이학 박사 학위논문

Homomorphic Encryption for Approximate Arithmetic

(근사계산을 위한 동형암호 구축에 관한 연구)

2018년 2월

서울대학교 대학원

수리과학부

송용수

Homomorphic Encryption for Approximate Arithmetic

(근사계산을 위한 동형암호 구축에 관한 연구)

지도교수 천정희

이 논문을 이학 박사 학위논문으로 제출함

2017년 10월

서울대학교 대학원

수리과학부

송용수

송용수의 이학 박사 학위논문을 인준함

2017년 12월

위 원 장	김 명 환	(인)
부 위 원 장	천 정 희	(인)
위 원	김 영 훈	(인)
위 원	현 동 훈	(인)
위 원	서 재 홍	(인)

Homomorphic Encryption for Approximate Arithmetic

A dissertation
submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
to the faculty of the Graduate School of
Seoul National University

by

Yongsoo Song

Dissertation Director : Professor Jung Hee Cheon

Department of Mathematical Sciences
Seoul National University

February 2018

Abstract

Homomorphic Encryption for Approximate Arithmetic

Yongsoo Song
Department of Mathematical Sciences
The Graduate School
Seoul National University

Homomorphic encryption is a cryptosystem which allows us to perform an arithmetic of encrypted data. The technology of homomorphic encryption has a tremendous possibilities in real world applications based on secure outsourcing of computation on public server. However, previous schemes had a common limitation in approximate arithmetic such as real number operations.

In this paper we suggest a method to construct a homomorphic encryption scheme for approximate arithmetic. It supports an approximate addition and multiplication of encrypted messages, together with a new *rescaling* procedure for managing the magnitude of plaintext. The main idea is to add a noise following significant figures which contain a main message. This noise is originally added to the plaintext for security, but considered to be a part of error occurring during approximate computations that is reduced along with plaintext by rescaling. Consequently, the bit size of ciphertext modulus grows linearly with the depth of the circuit being evaluated due to rescaling procedure, compared to an exponentially large size of previous works. We also propose a new batching technique for a RLWE-based construction. A plaintext polynomial will be mapped to a vector of complex numbers via complex canonical embedding map, which is an isometric ring homomorphism. We show that our scheme can be applied to the efficient evaluation of circuits of approximate numbers including transcendental functions such as multiplicative inverse, exponential function, logistic function and discrete Fourier transform.

We extend the leveled homomorphic encryption scheme into a fully homomorphic encryption. Namely, we propose a new technique to refresh low-level ciphertexts based on Gentry's bootstrapping process. The bootstrapping procedure is required to evaluate the decryption formula homomorphically using arithmetic operations over the integers, and the modular reduction becomes the main bottleneck in bootstrapping. We exploit a scaled sine function as an approximation of the modular reduction circuit and present an efficient strategy to evaluate trigonometric functions recursively. Our method requires only two homomorphic multiplications at each iteration and the computation cost grows linearly with the depth of the decryption formula. We also show how to bootstrap packed ciphertexts on RLWE construction with a proof of concept implementation.

We prove the efficiency of our scheme by applying it to real-world applications. Specifically we use our open source homomorphic encryption library to learn a model of logistic regression using biomedical data. We show that our scheme can evaluate the gradient descent method with $20 \sim 25$ iterations in a few hours.

Key words: homomorphic encryption, approximate arithmetic, bootstrapping, logistic regression

Student Number: 2012-23025

Contents

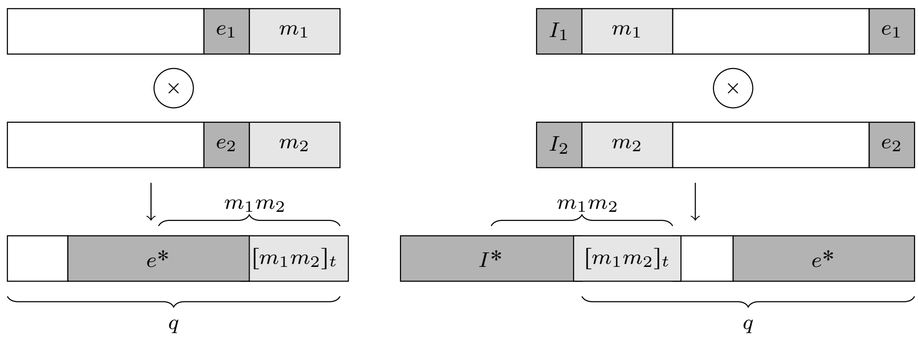
Chapter 1

Introduction

Homomorphic encryption (HE) is a cryptographic scheme that enables homomorphic operations on encrypted data without decryption. Many of HE schemes (e.g. [DGHV10, BV11a, BV11b, Bra12, BGV12, GHS12b, LATV12, BLLN13, GSW13, CLT14, CS15, DM15, DHS16]) have been suggested following Gentry's blueprint [Gen09]. HE can be applied to the evaluation of various algorithms on encrypted financial, medical, or genomic data [NLV11, LLAN14, CKL15, WZD$^+$16, KSC17].

Most of real-world data contain some errors from their true values. For instance, a measured value of quantity has an observational error from its true value and sampling error can be made as only a sample of the whole population is being observed in statistics. In practice, data should be discretized (quantized) to an approximate value (e.g. floating-point number), in order to be represented by a finite number of bits in computer systems. In this case, an approximate value may substitute the original data and a small rounding error does not have too much effect on computation result. For the efficiency of approximate arithmetic, we store a few numbers of significant digits (e.g. most significant bits, MSBs) and carry out arithmetic operations between them. The resulting value should be rounded again by removing some inaccurate least significant bits (LSBs) to maintain the bit size of significand (mantissa).

