposition ring. Kim and Song [32] also focus on the similar issue and construct HE based on another subring of the cyclotomic ring, called "conjugate-invariant ring", aiming for efficient homomorphic fixed-point number computation. Terada, Nakano, Okumura and Miyaji [48] conducted some experiments regarding lattice attack against Ring-LWE problem over the decomposition ring. They concluded that the Ring-LWE problem on the decomposition ring is expected to be as secure as the ordinal Ring-LWE problem on the cyclotomic ring.

1.3 Multilinear Maps

Cryptographic bilinear map was realized by elliptic curve and opened the path to new functional cryptography such as identity-based encryption (IBE), attribute-based encryption (ABE), group signature, and so on. The extension from bilinear map to multilinear map was desired and defined by Boneh and Silverberg, but the construction was log-time open problem.

With progress in study of homomorphic encryption, researchers considered whether the homomorphic property of homomorphic encryption has capability to realize multilinear map. Garg et al proposed first construction of multilinear map based on ideal lattice[18], which is leveled encoding scheme as they called *graded encoding scheme*. The multi-linearity is attractive property for cryptography, and high-functionality schemes are proposed such as multi-party key exchange, ABE for circuit, and it produces new cryptographic primitives like witness encryption and indistinguishability obfuscation.

The security of multilinear map is under studying, and knowledge has been accumulated by the works of attack against new schemes and works of improving those schemes more securely.

1.3.1 Our contributions

We study application of multilinear maps and propose two schemes: a group key exchange protocol and a witness encryption scheme based on the Hamilton Cycle Problem.

Our group key exchange protocol is very simple, each party multiplies his encoding to the received encoding and pass the result to the next party in upflow and downflow. The computational cost and communication size per party are constant with respect to the number of parties, this is an advantage brought using multilinear maps.

Witness encryption is a new type of cryptosystem that could be achieved using multilinear maps. Witness encryption is based on some NP-complete language. Only one with possession of the witness for the language can decrypt the ciphertext to its original message. The first construction by Garg et al is based on EXACT-COLVER Problem as NP-complete language. We construct a new witness encryption scheme based on another NP-complete problem, Hamilton Cycle Problem. We prove security of our scheme based on our generic cyclic colored matrix model that is a variant of a generic colored matrix model.

Chapter 2

Lattice-Based Cryptography

2.1 Preliminaries

2.1.1 Notation

For a positive integer m, $\mathbb{Z}_m$ denotes the ring of congruent integers mod m, and $\mathbb{Z}_m^*$ denotes its multiplicative subgroup. For an integer a (that is prime to m), $\mathrm{ord}_m^{\times}(a)$ denotes the order of $a \in \mathbb{Z}_m^*$. Basically vectors are supposed to represent column vectors. The symbol $\mathbf{1}$ denotes a column vector with all entries equal to 1. I_n denotes the $n \times n$ identity matrix. The symbol $\mathrm{Diag}(\alpha_1, \cdots, \alpha_n)$ means a diagonal matrix with diagonals $\alpha_1, \ldots, \alpha_n$. For vectors $\boldsymbol{x}, \boldsymbol{y}$ $(\in \mathbb{C}^n)$, $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^n x_i \overline{y}_i$ denotes the inner product of $\boldsymbol{x}$ and $\boldsymbol{y}$. $\|\boldsymbol{x}\|_2 = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle}$ denotes the l_2-norm and $\|\boldsymbol{x}\|_\infty = \max\{|x_i|\}_{i=1}^n$ denotes the infinity norm of $\boldsymbol{x}$. For vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, $\boldsymbol{a} \odot \boldsymbol{b} = (a_i b_i)_i$ denotes the component-wise product of $\boldsymbol{a}$ and $\boldsymbol{b}$. For a square matrix M over $\mathbb{R}$, $s_1(M)$ denotes the largest singular value of M. For a matrix A over $\mathbb{C}$, $A^* = \overline{A}^T$ denotes the transpose of complex conjugate of A.

2.2 LWE Encryption

2.2.1 Lattices

For n linearly independent vectors $B = \{b_j\}_{j=1}^n \subset \mathbb{R}^n$, $\Lambda = \mathcal{L}(B) = \left\{ \sum_{j=1}^n z_j b_j \mid z_j \in \mathbb{Z} \ (\forall j) \right\}$ is called an n-dimensional *lattice*. For a lattice $\Lambda \subset \mathbb{R}^n$, its *dual lattice* is defined by $\Lambda^\vee = \left\{ y \in \mathbb{R}^n \mid \langle x, y \rangle \in \mathbb{Z} \ (\forall x \in \Lambda) \right\}$. The dual lattice is itself a lattice. The dual of dual lattice is the same

as the original lattice: $(\Lambda^\vee)^\vee = \Lambda$. For a countable subset $A \subset \mathbb{R}^n$, the sum $D_s(A) \stackrel{\text{def}}{=} \sum_{x\in A} D_s(x)$ is well-defined.

2.2.2 Gaussian distributions and subgaussian random variables

For a positive real $s > 0$, the n-dimensional (spherical) Gaussian function $\rho_s : \mathbb{R}^n \to (0, 1]$ is defined as

$$\rho_s(x) = \exp(-\pi \|x\|_2^2 / s^2).$$

It defines the continuous Gaussian distribution D_s with density $s^{-n}\rho_s(x)$.

A random variable X over $\mathbb{R}$ is called *subgaussian with parameter* s (> 0) if $\mathbb{E}[\exp(2\pi t X)] \leq \exp(\pi s^2 t^2) \quad (\forall t \in \mathbb{R})$. A random variable X over $\mathbb{R}^n$ is called subgaussian with parameter s if $\langle X, u \rangle$ is subgaussian with parameter s for any unit vector $u \in \mathbb{R}^n$. A random variable X according to Gaussian distribution D_s is subgaussian with parameter s. A bounded random variable X (as $|X| \leq B$) with $\mathbb{E}[X] = 0$ is subgaussian with parameter $B\sqrt{2\pi}$.

A subgaussian random variable with parameter s satisfies the tail inequality:

$$\Pr[|X| \geq t] \;\leq\; 2\exp\left(-\pi \frac{t^2}{s^2}\right) \quad (\forall t \geq 0). \tag{2.1}$$

The discrete Gaussian distribution $D_{\Lambda+c,s}$ on a (coset of) lattice Λ is defined by restricting the continuous Gaussian distribution D_s on the (coset of) lattice Λ:

$$D_{\Lambda+c,s}(x) \stackrel{\text{def}}{=} \frac{D_s(x)}{D_s(\Lambda + c)} \quad (x \in \Lambda + c).$$

2.2.3 LWE Problem

The Learning With Errors Problem (LWE problem) was introduced by Regev [41] as a generalization of "learning parity with noise". Here we describe a definition by Brakerski, Gentry, and Vaikuntanathan [6].

Definition 1 *(LWE, definition 2.12 in [6]) For an integer $q = q(n)$ and an error distribution $\chi = \chi(n)$ over $\mathbb{Z}_q$, the learning with errors problem $LWE_{n,m,q,\chi}$ is defined as follows: Given m independent samples from $A_{s,\chi}$ (for some $s \in \mathbb{Z}_q^n$), output s with noticeable probability.*

The (average-case) decision variant of the LWE problem, denoted $DLWE_{n,m,q,\chi}$, is to distinguish (with non negligible advantage) m samples chosen according to $A_{s,\chi}$ (for uniformly random

$s \leftarrow \mathbb{Z}_q^n$), from m samples chosen according to the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. We denote by $DLWE_{n,q,\chi}$ the variant where the adversary gets oracle access to $A_{s,\chi}$ and is not apriori bounded in the number of samples.

For an algorithm $\mathcal{B}$ and security parameter λ, we denote

$DLWE_{n,q,\chi}Adv[\mathcal{B}] := |Pr[\mathcal{B}^{A_{s,\chi}}(1^\lambda) = 1] - Pr[\mathcal{B}^{U(\mathbb{Z}_q^n \times \mathbb{Z}_q)}(1^\lambda) = 1]|$.

2.3 Number Fields and Rings

2.3.1 Number Fields

A complex number $\alpha \in \mathbb{C}$ is called an *algebraic number* if it satisfies $f(\alpha) = 0$ for some nonzero polynomial $f(X) \in \mathbb{Q}[X]$ over $\mathbb{Q}$. For an algebraic number α, the monic and irreducible polynomial $f(X)$ satisfying $f(\alpha) = 0$ is uniquely determined and called the *minimum polynomial* of α. An algebraic number α generates a *number field* $K = \mathbb{Q}(\alpha)$ over $\mathbb{Q}$, which is isomorphic to $\mathbb{Q}[X]/(f(X))$, via $g(\alpha) \mapsto g(X)$. The dimension of K as a $\mathbb{Q}$-vector space is called the *degree* of K and denoted as $[K : \mathbb{Q}]$. By the isomorphism, $[K : \mathbb{Q}] = \deg f$.

An algebraic number α is called an *algebraic integer* if its minimum polynomial belongs to $\mathbb{Z}[X]$. All algebraic integers belonging to a number field $K = \mathbb{Q}(\alpha)$ constitutes a ring R, called an *integer ring* of K.

A number field $K = \mathbb{Q}(\alpha)$ has $n\,(= [K : \mathbb{Q}])$ isomorphisms ρ_i $(i = 1, \ldots, n)$ to subfields of the complex number field $\mathbb{C}$. The trace map $\mathrm{Tr}_{K|\mathbb{Q}} : K \to \mathbb{Q}$ is defined by $\mathrm{Tr}_{K|\mathbb{Q}}(a) = \sum_{i=1}^n \rho_i(a)$ $(\in \mathbb{Q})$. If all of the isomorphisms ρ_i induce automorphisms of K (i.e., $\rho_i(K) = K$ for any i), the field K is called a *Galois extension* of $\mathbb{Q}$ and the set of isomorphisms $\mathrm{Gal}(K|\mathbb{Q}) \stackrel{\text{def}}{=} \{\rho_1, \ldots, \rho_n\}$ constitutes a group, called the *Galois group* of K over $\mathbb{Q}$. By the Galois theory, all subfields L of K and all subgroups H of $G = \mathrm{Gal}(K|\mathbb{Q})$ corresponds to each other one-to-one:

$$
\begin{aligned}
L \;&\mapsto\; H = G_L = \{\rho \in G \mid \rho(a) = a \text{ for any } a \in L\} \\
&\qquad : \text{the stabilizer group of } L \\
H \;&\mapsto\; L = K^H = \{a \in K \mid \rho(a) = a \text{ for any } \rho \in H\} \\
&\qquad : \text{the fixed field by } H.
\end{aligned}
$$

Here, K is also a Galois extension of L with Galois group $\mathrm{Gal}(K|L) = H$ (since any isomorphism (of K into $\mathbb{C}$) that fixes L sends K to K). Especially, $[K : L] = |H|$. The trace map of K over L is

defined by $\mathrm{Tr}_{K|L}(a) = \sum_{\rho \in H} \rho(a)$ $(\in L)$ for $a \in K$.

2.3.2 Cyclotomic Fields and Rings

Let m be a positive integer. A primitive m-th root of unity $\zeta = \exp(2\pi\sqrt{-1}/\mathsf{m})$ has the minimum polynomial $\Phi_{\mathsf{m}}(X) \in \mathbb{Z}[X]$ of degree $n = \phi(\mathsf{m})$ that belongs to $\mathbb{Z}[X]$, called the *cyclotomic polynomial*. Especially, ζ is an algebraic integer. A number field $K = \mathbb{Q}(\zeta)$ generated by ζ is called the m-th *cyclotomic field* and its elements are called *cyclotomic numbers*. The integer ring R of the cyclotomic field $K = \mathbb{Q}(\zeta)$ is known to be $R = \mathbb{Z}[\zeta] = \mathbb{Z}[X]/\Phi_{\mathsf{m}}(X)$. In particular, as a $\mathbb{Z}$-module, R has a basis (called *power basis*) $\{1, \zeta, \ldots, \zeta^{n-1}\}$, i.e., $R = \mathbb{Z} \cdot 1 + \mathbb{Z} \cdot \zeta + \cdots + \mathbb{Z} \cdot \zeta^{n-1}$. The integer ring R is called the m-th *cyclotomic ring* and its elements are called *cyclotomic integers*. For a positive integer q, $R_q = R/qR = \mathbb{Z}_q[X]/\Phi_{\mathsf{m}}(X)$ is a ring of *cyclotomic integers mod* q.

The cyclotomic field $K = \mathbb{Q}(\zeta)$ is a Galois extension over $\mathbb{Q}$ since it has $n = [K : \mathbb{Q}]$ automorphisms ρ_i defined by $\rho_i(\zeta) = \zeta^i$ for $i \in \mathbb{Z}^*_{\mathsf{m}}$. Its Galois group $G = \mathrm{Gal}(K|\mathbb{Q})$ is isomorphic to $\mathbb{Z}^*_{\mathsf{m}}$ by corresponding ρ_i to i. Note that $\rho_i(\overline{b}) = \overline{\rho_i(b)}$, since $\overline{a} = \rho_{-1}(a)$.

The trace of ζ for the prime index m is simple:

Lemma 1 *If the index* m *is prime, we have*

$$\mathrm{Tr}_{K|\mathbb{Q}}(\zeta^i) = \begin{cases} \mathsf{m} - 1 & (i \equiv 0 \pmod{\mathsf{m}}) \\ -1 & (i \not\equiv 0 \pmod{\mathsf{m}}). \end{cases}$$

2.3.3 Tensorial Decomposition

Lyubashevsky, Peikert, and Regev provide a toolkit for Ring-LWE cryptography [39], in this work they propose a technique to construct Ring-LWE schemes that work in arbitrary cyclotomic rings, which is required for SIMD operations and for security reason, using tensorial decomposition. Mayer implements the toolkit in his work [40].

The tensorial decomposition technique views a m-th cyclotomic ring as a combination of decomposed m_l-th cyclotomic rings where $\mathsf{m} = \prod_l \mathsf{m}_l$. The decomposing ring technique enables us to compute simply and fast ring operations, in particularly canonical embedding of the ring elements.

In our homomorphic encryption schemes based on the decomposition ring, the conversion between a decomposition ring element and plaintext slots is the canonical embedding of the decomposition ring element. We adopt the tensorial decomposition technique to it. The canonical embedding of the m-th cyclotomic ring can be viewed as a tensor product of the canonical embedding of the m_l-th cyclotomic rings.

From Proposition 1.4.4 and 1.4.5 in [40], let m be any positive integer with prime power factorization $m = \prod_{l=0}^{s} m_l$, then $\zeta_m = \prod_{l=0}^{s} \zeta_{m_l}$ and $\mathbb{Q}(\zeta_m)$ is isomorphic as a field to the tensor product $\bigotimes_{l=0}^{s} \mathbb{Q}(\zeta_{m_l})$

$$\mathbb{Q}(\zeta_m) \simeq \bigotimes_{l=0}^{s} \mathbb{Q}(\zeta_{m_l}) \tag{2.2}$$

via the correspondence

$$\prod_{l=0}^{s} a_l \leftrightarrow \left(\bigotimes_{l=0}^{s} a_l \right), \tag{2.3}$$

where $a_l \in \mathbb{Q}(\zeta_{m_l})$.

When viewing $K = \mathbb{Q}(\zeta_m)$ as the tensor product $\bigotimes_{l=0}^{s} \mathbb{Q}(\zeta_{m_l})$ as above, the canonical embedding $\sigma = (\rho_i)_{i \in \mathbb{Z}_m^*}$ of $K = \mathbb{Q}(\zeta_m)$ is viewed as the tensor product of the canonical embeddings $\sigma^{(l)} = (\rho_j)_{j \in \mathbb{Z}_{m_l}^*}$ of $K_l = \mathbb{Q}(\zeta_{m_l})$ using the CRT of $\mathbb{Z}_m^* \simeq \prod_{l=0}^{s} \mathbb{Z}_{m_l}^*$,

$$\sigma\left(\bigotimes_{l=0}^{s} a_l \right) = \bigotimes_{l=0}^{s} (\rho_j(a_l))_{j \in \mathbb{Z}_{m_l}^*} = \bigotimes_{l=0}^{s} \sigma^{(l)}(a_l). \tag{2.4}$$

Applying this relation to the cyclotomic ring $R = \mathbb{Z}[\zeta_m]$ having the powerful basis $\boldsymbol{p}$ and its decomposed cyclotomic rings $R_l = \mathbb{Z}[\zeta_{m_l}]$ having the powerful bases $\boldsymbol{p_l}$, we have,

$$\sigma\left(\bigotimes_{l=0}^{s} \boldsymbol{p_l}^T \right) = \bigotimes_{l=0}^{s} \sigma^{(l)}(\boldsymbol{p_l}^T). \tag{2.5}$$

This shows that the canonical embedding of the cyclotomic ring $R = \mathbb{Z}[\zeta_m]$ is viewed as the tensor product of the decomposed cyclotomic rings $R_l = \mathbb{Z}[\zeta_{m_l}]$.

2.3.4 Geometry of numbers of cyclotomic ring: Ideal Lattices

Using the n automorphisms ρ_i ($i \in \mathbb{Z}_{\mathsf{m}}^*$), the cyclotomic field K is embedded into an n-dimensional complex vector space $\mathbb{C}^{\mathbb{Z}_{\mathsf{m}}^*}$, called the *canonical embedding* $\sigma : K \to H\,(\subset \mathbb{C}^{\mathbb{Z}_{\mathsf{m}}^*})$: $\sigma(a) = (\rho_i(a))_{i \in \mathbb{Z}_{\mathsf{m}}^*}$. Its image $\sigma(K)$ is contained in the space H defined as

$$H \stackrel{\text{def}}{=} \{x \in \mathbb{C}^{\mathbb{Z}_{\mathsf{m}}^*} : x_i = \overline{x}_{m-i} \quad (\forall i \in \mathbb{Z}_{\mathsf{m}}^*)\}.$$

Since $H = B\mathbb{R}^n$ with the unitary matrix $B = \frac{1}{\sqrt{2}}\begin{pmatrix} I & \sqrt{-1}J \\ J & -\sqrt{-1}I \end{pmatrix}$, the space H is isomorphic to $\mathbb{R}^n$ as an inner product $\mathbb{R}$-space (where J is the reversal matrix of the identity matrix I).

By the canonical embedding σ, one can regard the cyclotomic ring R (or its (fractional) ideals of R) as lattices in the n-dimensional real vector space H, called *ideal lattices*. Inner products or norms of elements $a \in K$ are defined through the embedding σ:

$$\begin{aligned}
&\langle a, b \rangle \stackrel{\text{def}}{=} \langle \sigma(a), \sigma(b) \rangle = \mathrm{Tr}_{K|\mathbb{Q}}(a\overline{b}), \\
&\|a\|_p \stackrel{\text{def}}{=} \|\sigma(a)\|_p = (\Sigma_{i=1}^n |\sigma_i(a)|^p)^{1/p} \text{ for } p < \infty, \\
&\|a\|_\infty \stackrel{\text{def}}{=} \|\sigma(a)\|_\infty = \max\{|\sigma_i(a)|\}_{i=1}^n.
\end{aligned}$$

For any ring elements $x, y \in R$, their norms satisfy the following inequality

$$\|x \cdot y\|_p \leq \|x\|_\infty \cdot \|y\|_p$$

Relation between canonical and polynomial embedding norms. The norms of the canonical embedding and the polynomial embedding of a ring element have a relation. $\boldsymbol{x}$ denotes a coefficient vector of $x \in R$ in the power basis and $\|\boldsymbol{x}\|_\infty$ denotes a polynomial embedding norm. The relation is as follows:
There is a ring constant c_m which depends only on m such that

$$\|\boldsymbol{x}\|_\infty \leq c_m \cdot \|x\|_\infty, \forall x \in R. \tag{2.6}$$

The constant c_m is studied in the work [16] by Damgård et al.

2.4 Ring-LWE Encryption

2.4.1 Ring-LWE Problem

The ring learning with errors (RLWE) problem was firstly introduced by Lyubashevsky, Peikert, and Regev [38]. They give the formal definition with dual ideal and informal definition without dual ideal.

Duality. Let K be a number field of degree n and R be its ring of integers $R = O_K$. For any lattice $\mathcal{L}$ in K (i.e. for the $\mathbb{Z}$-span of any $\mathbb{Q}$-basis of K), its dual is defined as $\mathcal{L}^\vee = \{x \in$

$K \mid \mathrm{Tr}(x\mathcal{L}) \subseteq \mathbb{Z}\ \Big\}$. The dual ideal $R^\vee$ of R is a fractional ideal called codifferent and it yields a nice relation between a dual ideal $\mathcal{I}^\vee$ of an ideal $\mathcal{I}$ and its inverse $\mathcal{I}^{-1}$, that is $\mathcal{I}^\vee = R^\vee \mathcal{I}^{-1}$.

Here we describe their formal definitions of search and decisional Ring-LWE Problem.

Definition 2 *(Ring-LWE Distribution, Definition 3.1 in [38]) Let K be a number field and R be its ring of integers whose dual ideal is $R^\vee$. Let $R_q^\vee = R^\vee/qR^\vee$. For a secret $s \in R_q^\vee$ and an error distribution ψ over $K_\mathbb{R}$, a sample from the Ring-LWE distribution $A_{s,\psi}$ over $R_q \times K_\mathbb{R}/R^\vee$ is generated by choosing $a \leftarrow R_q$ uniformly at random, choosing $e \leftarrow \psi$, and outputting $(a, b = (a \cdot s)/q + e \pmod{}^\vee R)$.*

Definition 3 *(Ring-LWE, Search, Definition 3.2 in [38]) For some arbitrary secret $s \in R_q^\vee$ and an error distribution ψ over $K_\mathbb{R}$, the search version of the Ring-LWE problem is defined as follows: given access to arbitrarily many independent samples from $A_{s,\psi}$, find s.*

Definition 4 *(Ring-LWE, Average-Case Decision, Definition 3.3 in [38]) For a uniformly random secret $s \in R_q^\vee$ and an error distribution ψ over $K_\mathbb{R}$, the average-case decision version of the Ring-LWE problem is to distinguish with non-negligible advantage between arbitrarily many independent samples from $A_{s,\psi}$ and the same number of uniformly random and independent samples from $R_q \times K_\mathbb{R}/R^\vee$.*

They informally defined a non-dual decisional Ring-LWE problem. Let the ring R be the m-th cyclotomic ring for $\mathsf{m} = 2^k$ (i.e., $R = \mathbb{Z}[x]/(x^n + 1)$ for $n = 2^{k-1}$).

Definition 5 *(Ring-LWE, Decision (informal), in [38]) Fix a certain error distribution over R and let $s \in R_q$ be uniformly random. The decisional Ring-LWE problem is to distinguish arbitrary many independent random noisy ring equations $(a, b \approx a \cdot s) \in R_q \times R_q$ where each a is uniformly random and each product $a \cdot s$ is perturbed by a term drawn independently from the error distribution over R.*

This non-dual Ring-LWE problem is equivalent to the dual Ring-LWE problem, because in the case of $\mathsf{m} = 2^k$, $R^\vee = n^{-1}R$ is simply a scaling of the ring and the dual Ring-LWE samples $(a, b = (a \cdot s)/q + e)$ can be transformed to the non-dual Ring-LWE samples $(a, b' = (a \cdot s')/q + e')$ where $s' = s \cdot n \in R_q$ and $e' = e \cdot n \in R$.

Security

In the work [38], the search version of Ring-LWE problem is proved to have a quantum polynomial time reduction from the approximate shortest vector problem of ideal lattices in *any number field*. They proved equivalence between the search and the decisional versions of the Ring-LWE problems only for cyclotomic rings.

2.5 Homomorphic Encryption

Homomorphic encryption (HE) scheme enables us computation on encrypted data. One can add or multiply (or more generally "evaluate") given ciphertexts and generate a new ciphertext that encrypts the sum or product (or "evaluation") of underlying data of the input ciphertexts. Such computation (called *homomorphic* addition or multiplication or evaluation) can be done without using the secret key and one will never know anything about the processed or generated data.

Since the breakthrough construction given by Gentry [22], many efforts have been dedicated to make such homomorphic encryption scheme more secure and more efficient.

2.5.1 Homomorphic encryption scheme

A homomorphic encryption scheme is a quadruple HE=(Gen, Encrypt, Decrypt, Evaluate) of probabilistic polynomial time algorithms. Gen generates a public key pk, a secret key sk and an evaluation key evk: $(\mathsf{pk}, \mathsf{sk}, \mathsf{evk}) \leftarrow \mathsf{Gen}(1^\lambda)$. Encrypt encrypts a plaintext $x \in \mathsf{X}$ to a ciphertext c under a public key pk: $c \leftarrow \mathsf{Encrypt}(\mathsf{pk}, x)$. Decrypt decrypts a ciphertext c to a plaintext x using the secret key sk: $x \leftarrow \mathsf{Decrypt}(\mathsf{sk}, c)$. Evaluate applies a function $f : \mathsf{X}^l \rightarrow \mathsf{X}$ (given as an arithmetic circuit) to ciphertexts $c_1, \ldots, c_l$ and outputs a new ciphertext c_f using the evaluation key evk : $c_f \leftarrow \mathsf{Evaluate}(\mathsf{evk}, f, c_1, \ldots, c_l)$.

A homomorphic encryption scheme HE is *L-homomorphic* for $L = L(\lambda)$ if for any function $f : \mathsf{X}^l \rightarrow \mathsf{X}$ given as an arithmetic circuit of depth L and for any l plaintexts $x_1, \ldots, x_l \in \mathsf{X}$, it holds that

$$\mathsf{Decrypt}_{\mathsf{sk}}(\mathsf{Evaluate}_{\mathsf{evk}}(f, c_1, \ldots, c_l)) \;=\; f(x_1, \ldots, x_l)$$

for $c_i \leftarrow \mathsf{Encrypt}_{\mathsf{pk}}(x_i)$ $(i = 1, \ldots, l)$ except with a negligible probability (i.e., $\mathsf{Decrypt}_{\mathsf{sk}}$ is ring homomorphic). A homomorphic encryption scheme is called *fully homomorphic* if it is

L-homomorphic for any polynomial function $L = poly(\lambda)$.

2.5.2 Two types of Ring-LWE homomorphic encryption scheme

In this section, we introduce representative two types of homomorphic encryption scheme based on Ring-LWE problem, FV-type scheme and BGV-type scheme. The FV-type scheme was proposed by Fan and Vercauteren [17], in which a message is placed in the most significant digits of the ciphertext (called MSD form). FV scheme is implemented in SEAL [37] library. The BGV-type scheme was proposed by Brakerski, Gentry, and Vaikuntanathan [6], in which a message is placed in the lowest significant digits of the ciphertext (called LSD form). BGV scheme is implemented in HElib [27] library.

Many Ring-LWE homomorphic encryption schemes including FV scheme and BGV scheme are constructed on the cyclotomic ring. We will construct FV-type and BGV-type homomorphic encryption schemes on the decomposition ring which is one of subrings of the cyclotomic ring. In Section 4, we will describe our proposal schemes.

Scheme Description

Here we describe these two types of schemes as symmetric encryption.

Setup. The system parameters are determined according a given security level, those are a ciphertext modulus q, a plaintext modulus t and a ring dimension n. The underlying cyclotomic ring is $R = \mathbb{Z}[\zeta_m] = \mathbb{Z}[x]/(\Phi(x))$ where the cyclotomic polynomial $\Phi(x)$ has degree $n = \phi(m)$. When the ring dimension m is chosen to be a power of 2, then the underlying cyclotomic ring is $R = \mathbb{Z}[x]/(x^n + 1)$. Set the n dimensional distributions, χ_{key} for a secret key and χ_{err} for a noise.

Secret key generation. A secret key $s \in R$ is sampled from the distribution χ_{key}.

Encryption. A noise $e \in R$ is sampled from the distribution χ_{err}, and using it a message $m \in R_t$ is encrypted in a LWE instance under the secret key s. The ciphertext is a pair of ring elements $ct = (a, b) \in R_q \times R_q$. In the FV-type scheme, this ciphertext forms below LWE instance

$$a + b \cdot s \equiv \left\lfloor \frac{q}{t} \right\rfloor m + e \pmod{q}.$$

In the BGV-type scheme, the ciphertext forms below LWE instance

$$a + b \cdot s \equiv m + t \cdot e \pmod{q}.$$

Decryption and Correctness. Given a ciphertext $ct = (a, b)$ and a secret key s, the noise in the ciphertext is removed using the secret key and a message m is recovered.

In the FV-type scheme, the decryption is performed by setting

$$m \leftarrow \left\lfloor \frac{t}{q}(a + b \cdot s \bmod q) \right\rceil \bmod t.$$

Note that this rounding is done for each coefficient of the ring element.

The decryption procedure will work when the rounding in the upper formula is correct. Let ϵ be a rounding error s.t. $0 \leq \epsilon = \frac{q}{t} - \left\lfloor \frac{q}{t} \right\rfloor < 1$ and we suppose $a + b \cdot s = \left\lfloor \frac{q}{t} \right\rfloor m + e + rq$ for some integer r. Then the decryption formula is

$$\begin{aligned} \frac{t}{q}(a + b \cdot s \bmod q) &= \frac{t}{q}\left(\left\lfloor \frac{q}{t} \right\rfloor m + e\right) \\ &= \frac{t}{q}\left(\left(\frac{q}{t} - \epsilon\right)m + e\right) \\ &= m + \frac{t}{q}\left(-\epsilon m + e\right) \end{aligned}$$

Let $\boldsymbol{x}$ be a coefficient vector of a ring element $(-\epsilon m + e)$ and ν be an upper bound of a canonical embedding norm $\left\|-\epsilon m + e\right\|_\infty$. By the relation (2.6), when $c_m \cdot \nu < \frac{q}{2t}$ we get

$$\frac{t}{q}\left\|\boldsymbol{x}\right\|_\infty \leq \frac{t}{q} c_m \cdot \nu < \frac{t}{q}\frac{q}{2t} = \frac{1}{2}.$$

Therefore taking q large enough to ensure $c_m \cdot \nu < \frac{q}{2t}$, the decryption procedure returns the message m correctly.

In the BGV-type scheme, the decryption is performed by setting

$$m \leftarrow (a + b \cdot s \bmod q) \bmod t.$$

When all coefficients of a ring element $a + b \cdot s$ is no wrap-around by $\bmod q$ then $a + b \cdot s = m + t \cdot e$ and decryption works. Let $\boldsymbol{x}$ be a coefficient vector of $a + b \cdot s$ and ν be an upper bound of a canonical embedding norm $\left\|a + b \cdot s\right\|_\infty$. By the relation (2.6), when $c_m \cdot \nu < \frac{q}{2}$ then $\boldsymbol{x} \leq c_m \cdot \nu < \frac{q}{2}$. Therefore, the decryption works correctly by taking q large enough to ensure $c_m \cdot \nu < \frac{q}{2}$.

Homomorphic operations

Addition. Let $ct_i = (a_i, b_i)$ for $i = 1, 2$ be two ciphertexts encrypted messages m_1, m_2 under a same secret key s. In both FV-type and BGV-type scheme, homomorphic addition generates a new ciphertext $ct' = (a', b')$ of the message $m_1 + m_2$ by just adding elements of ciphertexts as follows: $a' = a_1 + a_2 \bmod q$ and $b' = b_1 + b_2 \bmod q$. The noise of ct' is sum of noises in the added ciphertexts.

Multiplication. We start by observing a result of multiplying two equations with respect to ciphertext ct_1, ct_2. In the FV-type scheme,

$$\begin{aligned}\frac{t}{q}(a_1 + b_1 \cdot s)(a_2 + b_2 \cdot s) &\equiv \frac{t}{q}\left(\left\lfloor \frac{q}{t} \right\rfloor m_1 + e_1\right)\left(\left\lfloor \frac{q}{t} \right\rfloor m_2 + e_2\right) \pmod q \\ &\equiv \frac{t}{q}\left\lfloor \frac{q}{t} \right\rfloor^2 m_1 m_2 + \frac{t}{q}\left(\left\lfloor \frac{q}{t} \right\rfloor (m_1 e_2 + m_2 e_1) + e_1 e_2\right) \pmod q\end{aligned}$$

and in the BGV-type scheme,

$$\begin{aligned}(a_1 + b_1 \cdot s)(a_2 + b_2 \cdot s) &\equiv (m_1 + t \cdot e_1)(m_2 + t \cdot e_2) \pmod q \\ &\equiv m_1 m_2 + t(m_1 e_2 + m_2 e_1 + t e_1 e_2) \pmod q.\end{aligned}$$

The right side of those congruences form encryption of the message $m_1 m_2$ with added noise terms.

Next we observe the left side of those congruences and consider how to create new ciphertext from ct_1, ct_2 without using the secret key s.

$$(a_1 + b_1 \cdot s)(a_2 + b_2 \cdot s) \equiv a_1 a_2 + (a_1 b_2 + a_2 b_1)s + b_1 b_2 s^2 \pmod q$$

Key Switching (Linearization)

Using an encryption of a square of the secret key s^2 under the secret key s, called switching key swk, we can linearize the s^2 to s. For example in the BGV-type scheme, let switching key be $swk = (A, B)$ s.t. $A + Bs \equiv s^2 + te \pmod q$,

$$\begin{aligned}(a_1 + b_1 \cdot s)(a_2 + b_2 \cdot s) &\equiv a_1 a_2 + (a_1 b_2 + a_2 b_1)s + b_1 b_2 (A + Bs) \pmod q \\ &\equiv (a_1 a_2 + b_1 b_2 A) + (a_1 b_2 + a_2 b_1 + b_1 b_2 B)s - b_1 b_2 te \pmod q.\end{aligned}$$

Let $a' = a_1 a_2 + b_1 b_2 A$ and $b' = a_1 b_2 + a_2 b_1 + b_1 b_2 B$ then $c' = (a', b')$ can be regarded as a ciphertext of the message $m_1 m_2$ with additive noise term $e' = b_1 b_2 te$.