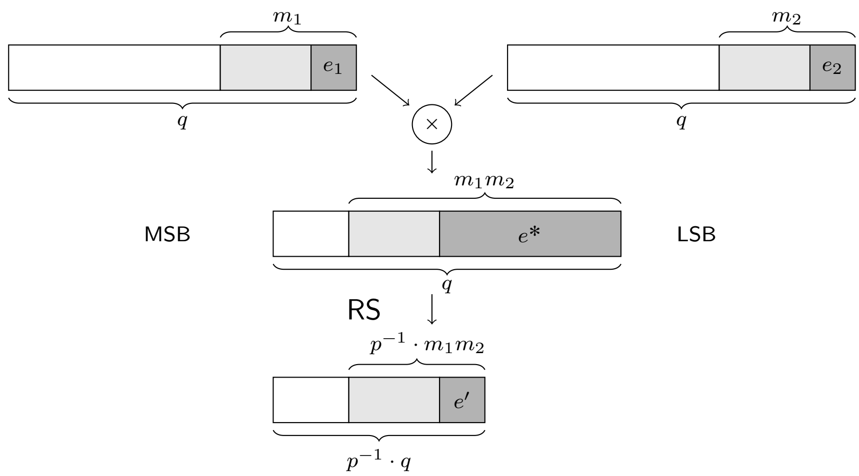
Figure 1.1: Multiplications of BGV-type (left) and FV-type (right) schemes



Unfortunately this rounding operation has been considered difficult to perform on HE since it is not simply represented as a small-degree polynomial. Previous approaches to approximate arithmetic require similar multiplicative depth and complexity to the case of bootstrapping for extraction of MSBs [AN16, JA16]. Other methods based on exact integer operations [DGBL$^+$17, CSVW16] require an exponentially large bit size of ciphertext modulus with the depth of the circuit to ensure correctness.

We point out that the decryption structures of existing HE schemes are not appropriate for arithmetic of indiscrete spaces. For a plaintext modulus t and a ciphertext modulus q, BGV-type HE schemes [BGV12, GHS12b, LATV12, DHS16] have a decryption structure of the form $\langle \mathbf{c}_i, sk\rangle = m_i + te_i \pmod q$. Therefore, the MSBs of m_1+m_2 and m_1m_2 are destroyed by inserted errors e_i during homomorphic operations. On the other hand, the decryption structure of FV-type HE schemes [Bra12, FV12, BLLN13] is $\langle \mathbf{c}_i, sk\rangle = qI_i + (q/t)m_i + e_i$ for some I_i and e_i. Multiplication of two ciphertexts satisfies $\langle \mathbf{c}^*, sk\rangle = qI^* + (q/t)m_1m_2 + e^*$ for $I^* = tI_1I_2 + I_1m_2 + I_2m_1$ and $e^* \approx t(I_1e_2+I_2e_1)$, so the MSBs of resulting message are also destroyed (see Fig.1.1 for an illustration). HE schemes with matrix ciphertexts [GSW13, DM15] support homomorphic operations over the integers (or integral polynomials) but the error growth depends on the size of plaintexts.As a result, previous HE schemes are required to have an exponentially large ciphertext modulus with the depth of a circuit for approximate arithmetic.

Figure 1.2: Homomorphic multiplication and rescaling



1.1 Homomorphic Encryption for Approximate Arithmetic

The purpose of this paper is to present a method for efficient approximate computation on HE. The main idea is to treat an encryption noise as part of error occurring during approximate computations. That is, an encryption $\mathbf{c}$ of message m by the secret key sk will have a decryption structure of the form $\langle \mathbf{c}, sk \rangle = m + e \pmod q$ where e is a small error inserted to guarantee the security of hardness assumptions such as the learning with errors (LWE), the ring- LWE (RLWE) and the NTRU problems. If e is small enough compared to the message, this noise is not likely to destroy the significant figures of m and the whole value $m' = m + e$ can replace the original message in approximate arithmetic. One may multiply a scale factor to the message before encryption to reduce the precision loss from encryption noise.

For homomorphic operations, we always maintain our decryption structure small enough compared to the ciphertext modulus so that computation result is still smaller than q. However, we still have a problem that the bit size of message increases exponentially with the depth of a circuit without rounding. To address this problem, we suggest a new technique - called *rescaling* - that manipulates the message of ciphertext. Technically it seems similar to the modulus-switching method suggested by Brakerski and Vaikuntanatan [BV11a], but it plays a completely different role in our construction. For an encryption $\mathbf{c}$ of m such that $\langle \mathbf{c}, sk \rangle = m+e \pmod q$, the rescaling procedure outputs a ciphertext $\lfloor p^{-1} \cdot \mathbf{c} \rceil \pmod{q/p}$, which is a valid encryption of m/p with noise about e/p. It reduces the size of ciphertext modulus and consequently removes the error located in the LSBs of messages, similar to the rounding step of fixed/floating-point arithmetic, while almost preserving the precision of plaintexts.

The composition of homomorphic operation and rescaling mimics the ordinary approximate arithmetic (see Fig.1.2). As a result, the bit size of a required ciphertext modulus grows linearly with the depth of a circuit rather than exponentially. We also prove that this scheme is almost optimal in the sense of precision: precision loss of a resulting message is at most one bit more compared to unencrypted floating-point arithmetic.

1.1.1 Packed Ciphertext

It is inevitable to encrypt a vector of multiple plaintexts in a single ciphertext for efficient homomorphic computation. The plaintext space of previous RLWE-based HE schemes is a cyclotomic polynomial ring $\mathbb{Z}_t[X]/(\Phi_M(X))$ of a finite characteristic. A plaintext polynomial could be decoded as a vector of plaintext values into a product of finite fields by a ring isomorphism [SV10, SV14]. An inserted error is placed separately from the plaintext space so it may be removed by using plaintext characteristic after carrying out homomorphic operations.