To reduce the noise of the switching key and the additive noise term e', the switching key swk is generated by using a word-decomposition technique. In the below procedure, w denotes a word and l_w denotes the number of the words.

SwitchKeyGen(s^2, s) :

For $j = 0$ to $l_w - 1$

$e_j \leftarrow \chi_{err}$

$h_j = w^j s^2 + te_j \bmod q$

$B_j \xleftarrow{\mathsf{u}} R_q$

$A_j = -B_j \odot s + h_j \bmod q$

return swk $= ((A_j, B_j)_{j=0}^{l_w-1})$

The multiplication procedure computes $\alpha = \left\lfloor \frac{t}{q}(a_1 a_2) \right\rceil \bmod q$, $\beta = \left\lfloor \frac{t}{q}(a_1 b_2 + a_2 b_1) \right\rceil \bmod q$ and $\gamma = \left\lfloor \frac{t}{q}(b_1 b_2) \right\rceil \bmod q$ for the FV-type scheme, or computes $\alpha = a_1 a_2 \bmod q$ and $\beta = a_1 b_2 + a_2 b_1 \bmod q$ and $\gamma = b_1 b_2 \bmod q$ for the BGV-type scheme.

The term γ is decomposed to word expression d'_j and used for computing $A' = b_1 b_2 A$ and $B' = b_1 b_2 B$ such as

$$A' = \Sigma_{j=0}^{l_w-1} A_j \odot d'_j \bmod q,$$
$$B' = \Sigma_{j=0}^{l_w-1} B_j \odot d'_j \bmod q.$$

Finally the procedure computes $a' = \alpha + A' \bmod q$ and $b' = \beta + B' \bmod q$ and returns the ciphertext $ct' = (a', b')$.

Modulus switching. The noise in a ciphertext can be reduced by scaling down the ciphertext. The outline of this technique is dividing the ciphertext by some prime p that does not change an encrypted message in the division operation.

Here we explain the rescaling technique in the case of BGV-type scheme. In the BGV-type scheme, the ciphertext modulus is generated as a product of above primes s.t. $q_l = \Sigma_{j=0}^{l} p_j$ for $l = 0$ to $L - 1$ where $p_j \equiv 1 \pmod t$. L is a system parameter. The following is a rescaling function that scales down a ciphertext from q_l to $q_{l'}$, which means to divide by $P = \dfrac{q_l}{q'_l}$.

Rescale$(ct = (a, b, l), l')$:

Fix δ_a s.t. $\delta_a \equiv a \pmod P$ and $\delta_a \equiv 0 \pmod t$,

δ_b s.t. $\delta_b \equiv b \pmod P$ and $\delta_b \equiv 0 \pmod t$

$$a' = \frac{a - \delta_a}{P}, b' = \frac{b - \delta_b}{P}$$

return $ct' = (a', b', l')$

The output ciphertext forms

$$a' + b's = \frac{1}{P}((a - \delta_a) + (b - \delta_b)s)$$

By the conditions $\delta_a \equiv a \pmod{P}$ and $\delta_b \equiv b \pmod{P}$, this equation is divisible exactly by P over the integers. To divide by P does not affect in modulo t since P is chosen to satisfy $P \equiv 1 \pmod{t}$. The condition $\delta_a \equiv 0 \pmod{t}$ and $\delta_b \equiv 0 \pmod{t}$ leads the relation $(a - \delta_a) + (b - \delta_b)s \equiv a + bs \pmod{t}$. Therefore, the output ciphertext ct' can recover the same massage m of ct.

Now we consider magnitude of the noise in the rescaled ciphertext. Recall that for correctness of decryption in the BGV-type scheme, the upper bound of the canonical embedding norm $\|a + bs\|_\infty$ is evaluated. The norm of the rescaled ciphertext holds the below inequation

$$\begin{aligned}\|a' + b's\|_\infty &= \frac{1}{P}(\|(a - \delta_a) + (b - \delta_b)s\|_\infty) \\ &< \frac{1}{P}\|a + bs\|_\infty + \frac{1}{P}\|\delta_a + \delta_b s\|_\infty.\end{aligned}$$

Observing briefly, the original canonical norm $\|a + bs\|_\infty$ is scale down to $1/P$ but additional noise $\frac{1}{P}\|\delta_a + \delta_b s\|_\infty$ is occurred. In the work [25] [15], they analyze the noise magnitude of the rescaled ciphertext.

2.5.3 Bootstrap

The rescaling technique reduces noise of the ciphertext, but it limits the number of homomorphic operation. The bootstrap technique changes a ciphertext including a large noise to a fresh ciphertext of the same message. the bootstrapping enables to remove the number constraint about homomorphic operation.

The original bootstrapping technique was proposed by Gentry [22]. The main idea is that, to apply decryption procedure homomorphically to a ciphertext under the encryption of the secret key, the noises in the original ciphertext are removed and the outer ciphertext becomes a fresh ciphertext of the same message.

Considering how to decrypt homomorphically, we don't have homomorphic decryption operation for LSD form which calculates a remainder by plaintext modulus. On the other hand,

we can remove lower noise homomorphically by right shifting, which is the decryption method of MSD form using homomorphic additive and multiplicative operations. Concretely speaking, extract digits from the original ciphertext (which equals to sum of message and noise implicitly) and calculate its noise and remove that from total. All of those operations are executed homomorphically.

Chapter 3

SIMD Operations of Homomorphic Encryption

3.1 Batching techniques

The message space of the basic Ring-LWE encryption is the cyclotomic ring R_t. Many applications may want to use messages in integers rather than ring elements. When use $\mathbb{Z}_t$ coefficients in polynomials with respect to the ring elements for the integer messages, we gain only one product of two messages from each constant term of the two polynomials. Using Chinese Remainder Theorem (CRT), which divides the ring to many slots, we can get many products of messages from every slots.

The CRT batching technique on the Ring-LWE homomorphic encryption was firstly introduced by Smart and Vercauteren [45]. This technique allows us to encrypt multiple messages in a single ciphertext and execute Single Instruction Multiple Data (SIMD) operations homomorphically on it. The CRT is an isomorphism among a residual ring R/pR and the product of residual fields $R/\mathfrak{P}_i$ where $\mathfrak{P}_i$ are prime ideals of the prime p. HEAAN [10] uses a complex canonical embedding for batching that maps an element of a residual ring R/pR to multiple complex numbers by conjugate maps.

3.2 Algebraic slots

3.2.1 Prime decomposition

The slot structure on some algebraic field is determined by how the prime p decomposes into prime ideals on the algebraic field. In this section, we observe the prime decomposition in various kinds of fields.

Algebraic number field. Let K be an algebraic number field of degree n and $\mathcal{O}_K$ be its ring of integers. A principal ideal (p) generated by prime p decomposes into prime ideals $\mathfrak{P}_0, \ldots, \mathfrak{P}_{g-1}$ of $\mathcal{O}_K$ uniquely

$$(p) = \mathfrak{P}_0^{e_0} \cdots \mathfrak{P}_{g-1}^{e_{g-1}}$$

where $\mathfrak{P}_i$ are distinct prime ideals. Let d_i be $d_i \stackrel{\text{def}}{=} [\mathcal{O}_K/\mathfrak{P}_i : \mathbb{F}_p]$ then the norm of $\mathfrak{P}_i$ is $N\mathfrak{P}_i = p^{d_i}$ and

$$n = [K : \mathbb{Q}] = e_0 d_0 + \cdots + e_{g-1} d_{g-1}.$$

Galois extension field. When K is a Galois extension field over $\mathbb{Q}$, e_i are same value and also d_i are same value, therefore

$$(p) = (\mathfrak{P}_0 \cdots \mathfrak{P}_{g-1})^e, N\mathfrak{P}_i = p^d, n = edg.$$

Cyclotomic ring. Furthermore, when K is a m-th cyclotomic field $K = \mathbb{Q}(\zeta)$ with a primitive m-th root of unity ζ and m is represented as $\mathsf{m} = \mathsf{m}_0\mathsf{p}^{\mathsf{t}}$, $e = \varphi(p^t)$ and d is a multiplicative order of p mod m_0 ($d = \mathrm{ord}^{\times}_{\mathsf{m}_0}(p)$).

Here, we consider the case that the p does not divide m ($p \nmid \mathsf{m}$), that is $\mathsf{m}_0 = \mathsf{m}$, $p^t = 1$ and $e = \varphi(p^t) = 1$. In this case, the factorization of prime p on the cyclotomic ring $R = \mathbb{Z}[\zeta]$ is as follows:

$$(p) = \mathfrak{P}_0 \cdots \mathfrak{P}_{g-1}, N\mathfrak{P}_i = p^d, n = \varphi(\mathsf{m}) = dg.$$

Totally split case

The case of $e_i = 1$ and $d_i = 1$ for all i is called *totally split* case. When $d = 1$ in the above

condition, that is $p \equiv 1 \pmod{\mathsf{m}}$ (i.e. $\mathsf{m}|(\mathsf{p}-1)$), the prime p decompose completely in the cyclotomic ring.

$$(p) = \mathfrak{P}_0 \cdots \mathfrak{P}_{g-1}, N\mathfrak{P}_i = p, n = \varphi(\mathsf{m}) = g.$$

3.2.2 Structure of prime ideals of prime p

Let K be an algebraic number field $K = \mathbb{Q}(\omega)$ of degree n with a minimal polynomial $F(X)$ of ω, and $\mathcal{O}_K$ be its ring of integers. As described above, a principal ideal (p) generated by prime p decomposes into prime ideals $\mathfrak{P}_0, \ldots, \mathfrak{P}_{g-1}$ of $\mathcal{O}_K$ uniquely

$$(p) = \mathfrak{P}_0^{e_0} \cdots \mathfrak{P}_{g-1}^{e_{g-1}}.$$

Although the minimal polynomial $F(X)$ is irreducible over $\mathbb{Q}$, by taking mod p, it will be factored into a product of several polynomials $F_i(X)$'s:

$$F(X) \equiv F_0(X)^{e_0} \cdots F_{g-1}(X)^{e_{g-1}} \pmod{p},$$

where all of $F_i(X) \in \mathbb{Q}[X]$ are irreducible mod p and their degree are d_i.

It is known that each prime factor $\mathfrak{P}_i$ is generated by p and $F_i(\omega)$ as ideals of $\mathcal{O}_K$:

$$\mathfrak{P}_i = (p, F_i(\omega)).$$

Therefore, we can know each prime factor $\mathfrak{P}_i$ by factorization of the minimal polynomial $F(X)$ mod p.

Cyclotomic ring. In the case of the m-th cyclotomic field $K = \mathbb{Q}(\zeta)$ where prime p does not divide m ($p \nmid m$), the cyclotomic polynomial $\Phi(X)$ decomposes into g polynomials by taking mod p:

$$\Phi(X) \equiv F_0(X) \cdots F_{g-1}(X) \pmod{p}, \tag{3.1}$$

where all of $F_i(X) \in \mathbb{Z}[X]$ has degree $d = \mathrm{ord}_{\mathsf{m}}^{\times}(p)$ which is a divisor of $n = \varphi(\mathsf{m})$. The number of factors is equal to $g = n/d$. The prime ideals on the cyclotomic ring $R = \mathbb{Z}[\zeta]$ are

$$\mathfrak{P}_i = (p, F_i(\zeta)).$$

Totally split case

In the case of p decomposes completely in the cyclotomic ring (i.e. $p \equiv 1 \pmod{\mathsf{m}}$),

$$\Phi(X) \equiv \Pi_{i \in \mathbb{Z}_{\mathsf{m}}^*}(X - \omega^i) \pmod{p}$$

where ω is some primitive m-th root of unity. The prime ideals are

$$\mathfrak{P}_i = (p, (\zeta - \omega^i)).$$

The condition $p \equiv 1 \pmod{\mathsf{m}}$ induces an important property, that is it guarantees the existence of a primitive m-th root of unity ω in $\mathbb{Z}_p$. This integral primitive m-th root of unity ω is useful for the Number Theoretic Transform (NTT).

3.2.3 Chinese Reminder Theorem

Theorem 1 (Chinese Reminder Theorem (CRT)) *Let $\mathcal{I}_1, \cdots, \mathcal{I}_k$ be k integral and pairwise co-prime ideals of an integral ring R. Let $\mathcal{I}$ be the product of $\mathcal{I}_1, \cdots, \mathcal{I}_k$, then the following ring homomorphism holds*

$$R/\mathcal{I} \simeq R/\mathcal{I}_1 \oplus \cdots \oplus R/\mathcal{I}_k.$$

From the prime factorization $(p) = \mathfrak{P}_0^{e_0} \cdots \mathfrak{P}_{g-1}^{e_{g-1}}$ and above CRT, we get an isomorphism among a residual ring $\mathcal{O}_K/(p)$ and the product of residual fields $\mathcal{O}_K/\mathfrak{P}_i$.

$$\mathcal{O}_K/(p) \simeq \mathcal{O}_K/\mathfrak{P}_0^{e_0} \oplus \cdots \oplus \mathcal{O}_K/\mathfrak{P}_{g-1}^{e_{g-1}}.$$

Cyclotomic ring. In the cyclotomic ring $R = \mathbb{Z}[\zeta]$, each corresponding residual field is given by

$$R/\mathfrak{P}_i \simeq R/(p, F_i(\zeta)) \simeq \mathbb{Z}_p[X]/F_i(X) \simeq \mathrm{GF}(p^d)$$

for $i = 0, \ldots, g-1$. Thus by CRT, we have

$$R_p \simeq R/\mathfrak{P}_0 \oplus \cdots \oplus R/\mathfrak{P}_{g-1} \simeq \mathrm{GF}(p^d) \oplus \cdots \oplus \mathrm{GF}(p^d).$$

Totally split case

When p decomposes completely, the CRT is

$$R_p \simeq \oplus_{i \in \mathbb{Z}_{\mathsf{m}}^*} R/(p, (\zeta - \omega^i))$$

where ω is a primitive m-th root of unity. By taking ω in $\mathbb{Z}_p$,

$$\oplus_{i\in\mathbb{Z}^*_{\mathsf{m}}} R/(p,(\zeta-\omega^i)) \;\simeq\; \mathbb{Z}_p \oplus \cdots \oplus \mathbb{Z}_p.$$

Let $\boldsymbol{a} = (a_i)_{i\in[0,n-1]}$ be a coefficient vector of a ring element $a = \Sigma_{i\in[0,n-1]} a_i\zeta^i$ and $\boldsymbol{b} \in \mathbb{Z}_p^n$ a integer slot vector. The transform from the ring element to the slots is represented as a linear transform such as:

$$\boldsymbol{b} \;=\; \begin{pmatrix} \omega^{i_0\cdot 0} & \omega^{i_0\cdot 1} & \cdots & \omega^{i_0\cdot(n-1)} \\ \omega^{i_1\cdot 0} & \omega^{i_1\cdot 1} & \cdots & \omega^{i_1\cdot(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^{i_{n-1}\cdot 0} & \omega^{i_{n-1}\cdot 1} & \cdots & \omega^{i_{n-1}\cdot(n-1)} \end{pmatrix} \boldsymbol{a}$$

where $i_k \in \mathbb{Z}^*_{\mathsf{m}}$ $(k = [0, n-1])$. This transformation is called Number Theoretic Transform (NTT).

3.3 Complex canonical embedding slots

Recall the canonical embedding for the m-th cyclotomic field $K = \mathbb{Q}(\zeta)$ (see Section 2.3.4). Define a space $H \stackrel{\text{def}}{=} \{x \in \mathbb{C}^{\mathbb{Z}^*_{\mathsf{m}}} \;:\; x_i = \overline{x}_{m-i} \quad (\forall i \in \mathbb{Z}^*_{\mathsf{m}})\}$. The canonical embedding map $\sigma : K \;\to\; H\,(\subset \mathbb{C}^{\mathbb{Z}^*_{\mathsf{m}}})$ is

$$\sigma(a) \;=\; (\rho_i(a))_{i\in\mathbb{Z}^*_{\mathsf{m}}}$$

where ρ_i's are automorphisms defined by $\rho_i(\zeta) = \zeta^i$ for $i \in \mathbb{Z}^*_{\mathsf{m}}$. Note that $\rho_i(a) = \overline{\rho_{m-i}(a)}$ for $i \in \mathbb{Z}^*_{\mathsf{m}}$.

The ring homomorphism of the automorphisms ρ_i makes slot-wise homomorphism for ring elements $a, b \in R$ such that

$$\begin{aligned} \sigma(a+b) \;&=\; (\rho_i(a+b))_{i\in\mathbb{Z}^*_{\mathsf{m}}} \\ &=\; (\rho_i(a)+\rho_i(b))_{i\in\mathbb{Z}^*_{\mathsf{m}}}. \end{aligned}$$

Using a coefficient vector $\boldsymbol{a} = (a_i)_{i\in[n]}$ of a ring element $a = \Sigma_{i\in[n]} a_i\zeta^i$, the canonical embedding of the ring element a into the complex numbers of the slots is represented as a linear transform such as:

$$\sigma(a) \;=\; \begin{pmatrix} \rho_{i_0}(a) \\ \rho_{i_1}(a) \\ \vdots \\ \rho_{i_{n-1}}(a) \end{pmatrix} = \begin{pmatrix} \zeta^{i_0\cdot 0} & \zeta^{i_0\cdot 1} & \cdots & \zeta^{i_0\cdot(n-1)} \\ \zeta^{i_1\cdot 0} & \zeta^{i_1\cdot 1} & \cdots & \zeta^{i_1\cdot(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \zeta^{i_{n-1}\cdot 0} & \zeta^{i_{n-1}\cdot 1} & \cdots & \zeta^{i_{n-1}\cdot(n-1)} \end{pmatrix} \boldsymbol{a}$$

where $i_k \in \mathbb{Z}_{\mathsf{m}}^*$ $(k = [n])$. In the case of m is a prime, this transformation is as same as Discrete Fourier Transform (DFT).

3.4 Utilization of the slot structures

The slot structure is used in two ways. The first way is SIMD operation and the second way is fast multiplication of ring elements. In the SIMD operation, we compute ring elements packing multiple messages. The resulting ring element includes the result of slot-wise computation of the messages. To multiply ring elements efficiently, we convert each ring element to the slots and multiply in slot-wise, and then pack the product in the slots back to the ring element. In this way, we can reduce computational cost of multiplication of the ring elements.

3.4.1 SIMD operation of messages

In the algebraic slot structure, each slot with structure $\mathrm{GF}(p^d)$ is represented as a degree d polynomial modulo p. We set d integer messages to coefficients of the polynomial for each slot. By the homomorphic addition, we gain sums of messages coefficient-wise. However in the homomorphic multiplication, we gain products of messages only from constant terms of polynomials of the slots. Because, the other terms of polynomials generate cross terms in the slot-wise multiplication. We focus on this issue caused by $\mathrm{GF}(p^d)$ slot structure, and shrink the slot structure to $\mathbb{Z}_p$ using the decomposition ring which is a subring of the cyclotomic ring. The detail of our method is explained in the next chapter. Note that the totally split case is inappropriate for SIMD operation of the message since the prime p, which would be of the form $mk + 1$ with some integer k, is quiet large for the plaintext modulus.

The complex canonical embedding slot structure has $n = \varphi(\mathsf{m})$ complex slots. The n complex number messages are packed into one ring element $a \in \mathbb{Z}[\zeta]$ using a inverse map of the canonical embedding map. The image of the inverse map is complex number, thus a rounding operation is required to be a ring element.

3.4.2 Fast multiplication of ciphertexts

The totally split case can be used for the fast multiplication of ciphertexts in the algebraic slot structure, since the ciphertext modulus is large enough to use it. Thus, we can compute slot-wise

multiplication in the simple $\mathbb{Z}_p$ slots.

For the complex canonical embedding slot structure, slot-wise multiplication is multiplying complex numbers.

3.5 Slot structures of various schemes

Here we summarize the slot structures of various number fields and plot the representative homomorphic encryption schemes having SIMD function in Figure 3.1.

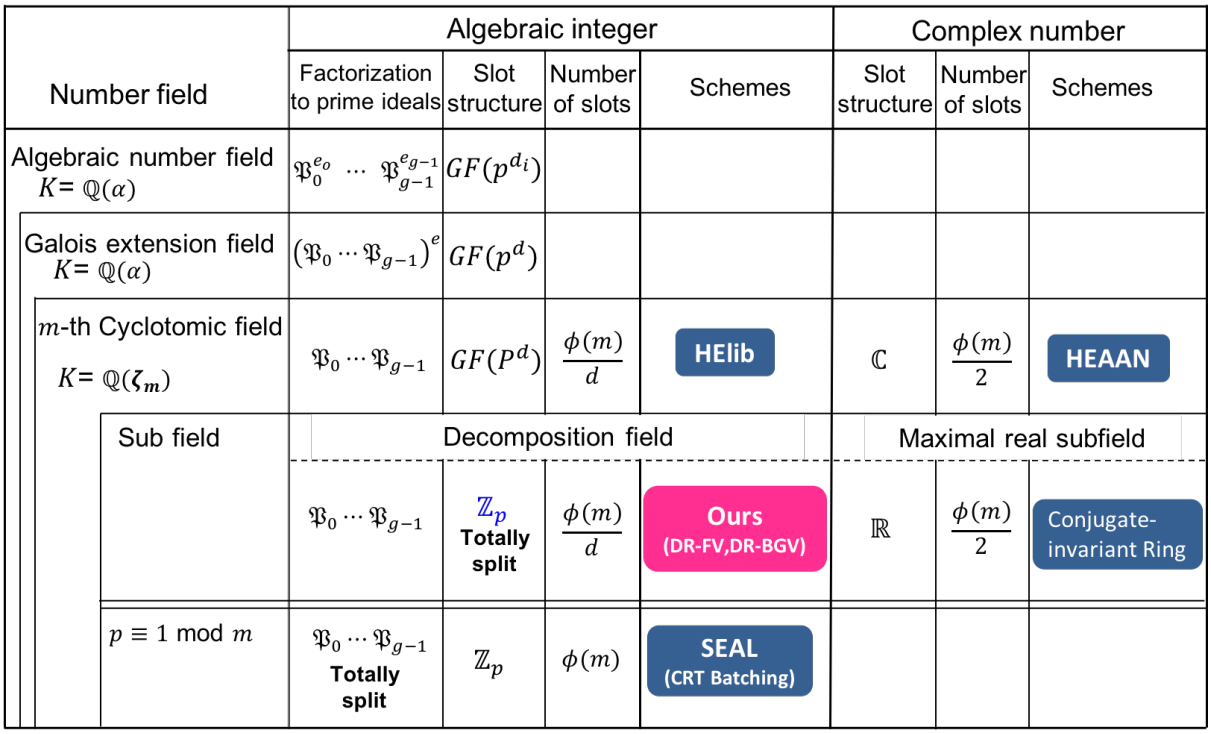

Figure 3.1: Slot structures of homomorphic encryption schemes

The pack/unpack system with homomorphism realizes SIMD operation and the fast multiplication.

One of those systems is CRT which is the isomorphism between a residual ring for a prime p and a direct product of residual fields by prime ideals of the prime p. CRT generates the algebraic slot structure. On the cyclotomic ring, each slot is a finite field GF(p^d) and only a constant term in d coefficients is available for homomorphic multiplication, though computation and space are needed for d dimensions, as we state before. HElib [27] use this structure. We construct new slot structure of $\mathbb{Z}_p$ using the decomposition ring. The slot structure obtains d times parallelism

against the structure of the cyclotomic ring in a same security level. The CRT batching of SEAL [37] use the case of the prime p totally split. In this case, the plaintext modulus is large and it requires the large ciphertext modulus, and it causes large noise growth.

Another pack/unpack system is the complex canonical embedding. That maps an element of a number field (or ring) to the complex numbers, which are evaluation results by the element of all conjugate maps of the number fields (or rings). Homomorphism of the conjugate map enables the homomorphic computation of SIMD operation. HEAAN [10] uses this method in their encoding. For real number messages, only real part of the complex number is used for each slot, though computation and space are needed for imaginary part. In the study [32], authors focus on this issue and improve HEAAN using a maximal real subfield $F = \mathbb{Q}(\xi)$ of a cyclotomic field $K = \mathbb{Q}(\zeta)$ where $\xi = \zeta + \zeta^{-1}$. The plaintext slot structure based on a conjugate-invariant ring, which is a ring of integers of the maximal real subfield, is real number, since the ξ and its conjugate elements are real number. Their scheme achieves 2 times parallelism of HEAAN in the same security level.

Chapter 4

Homomorphic Encryption Scheme Based on Decomposition Ring

4.1 Introduction

We propose the decomposition ring homomorphic encryption scheme, that is a homomorphic encryption scheme built on the decomposition ring, which is a subring of cyclotomic ring. By using the decomposition ring the structure of plaintext slot becomes $\mathbb{Z}_{p^l}$, instead of GF(p^d) in conventional schemes on the cyclotomic ring. For homomorphic multiplication of integers, one can use the full of $\mathbb{Z}_{p^l}$ slots using the proposed scheme, although in conventional schemes one can use only one-dimensional subspace GF(p) in each GF(p^d) slot. This allows us to realize fast and compact homomorphic encryption for integer plaintexts. In fact, our benchmark results indicate that our decomposition ring homomorphic encryption schemes are several times faster than HElib for integer plaintexts due to its higher parallel computation.

Our perspective: GF(p^d) versus $\mathbb{Z}_{p^l}$ slots. In the Ring-HE schemes based on cyclotomic ring, its structure of the plaintext slot is known to be Galois field GF(p^d) of some degree d. Such plaintext structure is good for applications that use data represented by elements of Galois field GF(p^d), such as error correcting codes or AES ciphers. However, many applications will use integers modulo a power of prime p^l (i.e., elements in $\mathbb{Z}_{p^l}$) rather than elements of Galois field GF(p^d).

We focus on the fact that restricting the cyclotomic ring to its subring called "Decomposition

Ring", the slot structure shrinks from $\mathrm{GF}(p^d)$ to $\mathbb{Z}_p$. Then, by using Hensel lifting, we can enlarge the modulus from $\mathbb{Z}_p$ to $\mathbb{Z}_{p^l}$. We believe in that such plaintext structure will be more natural, easy to handle, and significantly efficient for many applications.

Plaintext space of Ring-HE schemes based on cyclotomic ring is

$$R/\mathfrak{P}_0 \oplus \cdots \oplus R/\mathfrak{P}_{g-1} \;\simeq\; \mathrm{GF}(p^d) \oplus \cdots \oplus \mathrm{GF}(p^d).$$

Recall that we can use only 1-dimensional subspace $\mathrm{GF}(p) = \mathbb{Z}_p$ in each d-dimensional slot $\mathrm{GF}(p^d)$ for homomorphic multiplication of mod-p integers.

The decomposition ring R_Z with respect to prime p is the minimum subring of R in which the prime p has the same form of prime ideal factorization as in R , that is,

$$pR_Z = \mathfrak{p}_0\mathfrak{p}_1 \cdots \mathfrak{p}_{g-1} \tag{4.1}$$

with the same number g of factors. By the minimality of R_Z, the residual fields $R_Z/\mathfrak{p}_i$ of each factor $\mathfrak{p}_i$ must be one-dimensional, that is, isomorphic to $\mathbb{Z}_p$. So the plaintext space on R_Z will be

$$(R_Z)_p \;\simeq\; R_Z/\mathfrak{p}_0 \oplus \cdots \oplus R_Z/\mathfrak{p}_{g-1} \;\simeq\; \mathbb{Z}_p \oplus \cdots \oplus \mathbb{Z}_p.$$

Applying Hensel lift $l-1$ times, we get $(R_Z)_{p^l} \;\simeq\; \mathbb{Z}_{p^l} \oplus \cdots \oplus \mathbb{Z}_{p^l}$ for p^l. Thus, the decomposition ring R_Z realizes plaintext slots of integers modulo p^l, as desired.

Two bases. We investigate structure of the decomposition ring R_Z, following the way in cyclotomic cases given by Lyubashevsky, Peikert, and Regev [39]. Then, we will give two types of bases of R_Z, called η-basis and ξ-basis, which can substitute well for the power(ful) and CRT bases in cyclotomic cases, respectively.

Construction. Based on the above investigation, we construct two types of homomorphic encryption schemes over the decomposition ring: DR-FV and DR-BGV. The DR-FV and DR-BGV schemes realize the FV [17] and the BGV scheme [6] over the decomposition ring, respectively. We also construct the bootstrap function for our DR-BGV scheme. We show several bounds on the noise growth occurring among homomorphic computations and prove that both of DR-FV and DR-BGV schemes are fully homomorphic on modulus of magnitude $q = O(\lambda^{\log \lambda})$.

Related works. We focus on the wasteful slot structure of such HE schemes based on cyclotomic ring, and improve the slot structure using the decomposition ring. Kim and Song [32] also focus on the similar issue and construct HE based on another subring of the cyclotomic ring, called "conjugate-invariant ring", aiming for efficient homomorphic fixed-point number computation.

4.2 Decomposition Rings and Their Properties

4.2.1 Decomposition Field

Let $G = \mathrm{Gal}(K|\mathbb{Q})$ be the Galois group of the m-th cyclotomic field $K = \mathbb{Q}(\zeta)$ over $\mathbb{Q}$. Let p be a prime that does not divide m. Recall such p has the prime ideal decomposition of Eq (3.2.2). The *decomposition group* G_Z of K w.r.t. p is the subgroup of G defined as

$$G_Z \stackrel{\text{def}}{=} \{\rho \in G \mid \mathfrak{P}_i^{\rho} = \mathfrak{P}_i \ \ (i = 0, \ldots, g-1)\}.$$

That is, G_Z is the subgroup of automorphisms ρ of K that stabilize each prime ideal $\mathfrak{P}_i$ lying over p. Recall the Galois group $G = \mathrm{Gal}(K|\mathbb{Q})$ is isomorphic to $\mathbb{Z}_{\mathsf{m}}^*$ via ρ^{-1}. Since p does not divide m, $p \in \mathbb{Z}_{\mathsf{m}}^*$. It is known that the decomposition group G_Z is generated by the automorphism ρ_p corresponding to the prime p, called the Frobenius map w.r.t. p: $G_Z = \langle \rho_p \rangle \simeq \langle p \rangle \subseteq \mathbb{Z}_{\mathsf{m}}^*$. The order of G_Z is equal to $d = \mathrm{ord}_{\mathsf{m}}^{\times}(p)$. The fixed field $Z = K^{G_Z}$ by G_Z is called the *decomposition field* of K (w.r.t. p). The decomposition field Z can be characterized as the smallest subfield Z of K such that $\mathfrak{P}_i \cap Z$ does not split in K, so that the prime p factorizes into prime ideals in Z in much the same way as in K. By the Galois theory, $G_Z = \mathrm{Gal}(K|Z)$. For degrees, we have $[K : Z] = |G_Z| = d$, $[Z : \mathbb{Q}] = n/d = g$. The decomposition field Z is itself the Galois extension of $\mathbb{Q}$ and its Galois group $\mathrm{Gal}(Z|\mathbb{Q}) = G/G_Z$ is given by $\mathrm{Gal}(Z|\mathbb{Q}) \simeq \mathbb{Z}_{\mathsf{m}}^*/\langle p \rangle$.

4.2.2 Decomposition Ring

The integer ring $R_Z = R \cap Z$ of the decomposition field Z is called the *decomposition ring*. Primes ideals over p in the decomposition ring R_Z are given by $\mathfrak{p}_i = \mathfrak{P}_i \cap Z$ for $i = 0, \ldots, g-1$, and the prime p factors into the product of those prime ideals in much the same way as in K:

$$pR_Z = \mathfrak{p}_0 \cdots \mathfrak{p}_{g-1}. \tag{4.2}$$

This leads to the decomposition of $(R_Z)_p$: $(R_Z)_p \simeq R_Z/\mathfrak{p}_0 \oplus \cdots \oplus R_Z/\mathfrak{p}_{g-1}$.

For each prime ideal $\mathfrak{P}_i$ (of R) lying over $\mathfrak{p}_i$, the Frobenius map ρ_p acts as the p-th power automorphism $\mathrm{pow}_p(x) = x^p$ on $R/\mathfrak{P}_i$:

$$\begin{array}{ccc} R & \longrightarrow & R/\mathfrak{P}_i \\ {\scriptstyle \rho_p}\downarrow & & \downarrow{\scriptstyle \mathrm{pow}_p} \\ R & \longrightarrow & R/\mathfrak{P}_i \end{array}$$

Then, by definition of $R_Z = R^{\langle \rho_p \rangle}$, any element in $R_Z/\mathfrak{p}_i$ must be fixed by pow_p. By definition of d, $|\langle \mathrm{pow}_p \rangle| = d$ so $[R/\mathfrak{P}_i : R_Z/\mathfrak{p}_i] = d$. This means:

$$R_Z/\mathfrak{p}_i = (R/\mathfrak{P}_i)^{\langle \mathrm{pow}_p \rangle} = \mathbb{Z}_p.$$

Thus, we see that all slots of $(R_Z)_p$ must be one-dimensional: $(R_Z)_p \simeq \mathbb{Z}_p \oplus \cdots \oplus \mathbb{Z}_p$.

$$\begin{array}{ccccc} K = \mathbb{Q}(\zeta) & & R = \mathbb{Z}(\zeta) & & R/pR \xrightarrow{\sim} \mathrm{GF}(p^d) \oplus \cdots \oplus \mathrm{GF}(p^d) \\ \Big| \Big) G_Z & & \Big| & & \Big| \Big) \langle \mathrm{pow}_p \rangle \\ Z & & R_Z & & R_Z/pR_Z \xrightarrow{\sim} \mathbb{Z}_p \oplus \cdots \oplus \mathbb{Z}_p \\ \Big| G & & \Big| & & \Big| \\ \mathbb{Q} & & \mathbb{Z} & & \mathbb{Z}/p\mathbb{Z} \end{array}$$

Here we recall Hensel Lifting:

Lemma 2 ([23] Lemma 3, Hensel Lifting) *Let p be a prime, let $i \geq 1$ be an integer, and let $F, G, \Phi \in \mathbb{Z}[X]$ be monic integer polynomials, such that F, G are co-prime modulo p, and $F \cdot G = \Phi \pmod{p^i}$. Then there exist monic polynomials $\overline{F}, \overline{G} \in \mathbb{Z}[X]$ such that $F \equiv \overline{F} \pmod{p^i}$ and $G \equiv \overline{G} \pmod{p^i}$ and $\overline{F} \cdot \overline{G} = \Phi \pmod{p^{i+1}}$.*

By Hensel-lifting of the factorization of $\Phi_{\mathsf{m}}(X) \bmod p$ (Eq (3.1)) to modulus p^l, we get factorization of $\Phi_{\mathsf{m}}(X) \bmod p^l$: $\Phi_{\mathsf{m}}(X) \equiv \overline{F}_0(X) \cdots \overline{F}_{g-1}(X) \pmod{p^l}$. Here, note that the number g of irreducible factors and the degree d of each factor remain unchanged in the lifting. According to this factorization, the ideal $p^l R$ of R is factored as $p^l R = \mathfrak{Q}_0 \cdots \mathfrak{Q}_{g-1}$ with ideals $\mathfrak{Q}_i = (p^l, \overline{F}_i(\zeta))$ of R.

Then, on the decomposition ring, we get

$$p^l R_Z = \mathfrak{q}_0 \cdots \mathfrak{q}_{g-1} \tag{4.3}$$

$$(R_Z)_{p^l} \simeq \mathbb{Z}_{p^l} \oplus \cdots \oplus \mathbb{Z}_{p^l} \tag{4.4}$$

with $\mathfrak{q}_i = \mathfrak{Q}_i \cap Z$ and $R_Z/\mathfrak{q}_i \simeq \mathbb{Z}_{p^l}$. This structure of the decomposition ring $(R_Z)_{p^l}$ brings us the plaintext structure of our decomposition ring homomorphic encryption scheme, being composed of g mod-p^l integer slots.

4.2.3 Bases of the decomposition ring R_Z

To construct homomorphic encryption schemes using some ring R, we will need two types of bases of the ring R over $\mathbb{Z}$, one for decoding elements in $R \otimes \mathbb{R}$ into its nearest element in R, and another one that enables FFT-like fast computations among elements in R. In addition, we also need some quasi-linear time transformations among vector representations with respect to the two types of bases. Here, *assuming the index* m *of cyclotomic ring* R *is prime*, we construct such two types of bases for the decomposition ring R_Z, following the cyclotomic ring case given by Lyubashevsky, Peikert and Regev [39].

The η-basis

Let m be a prime and $K = \mathbb{Q}(\zeta)$ be the m-th cyclotomic field. For a prime $p\,(\neq \mathsf{m})$, let Z be the decomposition field of K with respect to p.

Fix any set of representatives $\{t_0, \ldots, t_{g-1}\}$ of $\mathbb{Z}_{\mathsf{m}}^*/\langle p\rangle \simeq \mathrm{Gal}(Z|\mathbb{Q})$. For $i = 0, \ldots, g-1$, define

$$\eta_i \stackrel{\text{def}}{=} \mathrm{Tr}_{K|Z}(\zeta^{t_i}) = \sum_{a \in \langle p\rangle} \zeta^{t_i a} \ (\in R_Z).$$

Lemma 3 *For* $i = 0, \ldots, g-1$, *we have* $\mathrm{Tr}_{Z|\mathbb{Q}}(\eta_i) = \sum_{i=0}^{g-1} \eta_i = -1$, $\mathrm{Tr}_{Z|\mathbb{Q}}(\overline{\eta}_i) = \sum_{i=0}^{g-1} \overline{\eta}_i = -1$.

Proof. $\mathrm{Tr}_{Z|\mathbb{Q}}(\eta_i) = \mathrm{Tr}_{Z|\mathbb{Q}}(\mathrm{Tr}_{K|Z}(\zeta^{t_i})) = \mathrm{Tr}_{K|\mathbb{Q}}(\zeta^{t_i})$. So, by Lemma 1, $\mathrm{Tr}_{Z|\mathbb{Q}}(\eta_i) = -1$ for any i. Similarly, $\mathrm{Tr}_{Z|\mathbb{Q}}(\overline{\eta}_i) = \mathrm{Tr}_{Z|\mathbb{Q}}(\mathrm{Tr}_{K|Z}(\zeta^{-t_i})) = \mathrm{Tr}_{K|\mathbb{Q}}(\zeta^{-t_i}) = -1$. □

Lemma 4 *For the prime index* m, *the set* $\{\eta_0, \ldots, \eta_{g-1}\}$ *is a basis of the decomposition ring* R_Z *(w.r.t.* $p\,(\neq \mathsf{m})$*) over* $\mathbb{Z}$, *i.e.,* $R_Z = \mathbb{Z}\eta_0 + \cdots + \mathbb{Z}\eta_{g-1}$.

Proof. Since the index m is prime, the cyclotomic ring R has a basis $B = \{1, \zeta, \ldots, \zeta^{\mathsf{m}-2}\}$ over $\mathbb{Z}$. Since ζ is a unit of R, $B' := \zeta B = \{\zeta, \zeta^2, \ldots, \zeta^{\mathsf{m}-1}\}$ is also a basis of R over $\mathbb{Z}$. The fixing group $G_Z = \langle \rho_p \rangle$ of Z acts on B' and decomposes it into g orbits $\zeta^{t_i\langle p\rangle} = \{\zeta^{t_i}, \zeta^{t_i p}, \ldots, \zeta^{t_i p^{d-1}}\}$ $(i = 0, \ldots, g-1)$. An element $z = \sum_{i=1}^{\mathsf{m}-1} z_i \zeta^i \in R_Z$ that is stable under the action of G_Z must

have constant integer coefficients over the each orbits $\zeta^{t_i\langle p\rangle}$. Hence, z is a $\mathbb{Z}$-linear combination of $\{\eta_1, \ldots, \eta_g\}$. □

Definition 6 *We call the basis* $\boldsymbol{\eta} := (\eta_0, \ldots, \eta_{g-1})$ η-basis *of* R_Z. *For any* $a \in R_Z$, *there exists unique* $\boldsymbol{a} \in \mathbb{Z}^g$ *satisfying* $a = \boldsymbol{\eta}^T \boldsymbol{a}$. *We call such* $\boldsymbol{a} \in \mathbb{Z}^g$ η-vector *of* $a \in R_Z$.

The ξ-basis

By the choice of t_i's, the Galois group $\mathrm{Gal}(Z|\mathbb{Q})$ of Z is given by

$$\mathrm{Gal}(Z|\mathbb{Q}) = \{\rho_{t_0}, \ldots, \rho_{t_{g-1}}\}.$$

Elements $a \in Z$ in the decomposition field are regarded as g-dimensional $\mathbb{R}$-vectors through the canonical embedding $\sigma_Z : Z \to H_Z\,(\subset \mathbb{C}^{\mathbb{Z}_{\mathsf{m}}^*/\langle p\rangle})$ defined as $\sigma_Z(a) = (\rho_i(a))_{i\in\mathbb{Z}_{\mathsf{m}}^*/\langle p\rangle}$. The g-dimensional $\mathbb{R}$-subspace H_Z is as

$$H_Z \stackrel{\text{def}}{=} \{x \in \mathbb{C}^{\mathbb{Z}_{\mathsf{m}}^*/\langle p\rangle} : x_i = \overline{x}_{g-i} \quad (\forall i \in \mathbb{Z}_{\mathsf{m}}^*/\langle p\rangle)\}.$$

Define a $g \times g$ matrix Ω_Z over R_Z as

$$\Omega_Z = \left(\rho_{t_i}(\eta_j)\right)_{0\le i,j<g} \quad (\in R_Z^{g\times g}).$$

Note that each column of Ω_Z is the canonical embedding $\sigma_Z(\eta_j)$ of η_j. Since the index m is prime, the Galois group $\mathrm{Gal}(Z|\mathbb{Q})$ is cyclic and we can take the representatives $\{t_0, \ldots, t_{g-1}\}$ so that $t_j \equiv t^j \pmod{\langle p\rangle}$ with some $t \in \mathbb{Z}_{\mathsf{m}}^*$ for $j = 0, \ldots, g-1$. Setting $\eta = \mathrm{Tr}_{K|Z}(\zeta)$, for any i and j,

$$\rho_{t_i}(\eta_j) = \rho_{t_i}(\rho_{t_j}(\eta)) = \rho_{t_i\cdot t_j}(\eta) = \rho_{t_{i+j}}(\eta) = \eta_{i+j}.$$

In particular, Ω_Z is symmetric. We can show that:

Lemma 5 $\Omega_Z^*\Omega_Z = (\mathrm{Tr}_{Z|\mathbb{Q}}(\overline{\eta}_i\eta_j))_{0\le i,j<g} = \mathsf{m}\mathsf{I}_{\mathsf{g}} - \mathsf{d}\mathbf{1}\cdot\mathbf{1}^{\mathsf{T}}$ $(\in \mathbb{Z}^{\mathsf{g}\times\mathsf{g}})$.

Proof. For $0 \le i, j < g$,

$$\begin{aligned}\overline{\eta}_i\eta_j &= \Big(\sum_{a\in\langle p\rangle}\zeta^{-at_i}\Big)\Big(\sum_{b\in\langle p\rangle}\zeta^{bt_j}\Big) = \sum_{a,b\in\langle p\rangle}\zeta^{-at_i+bt_j}\\ &= \sum_{a\in\langle p\rangle}\sum_{b\in\langle p\rangle}\rho_a(\zeta^{-t_i+ba^{-1}t_j})\\ &= \sum_{a\in\langle p\rangle}\sum_{b\in\langle p\rangle}\rho_a(\zeta^{-t_i+bt_j})\\ &= \sum_{b\in\langle p\rangle}\mathrm{Tr}_{K|Z}(\zeta^{-t_i+bt_j}).\end{aligned}$$

Here, Suppose $i \neq j$. Then, $-t_i + bt_j \not\equiv 0 \pmod{\mathsf{m}}$ for any $b \in \langle p\rangle$. Hence, by Lemma 1,

$$\mathrm{Tr}_{Z|\mathbb{Q}}(\overline{\eta}_i\eta_j) = \sum_{b\in\langle p\rangle}\mathrm{Tr}_{K|\mathbb{Q}}(\zeta^{-t_i+bt_j}) = |\langle p\rangle|\cdot(-1) = -d.$$

If $i = j$, since $\mathrm{Tr}_{K|\mathbb{Q}}(\zeta^{-t_i+bt_i}) = \mathsf{m}-1$ only if $b = 1$ and -1 otherwise by Lemma 1,

$$\begin{aligned}\mathrm{Tr}_{Z|\mathbb{Q}}(\overline{\eta}_i\eta_i) &= \sum_{b\in\langle p\rangle}\mathrm{Tr}_{K|\mathbb{Q}}(\zeta^{-t_i+bt_i})\\ &= \mathsf{m}-1+(\mathsf{d}-1)\cdot(-1)\\ &= m-d.\end{aligned}$$

□

Corollary 1 *The set* $\{\mathsf{m}^{-1}(\eta_0-\mathsf{d}),\cdots,\mathsf{m}^{-1}(\eta_{\mathsf{g}-1}-\mathsf{d})\}$ *is the dual basis of conjugate* η*-basis* $\{\overline{\eta_0},\cdots,\overline{\eta_{g-1}}\}$*, i.e. for any* $0 \le i, j < g$*,*

$$\mathrm{Tr}_{Z|\mathbb{Q}}\Big(\frac{\eta_i-d}{\mathsf{m}}\cdot\overline{\eta_j}\Big) = \delta_{ij}.$$

In particular, $R_Z^\vee = \mathbb{Z}\frac{\eta_0-d}{\mathsf{m}}+\cdots+\mathbb{Z}\frac{\eta_{g-1}-d}{\mathsf{m}}$.

Proof. For any i, by Lemma 3 and 5 we have

$$\mathrm{Tr}_{Z|\mathbb{Q}}\Big(\frac{\eta_i-d}{\mathsf{m}}\cdot\overline{\eta_i}\Big) = \frac{1}{\mathsf{m}}(\mathsf{m}-\mathsf{d}) - \frac{\mathsf{d}}{\mathsf{m}}\cdot(-1) = 1.$$

Similarly, for any $i \neq j$ we have

$$\mathrm{Tr}_{Z|\mathbb{Q}}\Big(\frac{\eta_i-d}{\mathsf{m}}\cdot\overline{\eta_j}\Big) = \frac{-d}{\mathsf{m}} - \frac{d}{\mathsf{m}}\cdot(-1) = 0$$

□

Define a $g\times g$ matrix Γ_Z over Z as

$$\Gamma_Z \overset{\text{def}}{=} \Big(\rho_{t_i}\big(\frac{\overline{\eta_j}-d}{\mathsf{m}}\big)\Big)_{0\le i,j<g} \quad (\in Z^{g\times g}).$$

Corollary 1 means that $\overline{\Gamma}_Z^T\overline{\Omega}_Z = I$. Since Ω_Z is symmetric,

$$\Gamma_Z\Omega_Z = \Omega_Z\Gamma_Z = I. \tag{4.5}$$

Lemma 6 *For any $\boldsymbol{b} = \Omega_Z \boldsymbol{a}$, we have*

$$\boldsymbol{a} \;=\; \Gamma_Z \boldsymbol{b} \;=\; \frac{1}{\mathsf{m}}\Big(\overline{\Omega}_Z \boldsymbol{b} - d\big(\sum_j b_j\big)\cdot \mathbf{1}\Big).$$

Proof.

$$\begin{aligned}
\boldsymbol{a} \;=\; \Gamma_Z \boldsymbol{b} \;&=\; \Big(\rho_{t_i}\big(\frac{\overline{\eta}_j - d}{\mathsf{m}}\big)\Big)_{ij} \boldsymbol{b} \\
&=\; \Big(\frac{1}{\mathsf{m}}\sum_j \rho_{t_i}(\overline{\eta}_j - d) b_j\Big)_i \\
&=\; \frac{1}{\mathsf{m}}\Big(\sum_j \rho_{t_i}(\overline{\eta}_j) b_j - d\sum_j b_j\Big)_i \\
&=\; \frac{1}{\mathsf{m}}\Big(\overline{\Omega}_Z \boldsymbol{b} - d\big(\sum_j b_j\big)\cdot \mathbf{1}\Big)
\end{aligned}$$

□

Let q be a power of the prime p. (Later we will use $q = p^l$ for the plaintext modulus and $q = p^r$ for the ciphertext modulus of the FV-type scheme.) Let $\mathfrak{q} = \mathfrak{q}_0$ be the first ideal that appears in the factorization of qR_Z (Eq (4.3)). Recall that $R_Z/\mathfrak{q} \;\simeq\; \mathbb{Z}_q$.

Let

$$\Omega_Z^{(q)} \;\stackrel{\text{def}}{=}\; \Omega_Z \bmod \mathfrak{q} \;\; (\in (R_Z)_{\mathfrak{q}}^{g\times g} \simeq \mathbb{Z}_q^{g\times g})$$

$\Omega_Z^{(q)}$ is invertible mod $\mathfrak{q}$.

Definition 7 *Define $\boldsymbol{\xi} = (\xi_0, \ldots, \xi_{g-1}) \in (R_Z)_{\mathfrak{q}}^g$ by $\boldsymbol{\eta}^T \equiv \boldsymbol{\xi}^T \Omega_Z^{(q)} \pmod{\mathfrak{q}}$. We call the basis $\boldsymbol{\xi}$ of $(R_Z)_{\mathfrak{q}}$ over $\mathbb{Z}_q$ ξ-*basis *of R_Z (with respect to $\mathfrak{q}$). For any $a \in (R_Z)_{\mathfrak{q}}$, there exists unique $\boldsymbol{b} \in \mathbb{Z}_q^g$ satisfying that $a = \boldsymbol{\xi}^T \boldsymbol{b}$. We call such $\boldsymbol{b} \in \mathbb{Z}_q^g$ as ξ-*vector *of $a \in (R_Z)_{\mathfrak{q}}$.*

Lemma 7 *For any $a \in R_Z$ it holds that*

$$\begin{aligned}
&a \equiv \boldsymbol{\eta}^T \cdot \boldsymbol{a} \;\Leftrightarrow\; a \equiv \boldsymbol{\xi}^T \cdot (\Omega_Z^{(q)} \cdot \boldsymbol{a}) \pmod{\mathfrak{q}} \\
&a = \boldsymbol{\eta}^T \cdot \boldsymbol{a} \;\Leftrightarrow\; \sigma_Z(a) = \Omega_Z \boldsymbol{a} \\
&a \equiv \boldsymbol{\xi}^T \cdot \boldsymbol{b} \pmod{\mathfrak{q}} \;\Leftrightarrow\; \sigma_Z(a) \equiv \boldsymbol{b} \pmod{\mathfrak{q}}
\end{aligned}$$

Proof. The first claim is the definition of $\boldsymbol{\xi}$.

Since $\Omega_Z = \Big(\sigma_Z(\eta_j)\Big)_{0\le j<g}$, $a = \boldsymbol{\eta}^T \cdot \boldsymbol{a}$ if and only if $\sigma_Z(a) = \Omega_Z \boldsymbol{a}$.

Next,

$$\begin{aligned}
a = \boldsymbol{\xi}^T \cdot \boldsymbol{b} \;&\Leftrightarrow\; a \equiv \boldsymbol{\eta}^T (\Omega_Z^{(q)})^{-1} \cdot \boldsymbol{b} \pmod{\mathfrak{q}} \\
&\Leftrightarrow\; \sigma_Z(a) \equiv \Omega_Z (\Omega_Z^{(q)})^{-1} \cdot \boldsymbol{b} \equiv \boldsymbol{b} \pmod{\mathfrak{q}}
\end{aligned}$$

□

Lemma 8 *If* $a_1 = \boldsymbol{\xi}^T \cdot \boldsymbol{b}_1$ *and* $a_2 = \boldsymbol{\xi}^T \cdot \boldsymbol{b}_2$, *then* $a_1 a_2 = \boldsymbol{\xi}^T \cdot (\boldsymbol{b}_1 \odot \boldsymbol{b}_2)$.

Proof. $\sigma_Z(a_1 a_2) = \sigma_Z(a_1) \odot \sigma_Z(a_2) = \boldsymbol{b}_1 \odot \boldsymbol{b}_2$ □

4.2.4 Conversion between η- and ξ-vectors

Resolution of unity in R_Z mod $\mathfrak{q}$

As stated before, by Hensel-lifting the factorization of $\Phi_{\mathsf{m}}(X)$ mod p (Eq (3.1)) to modulus q which is a power of p, we get factorization of $\Phi_{\mathsf{m}}(X)$ mod q: $\Phi_{\mathsf{m}}(X) \equiv \overline{F}_0(X) \cdots \overline{F}_{g-1}(X) \pmod{q}$. According to this factorization, the ideal qR of R is factored as $qR = \mathfrak{Q}_0 \cdots \mathfrak{Q}_{g-1}$ with ideals $\mathfrak{Q}_i = (q, \overline{F}_i(\zeta))$ of R.

For each $i = 0, \ldots, g-1$, let $G_i(X) \stackrel{\text{def}}{=} \prod_{j \neq i} \overline{F}_j(X) \pmod{q}$ and $P_i(X) \stackrel{\text{def}}{=} (G_i(X)^{-1} \bmod (q, \overline{F}_i(X))) \cdot G_i(X) \pmod{q}$. It is verified that the set $\{\tau_i = P_i(\zeta)\}_{i=0}^{g-1}$ constitutes a *resolution of unity* in R mod q, i.e.

$$\tau_i \equiv \begin{cases} 1 \pmod{\mathfrak{Q}_i} & (i = 0, \ldots, g-1) \\ 0 \pmod{\mathfrak{Q}_j} & (j \neq i) \end{cases}$$

and it holds that

$$\sum_{i=0}^{g-1} \tau_i \equiv 1, \ \tau_i^2 \equiv \tau_i, \ \tau_i \tau_j \equiv 0 \pmod{q} \ (j \neq i).$$

By the Chinese remainder theorem, the resolution of unity $\{\tau_i\}_{i=0}^{g-1}$ is uniquely determined mod qR. In the following we take coefficients of each τ_i from $[-q/2, q/2)$ over the basis $B' = \{\zeta, \zeta^2, \ldots, \zeta^{\mathsf{m}-1}\}$ of R.

Lemma 9 *For any* $0 \leq i < g$ *it is that* $\tau_i \in R_Z$, *and* $\{\tau_i\}_{i=0}^{g-1}$ *is also a resolution of unity in* R_Z *mod* q.

Proof. The ideal qR_Z factors in R_Z as

$$qR_Z = \mathfrak{q}_0 \mathfrak{q}_1 \cdots \mathfrak{q}_{g-1}$$

where $\mathfrak{q}_i = \mathfrak{Q}_i \cap R_Z$ for any i.

Let $\{\tau'_i\}_{i=0}^{g-1}$ be a resolution of unity in R_Z mod q. Here, we take the coefficients of each τ'_i from $[-q/2, q/2)$ over the η-basis $\{\eta_0, \ldots, \eta_{g-1}\}$ of R_Z.

Then,

$$\tau_i' \equiv \begin{cases} 1 \pmod{\mathfrak{q}_i} & (i = 0, \ldots, g-1) \\ 0 \pmod{\mathfrak{q}_j} & (j \neq i). \end{cases}$$

Since $\mathfrak{q}_i \subset \mathfrak{Q}_i$ for any i, $\{\tau_i'\}_{i=0}^{g-1}$ is also a resolution of unity in R mod q. Since the coefficients of each τ_i' over the η-basis are in $[-q/2, q/2)$, by definition of $\eta_i = \sum_{a \in \langle p \rangle} \zeta^{t_i a}$, their coefficients over the basis B' are trivially also in $[-q/2, q/2)$. Hence, by the uniqueness of resolution, it must be that $\tau_i' = \tau_i$ for all i. □

Using the resolution of unity $\{\tau_i\}_{i=0}^{g-1}$ in R_Z, we can compute $a_i \in \mathbb{Z}_q$ satisfying $a \equiv a_i \pmod{\mathfrak{q}_i}$ given $a \in R_Z$, as follows:

$$a \bmod \mathfrak{q}_i = a\tau_i \bmod q \;=\; a_i\tau_i \bmod q \overset{\text{dividing by } \tau_i}{\longmapsto} a_i.$$

Computation of $\Omega_Z^{(q)}$

Now we can compute the matrix $\Omega_Z^{(q)} = \left(\eta_{i+j} \bmod \mathfrak{q}\right)_{0 \le i,j < g}$ $(\in \mathbb{Z}_q^{g \times g})$ by computing the entities η_{i+j} in Ω_Z as cyclotomic integers and reducing them modulo $\mathfrak{q}$ $(= \mathfrak{q}_0)$ using the resolution of unity $\{\tau_i\}_{i=0}^{g-1}$. Since the matrix $\Omega_Z^{(q)}$ has cyclic structure (the $(i+1)$-th row is a left shift of the i-th row), it is sufficient to compute its first row. Here, we remark that once we have computed the matrix $\Omega_Z^{(q)}$, we can totally forget the original structure of cyclotomic ring R, and all we need is doing various computations among η- and ξ-vectors (of elements in R_Z) with necessary conversion between them using the matrix $\Omega_Z^{(q)}$.

Computation of $\boldsymbol{b} = \Omega_Z^{(q)} \cdot \boldsymbol{a}$

To convert η-vector $\boldsymbol{a}$ of an element $a = \boldsymbol{\eta}^T \cdot \boldsymbol{a} \in R_Z$ to its corresponding ξ-vector $\boldsymbol{b}$ (satisfying $a = \boldsymbol{\xi}^T \cdot \boldsymbol{b}$), by Lemma 7, we need to compute a matrix-vector product $\boldsymbol{b} = \Omega_Z^{(q)} \cdot \boldsymbol{a}$. By Lemma 6, the inverse conversion from ξ-vector $\boldsymbol{b}$ to its corresponding η-vector $\boldsymbol{a} = \Gamma_Z \cdot \boldsymbol{b}$ also can be computed using a similar matrix-vector product $\overline{\Omega}_Z^{(q)} \cdot \boldsymbol{b}$. Here, $\overline{\Omega}_Z^{(q)} \overset{\text{def}}{=} \overline{\Omega_Z} \bmod \mathfrak{q}$.

By definition of $\Omega_Z^{(q)}$, the j-th component b_j of the product $\boldsymbol{b} = \Omega_Z^{(q)} \cdot \boldsymbol{a}$ is $b_j = \sum_{i=0}^{g-1} a_i \eta_{i+j}$ (where indexes are mod g and we omit mod$\mathfrak{q}$). This means that $\boldsymbol{b}$ is the convolution product of vector $\boldsymbol{\eta}$ and the reversal vector $(a_0, a_{g-1}, a_{g-2}, \cdots, a_1)$ of $\boldsymbol{a}$, where $\boldsymbol{\eta} = (\eta_i)_{i=0}^{g-1}$ is the first row of $\Omega_Z^{(q)}$.

Define two polynomials $f(X) = \sum_{i=0}^{g-1} \eta_i X^i$ and $g(X) = a_0 + \sum_{i=1}^{g-1} a_{g-i} X^i$ over $\mathbb{Z}_q$. Since $\boldsymbol{b}$ is the convolution product of $\boldsymbol{\eta}$ and the reversal vector of $\boldsymbol{a}$, it holds that $f(X)g(X) = \sum_{i=0}^{g-1} b_i X^i \pmod{X^g - 1}$. The polynomial product $f(X)g(X) \pmod{X^g - 1}$ can be computed in quasi-linear time $\tilde{O}(g)$ using the FFT multiplication. Thus, we know that conversions between η-vectors $\boldsymbol{a}$ and ξ-vectors $\boldsymbol{b}$ can be done in quasi-linear time $\tilde{O}(g)$.

Remark In the BGV-type scheme, the ciphertext modulus makes a chain which contains L modulus $q_0, \cdots, q_{L-1}$ using L primes $p_0, \cdots, p_{L-1}$ s.t. $q_i = \Pi_{j=0}^{i} p_j$. For each modulus q_i, we generate a matrix $\Omega_Z^{(q_i)}$ in $\mathbb{Z}_{q_i}^{g \times g}$ to convert between η-vector and ξ-vector efficiently. More precisely, the reason why the method of Section 4.2.4 and 4.2.4 work is that the modulus q factors completely on the decomposition ring R_Z with respect to the prime p. When we choose each prime p_j to satisfy $p_j \equiv 1 \pmod{\mathsf{m}}$ then p_j factors completely on the cyclotomic ring and thus also factors completely on the decomposition ring R_Z. Therefore, we can generate $\Omega_Z^{(p_j)} (\in \mathbb{Z}_{p_j}^{g \times g})$ for such primes p_j using the method of Section 4.2.4 and 4.2.4, and we get $\Omega_Z^{(q_i)}$ by CRT-lifting the matrices $\Omega_Z^{(p_j)} (j = 0, \cdots, i)$ entity-wise:

$$\Omega_Z^{(q_i)} = CRT(\Omega_Z^{(p_0)}, \cdots, \Omega_Z^{(p_i)}) \in \mathbb{Z}_{q_i}^{g \times g}.$$

4.3 Homomorphic Encryption Based on Decomposition Ring

Now we construct two types of homomorphic encryption schemes over the decomposition ring: DR-FV and DR-BGV.

4.3.1 The Ring-LWE problem on the decomposition ring

For security of our homomorphic encryption scheme over the decomposition ring, we need hardness of a variant of the decisional Ring-LWE problem over the decomposition ring. Let m be a prime. Let R_Z be the decomposition ring of the m-th cyclotomic ring R with respect to some prime $p (\neq \mathsf{m})$. Let q be a positive integer. For an element $s \in R_Z$ and a distribution χ over R_Z, define a distribution $A_{s,\chi}$ on $(R_Z)_q \times (R_Z)_q$ as follows: First choose an element a uniformly from $(R_Z)_q$ and sample an element e according to the distribution χ. Then return the pair $(a, b = as + e \bmod q)$.

Definition 8 (LWE problem on the decomposition ring)

Let q, χ be as above. The R-DLWE$_{q,\chi}$ problem on the decomposition ring R_Z *asks to distinguish samples from $A_{s,\chi}$ with $s \xleftarrow{\mathsf{u}} \mathbb{Z}_q$ and (the same number of) samples uniformly chosen from $(R_Z)_q \times (R_Z)_q$.*

4.3.2 Security

Recall the search version of Ring-LWE problem is already proved to have a quantum polynomial time reduction from the approximate shortest vector problem of ideal lattices in *any number field* by Lyubashevsky, Peikert, and Regev [38]. They proved equivalence between the search and the decisional versions of the Ring-LWE problems only for cyclotomic rings. However, it is not difficult to see that the equivalence holds also over the decomposition rings, since those are subrings of cyclotomic rings and inherit good properties about prime ideal decomposition from them.

The key of their proof of equivalence is the existence of prime modulus q for Ring-LWE problem which totally decomposes into n prime ideal factors: $qR = \mathfrak{Q}_0 \cdots \mathfrak{Q}_{n-1}$. (Their residual fields $R/\mathfrak{Q}_i$ have polynomial order q and we can guess the solution of the Ring-LWE problem modulo ideal $\mathfrak{Q}_i$, and then we can verify validity of the guess using the assumed oracle for the decisional Ring-LWE problem.) Since the decomposition ring R_Z is a subring of the cyclotomic ring R, such modulus q totally decomposes into g prime ideals also in the decomposition ring R_Z: $qR_Z = \mathfrak{q}_0 \cdots \mathfrak{q}_{g-1}$. Using this decomposition, the proof of equivalence by [38] holds also over the decomposition rings R_Z, essentially as it is.

Terada, Nakano, Okumura and Miyaji [48] conducted some experiments regarding lattice attack against Ring-LWE problem over the decomposition ring. They concluded that the Ring-LWE problem on the decomposition ring is expected to be as secure as the ordinal Ring-LWE problem on the cyclotomic ring.

4.3.3 Parameters

Let m be a prime index of cyclotomic ring R. Choose a (small) prime p, distinct from m. Let $d = \mathrm{ord}_{\mathsf{m}}^{\times}(p)$ be the multiplicative order of p mod m, and $g = (\mathsf{m} - 1)/\mathsf{d}$ be the degree of the decomposition ring R_Z of R with respect to p. The plaintext modulus $t = p^l$ is a power of p and

the ciphertext modulus q will be chosen below.

4.3.4 Sampling

We will use the following three types of sampling algorithms regarding as R_Z.

The uniform distribution $\mathcal{U}_q$: This is uniform distribution over $(\mathbb{Z}/q\mathbb{Z})^g$ identified with $(\mathbb{Z} \cap (-q/2, q/2])$.

The discrete gaussian distribution $\mathcal{G}_q(\sigma^2)$: The discrete gaussian distribution $\mathcal{G}_q(\sigma^2)$ draws a g dimension integer vector which rounds a normal gaussian distribution with zero-mean and variance σ^2 to the nearest integer and outputs that integer vector reduced modulo q (into interval $(-q/2, q/2]$).

The $\mathcal{HWT}(h)$ distribution: The $\mathcal{HWT}(h)$ distribution chooses a vector uniformly at random from $\{0, \pm 1\}^g$ that has h nonzero entries.

Bounds on noises

We analyze bounds of noises according to the way of [15].

Let $a = \sum_{i=0}^{g} a_i \eta_i$ denote a random element of R_Z. The canonical embedding of each η_i has approximately Euclidean norm $\sqrt{\phi(\mathsf{m})}$ $(= \sqrt{gd})$. Letting V_a be the variance of each coefficient of $\boldsymbol{a}$, the variance of a is estimated as $V_a g d$.

When $\boldsymbol{a} \leftarrow \mathcal{U}_q$ then V_a is $(q-1)^2/12 \approx q^2/12$, hence the variance of a is $V_U = q^2 g d/12$. When $\boldsymbol{a} \leftarrow \mathcal{G}_q(\sigma^2)$ then the variance of a is $V_G = \sigma^2 g d$. When choosing $\boldsymbol{a} \leftarrow \mathcal{HWT}(h)$, the variance of a is $V_H = hd$.

The random element of R_Z is a sum of many independent distributed random variables, hence its distribution is approximated by a gaussian distribution of the same variance. We use six standard deviations as a bound on the size of a: $\left\|a\right\|_\infty \leq 6\sqrt{V}$. We use 16 $\sigma_a \sigma_b$ as a bound of product ab of two random variables both distributed closely to gaussian distributions of variances σ_a and σ_b, respectively. For a product of three random variables with variances σ_a, σ_b and σ_c, we use 40 $\sigma_a \sigma_b \sigma_c$.