On the other hand, a plaintext of our scheme is an element of a cyclotomic ring of characteristic zero and it embraces a small error which is inserted from encryption to ensure the security or occurs during approximate arithmetic. Hence we adapt an *isometric ring homomorphism* - the complex canonical embedding map. It preserves the size of polynomials so that a small error in a plaintext polynomial is not blow up during encoding/decoding procedures.

1.1.2 Evaluation of Circuits

One important feature of our method is that the precision loss during homomorphic evaluation is bounded by depth of a circuit and it is at most one more bit compared to unencrypted approximate arithmetic. Given encryptions of d messages with η bits of precision, our HE scheme of depth $\lceil \log d \rceil$ computes their product with $(\eta - \log d - 1)$ bits of precision in d multiplications while unencrypted approximate arithmetic such as floating-point multiplication can compute a significand with $(\eta - \log d)$ bits of precision. On the other hand, the previous methods require $\Omega(\eta^2 d)$ homomorphic computations by using bitwise encryption or need a large plaintext space of bit size $\Omega(\eta d)$ unless relying on expensive computations such as bootstrapping or bit extraction.

In our scheme, the required bit size of the largest ciphertext modulus can be reduced down to $O(\eta \log d)$ by performing the rescaling procedure after multiplication of ciphertexts. The parameters are smaller than for the previous works and this advantage enables us to efficiently perform the approximate evaluation of *transcendental* functions such as the exponential, logarithm and trigonometric functions by the evaluation of their Taylor series expansion. In particular, we suggest a specific algorithm for computing the multiplicative inverse with reduced complexity, which enables the efficient evaluation of rational functions.

1.1.3 Library

We provide an open-source implementation of our HE library (`HEAAN`) and algorithms in the C++ language. The source code is available at github [CKKS16]. We introduced `HEAAN` at a workshop for the standardization of HE hosted by Microsoft Research.*

*https://www.microsoft.com/en-us/research/event/homomorphic-encryption-standardization-workshop/

1.2 Bootstrapping

Our scheme has an advantage from the rescaling procedure for management of the plaintext magnitude. The required bit size of a ciphertext modulus can be reduced from $\Omega(2^L)$ down to $\mathcal{O}(L)$ where L is the depth of a circuit. However, it is a *leveled* HE scheme and can only evaluate a circuit of a fixed depth. Without bootstrapping, a ciphertext modulus decreases as computation progresses, so the HE scheme can no longer support any homomorphic operation at the lowest level.

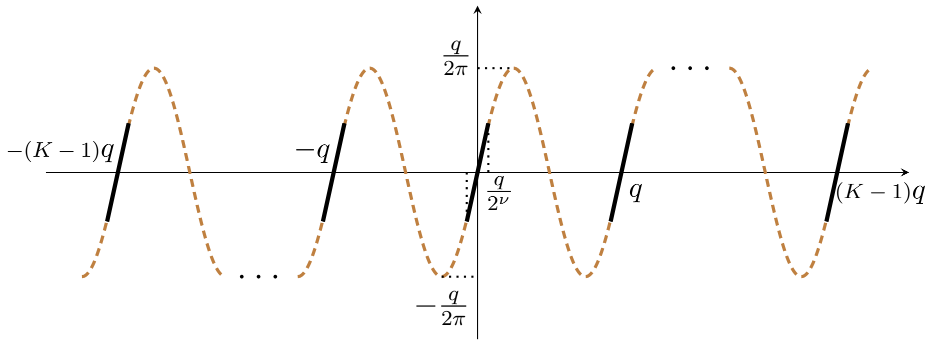
The bootstrapping procedure of the existing HE schemes can be understood as a homomorphic evaluation of decryption circuit. For example, the RLWE-based HE schemes have a common decryption structure $\langle \mathbf{c}, sk\rangle$. The BGV type schemes [BGV12, GHS12b] support an addition and multiplication with the reduction to a plaintext modulus, and they have a decryption circuit of the form $m = [[\langle \mathbf{c}, sk\rangle]_q]_t$ for the plaintext modulus t. To homomorphically evaluate the decryption circuit in a larger ciphertext modulus, they choose a temporary plaintext modulus close to q to simplify the modular reduction operation and represent the decryption formula as a lower degree polynomial over the plaintext space [GHS12a, HS15].

In the case of our HE scheme, it does not support any modulus reduction operation and this makes the bootstrapping much harder. To homomorphically evaluate the decryption procedure $[\langle \mathbf{c}, sk\rangle]_q$, we need to represent even the modulus reduction step $[\cdot]_q$ as a polynomial over the integers. For example, one of the naive approach for expression of a modular reduction is to use the polynomial interpolation of the modulus operation over the domain of $z = \langle \mathbf{c}, sk\rangle$, but it is a limiting factor for practical implementation due to its depth and complexity of evaluation.

1.2.1 Decryption Formula

We present a methodology to refresh a ciphertext of the `HEAAN` scheme and allows the evaluation of an arbitrary circuit. We take advantage of its intrin-

Figure 1.3: Modular reduction and scaled sine functions



sic characteristic - *approximate* computations on encrypted data. Since the decryption structure is already contains some error following the significant figures of a plaintext, the goal of bootstrapping is to evaluate the decryption formula approximately and compute an encryption of the original message in a large ciphertext modulus. Hence, the bootstrapping process can be reduced to a problem that finds an approximate circuit of the modular reduction. The approximation error should be small enough to preserve the precision of an input plaintext.

We first note that the modular reduction $F(z) = [z]_q$ is the identity function nearby zero and it is periodic with period q. If $\langle \mathbf{c}, sk\rangle$ is close to a multiple of q, a trigonometric function is a good candidate of approximation to the modular reduction. Namely, the decryption formula of `HEAAN` can be represented using a scaled *sine* function as

$$[\langle \mathbf{c}, sk\rangle]_q = \frac{q}{2\pi} \cdot \sin\left(\frac{2\pi}{q} \cdot \langle \mathbf{c}, sk\rangle\right) + O(\epsilon^3 \cdot q),$$

when $|[\langle \mathbf{c}, sk\rangle]_q| \leqslant \epsilon \cdot q$. Hence we may use the scaled sine function instead of the modular reduction in decryption formula.