4.3.5 Encoding methods of elements in R_Z

Basically, we use η-vectors $\boldsymbol{a} \in \mathbb{Z}^g$ to encode elements $a = \boldsymbol{\eta}^T \cdot \boldsymbol{a}$ in R_Z. To multiply two elements encoded by η-vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ modulo q, first we convert those η-vectors to corresponding ξ-vectors modulo q. We can multiply resulting ξ-vectors component-wise, and then re-convert the result into its corresponding η-vector modulo q. The functions eta_to_xi and xi_to_eta use the matrix $\Omega_Z^{(q)}$ computed in advance. $(\eta_i)_{i=0}^{g-1}$ denotes the first row of $\Omega_Z^{(q)}$.

mult_eta $(\boldsymbol{a}, \boldsymbol{b}, q)$:

$\boldsymbol{\alpha}_\xi =$ eta_to_xi$(\boldsymbol{a}, q)$

$\boldsymbol{\beta}_\xi =$ eta_to_xi$(\boldsymbol{b}, q)$

$\gamma_i = \alpha_i \beta_i \bmod q \ (i = 0, \ldots, g-1)$

return $\boldsymbol{c} =$ xi_to_eta$(\boldsymbol{\gamma}_\xi, q)$

eta_to_xi$(\boldsymbol{a}, q)$:

$a(X) = a_0 + \sum_{i=1}^{g-1} a_{g-i} X^i$

$c(X) = \sum_{i=0}^{g-1} \eta_i X^i$

$b(X) = a(X)c(X) \bmod (q, X^g - 1)$

return $\boldsymbol{b}_\xi = (b_0, \ldots, b_{g-1})$

xi_to_eta$(\boldsymbol{b}_\xi, q)$:

$b(X) = b_0 + \sum_{i=1}^{g-1} b_{g-i} X^i$

$c(X) = \sum_{i=0}^{g-1} \overline{\eta}_i X^i$

$a(X) = b(X)c(X) \bmod (q, X^g - 1)$

$t = b_0 + \cdots + b_{g-1} \bmod q$

return $\boldsymbol{a} = (m^{-1}(a_i - dt) \bmod q)_{i=0}^{g-1}$

These algorithms can be computed by $O(g \log g)$ operations of underlying mod q integers by using the FFT multiplications of degree g polynomials (see Section 4.2.4).

We regard plaintext vectors $\boldsymbol{m} \in \mathbb{Z}_t^g$ as ξ-vectors of corresponding elements $m_\xi = \boldsymbol{\xi}^T \boldsymbol{m} \in (R_Z)_t$. By Lemma 8 their products $m_\xi m'_\xi \in (R_Z)_t$ encodes the plaintext vector $\boldsymbol{m} \odot \boldsymbol{m'} \in \mathbb{Z}_t^g$. For a fixed integer base w, let $l_w = \lfloor \log_w(q) \rfloor + 1$. Any vector $\boldsymbol{a} \in \mathbb{Z}_q^g$ can be written as $\boldsymbol{a} = \sum_{k=0}^{l_w-1} \boldsymbol{a}_k w^k$ with vectors $\boldsymbol{a}_k \in \mathbb{Z}_w^{l_w}$ of small entries. Define $\mathsf{WD}(\boldsymbol{a}) \stackrel{\text{def}}{=} (\boldsymbol{a}_k)_{k=0}^{l_w-1} \left(\in (\mathbb{Z}_w^g)^{l_w}\right)$.

4.3.6 Two types of Homomorphic Encryption Schemes

We construct two types of homomorphic encryption schemes based on the decomposition ring. The first scheme DR-FV is based on the FV scheme and the second scheme DR-BGV is based on the BGV scheme.

FV-type

$$a + b \cdot s \equiv \left\lfloor \frac{q}{t} \right\rfloor \cdot m + e \pmod{q}$$

BGV-type

$$a + b \cdot s \equiv f \cdot m + t \cdot e \pmod{q}$$

Here, (a, b) denotes a ciphertext, s denotes a secret key, f is a factor of plaintext m, e denotes a noise, l is ciphertext level, and t and q are modulus of plaintext and ciphertext.

4.3.7 Scheme Description

Each scheme is given as symmetric key version. The public key version is easily derived as usual.

Our schemes have two functions for HE.Gen. SecretKeyGen() generates a secret key sk. Using the secret key sk, SwitchKeyGen($\boldsymbol{x}$, sk) generates a switching key swk from $\boldsymbol{x}$ to sk. Especially it gives the evaluation key evk taking $\boldsymbol{x} = \mathsf{sk}^2$: evk $\leftarrow$ SwitchKeyGen(sk^2, sk). Encrypt(sk, $\boldsymbol{m}$) encrypts a message $\boldsymbol{m}$ under the secret key sk and outputs a ciphertext ct, and Decrypt(sk, ct) decrypts the ciphertext ct under the secret key sk and outputs a message $\boldsymbol{m}$. Given an arithmetic circuit of function f, HE.Evaluate evaluates the circuit of f on given ciphertexts homomorphically. It uses Add(ct_1, ct_2) when evaluating an addition gate and uses Mult(swk, ct_1, ct_2) when evaluating a multiplication gate.

DR-FV scheme

We use a ciphertext modulus $q = p^r$ and a plaintext modulus $t = p^l$ $(r > l)$. Let $\Delta = \frac{q}{t} = p^{r-l}$.

Key Generation. The SecretKeyGen generates a secret key $\boldsymbol{s}_\xi$ as ξ-vector.

$\mathsf{SecretKeyGen}^{\mathsf{DR-FV}}(\mathsf{prm})$:

$\boldsymbol{s} \leftarrow \chi_{key}$

$\boldsymbol{s}_\xi \leftarrow \mathsf{eta\_to\_xi}(\boldsymbol{s}, q)$

return $\mathsf{sk} = \boldsymbol{s}_\xi \in \mathbb{Z}^g$

This can be computed by $O(g \log g)$ operations of underlying mod q integers.

The SwitchKeyGen takes as input a η-vector $\boldsymbol{x}$ and a secret key $\boldsymbol{s}_\xi$, and encrypts $\boldsymbol{x}$ under $\boldsymbol{s}_\xi$. The output switching key swk is used to linearize s^2 to s in homomorphic multiplication. $\mathsf{B}_{\mathsf{lin}}^{\mathsf{FV}}$ is the upper bound of canonical embedding norm of the noise included in the swk. The base-w decomposition technique is used to make swk be of less noise.

$\mathsf{SwitchKeyGen}^{\mathsf{DR-FV}}(\boldsymbol{x}, \mathsf{sk} = \boldsymbol{s}_\xi, \mathsf{prm})$:

For $j = 0$ to $l_w - 1$

$\boldsymbol{e}_j \leftarrow \chi_{err}$

$\boldsymbol{h}_j = w^j \boldsymbol{x} + \boldsymbol{e}_j \bmod q$

$\boldsymbol{B}_j \xleftarrow{\mathsf{u}} \mathbb{Z}_q^g$

$\boldsymbol{A}_j = -\boldsymbol{B}_j \odot \boldsymbol{s}_\xi + \mathsf{eta\_to\_xi}(\boldsymbol{h}_j, q) \bmod q$

return $\mathsf{swk} = ((\boldsymbol{A}_j, \boldsymbol{B}_j)_{j=0}^{l_w-1}, \nu = \mathsf{B}_{\mathsf{lin}}^{\mathsf{FV}})$

This can be computed by $O(l_w g \log g)$ operations of underlying mod q integers.

Encryption. The input plaintext $\boldsymbol{m}$ is interpreted as a ξ-vector. In the resulting ciphertext, $\mathsf{B}_{\mathsf{clean}}^{\mathsf{FV}}$ denotes the bound of its noise.

$\mathsf{Encrypt}^{\mathsf{DR-FV}}(\mathsf{sk} = \boldsymbol{s}_\xi \in \mathbb{Z}^g, \boldsymbol{m} \in \mathbb{Z}_t^g, \mathsf{prm})$:

$\boldsymbol{e} \leftarrow \chi_{err}$

$\boldsymbol{b}_\xi \xleftarrow{\mathsf{u}} \mathbb{Z}_q^g$

$\boldsymbol{a}_\xi = -\boldsymbol{b}_\xi \odot \boldsymbol{s}_\xi + \Delta \boldsymbol{m} + \mathsf{eta\_to\_xi}(\boldsymbol{e}, q) \bmod q$

return $ct = (\boldsymbol{a}_\xi, \boldsymbol{b}_\xi, \nu = \mathsf{B}_{\mathsf{clean}}^{\mathsf{FV}})$

This can be computed by $O(g \log g)$ operations of underlying mod q integers.

Decryption. Decryption removes the noise in the ciphertext by taking its right shift (by $\frac{1}{\Delta}$) and rounding.

$\mathsf{Decrypt}^{\mathsf{DR-FV}}(\mathsf{sk} = \boldsymbol{s}_\xi \in \mathbb{Z}^g, ct = (\boldsymbol{a}_\xi, \boldsymbol{b}_\xi, \nu), \mathsf{prm})$:

$\boldsymbol{h}_\xi = \boldsymbol{a}_\xi + \boldsymbol{b}_\xi \odot \boldsymbol{s}_\xi \bmod q$

$\boldsymbol{h} = \mathsf{xi\_to\_eta}(\boldsymbol{h}_\xi, q)$

$\boldsymbol{z} = \left\lfloor \frac{\boldsymbol{h}}{\Delta} \right\rceil \bmod t$

$\boldsymbol{m} = \mathsf{eta\_to\_xi}(\boldsymbol{z}, t)$

return $\boldsymbol{m}$

This can be computed by $O(g \log g)$ operations of underlying mod q integers.

Addition. Addition algorithm takes as input two ciphertext ct_1 and ct_2 and outputs a ciphertext ct encrypting the sum of underlying plaintexts.

$\mathsf{Add}^{\mathsf{DR-FV}}(ct_1 = (\boldsymbol{a}_1, \boldsymbol{b}_1, \nu_1), ct_2 = (\boldsymbol{a}_2, \boldsymbol{b}_2, \nu_2), \mathsf{prm})$:

$\boldsymbol{a} = \boldsymbol{a}_1 + \boldsymbol{a}_2 \bmod q$

$\boldsymbol{b} = \boldsymbol{b}_1 + \boldsymbol{b}_2 \bmod q$

$\nu = \nu_1 + \nu_2$

return $ct = (\boldsymbol{a}, \boldsymbol{b}, \nu)$

This can be computed by $O(g)$ operations of underlying mod q integers.

Multiplication. Multiplication algorithm takes as input two ciphertext ct_1 and ct_2 and outputs a ciphertext ct encrypting the product of underlying plaintexts.

$\mathsf{Mult}^{\mathsf{DR-FV}}(\mathsf{swk}, ct_1 = (\boldsymbol{a}_1, \boldsymbol{b}_1, \nu_1), ct_2 = (\boldsymbol{a}_2, \boldsymbol{b}_2, \nu_2), \mathsf{prm})$:

$\boldsymbol{\alpha} = \left\lfloor \frac{1}{\Delta} \cdot \mathsf{xi\_to\_eta}(\boldsymbol{a}_1 \odot \boldsymbol{a}_2 \bmod q^2/t) \right\rceil$

$\boldsymbol{\beta} = \left\lfloor \frac{1}{\Delta} \cdot \mathsf{xi\_to\_eta}(\boldsymbol{a}_1 \odot \boldsymbol{b}_2 + \boldsymbol{a}_2 \odot \boldsymbol{b}_1 \bmod q^2/t) \right\rceil$

$\boldsymbol{\gamma} = \left\lfloor \frac{1}{\Delta} \cdot \mathsf{xi\_to\_eta}(\boldsymbol{b}_1 \odot \boldsymbol{b}_2 \bmod q^2/t) \right\rceil$

$\boldsymbol{\alpha}' = \mathsf{eta\_to\_xi}(\boldsymbol{\alpha}, q)$

$\boldsymbol{\beta}' = \mathsf{eta\_to\_xi}(\boldsymbol{\beta}, q)$

$\boldsymbol{\gamma}' = \mathsf{eta\_to\_xi}(\boldsymbol{\gamma}, q)$

$\nu = \mathsf{B}^{\mathsf{FV}}_{\mathsf{direct\_mult}}(\nu_1, \nu_2)$

$ct = \mathsf{Linearize}^{\mathsf{DR-FV}}(\mathsf{swk}, (\boldsymbol{\alpha}', \boldsymbol{\beta}', \boldsymbol{\gamma}', \nu), \mathsf{prm})$

return ct

Here, Linearize switches the key in the ciphertext from s^2 to s using swk.

$\mathsf{Linearize}^{\mathsf{DR-FV}}(\mathsf{swk}, ct = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \nu), \mathsf{prm})$:

$(\boldsymbol{a}, \boldsymbol{b}) = \mathsf{KeySwitch}^{\mathsf{DR-FV}}(\mathsf{swk}, \boldsymbol{\gamma}, \mathsf{prm})$

$\boldsymbol{a}' = \boldsymbol{\alpha} + \boldsymbol{a} \bmod q$

$\boldsymbol{b}' = \boldsymbol{\beta} + \boldsymbol{b} \bmod q$

$\nu' = \nu + \mathsf{B}_{\mathsf{lin}}^{\mathsf{FV}}$

return $ct = (\boldsymbol{a}', \boldsymbol{b}', \nu')$

KeySwitch takes a switching key swk of $\boldsymbol{y}$ and a vector $\boldsymbol{x}$ as input, and returns a ciphertext encrypting the product $\boldsymbol{x} \odot \boldsymbol{y}$. In the multiplication, swk is an encryption of s^2 and $\boldsymbol{x} = \gamma$ so the output is a ciphertext of $s^2\gamma$.

$\mathsf{KeySwitch}^{\mathsf{DR-FV}}(\mathsf{swk} = (\{\boldsymbol{A}_j, \boldsymbol{B}_j\}_{j=0}^{l_w-1}, \nu), \boldsymbol{x}, \mathsf{prm})$:

$\boldsymbol{d} = \mathsf{xi\_to\_eta}(\boldsymbol{x}, q)$

Decompose $\boldsymbol{d} = \Sigma_{j=0}^{l_w-1} \boldsymbol{d}_j w^j$

$\boldsymbol{d}'_j = \mathsf{eta\_to\_xi}(\boldsymbol{d}_j, q)$ for $j = 0 \cdots l_w - 1$

$\boldsymbol{A} = \Sigma_{j=0}^{l_w-1} \boldsymbol{A}_j \odot \boldsymbol{d}'_j \bmod q$

$\boldsymbol{B} = \Sigma_{j=0}^{l_w-1} \boldsymbol{B}_j \odot \boldsymbol{d}'_j \bmod q$

return $(\boldsymbol{A}, \boldsymbol{B})$

As easily verified, the total complexity of $\mathsf{Mult}^{\mathsf{DR-FV}}$ is $O(l_w g \log g)$ of underlying mod q integers.

DR-BGV scheme

Ciphertext modulus chain: For scaling down ciphertext, the ciphertext modulus makes a chain which contains L modulus $q_0, \cdots, q_{L-1}$ using L primes $p_0, \cdots, p_{L-1}$ s.t. $q_i = \Pi_{j=0}^{i} p_j$ and primes are $p_i \equiv 1 \pmod{\mathsf{m}}$. We call a modulus q_i ciphertext as a level-i ciphertext. The level of a fresh ciphertext is $L - 1$ and one level is consumed by one multiplication. In the scheme, we use another special modulus q_s to reduce noise.

Key Generation. The SecretKeyGen generates a secret key of the maximum level $L - 1$.

$\mathsf{SecretKeyGen}^{\mathsf{DR-BGV}}(\mathsf{prm})$:

$\boldsymbol{s} \leftarrow \chi_{key}$

$\boldsymbol{s}_\xi \leftarrow \mathsf{eta\_to\_xi}(\boldsymbol{s}, q_{L-1})$

return $\mathsf{sk} = \boldsymbol{s}_\xi \in \mathbb{Z}^g$

This can be computed by $O(g \log g)$ operations of underlying mod q_{L-1} integers.

In the SwitchingKeyGen, we temporarily enlarge the modulus by multiplying q_s, to reduce the noise occurring.

$\mathsf{SwitchKeyGen}^{\mathsf{DR-BGV}}(\boldsymbol{x}, \mathsf{sk} = \boldsymbol{s}_\xi, \mathsf{prm})$:

For $j = 0$ to $l_w - 1$

$\boldsymbol{e}_j \leftarrow \chi_{err}$

$\boldsymbol{h}_j = q_s w^j \boldsymbol{x} + t\boldsymbol{e}_j \bmod q_{L-1}q_s$

$\boldsymbol{B}_j \xleftarrow{\mathsf{u}} \mathbb{Z}^g_{q_{L-1}q_s}$

$\boldsymbol{A}_j = -\boldsymbol{B}_j \odot \boldsymbol{s}_\xi + \mathsf{eta\_to\_xi}(\boldsymbol{h}_j, q_{L-1}q_s) \bmod q_{L-1}q_s$

return $\mathsf{swk} = ((\boldsymbol{A}_j, \boldsymbol{B}_j)_{j=0}^{l_w-1}, \nu = \mathsf{B}^{\mathsf{BGV}}_{\mathsf{lin}})$

This can be computed by $O(l_w g \log g)$ operations of underlying mod $q_{L-1}q_s$ integers.

Encryption.

The level of a fresh ciphertext is the maximum level $L - 1$.

$\mathsf{Encrypt}^{\mathsf{DR-BGV}}(\mathsf{sk} = \boldsymbol{s}_\xi \in \mathbb{Z}^g, \boldsymbol{m} \in \mathbb{Z}^g_t, \mathsf{prm})$:

$\boldsymbol{e} \leftarrow \chi_{err}$

$\boldsymbol{e'} = t\boldsymbol{e} \bmod q_{L-1}$

$\boldsymbol{b}_\xi \xleftarrow{\mathsf{u}} \mathbb{Z}^g_{q_{L-1}}$

$\boldsymbol{a}_\xi = -\boldsymbol{b}_\xi \odot \boldsymbol{s}_\xi + \boldsymbol{m} + \mathsf{eta\_to\_xi}(\boldsymbol{e'}, q_{L-1}) \bmod q_{L-1}$

return $ct = (\boldsymbol{a}_\xi, \boldsymbol{b}_\xi, f = 1, l = L - 1, \nu = \mathsf{B}^{\mathsf{FV}}_{\mathsf{clean}})$

This can be computed by $O(g \log g)$ operations of underlying mod q_{L-1} integers.

Decryption. Decryption algorithm removes the noise placed in upper digits by taking residue with respect to plaintext modulus t.

$\mathsf{Decrypt}^{\mathsf{DR-BGV}}(\mathsf{sk} = \boldsymbol{s}_\xi \in \mathbb{Z}^g, ct = (\boldsymbol{a}_\xi, \boldsymbol{b}_\xi, f, l, \nu), \mathsf{prm})$:

$\boldsymbol{h}_\xi = \boldsymbol{a}_\xi + \boldsymbol{b}_\xi \odot \boldsymbol{s}_\xi \bmod q_l$

$\boldsymbol{h} = \mathsf{xi\_to\_eta}(\boldsymbol{h}_\xi, q_l)$

$\boldsymbol{z} = f^{-1}\boldsymbol{h} \bmod t$

$\boldsymbol{m} = \mathsf{eta\_to\_xi}(\boldsymbol{z}, t)$

return $\boldsymbol{m}$

This can be computed by $O(g \log g)$ operations of underlying mod q_l integers.

Rescale. Rescale scales down a modulus of a given ciphertext from q_l to $q_{l'}$ ($q_l > q_{l'}$), to reduce its noise. Setting P be $q_l/q_{l'}$, the noise is scaled down by a factor of $1/P$, however an additional noise term $\mathsf{B}^{\mathsf{BGV}}_{\mathsf{scale}}$ appears.

$\mathsf{Rescale}(ct = (\boldsymbol{a}_\xi, \boldsymbol{b}_\xi, f, l, \nu), l')$:

$\boldsymbol{a} = \mathsf{xi\_to\_eta}(\boldsymbol{a}_\xi, q_l)$

$\boldsymbol{b} = \mathsf{xi\_to\_eta}(\boldsymbol{b}_\xi, q_l)$

Fix $\boldsymbol{\delta_a}$ s.t. $\boldsymbol{\delta_a} \equiv \boldsymbol{a} \pmod P$ and $\boldsymbol{\delta_a} \equiv 0 \pmod t$,

$\boldsymbol{\delta_b}$ s.t. $\boldsymbol{\delta_b} \equiv \boldsymbol{b} \pmod P$ and $\boldsymbol{\delta_b} \equiv 0 \pmod t$

$\boldsymbol{a}' = (\boldsymbol{a} - \boldsymbol{\delta_a})/P$, $\boldsymbol{b}' = (\boldsymbol{b} - \boldsymbol{\delta_b})/P$

$\boldsymbol{a}'_\xi = \mathsf{eta\_to\_xi}(\boldsymbol{a}', q_{l'})$

$\boldsymbol{b}'_\xi = \mathsf{eta\_to\_xi}(\boldsymbol{b}', q_{l'})$

$f' = f/P \bmod t$

$\nu' = \nu/P + \mathsf{B}^{\mathsf{BGV}}_{\mathsf{scale}}$

return $ct' = (\boldsymbol{a}'_\xi, \boldsymbol{b}'_\xi, f', l', \nu')$

This can be computed by $O(g \log g)$ operations of underlying mod q_l and mod $q_{l'}$ integers.

Addition. Assume the level of ct_2 is not greater than the level of ct_1.

$\mathsf{Add}^{\mathsf{DR-BGV}}(ct_1 = (\boldsymbol{a}_1, \boldsymbol{b}_1, f_1, l_1, \nu_1), ct_2 = (\boldsymbol{a}_2, \boldsymbol{b}_2, f_2, l_2, \nu_2), \mathsf{prm})$:

$ct'_2 = (\boldsymbol{a}'_2, \boldsymbol{b}'_2, f'_2, l_1, \nu'_2) = \mathsf{Rescale}(ct_{2\_msd}, l_1)$

$\boldsymbol{a} = f'_2\boldsymbol{a}_1 + f_1\boldsymbol{a}'_2 \bmod q_{l_1}$

$\boldsymbol{b} = f'_2\boldsymbol{b}_1 + f_1\boldsymbol{b}'_2 \bmod q_{l_1}$

$f = f_1 f_2'$

$\nu = \nu_1 + \nu_2'$

return $ct = (\boldsymbol{a}, \boldsymbol{b}, f, l_1, \nu)$

This can be computed by $O(g \log g)$ operations of underlying mod q_{l_1} and mod q_{l_2} integers.

Multiplication. Assume the level of ct_2 is not greater than the level of ct_1. At the last step in Mult, one level is consumed to reduce the incurred noise.

$\mathsf{Mult}^{\mathsf{DR-BGV}}(\mathsf{swk}, ct_1 = (\boldsymbol{a}_1, \boldsymbol{b}_1, f_1, l_1, \nu_1), ct_2 = (\boldsymbol{a}_2, \boldsymbol{b}_2, f_2, l_2, \nu_2), \mathsf{prm})$:

$ct_2' = (\boldsymbol{a}_2', \boldsymbol{b}_2', f_2', l_1, \nu_2') = \mathsf{Rescale}(ct_2, l_1)$

$\boldsymbol{\alpha} = \boldsymbol{a}_1 \odot \boldsymbol{a}_2' \bmod q_{l_1}$

$\boldsymbol{\beta} = \boldsymbol{a}_1 \odot \boldsymbol{b}_2' + \boldsymbol{a}_2' \odot \boldsymbol{b}_1 \bmod q_{l_1}$

$\boldsymbol{\gamma} = \boldsymbol{b}_1 \odot \boldsymbol{b}_2' \bmod q_{l_1}$

$f = f_1 f_2'$

$\nu = \mathsf{B}^{\mathsf{BGV}}_{\mathsf{direct\_mult}}(\nu_1, \nu_2')$

$(\boldsymbol{a}', \boldsymbol{b}', f, l_1, \nu'') = \mathsf{Linearize}^{\mathsf{DR-BGV}}(\mathsf{swk}, (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, f, l_1, \nu), \mathsf{prm})$

$ct = \mathsf{Rescale}((\boldsymbol{a}', \boldsymbol{b}', f, l_1, \nu''), l_1 - 1)$

return ct

Here, the key switching procedure is described below.

$\mathsf{Linearize}^{\mathsf{DR-BGV}}(\mathsf{swk}, ct = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, f, l, \nu), \mathsf{prm})$:

$(\boldsymbol{a}, \boldsymbol{b}, f, l, \nu') = \mathsf{KeySwitch}^{\mathsf{DR-BGV}}(\mathsf{swk}, (\boldsymbol{\gamma}, f, l), \mathsf{prm})$

$\boldsymbol{a}' = \boldsymbol{\alpha} + \boldsymbol{a} \bmod q_l$

$\boldsymbol{b}' = \boldsymbol{\beta} + \boldsymbol{b} \bmod q_l$

$\nu'' = \nu + \nu' + \mathsf{B}_{\mathsf{lin}}$

return $ct = (\boldsymbol{a}', \boldsymbol{b}', f, l, \nu'')$

In the last step of KeySwitch, the size of modulus is lowered from $q_l q_s$ to q_l to reduce noise in the key.

$\mathsf{KeySwitch}^{\mathsf{DR-BGV}}(\mathsf{swk} = (\{\boldsymbol{A}_j, \boldsymbol{B}_j\}_{j=0}^{l_w-1}, \nu), (\boldsymbol{x}, f, l), \mathsf{prm})$:

$\boldsymbol{d} = \mathsf{xi\_to\_eta}(\boldsymbol{x}, q_l)$ Decompose $\boldsymbol{d} = \Sigma_{j=0}^{l_w-1} \boldsymbol{d}_j w^j$

$\boldsymbol{d}'_j = \mathsf{eta\_to\_xi}(\boldsymbol{d}_j, q_l)$ for $j = 0 \cdots l_w - 1$

$\boldsymbol{A} = \Sigma_{j=0}^{l_w-1} \boldsymbol{A}_j \odot \boldsymbol{d}'_j \bmod q_l q_s$

$\boldsymbol{B} = \Sigma_{j=0}^{l_w-1} \boldsymbol{B}_j \odot \boldsymbol{d}'_j \bmod q_l q_s$

$(\boldsymbol{a}, \boldsymbol{b}, f, l, \nu') = \mathsf{Rescale}((\boldsymbol{A}, \boldsymbol{B}, f, l+s, \nu), l)$

return $(\boldsymbol{a}, \boldsymbol{b}, f, l, \nu')$

The total complexity of $\mathsf{Mult}^{\mathsf{DR-BGV}}$ is sum of $O(g \log g)$ operations with respect to the ciphertext modulus q_2 and $q_1 q_s$, and $O(l_w g \log g)$ operations with respect to q_1.

4.3.8 Tensorial Decomposition

Recall that the canonical embedding of the cyclotomic field $K = \mathbb{Q}(\zeta_m)$ is viewed as the tensor product of the canonical embedding of the prime-factored cyclotomic fields $K_l = \mathbb{Q}(\zeta_{m_l})$ where $m = \prod_{l=0}^{s} m_l$, as mentioned in Section 2.3.3. The relation of the equality 2.4 holds on the decomposition field $Z = K^{G_Z}$ and the prime-factored decomposition fields $Z_l = K_l^{G_Z}$.

Let $d = \mathrm{ord}_m^{\times}(p)$ be the multiplicative order of p mod m and $d_l = \mathrm{ord}_{m_l}^{\times}(p)$ be for m_l. We assume d_l's are pairwise coprime, then we have the CRT decomposition $\mathbb{Z}_m^*/\langle p \rangle \simeq \prod_{l=0}^{s} \mathbb{Z}_{m_l}^*/\langle p \rangle$. Using this CRT, we get the relation for the decomposition field,

$$\sigma_Z\left(\bigotimes_{l=0}^{s} a_l\right) = \bigotimes_{l=0}^{s} \sigma_Z^{(l)}(a_l)$$

where $a_l \in Z_l$.

Applying this relation to the η-bases vector $\boldsymbol{\eta}_l$ of the prime-factored decomposition rings R_{Zl} which is ring of integers of Z_l, we get

$$\sigma_Z\left(\bigotimes_{l=0}^{s} \boldsymbol{\eta_l}^T\right) = \bigotimes_{l=0}^{s} \sigma_Z^{(l)}(\boldsymbol{\eta_l}^T), \tag{4.6}$$

as same as the equation 2.5 for the cyclotomic ring bases.

In our schemes, we transform an element of the decomposition ring into plaintext slots by canonical embedding. We optimize this transformation using the tensorial decomposition technique. In the implementation, we split the index m into two primes s.t. $m = m_L m_R$. We generate two transformation matrices $\Omega_R \in \mathbb{Z}_q^{g_R \times g_R}$ for m_R and $\Omega_L \in \mathbb{Z}_q^{g_L \times g_L}$ for m_L in the same way as $\Omega_Z^{(q)} \in \mathbb{Z}_q^{g \times g}$ where $d_l = Ord_{m_l}(p), g_l = \phi(m_l)/d_l$ for $l = R$ and L. We use the tensor matrix $\Omega_R \bigotimes \Omega_L$ instead of the matrix $\Omega_Z^{(q)}$ in the basis transformation.

4.3.9 Bootstrap

The bootstrap function removes the noise in the ciphertext accumulated by the homomorphic operations, and enables the ciphertext to be multiplied or added unlimited times. The way of bootstrap is to evaluate the decryption circuit homomorphically to the ciphertext under the encryption of the secret key with respect to the ciphertext. Homomorphic computing of noise removing and basis conversion in the decryption circuit is very tough task, and this issue causes bottleneck to use fully homomorphic encryption.

We construct the bootstrap function on our homomorphic encryption schemes based on the decomposition ring, in which especially the homomorphic basis conversion become simple design because of the simple slot structure. We also construct homomorphic linear transformation for the tensor matrix used basis conversion of the bootstrap for our schemes optimized with the tensorial decomposition technique.

Parameters about modulus

To extract digits, the modulus of the original ciphertext is necessary to be converted to a power of the same base p to the base of plaintext modulus $t = p^r$, therefore, we convert the original ciphertext to MSD form and rescale its modulus from q to p^R. We take this new modulus p^R as small as it satisfies $t = p^r < p^R < q$ to lighten bootstrapping process. We define these modulus parameters as follows:

prm : public parameters for original ciphertext

pt_mod = $t(= p^r)$

ct_mod = q

$\mathsf{prm}_{\mathsf{in}}$: public parameters for inner ciphertext

pt_mod = $t(= p^r)$

ct_mod = p^R where $p^r < p^R < q$

$\mathsf{prm}_{\mathsf{out}}$: public parameters for outer ciphertext

pt_mod = p^R

ct_mod = Q (large enough)

Procedure Description

Key generation for Bootstrapping. Bootstrap key bsk is an encryption of the secret key sk under the large modulus secret key. The input $\boldsymbol{s}$ is η-vector of secret key $\boldsymbol{s}_\xi$.

BootstrapKeyGen($\boldsymbol{s}_\xi, \boldsymbol{s}$) :

$\boldsymbol{s}'_\xi$ = eta_to_xi($\boldsymbol{s}$) $\mod Q$

bsk = Encrypt(sk = $\boldsymbol{s}'_\xi, \boldsymbol{s}_\xi, \mathsf{prm}_{\mathsf{out}}$)

return bsk

Main procedure. The bootstrap process begins from converting to MSD and scaling down of the original ciphertext. The next step is to calculate $a+bs$ homomorphically which equals to $\lfloor \frac{q}{t} \rfloor m+e$ implicitly. Then clear lower digits and shift to right homomorphically (homRemoveNoise), as the result, only message m remains in the outer ciphertext.

In the following Bootstrap, ct denotes the ciphertext under prm, ct' denotes the ciphertext under $\mathsf{prm}_{\mathsf{in}}$, and CT denotes the ciphertext under $\mathsf{prm}_{\mathsf{out}}$.

CT_a and CT_b are dummy encryptions of the original ciphertext elements. Consider multiplication of a dummy ciphertext by a normal ciphertext, the secret key in the product does not become square of secret key, so we use a special multiplicative function Mult_without_linearization that omits linearization.