Now our goal is to homomorphically evaluate the trigonometric function $\frac{q}{2\pi} \cdot \sin\left(\frac{2\pi}{q} \cdot z\right)$ with an input $z = \langle \mathbf{c}, sk\rangle$, which is bounded by Kq for some constant $K = \mathcal{O}(\lambda)$ and a security parameter λ. In order to reduce the computation cost, we exploit the following identities

$$\cos 2\theta = \cos\theta^2 - \sin^2\theta, \ \sin 2\theta = 2\cos\theta\sin\theta.$$

It means that we can obtain some approximate values of $\cos 2\theta$ and $\sin 2\theta$ from approximate values of $\cos\theta$ and $\sin\theta$. By adapting this relation repeatedly, we can get an approximate value of $\sin(2^t \cdot \theta)$ from approximations of $\cos\theta$ and $\sin\theta$. From this point, the required number of homomorphic multiplications for the evaluation can be reduced from $\mathcal{O}(\sqrt{Kq})$ down to $\mathcal{O}(\log(Kq))$. For the efficient evaluation of a trigonometric function, we first compute the Taylor expansion of $\frac{q}{2\pi}\cos\left(\frac{2\pi}{q}\cdot\frac{z}{2^t}\right)$ and $\frac{q}{2\pi}\sin\left(\frac{2\pi}{q}\cdot\frac{z}{2^t}\right)$ of degree d_0. The choice of $d_0 = \Omega(Kq/2^t)$ is enough because $\frac{z}{2^t}$ belongs to the small interval $(-Kq/2^t, Kq/2^t)$. After that, we recursively repeat the above equation t times to get an approximate value of $\frac{q}{2\pi}\sin\left(\frac{2\pi}{q}z\right)$.

1.2.2 Results

Our bootstrapping method in HE for arithmetic of approximate number is a kind of new primitive. For a ciphertext $\mathbf{c}$ with a modulus q, our bootstrapping procedure generates a ciphertext $\mathbf{c}'$ with a larger modulus $Q \gg q$ satisfying the condition $[\langle \mathbf{c}', sk\rangle]_Q \approx [\langle \mathbf{c}, sk\rangle]_q$ while keeping an error small enough not to destroy the significant digits of a plaintext. The resulting ciphertext will have enough large modulus compared to a plaintext so that more homomorphic operations can be performed.

It is difficult to make a fair comparison with previous works [HS15], but it seems reasonable to see the depth and complexity of recryption in terms of the precision of a plaintext. So let us compare our bootstrapping process for $\log T$ significant bits with the previous bootstrapping for a HE scheme with a plaintext space modulo T. To preserve $\log T$ precision bits of an input message during our bootstrapping, our revised decryption formula is required to have a depth $2\log K + \frac{3}{2}\log T$. When we use the HElib method to bootstrap a ciphertext with $\log T$-bit plaintext modulus, the required depth becomes

$\log K + 2\log T$. Therefore the depth will be smaller compared to previous works when the plaintext modulus or the number of precision bits is larger than $2\log K$.

We exploit a trigonometric function, instead of polynomial interpolation, to reduce the complexity of bootstrapping process. We also make the use of evaluation method based on some identities of trigonometric functions. For the depth L of a decryption circuit, we could perform the bootstrapping procedure in $\mathcal{O}(L)$ multiplications, while the previous methods require $\mathcal{O}(L^2)$ multiplications to extract some digits recursively.

We also give specific implementation results to prove the performance of our bootstrapping. We apply the method of linear transformation in [HS15] for recryption of fulled packed ciphertexts and several optimization techniques. When we want to preserve 12 bits of precision of a message and use a single slot, our bootstrapping takes about 30 seconds. We also optimize the linear transform for sparsely packed ciphertexts and it takes about 140 seconds to recrypt a ciphertext that encrypts 128 complex numbers in the slots.

1.2.3 Implications of our bootstrapping method

One of the most prominent features of an approximate arithmetic is that every number contains an error that may increase as the computation progresses. The precision of a number is reduced by about one bit after multiplication, and finally we may not extract any meaningful information from the computation result if the depth of a circuit is larger than the bit precision of the input data. Meanwhile, the purpose of a bootstrapping is to construct a HE scheme for an arbitrary circuit. We refresh the ciphertexts in order to keep computing on encrypted data without any limitation on the depth of a circuit. This concept of *unlimited* computation may seem to contradict to the property of finite precision in approximate computation.

However, it can turn out a lot better in real-world applications which have a property of negative feedback (error correction). For example, a cyber-physical system (CPS) is a compromised mechanism of physical and computational components. A computational element commutes with the sensors and every signal contains a small error. The correctness of a CPS is guaranteed when it is stable because an error disappears as time goes on. Another example is gradient descent method, which is the most widely used algorithm for computation of a local minimum point. It has a number of applications in machine learning such as logistic regression and principal component analysis. Since it computes the gradient of a point to move it closer to an optimal point, a noise is not amplified during evaluation and the effect disappears after some iterations.

As in the above examples, we can continue without worrying about the precision of numbers when the overall system is stable. In fact, there are some proof-of-concept implementations about secure control of cyber-physical system [KLS$^+$16] and privacy-preserving logistic regression of biomedical data [KSW$^+$17]. We expect that our bootstrapping process can be applied to these real world applications.

1.3 Machine Learning

Our HE scheme for an approximate arithmetic has tremendous possibilities in real-world applications, especially when they require some real number operations. We take the logistic regression as a typical example that has been considered impractical over homomorphic encryption.