Bootstrap($ct = (\boldsymbol{a}_\xi, \boldsymbol{b}_\xi)$, bsk, swk_Q) :

ct_{msd} = ConvToMSD($ct, \mathsf{prm}_{\mathsf{in}}$)

$ct'_{msd} = (\boldsymbol{a}'_\xi, \boldsymbol{b}'_\xi)$ = Rescale($ct_{msd}, \mathsf{prm}_{\mathsf{in}}$.ct_mod)

$CT_a = (\boldsymbol{a}'_\xi \mod Q, 0)$

$CT_b = (\boldsymbol{b}'_\xi \mod Q, 0)$

CT_{bs} = Mult_without_linearization(swk_Q, CT_b, bsk, $\mathsf{prm}_{\mathsf{out}}$)

CT_{abs} = Add($CT_a, CT_{bs}, \mathsf{prm}_{\mathsf{out}}$)

CT_{abs_r} = hom_xi_to_eta($CT_{abs}, \mathsf{prm}_{\mathsf{out}}$)

CT_{m_r} = homRemoveNoise($CT_{abs\_r}, \mathsf{prm}_{\mathsf{out}}, \mathsf{prm}_{\mathsf{in}}, \mathsf{swk}_Q$)

CT_m = hom_eta_to_xi($CT_{m\_r}, \mathsf{prm}_{\mathsf{out}}$)

CT_{m_msd} = ConvToMSD($CT_m, \mathsf{prm}_{\mathsf{out}}$)

return ct_{m_msd}

Homomorphic noise remove. Some approaches are proposed for homomorphic noise remove, we select the method used in HElib by Halevi and Shoup [28]. The input ciphertext ct is supposed to be an encryption of message $\boldsymbol{m}$ which is extracted like $m_i = \Sigma_{j=0}^{e-1} a_j p^j$. This algorithm removes lower n digits.

homRemoveNoise(ct, prm_{ct}, prm_{pt}, p, swk):

$e = log_p(\text{prm}_{pt}.\text{ct\_mod})$

$n = e - log_p(\text{prm}_{pt}.\text{pt\_mod})$

$ct_z \leftarrow ct$

$w[0][0] = ct_z$

For $k = 0$ to $e - 1$

$ct_y \leftarrow ct_z$

For $j = 0$ to $\min(k\ ,\ n - 1)$

$ct_v = \text{Mult}(\text{swk}, w[j][k], w[j][k])$

$w[j][k+1] = ct_v$

If $k = e - 1$

$ct_d.\boldsymbol{a} = ct_v.\boldsymbol{a} * p^j$

$ct_d.\boldsymbol{b} = ct_v.\boldsymbol{b} * p^j$

$ct_{noise} = \text{Add}(ct_{noise}, ct_d)$

$ct'_y = \text{Sub}(ct_y, w[j][k+1])$

$ct_y = \text{div\_by\_p}(ct'_y)$

$w[k+1][k+1] = ct_y$

$ct_m = \text{Sub}(ct, ct_{noise})$

return homRightShift(ct_m, n)

Homomorphic linear transformation. Our decryption function converts the basis of the ciphertext before and after removing noise, those operations are $\Omega_Z^{(q)} \times \eta$-vector (hom_eta_to_xi) and $\Gamma_Z^{(q)} \times \xi$-vector (hom_xi_to_eta). The bootstrap procedure executes these linear transformation homomorphically to the given ciphertext. Halevi and Shoup propose a method to compute the homomorphic linear transformation in the study [27]. Their method is that, for each dimension

i, multiply an encryption of i-th diagonal vector of the matrix by an encryption of i times rotated data vector (η-vector or ξ-vector) homomorphically, and add the resulting product ciphertexts of all indexes homomorphically. We adopt this method to our schemes. The function $\mathsf{diagonal}(M, i)$ in the homLinearTrans outputs i-th diagonal vector of the matrix M.

$\mathsf{homLinearTrans}(\mathsf{swk}, \mathsf{rotk}, M \in \mathbb{Z}^{n\times n}, ct, \mathsf{prm})$:

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

$\boldsymbol{d}_i = \mathsf{diagonal}(M, i)$

$ct_{d\_i} = (\boldsymbol{d}_i \mod \mathsf{prm.ct\_mod}, 0)$ /*dummy encryption*/

$ct_{rot\_i} = \mathsf{homRotate}(\mathsf{rotk}_i, ct, i, \mathsf{prm})$

$ct_{prod} = \mathsf{Mult\_without\_linearization}(\mathsf{swk}, ct_{d\_i}, ct_{rot\_i})$

$ct_{sum} = \mathsf{Add}(ct_{sum}, ct_{prod})$

return ct_{sum}

Homomorphic rotation. In the below description, $\boldsymbol{v}^{(i)}$ denotes a vector rotated i times of an original vector $\boldsymbol{v}$. The homomorphic rotation function generates a ciphertext $ct' = (\boldsymbol{a}_\xi^{(i)}, \boldsymbol{b}_\xi^{(i)})$ from a given ciphertext $ct = (\boldsymbol{a}_\xi, \boldsymbol{b}_\xi)$ under the same secret key $\mathsf{sk} = \boldsymbol{s}_\xi$, which encrypts i times rotated message $\boldsymbol{m}^{(i)}$. For example BGV-type ciphertext, $ct = (\boldsymbol{a}_\xi, \boldsymbol{b}_\xi)$ satisfies

$$\boldsymbol{a}_\xi + \boldsymbol{b}_\xi \boldsymbol{s}_\xi \equiv \boldsymbol{m} + t\boldsymbol{e}_\xi \pmod{q}.$$

Rotating all vectors i times, the vectors satisfy

$$\boldsymbol{a}_\xi^{(i)} + \boldsymbol{b}_\xi^{(i)} \boldsymbol{s}_\xi^{(i)} \equiv \boldsymbol{m}^{(i)} + t\boldsymbol{e}_\xi^{(i)} \pmod{q}.$$

The rotated secret key $\boldsymbol{s}_\xi^{(i)}$ can be switched to the original secret key $\boldsymbol{s}_\xi$ by applying the key switching method, similar to key switching for homomorphic multiplication.

The RotateKeyGen generates the rotation key rotk, in which each key rotk_i is an encryption of i times rotated secret key $\boldsymbol{s}_\xi^{(i)}$.

$\mathsf{RotateKeyGen}(\boldsymbol{s}_\xi, \mathsf{sk} = \boldsymbol{s}_Q, \mathsf{prm}_{\mathsf{out}})$:

For $i = 1$ to $g - 1$

$\boldsymbol{s}_{\xi i} = \mathsf{rotate}(\boldsymbol{s}_\xi, i)$

$\boldsymbol{s}_i = \mathsf{xi\_to\_eta}(\boldsymbol{s}_{\xi i}, \mathsf{prm}_{\mathsf{out}})$

$\mathsf{rotk}_i = (\boldsymbol{A}_j, \boldsymbol{B}_j)_{j=0}^{l_w-1} = \mathsf{SwitchKeyGen}(\boldsymbol{s}_i, \boldsymbol{s}_Q, \mathsf{prm}_{\mathsf{out}})$

return $\mathsf{rotk} = (\mathsf{rotk}_i)_{i=1}^{g-1}$

The homRotate shifts $\boldsymbol{a}_\xi$ and $\boldsymbol{b}_\xi$ in the input ciphertext i times and switches the embraced rotated key $\boldsymbol{s}_\xi^{(i)}$ to the original key $\boldsymbol{s}_\xi$ using the rotation key rotk_i of i times rotation. The output is an encryption of the message rotated i times.

$\mathsf{homRotate}(\mathsf{rotk}_i = (\boldsymbol{A}_j, \boldsymbol{B}_j)_{j=0}^{l_w-1}), ct = (\boldsymbol{a}_\xi, \boldsymbol{b}_\xi), i, \mathsf{prm}) :$

$\boldsymbol{a}_\xi^{(i)} = \mathsf{rotate}(\boldsymbol{a}_\xi, i)$

$\boldsymbol{b}_\xi^{(i)} = \mathsf{rotate}(\boldsymbol{b}_\xi, i)$

$(\boldsymbol{a}^{(i)}, \boldsymbol{b}^{(i)}) = \mathsf{KeySwitch}(\mathsf{rotk}_i, \boldsymbol{b}_\xi^{(i)}, \mathsf{prm})$

$\boldsymbol{\alpha}_\xi^{(i)} = \boldsymbol{a}_\xi^{(i)} + \mathsf{eta\_to\_xi}(\boldsymbol{a}^{(i)}, \mathsf{prm})$

$\boldsymbol{\beta}_\xi^{(i)} = \mathsf{eta\_to\_xi}(\boldsymbol{b}^{(i)}, \mathsf{prm})$

return $ct = (\boldsymbol{\alpha}_\xi^{(i)}, \boldsymbol{\beta}_\xi^{(i)})$

Noise analysis of homomorphic rotation. Here we observe how the norm changes when ξ-vector is rotated. Let x be an element of R_Z. ξ-vector $\boldsymbol{y}$ of x corresponds to the canonical embedding $\sigma(x) = (\rho_{t_0}(x), \rho_{t_1}(x), \cdots, \rho_{t_{g-1}}(x))$. When $\boldsymbol{y}$ is rotated i times, the canonical embedding becomes $\sigma(x)^{(i)} = (\rho_{t_i}(x), \rho_{t_{1+i}}(x), \cdots, \rho_{t_{g-1+i}}(x))$, and equals to $(\rho_{t_0}(\rho_{t_i}(x)), \rho_{t_1}(\rho_{t_i}(x)), \cdots, \rho_{t_{g-1}}(\rho_{t_i}(x)))$. This is the canonical embedding of $\rho_{t_i}(x)$, that is $\sigma(x)^{(i)} = \sigma(\rho_{t_i}(x))$. Since ρ_{t_i}'s are Galois maps and the power basis of R_Z is composed of short and nearly orthogonal vectors to each other, the canonical embedding norm of rotated ξ-vector is almost as same as that of original ξ-vector, i.e. $\left\|\rho_{t_i}(x)\right\|_\infty \approx \left\|x\right\|_\infty$.

From this observation, we see that rotating ξ-vectors increases less noise.

Homomorphic linear transformation in tensorial decomposition case. When the scheme is optimized by the tensorial decomposition technique s.t. $m = \prod_{l=0}^{s} m_l$, the transformation matrix with respect to m is the tensor matrix of the transformation matrices with respect to m_l. Note that these matrices with respect to m_l are circulant matrices. The homomorphic linear transformation is realized by multiplying a ciphertext of diagonal vector of the tensor matrix and a ciphertext rotated partially along with the form of the tensor matrix.

We describe below the case of m factors m_R and m_L s.t. the transform matrix is $\Omega_Z^{(q)} = \Omega_L \bigotimes \Omega_R$.

The RotateKeyGen generates the rotation keys for Ω_R and Ω_L, those are encryptions of rotated keys of the original secret key $\boldsymbol{s}_\xi$ under the secret key sk. The keys in the rotk_R are rotated one by one in units of g_{R}, we call this baby-step rotation. And the keys in the rotk_L are jumped g_{R} steps in units of whole dimension $g_{\mathrm{R}} \cdot g_{\mathrm{L}}$, we call this giant-step rotation.

$\mathsf{RotateKeyGen}(\boldsymbol{s}_\xi, \mathsf{sk} = \boldsymbol{s}_Q, \mathsf{prm}_{\mathsf{out}})$:

$\mathsf{rotk}_R = \mathsf{RotateKeyGenUnit}(\boldsymbol{s}_\xi, \mathsf{sk} = \boldsymbol{s}_Q, \mathsf{jump} = 1, \mathsf{unit} = g_{\mathrm{R}}, \mathsf{prm}_{\mathsf{out}})$

$\mathsf{rotk}_L = \mathsf{RotateKeyGenUnit}(\boldsymbol{s}_\xi, \mathsf{sk} = \boldsymbol{s}_Q, \mathsf{jump} = g_{\mathrm{R}}, \mathsf{unit} = g_{\mathrm{R}} \cdot g_{\mathrm{L}}, \mathsf{prm}_{\mathsf{out}})$

return $(\mathsf{rotk}_R, \mathsf{rotk}_L)$

$\mathsf{RotateKeyGenUnit}(\boldsymbol{s}_\xi, \mathsf{sk} = \boldsymbol{s}_Q, \mathsf{jump}, \mathsf{unit}, \mathsf{prm}_{\mathsf{out}})$:

For $i = 0$ to $\mathsf{unit} - 1$

$\boldsymbol{s}_{\xi i} = \mathsf{rotateUnit}(\boldsymbol{s}_\xi, \mathsf{jump} \cdot i, \mathsf{unit})$

$\boldsymbol{s}_i = \mathsf{xi\_to\_eta}(\boldsymbol{s}_{\xi i}, \mathsf{prm}_{\mathsf{out}})$

$\mathsf{rotk}_i = (\boldsymbol{A}_j, \boldsymbol{B}_j)_{j=0}^{l_w - 1} = \mathsf{SwitchKeyGen}(\boldsymbol{s}_i, \boldsymbol{s}_Q, \mathsf{prm}_{\mathsf{out}})$

return $\mathsf{rotk} = (\mathsf{rotk}_i)_{i=0}^{\mathsf{unit}-1}$

$\mathsf{rotateUnit}(\boldsymbol{s}_\xi, k, \mathsf{unit})$:

For $j = 0$ to $\boldsymbol{s}_\xi.\mathsf{length} - 1$

If $j \bmod \mathsf{unit} < k$ Then $\boldsymbol{s}'_\xi[j - k + \mathsf{unit}] = \boldsymbol{s}_\xi[j]$

Else $\boldsymbol{s}'_\xi[j - k] = \boldsymbol{s}_\xi[j]$

return $\boldsymbol{s}'_\xi$

Combining baby-step and giant-step rotation, we can generate a ciphertext for linear transformation of the tensor matrix $\Omega_Z^{(q)}$. First, we perform baby-step rotation for Ω_R to the input ciphertext ct, and next we perform giant-step rotation for Ω_L to the resulting ciphertext ct_R from baby-step rotation.

$\mathsf{homLinearTrans}(\mathsf{swk}, \mathsf{rotk}, \Omega_Z^{(q)} \in \mathbb{Z}^{n \times n}, ct, \mathsf{prm})$:

$ct_R = \mathsf{homLinearTransRight}(\mathsf{swk}, \mathsf{rotk}_R, \Omega_R \in \mathbb{Z}^{g_{\mathrm{R}} \times g_{\mathrm{R}}}, ct, \mathsf{prm})$

$ct' = \mathsf{homLinearTransLeft}(\mathsf{swk}, \mathsf{rotk}_L, \Omega_L \in \mathbb{Z}^{g_{\mathrm{L}} \times g_{\mathrm{L}}}, ct_R, \mathsf{prm})$

return ct'

homLinearTransRight(swk, $\mathsf{rotk}_R, \Omega_R \in \mathbb{Z}^{g_{\mathrm{R}} \times g_{\mathrm{R}}}, ct$, prm):

For $i = 0$ to $g_{\mathrm{R}} - 1$

For $j = 0$ to $g_{\mathrm{L}} - 1$

For $k = 0$ to $g_{\mathrm{R}} - 1$

$\boldsymbol{d}_i = \boldsymbol{d}_i \parallel \mathsf{diagonal}(\Omega_R, i)[k]$

$ct_{d\_i} = (\boldsymbol{d}_i \mod \mathsf{prm.ct\_mod}, 0)$ /*dummy encryption*/

$ct_{rot\_i} = \mathsf{homRotateUnit}(\mathsf{rotk}_R[i], ct, i, g_{\mathrm{R}}, \mathsf{prm})$

$ct_{prod} = \mathsf{Mult\_without\_linearization}(\mathsf{swk}, ct_{d\_i}, ct_{rot\_i})$

$ct_{sum} = \mathsf{Add}(ct_{sum}, ct_{prod})$

return ct_{sum}

homLinearTransLeft(swk, $\mathsf{rotk}_L, \Omega_L \in \mathbb{Z}^{g_{\mathrm{L}} \times g_{\mathrm{L}}}, ct_R$, prm):

For $i = 0$ to $g_{\mathrm{L}} - 1$

For $k = 0$ to $g_{\mathrm{L}} - 1$

For $j = 0$ to $g_{\mathrm{R}} - 1$

$\boldsymbol{d}_i = \boldsymbol{d}_i \parallel \mathsf{diagonal}(\Omega_L, i)[k]$

$ct_{d\_i} = (\boldsymbol{d}_i \mod \mathsf{prm.ct\_mod}, 0)$ /*dummy encryption*/

$ct_{rot\_i} = \mathsf{homRotateUnit}(\mathsf{rotk}_L[i], ct_R, g_{\mathrm{R}} \cdot i, g_{\mathrm{R}} \cdot g_{\mathrm{L}}, \mathsf{prm})$

$ct_{prod} = \mathsf{Mult\_without\_linearization}(\mathsf{swk}, ct_{d\_i}, ct_{rot\_i})$

$ct_{sum} = \mathsf{Add}(ct_{sum}, ct_{prod})$

return ct_{sum}

homRotateUnit($\mathsf{rotk}_i = (\boldsymbol{A}_j, \boldsymbol{B}_j)_{j=0}^{l_w-1}), ct = (\boldsymbol{a}_\xi, \boldsymbol{b}_\xi), i$, unit, prm) :

$\boldsymbol{a}_\xi^{(i)} = \mathsf{rotateUnit}(\boldsymbol{a}_\xi, i, \mathsf{unit})$

$\boldsymbol{b}_\xi^{(i)} = \mathsf{rotateUnit}(\boldsymbol{b}_\xi, i, \mathsf{unit})$

$(\boldsymbol{a}^{(i)}, \boldsymbol{b}^{(i)}) = \mathsf{KeySwitch}(\mathsf{rotk}_i, \boldsymbol{b}_\xi^{(i)}, \mathsf{prm})$

$\boldsymbol{\alpha}_\xi^{(i)} = \boldsymbol{a}_\xi^{(i)} + \mathsf{eta\_to\_xi}(\boldsymbol{a}^{(i)}, \mathsf{prm})$

$\boldsymbol{\beta}_\xi^{(i)} = \mathsf{eta\_to\_xi}(\boldsymbol{b}^{(i)}, \mathsf{prm})$

return $ct = (\boldsymbol{\alpha}_\xi^{(i)}, \boldsymbol{\beta}_\xi^{(i)})$

Related works. For high dimensional ring, the homomorphic linear transformation algorithm rotates the data vector for many indexes using many switching keys of the indexes, it takes much time and consumes large key space. HElib improves the homomorphic linear transformation algorithm applying some strategies in the work [27], [28] and [29]. Here we introduce their strategies focus on the rotation of the data vector.

Let D be the dimension. As observed in the work [27], we can multiply an $D \times D$ matrix M by a D-dimensional column vector $\boldsymbol{v}$ by computing

$$M \times \boldsymbol{v} = \sum_{i \in [D]} \mathsf{diagonal}(M, i) \times \mathsf{rotate}(\boldsymbol{v}, i).$$

For computing every $\mathsf{rotate}(\boldsymbol{v}, i)$, the work [28] applies the baby-step/giant-step strategy. They divide the i-steps jump to one giant jump with gk-steps ($0 \leq k < h$) and one baby jump with j-steps ($0 \leq j < g$),

$$\mathsf{rotate}(\boldsymbol{v}, i) = \mathsf{rotate}(\boldsymbol{v}, j) \circ \mathsf{rotate}(\boldsymbol{v}, gk)$$

where $i = gk + j, h = \lceil D/g \rceil$. For all rotations ($0 \leq i < D$), h keys for giant-step rotation and g keys for baby-step rotation are required. In [28], they set $g = \lceil \sqrt{D} \rceil$, then $h = O(\sqrt{D})$ and $g = O(\sqrt{D})$, and the total key size is $O(\sqrt{D})$. They reduce the number of the rotation keys and also the number of the rotation operations from $O(D)$ to $O(\sqrt{D})$.

Next in [29], they study reducing the number of homomorphic addition. They change from adding rotations of all indexes to rotating sum of baby-step rotations with giant-step operation. Let the constant $c_i = \mathsf{diagonal}(M, i)$ in short.

$$M \times \boldsymbol{v} = \sum_{i \in [D]} c_i \times \mathsf{rotate}(\boldsymbol{v}, i) = \sum_{k \in [h]} \mathsf{rotate}\left(\sum_{j \in [g]} c' \times \mathsf{rotate}(\boldsymbol{v}, j), gk \right), \tag{4.7}$$

where $c' = \mathsf{rotate}(c_{j+gk}, -gk)$. The number of the homomorphic additions reduce from $O(D)$ to $O(\sqrt{D})$, and the number of the homomorphic rotation is also the same.

Here we observe the case when the matrix M is a tensor matrix. For example when m factors m_R and m_L s.t. $\Omega_Z^{(q)} = \Omega_L \bigotimes \Omega_R$ and the dimension of the Ω_R is g_{R}, the baby-step rotation in 4.7 ($\mathsf{rotate}(\boldsymbol{v}, j)$) rotates every g_{R} elements in the vector $\boldsymbol{v}$ partially, whereas rotates the vector in whole in the case of the normal matrix.

It is direct to see that:

Theorem 2 *The decomposition ring homomorphic encryption schemes DR-FV and DR-BGV are indistinguishably secure under the chosen plaintext attack if the* R-DLWE$_{q,\chi_{key},\chi_{err}}$ *problem on the decomposition ring R_Z is hard.*

For correctness we have the following theorem. (The proof is in 4.5)

Theorem 3 *The decomposition ring homomorphic encryption schemes DR-FV and DR-BGV will be fully homomorphic under circular security assumption (i.e., an encryption of secret key $\boldsymbol{s}$ does not leak any information about $\boldsymbol{s}$) by taking ciphertext modulus $q = O(\lambda^{\log \lambda})$ for DR-FV , and $p_i = \Omega(\sqrt{\lambda})$ $(i = 1, \ldots, L = \Omega(\log \lambda))$ and $q_s = \Omega(\sqrt{\lambda})$ for DR-BGV.*

4.4 Norms on the decomposition ring

Let $Z = Q(R_Z)$ be the quotient field of the decomposition ring R_Z. Norms of $a \in Z$ are defined by

$$\|a\|_2 \stackrel{\text{def}}{=} \|\sigma_Z(a)\|_2, \ \|a\|_\infty \stackrel{\text{def}}{=} \|\sigma_Z(a)\|_\infty.$$

Lemma 10 *For any $a, b \in Z$, we have*

$$\|ab\|_\infty \le \|a\|_\infty \cdot \|b\|_\infty.$$

Proof. $\|ab\|_\infty = \|\sigma_Z(ab)\|_\infty = \|\sigma_Z(a) \odot \sigma_Z(b)\|_\infty \le \|\sigma_Z(a)\|_\infty \cdot \|\sigma_Z(b)\|_\infty = \|a\|_\infty \cdot \|b\|_\infty.$ □

In the following, $\boldsymbol{a}$ means the η-vector of given $a = \boldsymbol{\eta}^T \cdot \boldsymbol{a} \in R_Z$.

Lemma 11 *(1) For any $a \in Z$, $\|a\|_2 \le \sqrt{\mathsf{m}}\|\boldsymbol{a}\|_2$.*

(2) For any $\boldsymbol{a} \in \mathbb{R}^g$, $\|\boldsymbol{a}\|_2 \le \|a\|_2$.

*(3) If $\boldsymbol{a} \in \mathbb{R}^g$ is far from being proportional to vector $\mathbf{1}$ (*far from constants *in short), we have $\|\boldsymbol{a}\|_2 \approx \frac{1}{\sqrt{\mathsf{m}}}\|a\|_2$.*

Proof. (1) By Lemma 7, $\sigma_Z(a) = \Omega_Z \boldsymbol{a}$ and by Lemma 5

$$\Omega_Z^* \Omega_Z = \mathsf{m}\mathsf{I}_\mathsf{g} - \mathsf{d}\mathbf{1} \cdot \mathbf{1}^\mathsf{T}.$$

The right-hand side matrix has eigenvalues $g-1$ times of m and 1 with corresponding eigenvectors $(1, -1, 0, \cdots, 0)$, $(1, 0, -1, 0, \cdots, 0)$, …, $(1, 0, \cdots, 0, -1)$, $(1, 1, \cdots, 1)$. So, the symmetric matrix

$\Omega_Z^*\Omega_Z$ can be diagonalized to $\mathrm{Diag}(\mathsf{m}, \cdots, \mathsf{m}, 1)$ by an orthogonal transformation, and we have $s_1(\Omega_Z) = \sqrt{\mathsf{m}}$. This means $\|a\|_2 \leq \sqrt{\mathsf{m}}\|\boldsymbol{a}\|_2$.
(2), (3) Conversely, $\boldsymbol{a} = (\Omega_Z)^{-1}\sigma_Z(a) = \Gamma_Z\sigma_Z(a)$. Similarly as above, the matrix $\Gamma_Z^*\Gamma_Z$ can be diagonalized to $\mathrm{Diag}(1/\mathsf{m}, \cdots, 1/\mathsf{m}, 1)$ by the orthogonal transformation. Hence, $s_1(\Gamma_Z) = 1$ and $\|\boldsymbol{a}\|_2 \leq \|a\|_2$. Since almost all of the eigenvalues of $\Gamma_Z^*\Gamma_Z$ are $1/m$, except 1 for eigenvector $(1, 1, \cdots, 1)$, if $\boldsymbol{a}$ is far from being proportional to the eigenvector $(1, 1, \cdots, 1)$, $\|\boldsymbol{a}\|_2 \approx \frac{1}{\sqrt{\mathsf{m}}}\|a\|_2$
□

Lemma 12 *(1) For any $a \in Z$, $\|a\|_\infty \leq \sqrt{\mathsf{mg}}\|\boldsymbol{a}\|_\infty$.*

(2) For any $\boldsymbol{a} \in \mathbb{R}^g$, $\|\boldsymbol{a}\|_\infty \leq \sqrt{g}\|a\|_\infty$.

(3) If a is far from constants, we have $\|\boldsymbol{a}\|_\infty \lessapprox \sqrt{g/\mathsf{m}}\|a\|_\infty$.

Proof. (1) By Lemma 11-(1), $\|a\|_\infty \leq \|a\|_2 \leq \sqrt{\mathsf{m}}\|\boldsymbol{a}\|_2 \leq \sqrt{\mathsf{mg}}\|\boldsymbol{a}\|_\infty$.
(2) By Lemma 11-(2), $\|\boldsymbol{a}\|_\infty \leq \|\boldsymbol{a}\|_2 \leq \|a\|_2 \leq \sqrt{g}\|a\|_\infty$.
(3) By Lemma 11-(3), $\|\boldsymbol{a}\|_\infty \leq \|\boldsymbol{a}\|_2 \approx \frac{1}{\sqrt{\mathsf{m}}}\|a\|_2 \leq \sqrt{g/\mathsf{m}}\|a\|_\infty$. □

Subgaussian elements. We call a random variable $a \in Z$ *subgaussian with parameter s* if corresponding random variable $\sigma_Z(a)$ on H_Z is subgaussian with parameter s.

Lemma 13 (Claim 2.1, Claim 2.4 [39]) *Let a_i be independent subgaussian random variables over Z with parameter s_i $(i = 1, 2)$. Then,*

1. *The sum $a_1 + a_2$ is subgaussian with parameter $\sqrt{s_1^2 + s_2^2}$.*
2. *For any a_2 fixed, the product $a_1 \cdot a_2$ is subgaussian with parameter $\|a_2\|_\infty s_1$.*

Lemma 14 *Let $\boldsymbol{a}$ be a subgaussian random variable over $\mathbb{R}^g$ of parameter s. Then, $a = \boldsymbol{\eta}^T \cdot \boldsymbol{a}$ is subgaussian over Z of parameter $\sqrt{\mathsf{m}}s$.*

Proof. By Lemma 7 $\sigma_Z(a) = \Omega_Z\boldsymbol{a}$. As seen in the proof of Lemma 11, $s_1(\Omega_Z) = \sqrt{\mathsf{m}}$. Hence, $\sigma_Z(a)$ is subgaussian of parameter $\sqrt{\mathsf{m}}s$. □

4.5 Correctness

We evaluate sizes of noises using their canonical embedding norms.

4.5.1 FV-type scheme DR-FV

Definition 9 (The noise term in FV-type scheme) *Let* $(a, b) \in R_Z^2$ *be a ciphertext pair designed for a message* $m \in R_Z$ *under a secret key* $s \in R_Z$*. When given* $((a, b), s, m)$*, the smallest noise term* $e \in R_Z$ *satisfying*

$$a + bs = \Delta m + e + kq$$

for some $k \in R_Z$ *called the inherent noise term of* (a, b) *for a message* m*.*

Noise bound for correctness. Set $y = a + bs = \Delta m + e \pmod q$. Then,

$$\frac{y}{\Delta} = \frac{t}{q}(\frac{q}{t}m + e) = m + \frac{t}{q}e \pmod q$$

If $\left\|\frac{t}{q}e\right\| < \frac{1}{2}$ then decryption works correctly. By 12-(3), to satisfy this inequality,

$$\sqrt{\frac{g}{\mathsf{m}}}\frac{t}{q}\left\|e\right\|_\infty < \frac{1}{2}$$

is required. We define $\mathsf{B}_{\mathsf{correct}}^{\mathsf{FV}} \stackrel{\text{def}}{=} \frac{q}{2t}\sqrt{\frac{\mathsf{m}}{g}}$. If the noise term e in a given ciphertext satisfies $\left\|e\right\|_\infty < \mathsf{B}_{\mathsf{correct}}^{\mathsf{FV}}$ then the ciphertext can be decrypted correctly.

Estimate of $\mathsf{B}_{\mathsf{clean}}^{\mathsf{FV}}$. Let e be a noise sampled in $\mathsf{Encrypt}^{\mathsf{DR-FV}}$. Then

$$\mathsf{B}_{\mathsf{clean}}^{\mathsf{FV}} = \left\|e\right\|_\infty = \sigma\sqrt{gd}.$$

Estimate of $\mathsf{B}_{\mathsf{direct\_mult}}^{\mathsf{FV}}$. Let (a', b') be the resulting ciphertext of $\mathsf{Mult}^{\mathsf{DR-FV}}(\mathsf{swk}, (a_1, b_1, \nu_1), (a_2, b_2, \nu_2))$, where swk is $((A_j, B_j)_{j=0}^{l_w-1}, \nu = \mathsf{B}_{\mathsf{lin}}^{\mathsf{FV}}) = \mathsf{SwitchKeyGen}^{\mathsf{DR-FV}}(s^2, \mathsf{sk} = s)$ satisfies $\Sigma_{j=0}^{l_w-1}(A_j + B_j s) = \Sigma_{j=0}^{l_w-1}(w^j s^2 + e_j) \bmod q$. Let α, β, γ be as in $\mathsf{Mult}^{\mathsf{DR-FV}}$ and $\epsilon_\alpha, \epsilon_\beta, \epsilon_\gamma$ be their rounding noises in

$(-\frac{1}{2}, \frac{1}{2}]$. Set $(d_0, \cdots, d_{l_w-1}) = \mathsf{WD}(\gamma)$. Then,

$$\begin{aligned}
\alpha + \beta s + \gamma s^2 &= \frac{1}{\Delta}(a_1 a_2 \mod q^2/t) + \epsilon_\alpha \\
&\quad + (\frac{1}{\Delta}(a_1 b_2 + a_2 b_1 \bmod q^2/t) + \epsilon_\beta)s \\
&\quad + (\frac{1}{\Delta}(b_1 b_2 \mod q^2/t) + \epsilon_\gamma)s^2 \\
&= \frac{1}{\Delta}(a_1 + b_1 s)(a_2 + b_2 s) + (\epsilon_\alpha + \epsilon_\beta s + \epsilon_\gamma s^2) \\
&= \frac{1}{\Delta}(\Delta m_1 + e_1)(\Delta m_2 + e_2) + (\epsilon_\alpha + \epsilon_\beta s + \epsilon_\gamma s^2) \\
&= \Delta m_1 m_2 + (m_1 e_2 + m_2 e_1) + e_1 e_2/\Delta \\
&\quad + (\epsilon_\alpha + \epsilon_\beta s + \epsilon_\gamma s^2)
\end{aligned}$$

Setting $e' = (m_1 e_2 + m_2 e_1) + e_1 e_2/\Delta + (\epsilon_\alpha + \epsilon_\beta s + \epsilon_\gamma s^2)$, we have $\mathsf{B}^{\mathsf{FV}}_{\mathsf{direct\_mult}}(\nu_1, \nu_2) = t\sqrt{3gd}(\nu_1 + \nu_2) + \frac{t}{q}\nu_1\nu_2 + \sqrt{3gd} + 8d\sqrt{\frac{gh}{3}} + 20hd\sqrt{\frac{gd}{3}}$, since

$$\begin{aligned}
\left\|e'\right\|_\infty &\le (\left\|m_1\right\|_\infty \nu_2 + \left\|m_2\right\|_\infty \nu_1) + \nu_1\nu_2/\Delta \\
&\quad + (\left\|\epsilon_\alpha\right\|_\infty + \left\|\epsilon_\beta s\right\|_\infty + \left\|\epsilon_\gamma s^2\right\|_\infty) \\
&\le t\sqrt{3gd}(\nu_1 + \nu_2) + \frac{t}{q}\nu_1\nu_2 \\
&\quad + \sqrt{3gd} + 8d\sqrt{\frac{gh}{3}} + 20hd\sqrt{\frac{gd}{3}}.
\end{aligned}$$

Estimate of $\mathsf{B}^{\mathsf{FV}}_{\mathsf{lin}}$. Suppose an input ciphertext (α, β, γ) of $\mathsf{Linearize}^{\mathsf{DR-FV}}$ satisfies $\alpha + \beta s + \gamma s^2 = \Delta m + e \bmod q$ and its noise bound is ν. Let (a', b') be the output ciphertext. Generate a switching key as $\mathsf{swk} = ((A_j, B_j)_{j=0}^{l_w-1}, \nu_{swk}) = \mathsf{SwitchKeyGen}^{\mathsf{DR-FV}}(s^2, \mathsf{sk} = s)$, which satisfies $\Sigma_{j=0}^{l_w-1}(A_j + B_j s) = \Sigma_{j=0}^{l_w-1}(w^j s^2 + e_j) \bmod q$. Let $(d_0, \cdots, d_{l_w-1}) = \mathsf{WD}(\gamma)$. Then,

$$\begin{aligned}
a' + b's &= (\alpha + \Sigma_{j=0}^{l_w-1} A_j d_j) + (\beta + \Sigma_{j=0}^{l_w-1} B_j d_j)s \bmod q \\
&= \alpha + \beta s + \Sigma_{j=0}^{l_w-1}(A_j + B_j s)d_j \bmod q \\
&= \alpha + \beta s + \Sigma_{j=0}^{l_w-1}(w^j s^2 + e_j)d_j \bmod q \\
&= \alpha + \beta s + \gamma s^2 + \Sigma_{j=0}^{l_w-1} e_j d_j \bmod q
\end{aligned}$$

So, it satisfies that

$$\left\|a' + b's\right\|_\infty \le \nu + \left\|\Sigma_{j=0}^{l_w-1} e_j d_j\right\|_\infty.$$

Therefore,

$\mathsf{B}^{\mathsf{FV}}_{\mathsf{lin}} = \left\|\Sigma_{j=0}^{l_w-1} e_j d_j\right\|_\infty = 16 l_w \sigma \sqrt{gd} w \sqrt{gd/12} = \frac{8}{\sqrt{3}} l_w \sigma g d w.$

Noise bound for $\mathsf{Mult}^{\mathsf{DR-FV}}$**.** The noise size of the output ciphertext of $\mathsf{Mult}^{\mathsf{DR-FV}}$ is

$$\begin{aligned}\nu_{mult}^{DR-FV} &= \mathsf{B}^{\mathsf{FV}}_{\mathsf{direct\_mult}} + \mathsf{B}^{\mathsf{FV}}_{\mathsf{lin}} \\ &= t\sqrt{3gd}(\nu_1+\nu_2) + \frac{t}{q}\nu_1\nu_2 \\ &+ \sqrt{3gd} + 8d\sqrt{\frac{gh}{3}} + 20hd\sqrt{\frac{gd}{3}} \\ &+ \frac{8}{\sqrt{3}} l_w \sigma g d w.\end{aligned}$$

Proof of Theorem 3 for DR-FV. By Lemma 4 of [5], we can implement $\mathsf{Decrypt}^{\mathsf{DR-BGV}}$ algorithm by some circuit of level $L_{dec} = O(\log \lambda)$.