Biomedical data are highly sensitive and often contain important personal information about individuals. In the United States, healthcare data sharing is protected by Health Insurance Portability and Accountability Act (HIPAA) [TJ12] while biomedical research practitioners are covered under federal regulation governing the “Common Rule”, a federal policy that protects people who volunteer for federally funded research studies [Rou17]. These policies set high standards on the protection of biomedical data and violations will lead to financial penalties and lost reputation. On the other hand, cloud computing, which significantly simplifies IT environments, is the trend for data management and analysis. According to a recent study by Microsoft, nearly a third of organizations work with four or more cloud vendors [Mic16]. The privacy concern, therefore, becomes a major hurdle for medical institutions to outsource data and computation to the commercial cloud. It is imperative to develop advanced mechanisms to assure the confidentiality of data to support secure analysis in the cloud environment.

An intuitive solution is to train a model without accessing the data and only obtain the estimated model parameters in a global manner. Assuming summary statistics can be shared, this can be done in a joint manner and we have developed the Grid Logistic Regression [WJKOM12, JLW+13, WJW+13] to show the feasibility of estimating the global parameters from distributed sources (e.g. by only exchange gradients and Hessian matrices). But there are still vulnerabilities in sharing even the summary statistics, for example, the difference in averaged age between a cohort of n patients and another cohort of $(n-1)$ overlapped patients can reveal the actual age of a single patient.

1.4 List of Papers

This thesis contains the results of the following papers. The main contribution is obtained jointly with Jung Hee Cheon, Andrey Kim, Miran Kim [CKKS17], which was presented in ASIACRYPT'17. Koohyung Han joined the development of bootstrapping technique for our scheme that originally appeared in [CHK+17] and it will appear in EUROCRYPT'18. Our source code for the `HEAAN` library is available at `github` [CKKS16]. The first application of our scheme was shown in [KLS+16], which is a joint work with Control & Dynamic Systems Lab of Prof. Hyungbo Shim. Kim et al. [KSW+17] shows an application of `HEAAN` to privacy-preserving logistic regression. This paper was done while Yongsoo Song was employed as an intern in the Division of Biomedical Informatics at University of California, San Diego, and it was accepted in JMIR Medical Informatics.

- [CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song: Homomorphic Encryption for Arithmetic of Approximate Numbers. In International Conference on the Theory and Application of Cryptology and Information Security, pages 409–437. Springer, 2017.

- [CKKS16] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song: Implementation of `HEAAN`, `https://github.com/kimandrik/HEAAN`, 2016.

- [CHK+17] Jung Hee Cheon, Koohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song: Bootstrapping for Approximate Number Homomorphic Encryption. To appear in EUROCRYPT'18.

- [KLS+16] Junsoo Kim, Chanhwa Lee, Hyungbo Shim, Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song: Encrypting Controller using Fully Homomorphic Encryption for Security of Cyber-Physical Systems. IFAC-PapersOnLine, 49(22):175-180, 2016.

- [KSW+17] Miran Kim, Yongsoo Song, Shuang Wang, Yuhou Xia, and Xiaoqian Jiang, Privacy-preserving Logistic Regression based on Homomorphic Encryption. JMIR Med Inform (forthcoming). doi:10.2196/medinform.8805, 2018.

Chapter 2

Preliminaries

2.1 Notation

All logarithms are base 2. We denote vectors in bold, e.g. $\mathbf{a}$, and every vector will be a column vector. We denote by $\langle \cdot, \cdot \rangle$ the usual dot product of two vectors. For a real number r, $\lfloor r \rceil$ denotes the nearest integer to r, rounding upwards in case of a tie. For an integer q, we identify $\mathbb{Z} \cap (-q/2, q/2]$ as a representative of $\mathbb{Z}_q$ and use $[a]_q$ to denote the reduction of the integer a modulo q into that interval. We use $x \leftarrow D$ to denote the sampling x according to a distribution D. It denotes the sampling from the uniform distribution over D when D is a finite set. We let λ denote the security parameter throughout the paper: all known valid attacks against the cryptographic scheme under scope should take $\Omega(2^\lambda)$ bit operations.

2.2 The Cyclotomic Ring

For a positive integer M, let $\Phi_M(X)$ be the M-th cyclotomic polynomial of degree $N = \phi(M)$. Let $\mathcal{R} = \mathbb{Z}[X]/(\Phi_M(X))$ be the ring of integers of a number field $\mathbb{Q}[X]/(\Phi_M(X))$. We write $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ for the residue ring of $\mathcal{R}$ modulo an integer q. An arbitrary element of the cyclotomic ring $\mathcal{P} = \mathbb{R}[X]/(\Phi_M(X))$ of real polynomials will be represented as a polynomial

$a(X) = \sum_{j=0}^{N-1} a_j X^j$ of degree less than N and identified with its coefficient vector $(a_0, \ldots, a_{N-1}) \in \mathbb{R}^N$. We define the relevant norms on the coefficient vector of a such as $\|a\|_\infty$ and $\|a\|_1$.

We write $\mathbb{Z}_M^* = \{x \in \mathbb{Z}_M : \gcd(x, M) = 1\}$ for the multiplicative group of units in $\mathbb{Z}_M$. Recall that the canonical embedding of $a(X) \in \mathbb{Q}[X]/(\Phi_M(X))$ into $\mathbb{C}^N$ is the vector $\sigma(a) = a(\zeta^j)_{j \in \mathbb{Z}_M^*}$ where $\zeta = \exp(-2\pi i/M)$ denotes a complex primitive M-th roots of unity. The ℓ_∞-norm of $\sigma(a)$ is called the *canonical embedding norm* of a, denoted by $\|a\|_\infty^{\mathsf{can}} = \|\sigma(a)\|_\infty$. The canonical embedding norm $\|\cdot\|_\infty^{\mathsf{can}}$ satisfies the following properties:

- For all $a, b \in \mathbb{Q}[X]/(\Phi_M(X))$, we have $\|a \cdot b\|_\infty^{\mathsf{can}} \leqslant \|a\|_\infty^{\mathsf{can}} \cdot \|b\|_\infty^{\mathsf{can}}$
- For all $a \in \mathbb{Q}[X]/(\Phi_M(X))$, we have $\|a\|_\infty^{\mathsf{can}} \leqslant \|a\|_1$.
- There is a ring constant c_M depending only on M such that $\|a\|_\infty \leqslant c_M \cdot \|a\|_\infty^{\mathsf{can}}$ for all $a \in \mathbb{Q}[X]/(\Phi_M(X))$.