Let ct' be the ciphertext after L_{dec} times multiplications and ν' be the bound of the canonical embedding noise of ct'. Then if $\nu' \le \mathsf{B}^{\mathsf{FV}}_{\mathsf{correct}}$, then the scheme can homomorphically evaluate its own $\mathsf{Decrypt}^{\mathsf{DR-FV}}$ circuit and will be fully homomorphic under circular security assumption.

Now we show $\nu' \le \mathsf{B}^{\mathsf{FV}}_{\mathsf{correct}}$ as follows: The size of each parameter is $g = O(\lambda), h = O(1), t = O(1)$, and we suppose $d = O(\log \lambda) = \tilde{O}(1)$. Two fresh ciphertexts have noises of size $\nu_1 = O(\sqrt{\lambda}), \nu_2 = O(\sqrt{\lambda})$. Then, by repeating the multiplication, the noise bound of the resulting ciphertext becomes as

$$\begin{aligned}\nu_{mult}^{DR-FV} &= \tilde{O}(\sqrt{\lambda})(\nu_1+\nu_2) + \frac{t}{q}\nu_1\nu_2 \\ &+ \tilde{O}(\sqrt{\lambda}) + \tilde{O}(\sqrt{\lambda}) + \tilde{O}(\sqrt{\lambda}) + \tilde{O}(\lambda)\end{aligned}$$

This shows that the increase ratio of ν_{mult}^{DR-FV} by one multiplication is $\tilde{O}(\sqrt{\lambda})$. It is because, the second term is $\frac{t}{q}\nu_1\nu_2 \le \frac{t}{q}\mathsf{B}^{\mathsf{FV}}_{\mathsf{correct}}\nu_i = \frac{1}{2}\sqrt{\frac{\mathsf{m}}{g}}\nu_i$ ($i = 1$ or 2), so the increase ratio of second term is $\tilde{O}(1)$ and it is smaller than that of the first term $\tilde{O}(\sqrt{\lambda})$.

Thus ν_{mult}^{DR-FV} increases by the factor of $\tilde{O}(\sqrt{\lambda})$ for each multiplication, and the factor is of $\log_2(\sqrt{\lambda}) = O(\log \lambda)$ bits. After L_{dec} times multiplication, the noise bound ν' is $O(\log(\lambda^{\log \lambda}))$ bit.

On the other hand, taking $q = O(\lambda^{\log \lambda})$ as assumption of the Theorem 3, $\mathsf{B}^{\mathsf{FV}}_{\mathsf{correct}} = \frac{q}{2t}\sqrt{\frac{\mathsf{m}}{g}} = O(\lambda^{\log \lambda})$ and it is $O(\log(\lambda^{\log \lambda}))$ bit. Therefore, we can take the modulus to satisfy $\nu' \le \mathsf{B}^{\mathsf{FV}}_{\mathsf{correct}}$.

□

4.5.2 BGV-type scheme DR-BGV

Definition 10 (The noise term in BGV-type scheme) *Let* $(a, b) \in R_Z^2$ *be a ciphertext pair designed for a message* $m \in R_Z$ *under a secret key* $s \in R_Z$*. When given* $((a, b), s, m, l)$ *where* l *is level, a noise* $e \in R_Z$ *is uniquely determined by the equation*

$$a + bs = m + te + kq_l$$

for some $k \in R_Z$*. Note that* $m + te$ *is not necessarily lower than* q_l*. We define the value* $m + te$ *as the noise term of* $((a, b), s, m, l)$*.*

Noise bound for correctness. Let $y = m + te$. If $\|y\| < \frac{q_l}{2}$ then decryption works correctly. By 12-(3), to satisfy this inequality,

$$\sqrt{\frac{g}{\mathsf{m}}}\|y\|_\infty < \frac{q_l}{2}$$

is required. We define $\mathsf{B}^{\mathsf{BGV}}_{\mathsf{correct}} \stackrel{\text{def}}{=} \frac{q_l}{2}\sqrt{\frac{\mathsf{m}}{g}}$. If the inherent noise y in a given level-l ciphertext satisfies $\|y\|_\infty < \mathsf{B}^{\mathsf{BGV}}_{\mathsf{correct}}$ then the ciphertext can be decrypted correctly.

Estimate of $\mathsf{B}^{\mathsf{BGV}}_{\mathsf{clean}}$. For a fresh ciphertext, the upper bound of its inherent noise is $\mathsf{B}^{\mathsf{BGV}}_{\mathsf{clean}} = t\sqrt{gd}(\sqrt{3} + 6\sigma)$ since

$$\begin{aligned}\|m + te\|_\infty &\le \|m\|_\infty + t\|e\|_\infty \\ &\le t\sqrt{3gd} + t6\sigma\sqrt{gd} = t\sqrt{gd}(\sqrt{3} + 6\sigma).\end{aligned}$$

Estimate of $\mathsf{B}^{\mathsf{BGV}}_{\mathsf{scale}}$. Let a scaled ciphertext from q_l to $q_{l'}$ be $(a', b', f, l', \nu') = \mathsf{Rescale}((a, b, f, l, \nu), l')$, and write its inherent noise as y. Set $P = \frac{q_l}{q_{l'}}$ and fix δ_a and δ_b s.t. $\delta_a \equiv a \pmod P$ and $\delta_a \equiv 0 \pmod t$, $\delta_b \equiv b \pmod P$ and $\delta_b \equiv 0 \pmod t$. Then, we have

$$\begin{aligned}\|a' + b's\|_\infty &\le \frac{1}{P}\|a + bs\|_\infty + \frac{1}{P}\|\delta_a + \delta_b s\|_\infty \\ &\le \frac{\nu}{P} + \frac{1}{P}(6\sqrt{P^2gd/12} + 16\sqrt{P^2gd/12}\sqrt{hd}) \\ &\le \frac{\nu}{P} + t(\sqrt{3gd} + 8d\sqrt{gh/3}).\end{aligned}$$

Thus, $\mathsf{B}^{\mathsf{BGV}}_{\mathsf{scale}} = t(\sqrt{3gd} + 8d\sqrt{gh/3})$.

Estimate of $\mathsf{B}^{\mathsf{BGV}}_{\mathsf{direct\_mult}}$. Let (a_1, b_1, f_1, ν_1) and (a_2, b_2, f_2, ν_2) be input ciphertexts of $\mathsf{Mult}^{\mathsf{DR-BGV}}$ and compute α, β, γ according to $\mathsf{Mult}^{\mathsf{DR-BGV}}$. Then,

$$\begin{aligned}\alpha + \beta s + \gamma s^2 &= a_1 a_2 + (a_1 b_2 + a_2 b_1)s + b_1 b_2 s^2 \bmod q_l \\ &= (a_1 + b_1 s)(a_2 + b_2 s) \bmod q_l\end{aligned}$$

This means that $\mathsf{B}_{\mathsf{direct\_mult}}(\nu_1, \nu_2) = \nu_1 \nu_2$.

Estimate of $\mathsf{B}^{\mathsf{BGV}}_{\mathsf{lin}}$. Suppose an input ciphertext (α, β, γ) of $\mathsf{Linearize}^{\mathsf{DR-BGV}}$ satisfies $\alpha + \beta s + \gamma s^2 \equiv m + te \pmod{q_l}$ and its noise is bound by ν. Let (a', b') be the output ciphertext of $\mathsf{Linearize}^{\mathsf{DR-BGV}}$ with input (α, β, γ). Generate a switching key $\mathsf{swk} = ((A_j, B_j)_{j=0}^{l_w-1}, \nu_{swk}) = \mathsf{SwitchKeyGen}^{\mathsf{DR-BGV}}(s^2, \mathsf{sk} = s)$, which satisfies $\Sigma_{j=0}^{l_w-1}(A_j + B_j s) = \Sigma_{j=0}^{l_w-1}(q_s w^j s^2 + te_j) \bmod q_{L-1} q_s$. Let $(d_0, \cdots, d_{l_w-1}) = \mathsf{WD}(\gamma)$ and $A = \Sigma_{j=0}^{l_w-1} A_j d_j \bmod q_l q_s$, $B = \Sigma_{j=0}^{l_w-1}$ and $B_j d_j \bmod q_l q_s$. Then,

$$\begin{aligned}A + Bs &= \Sigma_{j=0}^{l_w-1}(A_j + B_j s)d_j \bmod q_l q_s \\ &= \Sigma_{j=0}^{l_w-1}(q_s w^j s^2 + te_j)d_j \bmod q_l q_s \\ &= q_s \gamma s^2 + t\Sigma_{j=0}^{l_w-1} e_j d_j \bmod q_l q_s\end{aligned}$$

After scaling from $q_l q_s$ to q_l, i.e. $(a, b, f, l, \nu') = \mathsf{Rescale}((A, B, f, l + s, \nu), l)$, it satisfies that

$$a + bs = \gamma s^2 + \frac{t}{q_s}\Sigma_{j=0}^{l_w-1} e_j d_j + e_k \bmod q_l.$$

where e_k is a rounding noise added by $\mathsf{Rescale}$ satisfying $\left\|e_k\right\|_\infty < \mathsf{B}^{\mathsf{BGV}}_{\mathsf{scale}}$. Now we see that

$$\begin{aligned}a' + b's &= \alpha + a + (\beta + b)s \bmod q_l \\ &= \alpha + \beta s + a + bs \bmod q_l \\ &= \alpha + \beta s + \gamma s^2 + \frac{t}{q_s}\Sigma_{j=0}^{l_w-1} e_j d_j + e_k \bmod q_l\end{aligned}$$

Then,

$$\left\|a' + b's\right\|_\infty \le \nu + \frac{t}{q_s}\left\|\Sigma_{j=0}^{l_w-1} e_j d_j\right\|_\infty + \left\|e_k\right\|_\infty,$$

where

$$\begin{aligned}\frac{t}{q_s}\left\|\Sigma_{j=0}^{l_w-1} e_j d_j\right\|_\infty &\le \frac{t}{q_s} 16 l_w \sigma \sqrt{gd} w \sqrt{gd/12} \\ &\le 8 t l_w \sigma g d w / \sqrt{3} q_s.\end{aligned}$$

Thus, we have $\mathsf{B}_{\mathsf{lin}} = 8tl_w\sigma gdw/\sqrt{3}q_s$.

Noise bound for $\mathsf{Mult}^{\mathsf{DR-BGV}}$. In $\mathsf{Mult}^{\mathsf{DR-BGV}}$, the noise bound of output ciphertext of $\mathsf{Linearize}^{\mathsf{DR-BGV}}$ is $\nu'' = \nu_1\nu_2' + \mathsf{B}_{\mathsf{lin}}^{\mathsf{BGV}} + \mathsf{B}_{\mathsf{scale}}^{\mathsf{BGV}}$, where ν_2' is a noise bound of ct_2 rescaled from q_{l_2} to q_{l_1}. After linearization, the noise is reduced by $\mathsf{Rescale}((a', b', f, l_1, \nu''), l_1 - 1)$. Thus, the noise bound of one multiplication is as follows:

$$\begin{aligned}\nu_{mult}^{DR-BGV} &= \big(\nu_1(\frac{q_{l_1}}{q_{l_2}}\nu_2 + t(\sqrt{3gd} + 8d\sqrt{gh/3})) \\ &\quad + \frac{8tl_w\sigma gdw}{\sqrt{3}q_s} + t(\sqrt{3gd} + 8d\sqrt{gh/3})\big)\frac{q_{l_1-1}}{q_{l_1}} \\ &\quad + t(\sqrt{3gd} + 8d\sqrt{gh/3}).\end{aligned}$$

Proof of Theorem 3 for DR-BGV. By Lemma 4 of [5], we can implement $\mathsf{Decrypt}^{\mathsf{DR-BGV}}$ algorithm by some circuit of level $L_{dec} = O(\log\lambda)$.

Let ct' be the ciphertext after L_{dec} times multiplications, L be the maximum level in the system parameter, $l' = L - L_{dec}$ be the level of ct' and ν' be the bound of the canonical embedding noise of ct'. Then if $\nu' \le \mathsf{B}_{\mathsf{correct}}^{\mathsf{BGV}}(l')$, then the scheme can homomorphically evaluate its own $\mathsf{Decrypt}^{\mathsf{DR-BGV}}$ circuit and will be fully homomorphic under circular security assumption.

Now we show $\nu' \le \mathsf{B}_{\mathsf{correct}}^{\mathsf{BGV}}(l')$ as follows:

The size of each parameter is $g = O(\lambda), h = O(1), t = O(1)$, and we suppose $d = O(\log\lambda) = \tilde{O}(1)$, then the size of noise of a fresh ciphertext is $3.2\sqrt{gd} = \tilde{O}(\sqrt{\lambda})$.

From assumption of Theorem 3, $p_i = \Omega(\sqrt{\lambda})$ (i.e. $\frac{q_{l-1}}{q_l} = \frac{1}{\Omega(\sqrt{\lambda})}$) and $q_s = \Omega(\sqrt{\lambda})$. Two fresh ciphertexts have noises of size $\nu_1 = \tilde{O}(\sqrt{\lambda}), \nu_2 = \tilde{O}(\sqrt{\lambda})$. Then, by repeating the multiplication, the noise bound of the resulting ciphertext becomes as

$$\begin{aligned}\nu_{mult}^{DR-BGV} &= \big(\tilde{O}(\sqrt{\lambda})(\frac{q_{l_1}}{q_{l_2}}\tilde{O}(\sqrt{\lambda}) + \tilde{O}(\sqrt{\lambda})) + \frac{\tilde{O}(\lambda)}{\Omega(\sqrt{\lambda})} \\ &\quad + \tilde{O}(\sqrt{\lambda})\big)\frac{1}{\Omega(\sqrt{\lambda})} + \tilde{O}(\sqrt{\lambda}) \\ &= \tilde{O}(\sqrt{\lambda}).\end{aligned}$$

This shows that after multiple multiplications the noise bound of result ciphertext always keeps $\tilde{O}(\sqrt{\lambda})$.

We denote the bound for correctness of level-l ciphertext $\mathsf{B}_{\mathsf{correct}}^{\mathsf{BGV}}(l)$ $(= \frac{q_l}{2}\sqrt{\frac{\mathsf{m}}{g}})$. Since $\sqrt{\frac{\mathsf{m}}{g}} =$

$\tilde{O}(1)$ and $q_0 = O(\sqrt{\lambda})$, $\mathsf{B}^{\mathsf{BGV}}_{\mathsf{correct}}(0) = \tilde{O}(\sqrt{\lambda})$. Thus, $\nu^{DR-BGV}_{mult} = \tilde{O}(\sqrt{\lambda}) < \tilde{O}(\sqrt{\lambda}) = \mathsf{B}^{\mathsf{BGV}}_{\mathsf{correct}}(0) \leq \mathsf{B}^{\mathsf{BGV}}_{\mathsf{correct}}(l)$ for any level l. Therefore $\nu' \leq \mathsf{B}^{\mathsf{BGV}}_{\mathsf{correct}}(l')$. □

4.5.3 Efficiency

Here, we compare efficiency of DR-FV (or DR-BGV) to the conventional HE scheme on the cyclotomic ring (CR-HE for short). In CR-HE, the ring dimension is $n = \phi(\mathsf{m})$, the number of slots is $g = n/d$ and the dimension of each slot is d. So, one can encrypt g integer plaintexts into a single ciphertext of n $(= gd)$ dimension. On the contrary, in DR-FV (or DR-BGV), the ring dimension and the number of slots are both g and the dimension of each slot is 1. One can encrypt g integer plaintexts into a single ciphertext of the same dimension g. Thus, on the same level of security (i.e. same dimension), DR-FV (or DR-BGV) can handle d times as many plaintexts as CR-HE in a single ciphertext. This means that DR-FV (or DR-BGV) achieves more faster and compact HE than conventional CR-HE for integer plaintexts. More concrete benchmark results are given in Section 4.6.

4.6 Benchmark Results

Results of decomposition ring homomorphic encryption scheme. To verify the effect of our slot structure on the decomposition ring, we implemented our two decomposition ring homomorphic encryption scheme DR-FV and DR-BGV, using the C++ language and performed several experiments using different parameters, comparing efficiency of our implementation of DR-FV, DR-BGV and the homomorphic encryption library HElib by Halevi and Shoup [27], which is based on the BGV scheme over ordinal cyclotomic rings [6]. SEAL [37] is a homomorphic encryption library of FV-type. Recall that our target plaintext space is a power of small prime since we think many applications will use such plaintext modulus (e.g. 2^l), however in the CRT batching of SEAL, the plaintext modulus t is required to be $t \equiv 1 \pmod{\mathsf{m}}$ and cannot be a power of small prime. For this reason, we do not compare our schemes with SEAL.

For notation of parameters, see Section 4.3.3. As common parameters, we choose four values of prime m so that the m-th cyclotomic ring R will have as many numbers of plaintext slots (i.e., large g and small d values) as possible. The plaintext modulus $t = 2^l$ is fixed as $l = 8$. The noise parameter $s_{err} = \sqrt{2\pi}\sigma_{err}$ is fixed as $\sigma_{err} = 3.2$. The ciphertext modulus q of bit-length r

is chosen as small as possible so that it enables homomorphic evaluation of exponentiation by 2^8 (i.e., $\mathsf{Enc}(\boldsymbol{s}, \boldsymbol{m})^{2^8}$) with respect to each implementation. In the DR-FV, the modulus is $q = 2^r$ with r in Table 1. In the BGV-type schemes (DR-BGV and HElib), the modulus is $q_8 = p_0 \cdots p_8 q_s$ and we describe the bit-length r of q_8 in Table 4.1. Table 4.1 summarizes the chosen parameters.

Assuming that there is no special attack utilizing the particular algebraic structure of involving rings, corresponding security parameters λ are estimated using the lwe-estimator-9302d4204b4f by [4, 2].

	m	g	d	l	r (DR-FV)	r (DR-BGV)	r (HElib)
par-127	127	18	7	8	162	189	135
par-8191	8191	630	13	8	210	247	250
par-43691	43691	1285	34	8	234	258	256
par-131071	131071	7710	17	8	242	261	-

Table 4.1: Chosen parameters.

Table 4.2 shows timing results for HElib in milliseconds on Intel Celeron(R) CPU G1840 @ 2.80GHz × 2. (We could not perform the test for par-131071 due to shortage of memory.)

	λ	Enc	Dec	Add	Mult	Exp-by-2^8
par-127	26	0.23	0.18	0.00	0.66	4.78
par-8191	92	30.45	210.77	0.84	107.53	512.64
par-43691	237	268.00	5158.44	4.74	634.69	4187.81
par-131071	-	-	-	-	-	-

Table 4.2: Timing results of HElib on mod-2^l integer plaintexts.

The secret key is chosen uniformly random among binary vectors of Hamming weight 64 over the power basis (default of HElib) and we encrypt g number of mod-2^l integer plaintexts into a single HElib ciphertext using plaintext slots. As seen in Section 2.3.2, HElib basically realizes $GF(2^d)$ arithmetic in each of g slots. If we want to encrypt mod-2^l integer plaintexts on slots and to homomorphically evaluate on them, we can use only 1-dimensional constant polynomials in each $d(= n/g)$-dimensional slots. This should cause certain waste in time and space. In fact, for

example, timings for par-43691 ($g = 1285$) is much larger than two times of those for par-8191 ($g = 630$) while being the ratio of g is $1285/630 \approx 2$. This indicates that the HElib scheme cannot handle many mod-2^l integer slots with high parallelism. So, to encrypt large number of mod-2^l integer plaintexts using HElib, we have no choice but to prepare many ciphertexts, each of which encrypts a divided set of small number of plaintexts on their slots.

On the other hand, Table 4.3 and Table 4.4 shows timing results (also in milliseconds on Intel Celeron(R) CPU G1840 @ 2.80GHz × 2) for our DR-FV scheme and DR-BGV scheme, respectively.

	λ	Enc	Dec	Add	Mult	Exp-by-2^8
par-127	-	0.14	0.12	0.00	0.57	4.47
par-8191	29	7.39	7.37	0.03	39.43	318.65
par-43691	32	17.38	17.19	0.11	92.14	741.42
par-131071	91	104.33	103.93	0.97	574.44	4620.22

Table 4.3: Timing results of DR-FV on mod-2^l integer plaintexts.

	λ	Enc	Dec	Add	Mult	Exp-by-2^8
par-127	-	0.06	0.08	0.00	0.54	3.53
par-8191	29	2.49	2.35	0.24	21.23	127.34
par-43691	32	5.17	5.19	0.59	50.85	293.52
par-131071	84	30.14	29.35	3.70	282.11	1678.52

Table 4.4: Timing results of DR-BGV on mod-2^l integer plaintexts.

The secret key is chosen uniformly random among binary vectors of Hamming weight 64 over η-basis and we encrypt g number of mod-2^l integer plaintexts into a single DR-FV or DR-BGV ciphertext. As seen, DR-BGV scheme is a little bit faster than DR-FV scheme, due to the effect of rescaling ciphertext modulus to the smaller ones after linearization. In both schemes, timings are approximately linear with respect to the number of slots g. This means that the DR-FV and DR-BGV schemes can handle many mod-2^l slots with high parallelism, as expected. We can encrypt large number of mod-2^l integer plaintexts into a single DR-FV or DR-BGV ciphertext using mod-2^l slots without waste, and can homomorphically compute on them with high parallelism.

Then, which is faster to encrypt many numbers of mod-2^l integer plaintexts between the following two cases?

(1) A single DR-BGV ciphertext with many plaintext slots.

(2) Many HElib ciphertexts with small number of plaintext slots.

The result for par-131071 of Table 4.4 shows we can encrypt 7710 mod-2^l integer slots in a single DR-BGV ciphertext with security parameter $\lambda = 84$ with timing:

$$(30.14, 29.35, 3.70, 282.11, 1678.52)$$

On a while, the result for par-8191 of Table 4.2 shows one can encrypt the same number of 7710 mod-2^l integer slots using $\lceil 7710/630 \rceil = 13$ ciphertexts with security parameter $\lambda = 92$. The 13 times of the line par-8191 of Table 4.2 is

$$(395.85, 2740.01, 10.92, 1397.89, 6664.32).$$

Thus, our benchmark indicates that Case (1) (a single DR-BGV ciphertext with many slots) is significantly faster than Case (2) (many HElib ciphertexts with small number of plaintext slots) under the similar level of security parameters.

Result of Bootstrap. We implemented our bootstrap procedure using C++ language and performed that on Intel Xeon(R) CPU E3-1505M v5 @ 2.80GHz × 8 in combination with our DR-BGV scheme. The noise parameter $s_{err} = \sqrt{2\pi}\sigma_{err}$ is fixed as $\sigma_{err} = 3.2$. We used small parameters as an initial experiment and we used a prime $p = 2$. Table 4.5 shows parameters of the rings composing the tensorial ring.

	m	g	d	t
Right Ring	257	16	16	3
Left Ring	127	18	7	3

Table 4.5: Chosen ring parameters for bootstrap.

In the bootstrap procedure, we use three parameter sets for plaintext modulus and ciphertext modulus. Those are prm for original ciphertext, prm_{in} for the inner ciphertext and prm_{out} for

the outer ciphertext. In this experiment, we fixed the plaintext modulus as 2^8. The ciphertext modulus 2^{19} in $\mathsf{prm}_{\mathsf{in}}$ is the smallest size to success rescaling. To bootstrap this ciphertext, the number of multiplications in bootstrap procedure is less than 50, so use ciphertext modulus chain with size 50 for $\mathsf{prm}_{\mathsf{in}}$ and $\mathsf{prm}_{\mathsf{out}}$. The size of each primes in the ciphertext modulus chain is 2^{42}, this is the smallest magnitude for success at these parameters.

Table 4.6 summarizes parameters of the plaintext modulus and the ciphertext modulus.

	pt_mod	ct_mod
prm	2^8	$2^{42\times 50}$
$\mathsf{prm}_{\mathsf{in}}$	2^8	2^{19}
$\mathsf{prm}_{\mathsf{out}}$	2^{19}	$2^{42\times 50}$

Table 4.6: Chosen modulus parameters for bootstrap.

In these parameters, our bootstrap procedure takes 18,026 millisecond for one ciphertext, this is average of 20 experiments including one experiment ending in failure. The bootstrap procedure needs many homomorphic transformations between the ring and the plaintext slots. We think our simple transformation, which is just multiplying a matrix to the slot vector, makes our bootstrap efficient.

Chapter 5

Two Applications of Multilinear Maps: Group Key Exchange and Witness Encryption

Constructing multilinear maps have been long-standing open problem, the first construction based on ideal lattices has been proposed by Garg et al. After this breakthrough, various new cryptographic systems have been proposed. Garg et al. introduce the concept of level into the encodings. The system has properties: we can extract a deterministic value from the encodings at only a specific level and the encodings are not allowed to downgrade to the lower levels. These properties are useful for cryptography. We study how this graded encoding system be applied to cryptosystems, and we propose two protocols: group key exchange and witness encryption. In our group key exchange, we achieve the communication size and the computation costs per party are both O(1) with respect to the number of parties in the group by piling up the encodings of passed parties in one encoding. A witness encryption is a new type cryptosystem using NP-complete problem. The first construction is based on EXACT-COVER problem. We construct it based on another NP-complete Hamilton Cycle problem and prove its security under the Generic Cyclic Colored Matrix Model.

5.1 Introduction

Multilinear maps have been desired to be constructed, because if exists it would enable many interesting cryptographic applications. Boneh and Silverberg attempted to construct multilinear maps from abelian varieties, but they concluded that such maps should be hard to construct [9]. The first construction of multilinear maps has been proposed by Garg, Gentry and Halevi based on ideal lattices [18]. After this breakthrough, various new cryptographic systems have been proposed. Multilinear maps has been realized a system called "graded encoding system". The encodings have homomorphic addition and multiplication functions, and they introduce the concept of *level* into the encodings that makes the encoding system interesting. Here we describe two of its interesting properties. The first property is that, an encoding is probabilistic, that is, encodings of a same plaintext are different because of included randomness, but at only a specified level, a deterministic value can be extracted from the random encodings of the same plaintext. The second property is that, an encoding is unable to downgrade to the lower levels, since the rerandomized encoding is not allowed to be divided by some other encodings. Cryptography applications use these properties. For example, in the group key exchange on N parties [18] [13], each party creates a level-$(N-1)$ encoding from own encoding and others, and they extract a same value as a sheared group key at the level-$(N-1)$. In witness encryption [21], an encryptor and a decryptor generate same level encodings for each by different way and extract same values from the encodings. The second property brings one-wayness and means a strong tool for cryptosystems. In the work of attribute based encryption for circuit [20], authors uses this one-wayness to prevent a back tracking attack and achieved to treat circuits with multi fun-out gates.

Our result. We have studied applications of multilinear maps. In this work we propose two such applications: Group Key Exchange and Witness Encryption. A one-round group key exchange protocol is described in the work [18] [13], in which each party collects encodings of all other parties and multiplies own encoding and theirs one-by-one. We design a GKE protocol, in the protocol one encoding is communicated on each upflow and downflow. The parties's secret are piled up gradually. The number of encodings communicated in a session of the GKE protocol per party does not depend on the number of parties in the group. Also, its computation cost per party is independent of the number of parties because each party only multiplies a received encoding by own encoding.

Witness encryption is a new type of cryptosystem that can be achieved by using multilinear

maps [21]. The witness encryption of [21] uses EXACT-COLVER Problem as NP-complete language. We try to construct based on another NP-complete problem, Hamilton Cycle Problem. The two problems have in common with a task of collecting just N element with no duplication, and this situation matches the property of the graded encoding system that extracts a deterministic value only from the specified level-N encodings. The difference is that, when use Hamilton Cycle Problem, elements of the cycle are connected by edges and the cycle follows the adjacent edges sequentially. We take in a tool of adjacent matrices for managing edge adjacency and manages non-commutativity of vertices in a cycle by matrix's non-commutativity. We proved the security of our scheme based on our generic cyclic colored matrix model that is a variant of a generic colored matrix model defined by the work of indistinguishability obfuscation [19].

5.2 Preliminaries

5.2.1 Approximate Multilinear Maps

Gentry et al. defined their notion of a approximate multilinear maps they call *graded encoding schemes* [18]. They view group elements in multilinear map schemes as just a convenient mechanism of encoding the exponent. Typical applications of bilinear maps use $\alpha \cdot g_i$ as an encoding of a plaintext integer $\alpha \in \mathbb{Z}_p$, in contrast their setting, they retain the concept of a somewhat homomorphic encoding and make an algebraic ring R (or field) play the role of the exponent space $\mathbb{Z}_p$.

Definition of Graded Encoding Schemes

Definition 11 *(κ-Graded Encoding System [18]). A κ-Graded Encoding System consists of a ring R and a system of sets $\mathsf{S} = \{S_i^{(\alpha)} \subset \{0,1\}^* : i \in [0,\kappa], \alpha \in R\}$, with the following properties:*
1. For every fixed index $i \in [0,\kappa]$, the sets $\{S_i^{(\alpha)} : \alpha \in R\}$ are disjoint. The set $S_i^{(\alpha)}$ consists of the "level-i encodings of α".
2. There is an associative binary operation "+" and a self-inverse unary operation "-" (on $\{0,1\}^*$) such that for every $\alpha_1, \alpha_2 \in R$, every index $i \leq \kappa$, and every $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_i^{(\alpha_2)}$, it holds that $u_1 + u_2 \in S_i^{(\alpha_1+\alpha_2)}$ and $-u_1 \in S_i^{-\alpha_1}$ where $\alpha_1 + \alpha_2$ and $-\alpha_1$ are addition and negation in R.
3. There is an associative binary operation "$\times$" (on $\{0,1\}^*$) such that for every $\alpha_1, \alpha_2 \in R$, every

i_1, i_2 with $i_1 + i_2 \leq \kappa$, and every $u_1 \in S_{i_1}^{(\alpha_1)}$ and $u_2 \in S_{i_2}^{(\alpha_2)}$, it holds that $u_1 \times u_2 \in S_{i_1+i_2}^{(\alpha_1\alpha_2)}$. Here $\alpha_1 \cdot \alpha_2$ is multiplication in R, and $i_1 + i_2$ is integer addition.

The n-graded encoding system of [18] for a ring R includes a system of sets $S = \{S_i^{(\alpha)} \subset \{0,1\}^* : i \in [0,n], \alpha \in R\}$ such that, for every fixed $i \in [0,n]$, the sets $\{S_i^{(\alpha)} : \alpha \in R\}$ are disjoint. The set $S_i^{(\alpha)}$ consists of the "level-i encodings of α".

Instance Generation. The randomized $\mathsf{InstGen}(1^\lambda, 1^n)$ takes as inputs the security parameter λ and integer n. The procedure outputs $(\mathsf{params}, p_{zt})$ where params is a description of an n-graded encoding system and p_{zt} is a level-n "zero-test parameter".

Ring Sampler. The randomized $\mathsf{samp}(\mathsf{params})$ outputs a "level-zero encoding" $a \in S_0$. The induced distribution on α such that $a \in S_0^{(\alpha)}$ is statistically uniform.

Encoding. The $\mathsf{enc}(\mathsf{params}, i, a)$ takes $i \in [n]$ and a level-zero encoding $a \in S_0^{(\alpha)}$ for some $\alpha \in R$, and outputs a level-i encoding $u \in S_i^{(\alpha)}$ of α.

Re-Randomization. The $\mathsf{reRand}(\mathsf{params}, i, u)$ re-randomizes encoding u to the same level i encoding, as long as the noise of the initial encoding u is under a given noise bound.

Addition and negation. Given params and two encodings at the same level, $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_i^{(\alpha_2)}$, it holds $\mathsf{add}(\mathsf{params}, u_1, u_2) \in S_i^{(\alpha_1+\alpha_2)}$, and $\mathsf{neg}(\mathsf{params}, u_1) \in S_i^{(-\alpha_1)}$, subject to bounds on the noise.

Multiplication. For $u_1 \in S_{i_1}^{(\alpha_1)}$,$u_2 \in S_{i_2}^{(\alpha_2)}$, such that $i_1 + i_2 \leq n$, we have $\mathsf{mult}(\mathsf{params}, u_1, u_2) \in S_{i_1+i_2}^{(\alpha_1\alpha_2)}$.

Zero-test. The procedure $\mathsf{isZero}(\mathsf{params}, p_{zt}, u)$ outputs 1 if $u \in S_n^{(0)}$ and 0 otherwise. Note that in conjunction with the procedure for subtracting encodings, this gives us an equality test.

Extraction. This procedure extracts a "canonical" and "random" representation of ring elements from their level-n encoding. Namely $\mathsf{ext}(\mathsf{params}, p_{zt}, u)$ outputs (say) $K \in \{0,1\}^\lambda$, such that:

(a) With overwhelming probability over the choice of $\alpha \in R$, for any two $u_1, u_2 \in S_n^{(\alpha)}$, $\mathsf{ext}(\mathsf{params}, p_{zt}, u_1) = \mathsf{ext}(\mathsf{params}, p_{zt}, u_2)$,

(b) The distribution $\{\mathsf{ext}(\mathsf{params}, p_{zt}, u) : \alpha \in R, u \in S_n^{(\alpha)}\}$ is statistically uniform over $\{0,1\}^\lambda$.

Graded Decisional Diffie-Hellman Problem

Garg et al. [18] and Coron et al. [13] define Graded DDH Problem (GDDHProblem), as following process.

1. $(\mathsf{params}, p_{zt}) \leftarrow \mathsf{InstGen}(1^\lambda, 1^\kappa)$
2. Choose $a_j \leftarrow \mathsf{samp}(\mathsf{params})$ for all $1 \le j \le \kappa + 1$
3. Set $u_j \leftarrow \mathsf{reRand}(\mathsf{params}, 1, \mathsf{enc}(\mathsf{params}, 1, a_j))$ for all $1 \le j \le \kappa + 1$
4. Choose $b \leftarrow \mathsf{samp}(\mathsf{params})$
5. Set $\tilde{u} = a_{\kappa+1} \times \Pi_{i=1}^{\kappa} u_i$
6. Set $\hat{u} = b \times \Pi_{i=1}^{\kappa} u_i$

The GDDH distinguisher is given as input the $\kappa + 1$ level-one encodings u_j and either $\tilde{u}$ (encoding the right product) or $\hat{u}$ (encoding a random product), and must decide which is the case.

Graded Decisional Diffie-Hellman Assumption (GDDH Assumption) The Graded Decisional Diffie-Hellman Assumption is that the advantage of any efficient adversary is negligible in the security parameter against Graded Decisional Diffie-Hellman Problem.

5.3 Group Key Exchange using Multilinear Maps

The graded encoding system has a function that extracts a deterministic value associated with a plaintext from specified level encodings of the plaintext, the specified level is specified at the instance generation. This functionality fits with a group key exchange (GKE), because the shared key should be generated from secrets of all parties. We construct a GKE protocol its communication type is (N-1)-round 2-way of upflow and downflow. The construction is simple, each party multiplies a received encoding by own encoding and sends the result encoding to the next party. Since in the communicated encoding secrets of passed parties are piled up, the message complexity per party does not depend on the number of parties. The computation cost for each party is also $O(1)$.

In Table 5.1, we compare our GKE scheme with two other schemes in terms of their performance with respect to the number of parties, 'N' means the number of parties, 'Comm.' is communication size and 'Comp.' is computation cost. The first scheme is one-round GKE using Multilinear Maps in [18]. In this scheme, communicated encodings are not product, the number

Table 5.1: Comparison of performance with respect to the number of parties (per party)

Scheme	Comm.	Comp.
One-round using MM [18]	$O(N)$	$O(N)$
upflow/downflow no MM [46]	$O(N)$	$O(N)$
upflow/downflow using MM (ours)	$O(1)$	$O(1)$

of encodings per party is $O(N)$ and the cost one party multiplies received encodings is $O(N)$. The second scheme [46] is upflow/downflow type like ours, but it does not use Multilinear Maps. The number of communicated group elements per party is $O(N)$ and the computation cost per party is also $O(N)$.

In our protocol, we design the shared group key as output value of pseudo-random function which seed is a level-$(N-1)$ encoding generated by above process and which input value is a session ID. We use graded encoding system for the session ID to achieve theoretically that the computational cost and communication size per party are $O(1)$. For this reason, we make two instances of graded encoding system for level-$(N-1)$ and level-N.

We prove our protocol under GDDH assumption using security model proposed by Bresson et al [8].

5.3.1 BCPQ Model : Security Model for GKE

BCPQ Model is a formal model for Group Key Exchange protocol (GKE) proposed by Bresson, Chevassut, Pointcheval, and Quisquater [8]. In their model, the adversary $\mathcal{A}$ controls all communication between player instances and asks an instance to release session key or long-lived key. A set ID of n players in protocols P is fixed. A player $U_i \in ID$ can have many instances called oracles, involved in distinct concurrent execution of P. An instance s of player U_i is denoted as oracle Π_i^s.

SessionIDS. Session id for oracle Π_i^s is defined as $SIDS(\Pi_i^s) = \{SIDS_{ij} : j \in ID\}$ where $SIDS_{ij}$ is the concatenation of all flows that Π_i^s exchanges with Π_j^t in an execution of P. Adversary $\mathcal{A}$ listens on the wire and can constructs SIDS.

Oracle Queries. Adversaries can send following queries to oracles:

·$\mathsf{Send}(\Pi_U^s, m)$:Adversary $\mathcal{A}$ gets the response which Π_U^s have generated in processing incoming-message m.

·Reveal(Π_U^s):This query forces Π_U^s to release its session key.

·Corrupt(U):Adversary $\mathcal{A}$ gets long-lived key LL_U of U but does not get the internal data of instance of U.

·Test(Π_U^s):$\mathcal{A}$ gets back session key or random string from Π_U^s.

Definition 12 (Correctness) *A GKE protocol is* correct *if for any operation execution between the oracles* $\Pi_{U_1}^s, \cdots, \Pi_{U_N}^s$ *with the same session ids* sid = $\mathrm{SIDS}(\Pi_{\mathrm{i}}^{\mathrm{s}})$ *all oracles accept with the same session group key.*

Definition 13 (AKE Security) *Protocol P is called* AKE-secure *if advantage of any efficient* $\mathcal{A}$ *in the following game* $Game^{AKE}(\mathcal{A})$ *is negligible.*

$Game^{AKE}$: Adversary $\mathcal{A}$ can ask queries except Test many times, and once, $\mathcal{A}$ sends a Test-query to a *fresh* oracle (a oracle is fresh if nobody has been asked for Corrupt-query at that moment, and no Reveal-query is asked to the oracle or its partners). $\mathcal{A}$ gets back its session key or a random string. The adversary wins if she correctly guesses the bit b used in the above game, and the advantage is probability of win minus 1/2, taken over all bit tosses.

5.3.2 Our Construction

Our scheme consists of PPT algorithms Setup, Upflow, Downflow and KeyGen, and parties are indexed Party_1 to Party_N, and Party_i is connected to Party_{i+1} $(1 \leq i \leq N-1)$. This GKE protocol first executes the Setup algorithm and the resulting public information should be shared among parties. Party_1 executes Upflow_1 without input value. Then, Party_i executes Upflow_i, receiving the outputs $(c_{ur}, d_{ur}, \sigma_{ur})$ of its precedent Upflow_{i-1} as inputs, sequentially for $i = 2$ to N. Then, Party_N executes $\mathrm{Downflow}_N$ without input value and Party_i executes $\mathrm{Downflow}_i$, receiving the outputs $(c_{dr}, d_{dr}, \sigma_{dr})$ of the precedent $\mathrm{Downflow}_{i+1}$ as inputs, also sequentially for $i = (N-1)$ to 1. In the Downflow sequence, each $\mathrm{Downflow}_i$ invokes the algorithm Keygen to compute the shared session-key ξ among the parties.

Building blocks are multilinear maps MM, EUF-CMA secure signature scheme Σ and pseudo random function H_{seed}. In the following algorithm, $a \cdot b$ denotes mult(params, a, b).

Setup. The algorithm Setup takes security parameter λ and number of parties N as input, makes two sets of MM parameter by instance generation specifying level $N-1$ and N, respectively. Then it generates N pairs of signing key sk_i and verification key vk_i by key generation of Σ, and sets signing keys to each party. The algorithm outputs two sets of parameter of MM and all verification keys $\{vk_i\}$ as public parameter.

$\mathsf{Setup}(1^\lambda, 1^N)$
  $(\mathsf{params}_1, p_{zt1}) \leftarrow \mathsf{MM.InstGen}(1^\lambda, 1^{N-1})$
  $(\mathsf{params}_2, p_{zt2}) \leftarrow \mathsf{MM.InstGen}(1^\lambda, 1^N)$
  for $i = 1$ to N
    $(vk_i, sk_i) \leftarrow \Sigma.\mathsf{KeyGen}(1^\lambda)$
    Sets sk_i to U_i
  return $(\mathsf{params}_1, p_{zt1}, \mathsf{params}_2, p_{zt2}, vk_i(1 \le i \le N))$

Upflow. The algorithm Upflow takes encoding c_{ur}, d_{ur} and signature σ_{ur} as input and verifies them on sender's verification key vk_{i-1}. Then it creates a secret level-0 encoding a_i and upgrades to level-1 and rerandomizes that using param_1. The algorithm creates one more level-1 encoding d_1 for sessionID using param_2. Then, multiplies them by received encoding and outputs the two encodings with its signature σ_{us}.

$\mathsf{Upflow_i}(c_{ur}, d_{ur}, \sigma_{ur})$
  if $i \neq 1$ and $\Sigma.\mathrm{verify}(vk_{i-1}, c_{ur}\|d_{ur}, \sigma_{ur})$=false then abort
  $a_i \leftarrow \mathsf{MM.samp}(\mathsf{params}_1)$
  $c_i \leftarrow \mathsf{MM.reRand}(\mathsf{params}_1, 1, \mathsf{MM.enc}(\mathsf{params}_1, 1, a_i))$
  $d_i \leftarrow \mathsf{MM.reRand}(\mathsf{params}_2, 1, \mathsf{MM.enc}(\mathsf{params}_2, 1, \mathsf{MM.samp}(\mathsf{params}_2)))$
  if $i \neq N$ then
    if $i = 1$ then $c_{us} := c_i$, $d_{us} := d_i$
    else $c_{us} := c_{ur} \cdot c_i$, $d_{us} := d_{ur} \cdot d_i$
    $\sigma_{us} \leftarrow \Sigma.\mathrm{sign}(sk_i, c_{us}\|d_{us})$
    return $(c_{us}, d_{us}, \sigma_{us})$

Downflow. The algorithm Downflow takes encodings c_{dr},d_{dr} and signature σ_{dr} as input and verifies them on sender's verification key vk_{i+1}. Then it generates a session key ξ by key generation algorithm, and multiplies own two encoding c_i and d_i by received encoding c_{dr} and d_{dr} respectively, and outputs the result encodings with its signature σ_{ds}.

$\mathsf{Downflow_i}(c_{dr}, d_{dr}, \sigma_{dr})$

if $i \neq N$ and Σ.verify$(vk_{i+1}, c_{dr}||d_{dr}, \sigma_{dr})$=false then abort

if $i \neq 1$ then

if $i = N$ then $c_{ds} := c_i,\ d_{ds} := d_i$

else $c_{ds} := c_{dr} \cdot c_i,\ d_{ds} := d_{dr} \cdot d_i$

$\sigma_{ds} \leftarrow \Sigma.\mathrm{Sign}(sk_i, c_{ds}||d_{ds})$

$\xi \leftarrow \mathsf{KeyGen}(a_i, c_{ur}, c_{dr}, d_i, d_{ur}, d_{dr})$

return $(c_{ds}, d_{ds}, \sigma_{ds})$ as message and ξ as local output.

KeyGen. The algorithm KeyGen takes encodings $a_i, c_{ur}, c_{dr}, d_i,\ d_{ur}, d_{dr}$ as input, multiplies a_i, c_{ur}, c_{dr} for seed of H and d_i, d_{ur}, d_{dr} for sessionID. MM.ext extracts a deterministic value from level-$(N-1)$ encoding c'_i, and KeyGen algorithm sets it to seed of H. A deterministic value from level-N encoding d'_i is sessionID and input to H.

$\mathsf{KeyGen_i}(a_i, c_{ur}, c_{dr}, d_i, d_{ur}, d_{dr})$

$c'_i := a_i \cdot c_{ur} \cdot c_{dr}$

$d'_i := d_i \cdot d_{ur} \cdot d_{dr}$

$\delta \leftarrow \mathsf{MM.ext}(\mathsf{params_1}, p_{zt1}, c'_i)$

$sessionID \leftarrow \mathsf{MM.ext}(\mathsf{params_2}, p_{zt2}, d'_i)$

return $\xi \leftarrow \mathsf{H}_\delta(sessionID)$

Correctness. We show all parties share a same session key string. Considering output string from H in KeyGen algorithm, seed δ of H is the product of own private level-0 encoding and level-1 encodings of all other parties. MM.ext outputs the same string that is a deterministic value of product of plaintexts a_i of all parties. SessionID is product of level-1 encodings d_i of all parties, thus MM.ext outputs the same value for each party.

5.3.3 Proof of Security

Theorem 4 *Our protocol is AKE secure under the GDDH Assumption (see Section A.2), and the assumptions that the signature scheme* Σ *is EUF-CMA and the function H is secure PSF.*

Proof. We prove the security of our protocol based on BCPQ security model.

Game0: Original Security game for our protocol

–Send(null,"$start$")

Follow the instruction of the Setup algorithm to compute and return ($\mathsf{params}_1, p_{zt1}, \mathsf{params}_2, p_{zt2}, vk_i (1 \leq i \leq N)$).

–Send(Π_i^s, Upflow($c_{ur}, d_{ur}, \sigma_{ur}$))

Follow the instruction of the $\mathsf{Upflow_i}$ algorithm to compute and return ($c_{us}, d_{us}, \sigma_{us}$).

–Send(Π_i^s, Downflow($c_{dr}, d_{dr}, \sigma_{dr}$))

Follow the instruction of the $\mathsf{Downflow_i}$ algorithm to compute and return ($c_{ds}, d_{ds}, \sigma_{ds}$) and local output ξ.

–Reveal(Π_i^s): return ξ

–Corrupt(U_i): return sk_i

–Test(Π_i^s)

$b \overset{\$}{\leftarrow} \{0,1\}$

if $b = 0$ then return ξ else return $r \overset{\$}{\leftarrow} \{0,1\}^\lambda$

Then adversary $\mathcal{A}$ outputs guess of b as $\hat{b}$.

We define S_i to be the event that $b = \hat{b}$ in Game$i$.

Game1: We make a transition into Game1: Abort if message is fabricated.

–Send(Π_i^s, Upflow(c_{ur}, d_{ur}, σ))

if $c_{ur}\|d_{ur} \notin m'$ and Σ.verify$(vk_{i-1}, c_{ur}\|d_{ur}, \sigma_{ur})$=true

then abort

if $i \neq 1$ and Σ.verify$(vk_{i-1}, c_{ur}\|d_{ur}, \sigma_{ur})$=false then abort

$m' := m'\|c_{ur}\|d_{ur}$

// following is same as Game0

–Send(Π_i^s, Downflow(c_{dr}, d_{dr}, σ))

if $c_{dr}\|d_{dr} \notin m'$ and Σ.verify$(vk_{i+1}, c_{dr}\|d_{dr}, \sigma_{dr})$=true

then abort

if $i \neq N$ and Σ.verify$(vk_{i+1}, c_{dr}\|d_{dr}, \sigma_{dr})$=false then abort

$m' := m'\|c_{dr}\|d_{dr}$

// following is same as Game0

Claim 1 *If the signature scheme* Σ *is EUF-CMA secure, then* $|\Pr[S_0] - \Pr[S_1]| \leq negl(\lambda)$.

Sketch of Proof. Let c and d be received encodings, m' be a set of messages that had been verified, vk be a verification key and σ be a signature. We define failure event F that "$(c, d) \notin m'$ and Σ.verify$(vk, c\|d, \sigma)$ = true". Then, $S_0 \wedge \neg F \Leftrightarrow S_1 \wedge \neg F$. Let $\mathcal{A}$ be an adversary that can get an advantage in distinguishing between Game0 and Game1. We construct a forger $\mathcal{F}$ from $\mathcal{A}$.

Forger $\mathcal{F}$. $\mathcal{F}$ guesses a party j whose signature $\mathcal{A}$ will forge. In Setup algorithm, $\mathcal{F}$ sets input vk to vk_j of party j. $\mathcal{A}$ sends Send-query to Π_i^s, if i=j and event F then $\mathcal{F}$ outputs the $((c, d), \sigma)$. $\mathcal{F}$ calls signing oracle to get signature of j.

The probability of $\mathcal{F}$ successes in above game is 1/N · Pr[F] and signature scheme Σ is EUF-CMA, thus Pr[F] $\leq negl(\lambda)$. From Difference Lemma defined by Shoup in Sequence of Games [44], $|\Pr[S_0]-\Pr[S_1]| \leq \Pr[F]$.

$\therefore |\Pr[S_0]-\Pr[S_1]| \leq negl(\lambda)$. □

Game2: We make a transition into Game2. In this game, seed of H is changed into random encoding which is formed by GDDH Problem instance.

–Test(Π_i^s)

$b \xleftarrow{\$} \{0, 1\}$

if $b = 0$ then

$\boxed{a' \leftarrow \mathsf{MM.samp}(\mathsf{params}_1)}$

$c_i' := a' \cdot c_{ur} \cdot c_{dr}$

$d_i' := d_i \cdot d_{ur} \cdot d_{dr}$

$\delta \leftarrow \mathsf{MM.ext}(\mathsf{params}_1, p_{zt1}, c_i')$

$sessionID \leftarrow \mathsf{MM.ext}(\mathsf{params}_2, p_{zt2}, d_i')$

$\xi \leftarrow \mathrm{H}_\delta(sessionID)$

return ξ

else return $r \xleftarrow{\$} \{0,1\}^\lambda$

Claim 2 *Under GDDH the assumption,* $|\Pr[\mathrm{S}_1] - \Pr[\mathrm{S}_2]| \leq negl(\lambda)$.

Proof. We construct a distinguisher $\mathcal{D}$ of GDDH Problem from adversary $\mathcal{A}$ against our protocol. $\mathcal{D}$ guesses a party whom $\mathcal{A}$ will ask Test-query. Its instance and partner instances are denoted by Π_i^{s*}. We define a event Guess that is "All Π_i^{s*} are correct". If input of $\mathcal{D}$ is $\tilde{u}$ then the view of $\mathcal{A}$ in $\mathcal{D}$ is view of $\mathcal{A}$ in Game1, and if input of $\mathcal{D}$ is $\hat{u}$ then that is view of $\mathcal{A}$ in Game2. Algorithm $\mathcal{D}(u_j(1 \leq j \leq N), T = \tilde{u} \text{ or } \hat{u})$

$s_i^* \xleftarrow{\$}$[1,S] //S is upper bound of numbers of instances $\mathcal{A}$ creates.

–Send(Π_i^s, $\mathsf{Upflow}(c_{ur}, d_{ur}, \sigma)$)

$\boxed{\text{if } s = s_i^* \text{ then}}$

if $c_{ur} \| d_{ur} \notin m'$ and Σ.verify$(vk_{i-1}, c_{ur} \| d_{ur}, \sigma_{ur})$=true then abort

if $i \neq 1$ and Σ.verify$(vk_{i-1}, c_{ur} \| d_{ur}, \sigma_{ur})$=false then abort

$m' := m' \| c_{ur} \| d_{ur}$

$\boxed{c_i \leftarrow u_i}$

$d_i \leftarrow \mathsf{MM.reRand}(\mathsf{params}_2, 1, \mathsf{MM.enc}(\mathsf{params}_2, 1, \mathsf{MM.samp}(\mathsf{params}_2)))$

if $i \neq N$ then

if $i = 1$ then $c_{us} := c_i, d_{us} := d_i$

else $c_{us} := c_{ur} \cdot c_i, d_{us} := d_{ur} \cdot d_i$

$\sigma_{us} \leftarrow \Sigma.\mathrm{Sign}(sk_i, c_{us} \| d_{us})$

return $(c_{us}, d_{us}, \sigma_{us})$

$\boxed{\text{else}}$ // $s \neq s_i^*$, following is same as Game1

–Send(Π_i^s, Downflow(c_{dr}, d_{dr}, σ))

if $s = s_i^*$ then

if $c_{dr}\|d_{dr} \notin m'$ and Σ.verify($vk_{i+1}, c_{dr}\|d_{dr}, \sigma_{dr}$)=true then abort

if $i \neq N$ and Σ.verify($vk_{i+1}, c_{dr}\|d_{dr}, \sigma_{dr}$)=false then abort

$m' := m'\|c_{dr}\|d_{dr}$

if $i \neq 1$ then

if $i = N$ then $c_{ds} := c_i, d_{ds} := d_i$

else $c_{ds} := c_{dr} \cdot c_i, d_{ds} := d_{dr} \cdot d_i$

$\sigma_{ds} \leftarrow \Sigma.\text{Sign}(sk_i, c_{ds}\|d_{ds})$

// MM.KeyGen is not called.

return (c_{ds}, σ_{ds})

else // $s \neq s_i^*$, following is same as Game2

–Reveal(Π_i^s)

if $s = s_i^*$ then abort with a random bit

return ξ

–Corrupt(U_i): return sk_i

–Test(Π_i^s)

if $s \neq s_i^*$ then abort with a random bit

$b \xleftarrow{\$} \{0, 1\}$

if $b = 0$ then

$d_i' := d_i \cdot d_{ur} \cdot d_{dr}$

$\delta \leftarrow \mathsf{MM.ext}(\mathsf{params}_1, p_{zt1}, T)$

$sessionID \leftarrow \mathsf{MM.ext}(\mathsf{params}_2, p_{zt2}, d_i')$

return $\xi \leftarrow \mathrm{H}_\delta(sessionID)$

else return $r \xleftarrow{\$} \{0, 1\}^\lambda$

The advantage of D is Adv_D = | Pr[D(u_j, T)=1 | $T = \tilde{u}$] - Pr[D(u_j, T)=1| $T = \hat{u}$] |. Now we describe Pr[D(u_j, T)=1 | $T = \tilde{u}$] as Pr[D($\tilde{u}$)=1], Pr[D(u_j, T)=1| $T = \hat{u}$] as Pr[D($\hat{u}$)=1].

$\Pr[D(\tilde{u})=1] = \Pr[D(\tilde{u})=1 \wedge \neg \text{ Guess}] + \Pr[D(\tilde{u})=1 \wedge \text{Guess}]$

$=1/2 + \Pr[\text{Guess}]\Pr[D(\tilde{u})=1 \mid \text{Guess}]$

$=1/2 + \Pr[\text{Guess}]\Pr[S_1]$

$\Pr[D(\hat{u})=1] = \Pr[D(\hat{u})=1 \wedge \neg \text{ Guess}] + \Pr[D(\hat{u})=1 \wedge \text{Guess}]$

$=1/2 + \Pr[\text{Guess}]\Pr[D(\hat{u})=1 \mid \text{Guess}]$

$=1/2 + \Pr[\text{Guess}]\Pr[S_2]$

$\text{Adv}_D = |\Pr[D(\tilde{u})=1] - \Pr[D(\hat{u})=1]| = \Pr[\text{Guess}]\cdot(\Pr[S_1]-\Pr[S_2])$

From GDDH assumption $\text{Adv}_D \leq negl(\lambda)$, and Pr[Guess] is 1/poly(λ). Therefore $|\Pr[S_1]-\Pr[S_2]| \leq negl(\lambda)$. □

Game3: We make a transition into Game3. In this game, the value of H is changed into a random string.

–Test(Π_i^s)

$b \xleftarrow{\$} \{0,1\}$

if $b = 0$ then return $\boxed{r' \xleftarrow{\$} \{0,1\}^\lambda}$

else return $r \xleftarrow{\$} \{0,1\}^\lambda$

Claim 3 *If H is secure then* $|\Pr[S_2] - \Pr[S_3]| \leq negl(\lambda)$.

Proof. Immediate from security of pseudo-random function. □

The advantage of $\mathcal{A}$ is clearly negligible in Game3 and from Claim1, Claim2, Claim3, the advantage of $\mathcal{A}$ of real is also negligible. □

5.4 Witness Encryption using Multilinear Maps

5.4.1 Preliminaries

Witness Encryption

A witness encryption scheme [21] for an NP language L(with corresponding witness relation R) consists of the following two polynomial-time algorithms:

Encryption. The algorithm $\mathsf{Encrypt}(1^\lambda, x, m)$ takes as input a security parameter λ, an unbounded-length string x, and a message $m \in \mathcal{M}$, and outputs a ciphertext CT.

Decryption. The algorithm $\mathsf{Decrypt}(CT, w)$ takes as input a ciphertext CT and an unbounded-length string w, and outputs a message m or the symbol $\perp$.

· Correctness. For any security parameter λ, for any m and for any $x \in L$ s.t. $R(x, w)$ holds, we have that Pr[$\mathsf{Decrypt}(\mathsf{Encrypt}(1^\lambda, x, m), w) = m$] $= 1 - negl(\lambda)$.
· Soundness Security. For any $x \notin L$, for any PPT adversary $\mathcal{A}$ and messages $m_0, m_1 \in \mathcal{M}$,
| Pr[$\mathcal{A}(\mathsf{Encrypt}(1^\lambda, x, m_0)) = 1$] - Pr[$\mathcal{A}(\mathsf{Encrypt}(1^\lambda, x, m_1)) = 1$] $| \leq negl(\lambda)$.

The Security-Correctness Gap. Correctness requires that an algorithm can decrypt if $x \in L$ and it knows w s.t. $R(x, w)$. Soundness Security requires that if $x \notin L$ then no PPT algorithm can decrypt. This security does not define on the case when $x \in L$. But since distinguishing $x \in L$ and $x \notin L$ is NP-complete, decryption would be difficult even if $x \in L$.

Definition of Graph and Hamilton Cycle Problem

A graph $G = (V, E)$ is a pair of a set of vertices V and a set of edges E associated with pairs of vertices, both being assumed finite. Let $V(G) = \{v_1, v_2, \cdots, v_n\}, E(G) = \{e_1, e_2, \cdots, e_m\}, |V| = n, |E| = m$. We will describe an edge between vertex v_i and vertex v_j as $e_{i,j}$. A walk of length k from v_0 to v_k is $P = (V, E)$ with $V = \{v_0, v_1, \cdots, v_k\}, E = \{e_{0,1}, \cdots, e_{k-1,k}\}$. A path is a walk with all different vertices. A cycle is a path which start vertex is equal to the goal vertex. A cycle that visits all vertices of V is called a Hamilton Cycle (HC). A directed graph is a graph where each edge has a direction, in such graph, $e_{i\to j}$ denotes an edge from vertex v_i to vertex v_j. An undirected graph is one where every edge has both $e_{i\to j}$ and $e_{j\to i}$. A simple graph is one which has no self-loop or multiple edges.

The Hamilton Cycle Problem (HCP) is that, given a graph $G = (V, E)$, decide whether G has a HC or not. A graph with a HC is called Hamiltonian Graph. HCP is NP-complete both for directed graphs and undirected graphs.

Generic Colored Matrix Model

Garg et al. defined a generic colored matrix model in [19] that captures attacks where the adversary is only allowed to add/multiply matrices in the correct order. They represent this restriction as color assigned to left and right of matrix and handle. For every matrix M, the corresponding record is $(h, M, (m, LC), (n, RC))$ where h is handle, M is $m \times n$ matrix, LC is left color, and RC is right color.

Setup. The represent oracle choose an initial set of l colored matrices and assigns to them the handle, inserts into the database the record $\{(h_i, M_i, (m_i, LC_i), (n_i, RC_i))\}_{i=1}^{l}$, and sends $\{(h_i, (m_i, LC_i), (n_i, RC_i))\}_{i=1}^{l}$ to the adversary but not the matrix M.

The adversary sends queries through two handles. A represent oracle, who performs generic computation to the adversary, looks up records corresponding the two handles in the database and if their color and row/column size satisfy the order restriction of addition or multiplication then the oracle adds or multiplies the two matrices. If the result record is not in the database then the oracle inserts the record with new handle h' into database. Then the oracle returns a contained or new handle. The adversary being unable to get matrix, he can only add and multiply through given handles. The following is the process of representation oracle.

Addition. When the adversary makes query $\mathsf{add}(h_1, h_2)$, the oracle looks up the records corresponding to h_1, h_2 in the database. If such records exist, then if $(m_1, LC_1) = (m_2, LC_2)$ and $(n_1, RC_1) = (n_2, RC_2)$ then the oracle computes their sum $M = M_1 + M_2$. If the database already contains the matrix $(M, (m_1, LC_1), (n_1, RC_1))$ then oracle returns its handle. Otherwise, the oracle assigns a new handle h' and inserts into the database $(h', M, (m_1, LC_1), (n_1, RC_1))$ and returns the new handle.

Multiplication. When the adversary makes query $\mathsf{mult}(h_1, h_2)$, the oracle looks up the records corresponding to h_1, h_2 in the database. If such records exist, then if $(n_1, RC_1) = (m_2, LC_2)$ then the oracle computes their product $M = M_1 M_2$. If the database already contains the matrix $(M, (m_1, LC_1), (n_2, RC_2))$ then oracle returns its handle. Otherwise, the oracle assigns a new handle h' and inserts into the database $(h', M, (m_1, LC_1), (n_2, RC_2))$ and returns the new handle.

5.4.2 Design Principle

We use a directed hamiltonian graph to construct our witness encryption. First, we consider design of blinding factor which hides plain text. HC passes every vertex in a graph exactly once, for this, we assign a secret to every vertex and set some value storing the secret to every edge. We generate a blinding factor from these secrets of all vertices and hide a message by the blinding factor and make it a ciphertext. Other components of ciphertext are edge secrets. One who knows a HC can collect all secrets from ciphertext using the knowledge of HC and he can recover the blinding factor. Difficulty of designing such blinding factor is that, during decrypt one who knows HC can generate the blinding factor and during encrypt the encryptor creates the same value without knowledge of HC.

We adopt to use high-dimensional matrix for achieving this restriction of ordering on decryption because of matrix holds non-commutative. We assign an adjacency matrix to each edge, in which we set a secret of starting vertex, and put these adjacency matrices of all edges in the ciphertext. A decryptor who knows the HC can multiply all adjacency matrices in the order of the given HC and can make a product of the secrets of all vertices in the graph. The product value has no order, therefore, the encryptor also can make the same value.

Detail of the design. We name vertices by its index like $1, 2, \cdots, n$ and let $P_{i\to j}$ be a matrix assigned to an edge $e_{i\to j}$. We set a secret s_i of vertex i to the (i, j)-th element in $P_{i\to j}$. An element of (i, j) in a product matrix of two adjacent matrices P_i and P_j is a product $s_i s_j$. By multiplying matrices of edge in order of the given HC which starts and goals at vertex i, the element of (i, i) in the product matrix is $\Pi_{j\in[n]} s_j$. This value is independent of the order of vertices, therefore encryptor also can make this value from secrets assigning to vertices.

To prevent pulling out a secret s_i in the edge matrix $P_{i\to j}$, it is necessary to randomize edge matrices. For this, we take in a technique of oblivious transfer proposed by Joe Kilian [31]. We assign a randomized matrix R_i and its inverse R_i^{-1} to each vertex and set $R_i^{-1} P_{i\to j} R_j$ be a new edge matrix. When multiply these new matrices of adjacency edge, e.g. $R_i^{-1} P_{i\to j} R_j$ and $R_j^{-1} P_{j\to k} R_k$, the product $R_j R_j^{-1}$ becomes identity matrix, thus the connected edge matrices becomes $R_i^{-1} P_{i\to j} P_{j\to k} R_k$. This works sequentially, a product matrix of connected edge matrices in a path that starts vertex i and goals vertex x becomes $R_i^{-1} P_{i\to j} \cdots P_{(x-1)\to x} R_x$. If $i \neq x$ then, the secret $\Pi_{k=i}^{x-1} s_k$ in the matrix $\Pi_{k=i}^{x-1} P_k$ is randomized, therefore it is impossible to pull out the secret. The other hand, if $i = x$ then, the product is $R_i^{-1} P_{i\to j} \cdots P_{y\to i} R_i$ and it is possible to pull

out a trace $\Pi_{k=i}^{y} s_k$ of this product. This means that this construction has a problem that one who does not know HC can make blinding factor $\Pi_{j\in[n]} s_j$ from traces of partial cycles. We call partial cycle as "short cycle".

Definition 14 (Short Cycle) *We call a cycle "short cycle" if the cycle satisfies:*

1. *The cycle does not pass all vertices in a graph.*
2. *Each vertex in the cycle is passed only once.*

In other words, "short cycle" is a cycle except HC.

We assign one more secret r_i to each vertex i and set that to the $(1, 1)$-th element of every edge matrix $P_{i\to j}$. A $(1, 1)$-th element of a product matrix of edge matrices in a cycle that starts and goals vertex 1 becomes $\Pi r_i + \Pi s_i$. We regard HC as a cycle that starts and goals vertex 1, and then a element $(1, 1)$ in product of edge matrices in HC becomes $\Pi_{j\in[n]} r_j + \Pi_{j\in[n]} s_j$. This value is unable to be calculated from traces, that are $\Pi r + \Pi s$ from a short cycle not including vertex 1.

For the purpose of security proof, we assign another matrix R_1' instead of R_1^{-1} s.t. $R_1' R_1 \neq$ identity matrix. In a product of in-edge matrix and out-edge matrix about vertex 1, $R_1^{-1} R_1'$ is not canceled, therefore it prevents pulling out a trace. The blinding factor is now changed to $R_1'(\Pi_{j\in[n]} r_j + \Pi_{j\in[n]} s_j) R_1$.