The ring constant is obtained by $c_M = \|\mathsf{CRT}_M^{-1}\|_\infty$ where CRT_M is the CRT matrix for M, i.e., the Vandermonde matrix over the complex primitive M-th roots of unity, and the norm for a matrix $U = (u_{ij})_{0 \leqslant i,j < N}$ is defined by $\|U\|_\infty = \max_{0 \leqslant i < N} \left\{ \sum_{j=0}^{N-1} |u_{ij}| \right\}$. Refer [DPSZ12] for a discussion of c_M.

We use a natural extension of the canonical embedding to $\mathcal{P}$ which sends a real polynomial to the vector of evaluations at ζ^j's in $\mathbb{C}^N$. We abuse the notation and denote σ this extended mapping.

2.3 Ring Learining with Errors

We first define the space

$$\mathbb{H} = \{\mathbf{z} = (z_j)_{j \in \mathbb{Z}_M^*} \in \mathbb{C}^N : z_j = \overline{z_{-j}}, \forall j \in \mathbb{Z}_M^*\},$$

which is isomorphic to $\mathbb{R}^N$ as an inner product space via the unitary basis matrix

$$U = \begin{pmatrix} \frac{1}{\sqrt{2}}I & \frac{i}{\sqrt{2}}J \\ \frac{1}{\sqrt{2}}J & \frac{-i}{\sqrt{2}}I \end{pmatrix}$$

where I is the identity matrix of size $N/2$ and J is its reversal matrix.

For $r > 0$, we define the Gaussian function $\rho_r : \mathbb{H} \to (0, 1]$ as $\rho_r(\mathbf{z}) = \exp(-\pi \cdot \|\mathbf{z}\|_2^2/r^2)$. Denote by Γ_r the continuous Gaussian probability distribution whose density is given by $r^{-N} \cdot \rho_r(\mathbf{z})$. Now one can extend this to an elliptical Gaussian distribution $\Gamma_{\mathbf{r}}$ on $\mathbb{H}$ as follows: let $\mathbf{r} = (r_1, \ldots, r_N)$ be a vector of positive real numbers, then a sample from $\Gamma_{\mathbf{r}}$ is given by $U \cdot \mathbf{z}$ where each entry of $\mathbf{z} = (z_i)_{1 \leqslant i \leqslant N}$ is chosen independently from the (one-dimensional) Gaussian distribution Γ_{r_i} for $i = 1, 2, \ldots, N$. This also gives a distribution $\Psi_{\mathbf{r}}$ on $\mathbb{Q}[X]/(\Phi_M(X)) \otimes \mathbb{R}$. That is, $\mathsf{CRT}_M^{-1} \cdot U \cdot \mathbf{z}$ gives us the coordinates with respect to the polynomial basis $1, X, X^2, \ldots, X^{N-1}$.

In practice, one can discritize the continuous Gaussian distribution $\Psi_{\mathbf{r}}$ by taking a valid rounding $\lfloor \Psi_{\mathbf{r}} \rceil_{\mathcal{R}^\vee}$. Refer [LPR10, LPR13] for explaining the methods in more details. We use it as the discrete error distribution of RLWE.

Here we define the RLWE distribution and decisional problem associated with it. Let $\mathcal{R}^\vee$ be the dual fractional ideal of $\mathcal{R}$ and write $\mathcal{R}_q^\vee = \mathcal{R}^\vee/q\mathcal{R}^\vee$. For a positive integer modulus $q \geqslant 2$, $s \in \mathcal{R}_q^\vee$, $\mathbf{r} \in (\mathbb{R}^+)^N$ and an error distribution $\chi := \lfloor \Psi_{\mathbf{r}} \rceil_{\mathcal{R}^\vee}$, we define $A_{N,q,\chi}(s)$ as the RLWE distribution obtained by sampling $a \leftarrow \mathcal{R}_q$ uniformly at random, $e \leftarrow \chi$ and returning $(a, a \cdot s + e) \in \mathcal{R}_q \times \mathcal{R}_q^\vee$.

The (decision) ring learning with errors, denoted by $\mathsf{RLWE}_{N,q,\chi}(\mathcal{D})$, is a problem to distinguish arbitrarily many independent samples chosen according to $A_{N,q,\chi}(s)$ for a random choice of s sampled from the distribution $\mathcal{D}$ over $\mathcal{R}^\vee$ from the same number of uniformly random and independent samples from $\mathcal{R}_q \times \mathcal{R}_q^\vee$.

Chapter 3

Homomorphic Encryption for Arithmetic of Approximate Numbers

In this section, we describe a method to construct a HE scheme for approximate arithmetic on encrypted data. Given encryptions of m_1 and m_2, this scheme allows us to securely compute encryptions of approximate values of $m_1 + m_2$ and $m_1 m_2$ with a predetermined precision. The main idea of our construction is to treat an inserted noise of RLWE problem as part of am error occurring during approximate computation. The most important feature of our scheme is the *rounding* operation of plaintexts. Just like the ordinary approximate computations using floating-point numbers, the rounding operation removes some LSBs of message and makes a trade-off between size of numbers and precision loss. Our concrete construction is based on the BGV scheme [BGV12] with a multiplication method by raising the ciphertext modulus [GHS12b], but our methodology can be applied to most of existing HE schemes.

3.1 Main Idea

Previous HE schemes have discrete plaintext spaces and support an exact arithmetic between encrypted data. For example, the BGV-type schemes [BGV12, GHS12b] satisfy the equation $[\langle \mathbf{c}, sk\rangle]_q = m + te$ for some small e where $\mathbf{c}$ is an encryption of m with respect to the secret sk and t denotes the plaintext modulus. Meanwhile, the FV-type (also called the scale-invariant) schemes [Bra12, FV12] have a decryption structure of the form $[\langle \mathbf{c}, sk\rangle]_q = \frac{q}{t}m + e$. It is not efficient to perform the computation of approximate numbers (e.g. real numbers) based on these schemes because their plaintexts are contained in a space with a finite characteristic such as $\mathbb{Z}_t$ and $\mathbb{Z}_t[X]/(\Phi_M(X))$. Another approaches with matrix ciphertexts [GSW13, CGGI16] supports an arithmetic betwwen integers (or integral polynomials), but the noise growth depends on the size of plaintexts and there have not been suggested any rounding operation.

Our goal is to carry out approximate arithmetic over encrypted data, or equivalently, compute the MSBs of a resulting message after homomorphic operations. The main idea is to add an encryption noise following significant figures of an input message. More precisely, a ciphertext of our scheme has a decryption structure of the form $\langle \mathbf{c}, sk\rangle = m + e \pmod q$ for some small error e. This error is inserted from RLWE to guarantee the security of scheme, but it is be considered as an error that arises during approximate computations. That is, the output of decryption algorithm will be treated as an approximate value of the original message with a high precision. The size of a plaintext will be small enough compared to the ciphertext modulus, i.e., $\|[\langle \mathbf{c}, sk\rangle]_q\| \ll q$, so that the result of a homomorphic operation is still smaller than the ciphertext modulus.

There are some issues that we need to consider more carefully. In unencrypted approximate computations, small errors may blow up when applying operations in succession, so it is valuable to consider the proximity of a calculated result to the exact value of an algorithm. Similarly, encrypted plaintexts in our scheme will contain some errors and they might be increased during homomorphic evaluations. Thus we compute an upper bound of errors and predict the precision of resulting values.

The management of the size of messages is another issue. If we compute a circuit of multiplicative depth L without rounding of messages, then the bit size of an output value will exponentially grow with L. This naive method is inappropriate for practical usage because it causes a huge ciphertext modulus. To resolve this problem, we suggest a new technique which divides intermediate values by a base to reduce the size of plaintexts and ciphertext modulus. It allows us to discard some inaccurate LSBs of a message while an error is still kept relatively small compared to the message. Consequently it yields a leveled HE scheme which achieves a linearly growing size of ciphertext modulus in the depth of an evaluation circuit.

3.2 Packing Method

The batching technique in HE system allows us to encrypt multiple messages in a single ciphertext and enables a parallel processing in SIMD manner. We take its advantage to parallelize computations and reduce the memory and complexity. Currently all the practical implementations of HE schemes including HElib [HS13] and SEAL [LP16] are based on the ring structure $\mathcal{R} = \mathbb{Z}[X]/(\Phi_M(X))$ over a cyclotomic polynomial $\Phi_M(X)$. For a plaintext modulus t, a polynomial in the plaintext space $\mathbb{Z}_t[X]/(\Phi_M(X))$ could be decoded to a vector of elements in a ring of characteristic t with respect to the CRT-based encoding technique [SV10, SV14]. However, our plaintext space does not have a finite characteristic and every plaintext embraces an error for security that cannot be removed after decryption.

In this section, we suggest a new encoding/decoding technique to apply the batching technique in our scheme. For simplicity and efficiency, we assume that M is a power of two. Recall that $\Phi_M(X) = X^N + 1$ for $N = M/2$. The set of integers one modulo four forms a subgroup of index two of the multiplicative group $\mathbb{Z}_M^*$ of units is $N/2$. Hence $\{\zeta_j, \overline{\zeta_j} : 0 \leqslant j < N/2\}$ forms the set of the primitive M-th roots of unity where $\zeta = \exp(-2\pi i/M)$ and $\zeta_j := \zeta^{5^j}$ for $0 \leqslant j < N/2$. The native plaintext space of our RLWE-based construction can be understood as the set of small polynomials in $\mathcal{P}$. Our idea is to use the roots of cyclotomic polynomial to evaluate a plaintext polynomial and transform it into a vector of complex numbers. More precisely, let τ be a map from $\mathcal{P}$ to $\mathbb{C}^{N/2}$ defined by $m(X) \mapsto \mathbf{z} = (z_j = m(\zeta_j))_{0 \leqslant j < N/2}$. Note that τ is an isometric ring isomorphism between metric spaces $(\mathcal{P}, \|\cdot\|_\infty^{\mathsf{can}})$ and $(\mathbb{C}^{N/2}, \|\cdot\|_\infty)$. It plays a role of decoding algorithm for the packing of multiple complex numbers in a single polynomial. An arithmetic operation between cyclotomic polynomial can be identified with the (Hadamard) operation in a SIMD manner with respect to this mapping.

The decoding map τ can be understood as a linear transform from $\mathbb{R}^N$ to $\mathbb{C}^{N/2}$ by identifying a polynomial $m(X) = \sum_{i=0}^{N-1} m_i X^i \in \mathcal{P}$ with its coefficient vector $\mathbf{m} = (m_0, \ldots, m_{N-1})$. Its matrix representation is given by

$$U = \begin{bmatrix} 1 & \zeta_0 & \zeta_0^2 & \cdots & \zeta_0^{N-1} \\ 1 & \zeta_1 & \zeta_1^2 & \cdots & \zeta_1^{N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \zeta_{\frac{N}{2}-1} & \zeta_{\frac{N}{2}-1}^2 & \cdots & \zeta_{\frac{N}{2}-1}^{N-1} \end{bmatrix}$$

which is the $(N/2) \times N$ Vandermonde matrix generated by $\{\zeta_j : 0 \leqslant j < N/2\}$.