Every values assigned to vertex (i.e., secrets s_i, r_i and all elements of random matrices R_i, R_i^{-1} and R_1') are encoded at level-0 by sampling algorithm of multilinear maps. During making edge matrix in encrypt algorithm, all components in edge matrix $R_i^{-1} P_{i\to j} R_j$ are encoded to level-1 and rerandomized, the rerandomized encoding keeps additive and multiplicative homomorphism but does not keep homomorphism divisionally. We set up this encoding system with level-n. Using this zero-test parameter, we can extract a deterministic string from level-n encoding of each element in $R_1'(\Pi_{j\in[n]} r_j + \Pi_{j\in[n]} s_j) R_1$. We concatenate these extracted values and extract a random value of this concatenation by a strong randomness extractor. The extracted value is finally our blinding factor.

5.4.3 Our Construction

In our scheme of witness encryption, a NP-complete language is Hamilton Cycle Problem, and a statement is a graph G. As building blocks, we use a multilinear map MM and a randomness extractor ext. Notations: The symbol || denotes to concatenate strings. We omit "params" for

simplicity in the following algorithms.

Concrete Construction

Encryption. Encryption algorithm takes as input security parameter λ, a graph G of n vertices as statement of language, and a message m. First, it generates parameters of MM specifying level-n. Next, creates a level-0 encoded random matrix R_i and samples two level-0 encodings r_i and s_i for every vertices i in G. Then it creates matrix $P_{i\to j}$ for edge $e_{i\to j}$ and set r_i and s_i to the matrix. A public matrix $\hat{E}_{i\to j}$ is product of R_i^{-1}, $P_{i\to j}$ and R_i, whose elements are encoded level-1. Last, it creates blinding factor from all s_i and r_i and hides a message m.

$\mathsf{Encrypt}(1^\lambda, G, m) \to CT$

$(\mathsf{params}, p_{zt}) \leftarrow \mathsf{MM.InstGen}(1^\lambda, 1^n)$

for $i \in [n]$ // `all vertices`

$R_i \leftarrow \mathsf{sampMatrix}(n)$

if $i = 1$ then $R'_1 \leftarrow \mathsf{sampMatrix}(n)$

else create a invert matrix R_i^{-1} from R_i

$r_i \leftarrow \mathsf{samp}()$, $s_i \leftarrow \mathsf{samp}()$

for $i \in [n]$

for j s.t. edges $i \to j$ are out-edges of vertex i in G

create a $n \times n$ matrix $P_{i\to j}$

$(P_{i\to j})_{1,1} := r_i$, $(P_{i\to j})_{i,j} := s_i$

$(P_{i\to j})_{x,y} := 0$ if $(x, y) \neq (1, 1)$ and (i, j)

if $i = 1$ then $E_{i\to j} := R'_1 P_{1\to j} R_j$ // `level-0`

else $E_{i\to j} := R_i^{-1} P_{i\to j} R_j$ // `level-0`

$\hat{E}_{i\to j} \leftarrow \mathsf{encodeMatrix}(1, E_{i\to j})$ // `level-1`

Create an $n \times n$ matrix P'

$(P')_{1,1} := \Pi_{i\in[n]} r_i + \Pi_{i\in[n]} s_i$

$(P')_{x,y} := 0$ if $(x, y) \neq (1, 1)$

for $i, j \in [n]$

$u := u || \mathsf{MM.ext}(p_{zt}, \mathsf{MM.enc}(n, (R'_1 P' R_1)_{i,j}))$

$C := m \oplus \mathsf{ext}(u)$

return $CT = (C, \{\hat{E}_{i\to j}\}_{(i,j)\in E(G)}, G)$

$\mathsf{sampMatrix}(n) \rightarrow M$

create a $n \times n$ matrix M

for $i, j \in [n]$

$M_{i,j} \leftarrow \mathsf{MM.samp}()$ // `level-0`

return M

$\mathsf{encodeMatrix}(k, M) \rightarrow M$

for $i, j \in [n]$

$M_{i,j} \leftarrow \mathsf{MM.reRand}(k, \mathsf{MM.enc}(k, M_{i,j}))$ //`level-k`

return M

Decryption. Decryption algorithm takes as input a ciphertext CT and a hamilton cycle HC as witness. We can suppose the first vertex of HC is vertex 1. The algorithm chooses edge matrix in CT in the order of HC and multiplies them. Then it extracts a deterministic string for every element in the product by MM.ext and it concatenates all of the strings. Here $j = \mathsf{HC}(i)$ denotes a next vertex to i in HC $\{\cdots, i, j, \cdots\}$.

$\mathsf{Decrypt}(CT, HC)$

$M := \hat{E}_{1 \rightarrow HC(1)} \hat{E}_{HC(1) \rightarrow HC^2(1)} \cdots \hat{E}_{HC^{n-1}(1) \rightarrow 1}$

for $i, j \in [n]$

$u := u || \mathsf{MM.ext}(p_{zt}, \mathsf{MM.enc}(n, M_{i,j}))$

return $m := C \oplus \mathsf{ext}(u)$

Correctness

Now we show a blinding factor u recovered in the decryption algorithm is the same as u in the encryption algorithm. First, we evaluate M in the decryption. Every element in M is rerandomized encoding and we can homomorphically add and multiply them. Therefore, M is equivalent to level-n encoding of the following product.

$$(R'_1 P_{1 \rightarrow 2} R_2)(R_2^{-1} P_{2 \rightarrow 3} R_3) \cdots (R_{(n-1)}^{-1} P_{(n-1) \rightarrow 1} R_1) = R'_1 P_{1 \rightarrow 2} P_{2 \rightarrow 3} \cdots P_{(n-1) \rightarrow 1} R_1$$

Let P'' be $P_{1 \rightarrow 2} P_{2 \rightarrow 3} \cdots P_{(n-1) \rightarrow 1}$, then $P''_{1,1} = \Pi_{i \in [n]} r_i + \Pi_{i \in [n]} s_i$ and $P''_{x,y} = 0$ for $(x, y) \neq (1, 1)$.

Thus P'' is equal to P' calculated in the encryption algorithm. For each element in P' and P'', their output values by MM.ext are the same. Therefore, the blinding factor made by the encryption algorithm and the decryption algorithm are the same.

5.4.4 Proof of Security: Soundness Security

First, we define a variant of generic colored matrix model: *a generic cyclic colored matrix model*, and then define a security game based on the model.

Generic Cyclic Colored Matrix Model

We define a generic cyclic colored matrix model by expanding a generic colored matrix model defined by Garg et al. in [19] that captures attacks where the adversary only adds and multiplies matrices in the correct order.

A matrix in our ciphertext is sandwiched in between two random matrices, when multiply adjacent edge matrices, such random matrices are canceled. But when multiply not-adjacent edge matrices, they are not canceled and randomization remains. The generic colored matrix model [19] modeled this situation as colored matrix: an adversary multiplies matrices only if he queries two matrices such that RC_1 is the same as LC_2. Since our ciphertexts are encoded, we can add and multiply elements in matrices homomorphically but we cannot divide one by another. In the generic colored matrix model [19], an adversary queries add and mult using handle, therefore, we use this models for our encoding restriction.

If the LC and RC of a result matrix M are same then a trace of matrix M is possible to be pulled out. In the oracle's process in our model, when such case, the oracle computes a trace of the result matrix M, and returns a record additionally for the trace. In our model, there are two record types: "Matrix Type" and "Scalar Type". We add a query scalar_mult that multiplies a trace value in a "Scalar Type" record by a matrix in a "Matrix Type" record.

We do not return a handle of a eigenvalue of a matrix, because computing the eigenvalue needs division during Gaussian elimination, but the division is not provided on the encoding system used in our scheme.

Detail of The Generic Cyclic Colored Matrix Model

Setup and Addition algorithms are the same as the original generic colored matrix model. In Multiplication algorithm, the following process is executed after Multiplication process of the

original model. If the types of record corresponding h_1 and h_2 are both matrix type and LC_1 and RC_2 of result matrix are the same then the oracle computes a trace M' of the result matrix. If the trace is in the database then returns its handle, otherwise inserts into the database the new record with new handle $(h'', M', (1, 0), (1, 0))$ and returns its handle.

Scalar Multiplication. When the adversary makes query $\mathsf{scalar\_mult}(h_1, h_2)$, the oracle looks up the records corresponding to h_1, h_2 in the database. If such records exist, we view the first record as matrix type and the second record as scalar type, then the oracle computes product M =trace$\times M_1$. If the database already contains the matrix $(M, (m, LC), (n, RC))$ then oracle returns its handle. Otherwise, the oracle assigns a new handle h' and inserts into the database $(h', M, (m_1, LC_1), (n_1, RC_1))$ and returns the new handle.

Definition of Security Game for Witness Encryption using Multilinear Maps

We define a security game based on the generic cyclic colored matrix model for our witness encryption using multilinear maps. In our witness encryption, a matrix assigned to each edge corresponds to one record in the generic cyclic colored matrix model. To multiply two colored matrices$(h_1, M_1, (m_1, LC_1), (n_1, RC_1))$ and $(h_2, M_2, (m_2, LC_2), (n_2, RC_2))$ by $\mathsf{mult}(h_1, h_2)$ means to make a path by connecting two edges, and if $LC_1 = RC_2$, it means that the start vertex and the goal vertex are the same, in other words, a path becomes a cycle in our scheme. Therefore, in the security game, the representation oracle returns a handle of trace if the result of $\mathsf{mult}(h_1, h_2)$ becomes a cycle. Exceptionally, a random matrix R'_1, which is not a inverse of R_1, is assigned to out-edges of vertex 1, therefore, the left and right colors are different and the oracle does not return a handle of trace in this case.

Details of the Security Game for our Witness Encryption.

Setup. The oracle is given a graph G, it makes matrix type records for all edges in G and inserts into database them and returns their representations $\{(h, (m, LC), (n, RC))\}$ without giving matrix M itself to the adversary. To say more details, the oracle samples two level-0 encoded secrets r_i and s_i by $\mathsf{MM.samp}()$ for each vertex i. For every edge $e_{i\to j}$, it sets the secrets to a adjacent matrix $M_{i\to j}$ where $(M_{i\to j})_{1,1} = r_i$ and $(M_{i\to j})_{i,j} = s_i$, and inserts into the database those records $\{(h, M_{i\to j}, (n, i), (n, j))\}$. Then it gives $\{(h, (n, i), (n, j))\}$ to the adversary.

Addition. When the adversary makes query $\mathsf{add}(h_1, h_2)$, the oracle looks up the records corresponding to h_1, h_2 in the database. If such records exist then, if $(m_1 \neq m_2)$ or $(LC_1 \neq LC_2)$

or $(n_1 \neq n_2)$ or $(RC_1 \neq RC_2)$ then return bot. Otherwise, it adds $M = M_1 + M_2$ and looks up the record $(M, (m, LC), (n, RC))$, if it exists in the database then returns the handle, otherwise the oracle inserts the record $(h, M, (m, LC), (n, RC))$ with new handle h and returns the handle.

Multiplication. When the adversary makes query $\mathsf{mult}(h_1, h_2)$, the oracle looks up the records corresponding to h_1, h_2 in the database. If such records exist, if $n_1 \neq m_2$ or $RC_1 \neq LC_2$ then return bot. Otherwise, it multiplies $M = M_1 M_2$ and looks up the record $(M, (m_1, LC_1), (n_2, RC_2))$, if it exists in the database then returns the handle, otherwise the oracle inserts the record $(h, M, (m_1, LC_1), (n_2, RC_2))$ with new handle h and returns the handle. If the matrices corresponding to h_1, h_2 are both matrix type and they starts and goals at same vertex such that the vertex is not 1 then the oracle computes a trace of M and puts the trace to 1×1 matrix M and inserts into the scalar type record $(h, M, (1, 0), (1, 0))$ and returns its handle to the adversary.

Scalar Multiplication. When the adversary makes query $\mathsf{scalar\_mult}(h_1, h_2)$, the oracle looks up the records corresponding to h_1, h_2 in the database. If such records exist, let first record to be matrix type, second record to be scalar type, then the oracle multiplies $M = (M_2)_{1,1} M_1$ and looks up the record $(M, (m_1, LC_1), (n_1, RC_1))$, if it exists in the database then returns the handle, otherwise the oracle inserts the record $(h, M, (m_1, LC_1), (n_1, RC_1))$ with new handle h and returns the handle.

Proof of Security

Theorem 5 *Our witness encryption based on HC is soundness secure in the generic cyclic colored matrix model.*

More precisely, the probability that an adversary distinguishes a real model and a simulation is bounded above T^3/p from the Schwartz-Zippel lemma [42] [50] where an adversary receives at most T handles from the oracle and the secrets r, s are independently and uniformly chosen from $\mathbb{Z}_p$. T^3 means a number of handles $\times$ a number of combinations of r, s.

Proof. We make simulator which input is a graph which does not have any HC, and show that there is no handle of the matrix corresponding to the "full cycle":$\Pi_{j\in[n]} r_j + \Pi_{j\in[n]} s_j$ in a view of the adversary.

Algorithm of Simulator. We assume there are t short cycles which does include vertex 1 in a given graph G. Let sc be a set of vertices of a short cycle. Input values to a simulator is a graph

G. When simulator needs to return a handle of trace, it calculates a trace from the result matrix, then returns the result's handle.

Notations:

A given graph G has n vertices. A record rec_i denotes the record $(h_i, M_i, (m_i, LC_i), (n_i, RC_i))$ in the database.

Setup. A simulator makes edge matrices for all edges in given graph G, and inserts into the database those edge matrices and sends representations of edge without matrix M to the adversary.

for $i \in [n]$ // all vertices
  create a variable R_i S_i
  for j s.t. edges $i \to j$ are out-edges of vertex i
    create a $n \times n$ matrix $M_{i \to j}$
    $(M_{i \to j})_{1,1} := R_i$ $(M_{i \to j})_{i,j} := S_i$
    $(M_{i \to j})_{x,y} := 0$ if $(x, y) \neq (1, 1)$ and (i, j)
    $h_k :=$ new handle
    db.insert$(h_k, M_{i \to j}, (n, i), (n, j))$
give $\{(h_k, (n, i), (n, j))\}$ to the adversary.

Addition. If types of associated h_1, h_2 are both matrix type, if their start vertices are same and their goal vertices are also same then the simulator adds two matrices and inserts its record into database and returns the handle.

add(h_1, h_2)
  $rec1 =$ db.select(h_1) $rec2 =$ db.select(h_2)
  if $(m_1 \neq m_2)$ or $(LC_1 \neq LC_2)$ or $(n_1 \neq n_2)$ or $(RC_1 \neq RC_2)$ then return $\perp$
  $M = M_1 + M_2$
  $h \leftarrow$ db.select$(M, (m_1, LC_1), (n_1, RC_1))$
  if h does not exist in database then
    $h :=$ new handle
    db.insert$(h, M, (m_1, LC_1), (n_1, RC_1))$

return h

Multiplication. If types of associated h_1, h_2 are both matrix type, if first goal vertex and second start vertices are same, and first col size and second row size are same then the simulator multiplies the two matrices. If start and goal are same but not 1 then it calculates a trace from the result matrix M, and inserts its record into database and returns the handle.

mult(h_1, h_2)
  $rec1$ = db.select(h_1) $rec2$ = db.select(h_2)
  if $(n_1 \neq m_2)$ or $(RC_1 \neq LC_2)$ then return $\perp$
  $M = M_1 \cdot M_2$
  $h \leftarrow$ db.select($M, (m_1, LC_1), (n_2, RC_2)$)
  if h does not exist in database then
    h := new handle
    db.insert($h, M, (m_1, LC_1), (n_2, RC_2)$)
  if types of h_1, h_2 are matrix type and $(LC_1 = RC_2)$ and $(LC_1 \neq 1)$ then
    create 1×1 matrix M_t
    $(M_t)_{1,1} = M_{1,1} + M_{LC_1,RC_2}$
    $h_t \leftarrow$ db.select($M_t, (1, 0), (1, 0)$)
    if h_t does not exist in database then
      h_t := new handle
      db.insert($h_t, M_t, (1, 0), (1, 0)$)
    return (h, h_t)
  else return h

Scalar Multiplication. It multiplies a trace value in scalar type record by a matrix in a matrix type record and inserts the result record into database and returns the handle.

scalar_mult(h_1, h_2)
  $rec1$ = db.select(h_1) $rec2$ = db.select(h_2)
  let $rec1$ be a $n \times n$ edge matrix.

let $rec2$ be a 1×1 trace matrix.

if not above then return $\perp$

$M = (M_2)_{1,1} \cdot M_1$

$h \leftarrow \mathsf{db.select}(M, (m_1, LC_1), (n_1, RC_1))$

if h does not exist in database then

$h :=$ new handle

$\mathsf{db.insert}(h, M, (m_1, LC_1), (n_1, RC_1))$

return h

Analysis. We discuss about difference of distributions of adversary's view between real model and simulation. The simulator replaces random elements of edge matrices into variables, therefore it's possible that result of calculation of the elements is in database at real model but not in database at simulation. In such a case, he returns existing handle at real model and returns new handle at simulation, therefore, the view of adversary is different. But its provability is bounded above T^3/p from the Schwartz-Zippel lemma [42] [50] where an adversary receives at most T handles from the oracle and the secrets r, s are independently and uniformly chosen from any finite set$S$, $|S| = p$.

We show there is no record of our blinding factor in a simulator's database. Now we recall that our blinding factor is concatenation of all elements in the product matrix $R'_1(\Pi_{j\in[n]}r_j + \Pi_{j\in[n]}s_j)R_1$. We confirm about edge matrix records at each query. After setup, value in matrix column is M s.t. $M_{1,1} = R_i$ and $M_{i,j} = S_i$ for every edge $e_{i\to j}$. When two matrices M_1 and M_2 are added by add query, the result matrix M is such that $M_{1,1} = (M_1)_{1,1} + (M_2)_{1,1}$, $M_{i,j} = (M_1)_{i,j} + (M_2)_{i,j}$ and other elements are all zero. When two matrices M_1 and M_2 are multiplied by mult query, if $LC_1 \neq 1'$ or $RC_2 \neq 1$ then the result matrix M is such that $M_{1,1} = (M_1)_{1,1}(M_2)_{1,1}$, $M_{i,k} = (M_1)_{i,j}(M_2)_{j,k}$ and other elements are all zero. If $LC_1 = 1'$ and $RC_2 = 1$ then the result matrix M is such that $M_{1,1} = (M_1)_{1,1}(M_2)_{1,1} + (M_1)_{i,j}(M_2)_{j,k}$ and other elements are all zero. When a trace v and matrix M_1 are multiplied by $\mathsf{scalar\_mult}$, the result matrix M is such that $M_{1,1} = v(M_1)_{1,1}$, $M_{i,j} = v(M_1)_{i,j}$ and other elements are all zero. We notice that elements in M_1 and M_2 in above queries may be multiplied by some traces by $\mathsf{scalar\_mult}$ queries.

Now we consider whether there is a record of matrix M' s.t. $M'_{1,1} = \Pi_{i\in[n]}R_i + \Pi_{i\in[n]}S_i$ and other elements are all zero for our blinding factor.

Claim 4 *In the database, if there is records where $LC = 1'$ and $RC = 1$ and $M_{1,1}$ has some value v and $M_{i,j}$ $(i, j \neq 1)$ is zero then*

$$v = \Sigma_{sc}(\Pi_{sc_i'}(\Pi_{j \in sc_i'} R_j + \Pi_{j \in sc_i'} S_j))(\Pi_{j \in sc} R_j + \Pi_{j \in sc} S_j), \cdots Eq(1)$$

where sc denotes a short cycle which starts and goals at vertex 1, sc' denotes a short cycle which does not include vertex 1.

Proof. A trace is registered into database when a short cycle which does not include vertex 1 is made by mult query. After the trace is registered, a matrix may be scalar multiplied by scalar_mult query. Those matrices may be added and multiplied by add and mult. Using such two matrices, when a short cycle which starts and goals at vertex 1 is made by mult, a record where $LC = 1'$ and $RC = 1$ and $M_{1,1}$ has some value v and $M_{i,j}$ $(i, j \neq 1)$ is zero is registered. The form of v is $v = (\Sigma_{sc}\Pi_{sc_i'}(\Pi_{j \in sc_i'} R_j + \Pi_{j \in sc_i'} S_j))(\Pi_{j \in sc} R_j + \Pi_{j \in sc} S_j)$. From the observation before Claim 4, add and scalar_mult does not change the color and element position of the result matrix, therefore records where $LC = 1'$ and $RC = 1$ must be made from records where $LC = 1'$ and $RC = 1$ by add or scalar_mult. The result $M_{1,1}$ has a form $v = \Sigma_{sc}(\Pi_{sc_i'}(\Pi_{j \in sc_i'} R_j + \Pi_{j \in sc_i'} S_j))(\Pi_{j \in sc} R_j + \Pi_{j \in sc} S_j)$.□

Claim 5 *The Eq(1) never equals to $\Pi_{i \in [n]} R_i + \Pi_{i \in [n]} S_i$.*

Proof. To make $\Pi_{i \in [n]} R_i + \Pi_{i \in [n]} S_i$, it is necessary to multiply one $(\Pi_{j \in sc} R_j + \Pi_{j \in sc} S_j)$ and some $(\Pi_{j \in sc_i'} R_j + \Pi_{j \in sc_i'} S_j)$ because the input graph does not contain any Hamilton Cycle. The result has cross terms of $R_j S_j$, and for canceling this cross terms it needs a new product of $(\Pi_{j \in sc} R_j + \Pi_{j \in sc} S_j)$ and $(\Pi_{j \in sc_i'} R_j + \Pi_{j \in sc_i'} S_j)$ of smaller short cycles. Then the result has their new cross terms. To cancel those cross terms recursively, finally it becomes sums of products of $(\Pi_{j \in sc} R_j + \Pi_{j \in sc} S_j)$ and $(\Pi_{j \in sc_i'} R_j + \Pi_{j \in sc_i'} S_j)$ of minimum short cycles, and their cross terms are unable to be canceled. Therefor the Eq(1) never equals to $\Pi_{i \in [n]} R_i + \Pi_{i \in [n]} S_i$. □

Now the proof of Theorem 2 is complete. □

Chapter 6

Conclusions

Cryptography is now required to have the ability not only to hide secret data but also to gain added-value by analyzing or calculating the hidden secret safely. We studied homomorphic property that enables us to perform arithmetic calculation through privacy-preserving encodings of secrets. Our research objects are homomorphic encryption as an encryption scheme which recovers calculated secret, and multilinear maps as an encoding scheme which outputs a deterministic value corresponding to calculation result.

One of reasons why computation of homomorphic encryption is very heavy is its multi-dimensional slot structure based on the cyclotomic ring. We improved the slot structure by making it one dimensional by using the decomposition ring. We constructed the BGV and FV homomorphic schemes on the decomposition ring. We confirmed that our schemes are several times faster than HElib which is based on cyclotomic ring by some experiments. This simple slot structure based on decomposition ring is effective to parallel processing in bootstrap, we constructed fast bootstrap method for our scheme.

We studied multilinear map, in particular graded encoding system with useful properties: to extract a deterministic value at only a specific level and one-wayness regarding the level of encodings. We proposed two applications using multilinear maps: a group key exchange and witness encryption based on Hamilton Cycle Problem. The multi-linearity makes our group key exchange be constant communication size and constant computational cost with respect to the number of parties.

Bibliography

[1] S. Arita and S. Handa, “Subring Homomorphic Encryption,” Information Security and Cryptology - ICISC 2017, ed. H. Kim and D.C. Kim, Cham, pp.112-136, Springer International Publishing, 2018.

[2] M.R. Albrecht, “On Dual Lattice Attacks Against Small-Secret LWE and Parameter Choices in HElib and SEAL,” Advances in Cryptology - EUROCRYPT 2017, ed. J.S. Coron and J.B. Nielsen, Cham, pp.103-129, Springer International Publishing, 2017.

[3] J. Alperin-Sheriff and C. Peikert, “Practical Bootstrapping in Quasilinear Time,” Advances in Cryptology - CRYPTO 2013, ed. R. Canetti and J.A. Garay, Berlin, Heidelberg, pp.1-20, Springer Berlin Heidelberg, 2013.

[4] R.P. Martin R. Albrecht and S. Scott, “On the concrete hardness of Learning with Errors,” Journal of Mathematical Cryptology. Volume 9, Issue 3, pp.169-203, 2015.

[5] Z. Brakerski, “Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP,” Advances in Cryptology - CRYPTO 2012, ed. R. Safavi-Naini and R. Canetti, Berlin, Heidelberg, pp.868- 886, Springer Berlin Heidelberg, 2012.

[6] Z. Brakerski, C. Gentry and V. Vaikuntanathan, “(Leveled) Fully Homomorphic Encryption Without Bootstrapping,” Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS ’12, New York, NY, USA, pp.309-325, ACM, 2012.

[7] Z. Brakerski and V. Vaikuntanathan, “Efficient Fully Homomorphic Encryption from (Standard) LWE,” Proceedings of the 2011 IEEE 52Nd Annual Symposium on Foundations of Computer Science, FOCS ’11, Washington, DC, USA, pp.97-106, IEEE Computer Society, 2011.

[8] E. Bresson, O. Chevassut, D. Pointcheval and J. J. Quisquater, "Provably Authenticated Group Diffie-Hellman Key Exchange," In proceedings of 8th ACM Conference on CCS E. , pp. 255-264, 2001.

[9] D. Boneh and A. Silverberg, "Applications of multilinear forms to cryptography," Contemporary Mathematics, 324:71-90, 2003. 1, 2, 5, 10, 74, 75.

[10] J. H, Cheon, A. Kim, M. Kim and Y. Song, "Homomorphic Encryption for Arithmetic of Approximate Numbers," International Conference on the Theory and Application of Cryptology and Information Security, pp.409-437, Springer, 2017.

[11] J. H. Cheon, M. Kim and K. Lauter, "Homomorphic Computation of Edit Distance," Financial Cryptography and Data Security 2015, LNCS 8976, pp.194-212, 2015.

[12] H. Chen, K. Laine, P. Rindal, "Fast Private Set Intersection from Homomorphic Encryption," Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 1243-1255, ACM, 2017.

[13] J.S. Coron, T. Lepoint and M. Tibouchi, "Practical Multilinear Maps over the Integers," CRYPTO 2013, pp.476-493.

[14] E. Crockett and C. Peikert, "Λ o λ : A Functional Library for Lattice Cryptography," IACR Cryptology ePrint Archive, 2015-1134, 2015.

[15] A. Costache and N. P. Smart, “Which Ring Based Somewhat Homomorphic Encryption Scheme is Best?,” Proceedings of the RSA Conference on Topics in Cryptology - CT-RSA 2016 - Volume 9610, New York, NY, USA, pp.325-340, Springer-Verlag New York, Inc., 2016.

[16] J. Fan and F. Vercauteren, “Somewhat Practical Fully Homomorphic Encryption,” Cryptology ePrint Archive, Report 2012/144, 2012. https://eprint.iacr.org/2012/144.

[17] I. Damgård, V. Pastro, N. Smart and S. Zakarias, “Multiparty Computation from Somewhat Homomorphic Encryption,” Annual Cryptology Conference, Springer, pp.643-662, 2012.

[18] S. Garg, C. Gentry and S. Halevi, "Candidate Multilinear Maps from Ideal Lattices and Applications," In EUROCRYPT 2013, Lecture Notes in Computer Science. Springer, 2013. Cryptology ePrint Archive, Report 2012/610.

[19] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai and B. Waters, "Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits," FOCS 2013, pp.40-49.

[20] S. Garg, C. Gentry, S. Halevi, A. Sahai and B. Waters, "Attribute-based encryption for circuits from multilinear maps," Cryptology ePrint Archive, Report 2013/128, 2013. http://eprint.iacr.org/.

[21] S. Garg, C. Gentry, A. Sahai and B. Waters, "Witness Encryption and its Applications," STOC '13, Proceedings of the forty-fifth annual ACM symposium on Theory of Computing, pp.467-476.

[22] C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing, STOC '09, New York, NY, USA, pp.169-178, ACM, 2009.

[23] C. Gentry, S. Halevi and N.P. Smart, "Better Bootstrapping in Fully Homomorphic Encryption," Public Key Cryptography – PKC 2012, ed. Fischlin, Marc and Buchmann, Johannes and Manulis, Mark, Berlin, Heidelberg, pp.1-16, Springer Berlin Heidelberg, 2012.

[24] C. Gentry, S. Halevi and N.P. Smart, "Homomorphic Evaluation of the AES Circuit," Advances in Cryptology - CRYPTO 2012, ed. R. Safavi-Naini and R. Canetti, Berlin, Heidelberg, pp.850-867, Springer Berlin Heidelberg, 2012.

[25] C. Gentry, S. Halevi and N.P. Smart, "Fully Homomorphic Encryption with Polylog Overhead," Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp.465-482, Springer, 2012.

[26] T. Graepel, K. Lauter and M. Naehrig, "ML Confidential: Machine Learning on Encrypted Data," Information Security and Cryptology - ICISC 2012, ed. T. Kwon, M.K. Lee, and D. Kwon, Berlin, Heidelberg, pp.1-21, Springer Berlin Heidelberg, 2013.

[27] S. Halevi and V. Shoup, "Algorithms in HElib," Advances in Cryptology - CRYPTO 2014, ed. J.A. Garay and R. Gennaro, Berlin, Heidelberg, pp.554-571, Springer Berlin Heidelberg, 2014.

[28] S. Halevi and V. Shoup, “Bootstrapping for HElib,” Advances in Cryptology - EUROCRYPT 2015, ed. E. Oswald and M. Fischlin, Berlin, Heidelberg, pp.641-670, Springer Berlin Heidelberg, 2015.

[29] S. Halevi and V. Shoup, “Faster Homomorphic Linear Transformations in HElib,” Annual International Cryptology Conference, pp.93-120, Springer, 2018.

[30] R.M. Karp, "Reducibility among Combinatorial Problems," In Complexity of Computer Computations, pp.85-103, 1972.

[31] J. Kilian, "Founding cryptography on oblivious transfer," In Janos Simon, editor, STOC, pp.20-31. ACM, 1988.

[32] D. Kim and Y. Song, “Approximate Homomorphic Encryption over the Conjugate-Invariant Ring,” Information Security and Cryptology - ICISC 2018, ed. K. Lee, Cham, pp.85-102, Springer International Publishing, 2019.

[33] T. Lepoint, "Proof-of-concept implementation of a cryptographic multilinear maps on the integers," https://github.com/tlepoint/multimap.

[34] J. Liu, J. Li, S. Xu and B.C. Fung, “Secure Outsourced Frequent Pattern Mining by Fully Homomorphic Encryption,” Big Data Analytics and Knowledge Discovery, ed. S. Madria and T. Hara, Cham, pp.70-81, Springer International Publishing, 2015.

[35] K. Lauter, A. Lopez-Alt and M. Naehrig, “Private Computation on Encrypted Genomic Data,” Progress in Cryptology - LATINCRYPT 2014, ed. D.F. Aranha and A. Menezes, Cham, pp.3-27, Springer International Publishing, 2015.

[36] W. Lu, S. Kawasaki and J. Sakuma, "Using Fully Homomorphic Encryption for Statistical Analysis of Categorical, Ordinal and Numerical Data," Network and Distributed System Security Symposium (NDSS) in February, 2017.

[37] K. Laine and R. Player, "Simple Encrypted Arithmetic Library - SEAL(v2.0)," Technical report, Microsoft Research, September 2016. MSR-TR-2016-52.

[38] V. Lyubashevsky, C. Peikert, and O. Regev, “On Ideal Lattices and Learning with Errors over Rings,” Advances in Cryptology EUROCRYPT 2010, ed. H. Gilbert, Berlin, Heidelberg, pp.1-23, Springer Berlin Heidelberg, 2010.

[39] V. Lyubashevsky, C. Peikert and O. Regev, "A Toolkit for Ring-LWE Cryptography," EUROCRYPT 2013, LNCS 7881, pp 35-54, Springer, 2013.

[40] C. M. Mayer, "Implementing a Toolkit for Ring-LWE Based Cryptography in Arbitrary Cyclotomic Number Fields," Cryptology ePrint Archive, Report 2016/049, 2016. http://eprint.iacr.org/2016/049.

[41] O.Regev, "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography," In Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC ' 05) Harold N. Gabow and Ronald Fagin Eds., ACM, 84-93, 2005.

[42] T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities," J. ACM, 27(4):pp.701-717, 1980.

[43] A. Shamir, "How to share a secret," Communications of the ACM, Vol.22, no.11, pp.612-613, 1979.

[44] V. Shoup, "Sequences of Games: A Tool for Taming Complexity in Security Proofs," Cryptology ePrint Archive, Report 2004/332, 2004. http://eprint.iacr.org/ 2004/332.pdf. 42, 59, 60, 206, 209.

[45] N.P. Smart and F. Vercauteren, " Fully homomorphic SIMD operations, " Designs, Codes and Cryptography, vol.71, no.1, pp.57-81, Apr 2014.

[46] M. Steiner, G. Tsudik and M. Waidner, "Diffie-Hellman Key Distribution Extended to Group Communication," In Proceedings of the 3rd ACM Conference on Computer and Communications Security (CCS ' 96), pp.31.37. ACM Press, 1996. 13, 92, 93, 96, 104.

[47] W. Henecka, S. Kögl, A.-R. Sadeghi, T. Schneider and I. Wehrenberg, "TASTY: Tool for Automating Secure Two-partY computations," Proceedings of the 17th ACM conference on Computer and communications security, pp.451-462, ACM, 2010.

[48] S.Terada, H.Nakano, S.Okumura and A.Miyaji, "On the security of Ring-LWE with Homomorphic Encryption," SCIS2018.

[49] A.C. Yao, "How to generate and exchange secrets," 27th Annual Symposium on Foundations of Computer Science, pp.162-167, 1986.

[50] R. Zippel, "Probabilistic algorithms for sparse polynomials," In E. W. Ng, editor, EUROSAM, volume 72 of Lecture Notes in Computer Science, pp. 216-226.Springer, 1979.