For the computation of its inverse (i.e., encoding map), we first note that the relation $\overline{\mathbf{z}} = \overline{U} \cdot \mathbf{m}$ is obtained from $\mathbf{z} = U \cdot \mathbf{m}$ by taking its conjugation. If we write $\mathsf{CRT} = (U/\overline{U})$ as the Vandemonde matrix generated by the set $\{\zeta_j, \overline{\zeta_j} : 0 \leqslant j < N/2\}$ of M-th primitive roots of unity, then we get the identities $(\mathbf{z}/\overline{\mathbf{z}}) = \mathsf{CRT} \cdot \mathbf{m}$ and $\mathsf{CRT}^{-1} = \frac{1}{N} \overline{\mathsf{CRT}}^T$. It implies that the inverse linear transformation τ^{-1} can be computed by $\mathbf{m} = \frac{1}{N}(\overline{U}^T \cdot \mathbf{z} + U^T \cdot \overline{\mathbf{z}})$. We will identify two spaces $\mathcal{P}$ and $\mathbb{C}^{N/2}$ in the rest of paper, so that a ciphertext $\mathbf{c}$ will be called an encryption of $\mathbf{z} \in \mathbb{C}^{N/2}$ if $\mathbf{c}$ encrypts the corresponding polynomial $m(X) = \tau^{-1}(\mathbf{z})$.

A plaintext polynomial should have integer coefficients to be encrypted, but the encoded polynomial $m(X) = \tau^{-1}(\mathbf{z})$ might not be integral for an arbitrary vector $\mathbf{z} \in \mathbb{C}^N$. Hence we need to perform a rounding operation to find and send it to a polynomial $m'(X)$ in $\tau(\mathcal{R})$ with a small rounding error $\|m' - m\|_\infty^{\mathsf{can}}$. There are several round-off algorithms including the coordinate-wise randomized rounding. See [LPR13] for details.

We recommend to multiply a scaling factor $\Delta \geqslant 1$ before rounding to prevent the loss of precision and preserve the significant digits of a plaintext. Our encoding/decoding algorithms are explicitly given as follows:

- $\mathsf{Ecd}(\mathbf{z}; \Delta)$. For a complex vector $\mathbf{z} = (z_i)_{0 \leqslant i < N/2}$, the encoding process computes the polynomial $m(X) = \Delta \cdot \tau^{-1}(\mathbf{z}) \in \mathcal{P}$ and transforms it into an integral polynomial in $\mathcal{R}$ by sending its coefficients to the nearest integers.

- $\mathsf{Dcd}(m; \Delta)$. For an input polynomial $m \in \mathcal{R}$, output the vector $\mathbf{z} = \Delta^{-1} \cdot \tau(m)$, i.e., $\mathbf{z} = (z_j)_{0 \leqslant j < N/2}$ for $z_j = \Delta^{-1} \cdot m(\zeta_j)$.

As a toy example, let $M = 8$ and $\Delta = 64$. The decoding map has two evaluation points $\zeta_0 = \zeta$ and $\zeta_1 = \zeta^5$. For a given vector $\mathbf{z} = (3+4i, 2-i)$, the encoding algorithm finds the corresponding polynomial $\frac{1}{2}(5 + 3\sqrt{2}X + 3X^2 + 2\sqrt{2}X^3)$ has evaluation values $3+4i$ and $2-i$ at ζ_0 and ζ_1, respectively. Then the output of encoding algorithm is $m(X) = 160 + 136X + 96X^2 + 91X^3 \leftarrow \mathsf{Ecd}(\mathbf{z}; \Delta)$, which is the closest integral polynomial to $64 \cdot \frac{1}{2}(15 + 3\sqrt{2}X + 3X^2 + 2\sqrt{2}X^3)$. Note that $64^{-1} \cdot (m(\zeta_0), m(\zeta_1)) \approx (2.9972 + 4.0080i, 2.0028 - 1.0080i)$ is approximate to the input vector $\mathbf{z}$ with a high precision.

3.3 Scheme

The purpose of this subsection is to construct a leveled HE scheme for an approximate arithmetic. For convenience, we fix a base $p > 0$ and a modulus q_0, and let $q_\ell = p^\ell \cdot q_0$ for $0 < \ell \leqslant L$. The integer p will be used as a base for scaling in approximate computation. For a security parameter λ, we also choose a parameter $M = M(\lambda, q_L)$ for cyclotomic polynomial. For a level $0 \leqslant \ell \leqslant L$, a ciphertext of level ℓ will be a pair of polynomials in $\mathcal{R}_{q_\ell}$. Our scheme consists of five algorithm ($\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add}, \mathsf{Mult}$) with constants B_{enc} and $B_{\mathsf{mult}}(\ell)$ for noise estimation. We also choose the ring of Gaussian integers $\mathbb{Z}[i]$ as a discrete subspace of $\mathbb{C}$ for implementation.

The performance of our construction and the noise growth depend on the base HE scheme. We take the BGV scheme [BGV12] with multiplication method by raising ciphertext modulus [GHS12b] as the underlying scheme of our concrete construction and implementation which seems to be the most efficient among the existing RLWE-based schemes in most parameter sets from Costache and Smart's comparison [CS16].

We have some advantages in security and simplicity from the choice of a *power-of-two* degree cyclotomic ring. In this case, the dual ideal $\mathcal{R}^\vee = N^{-1}\cdot\mathcal{R}$ of $\mathcal{R} = \mathbb{Z}[X]/(X^N+1)$ is simply a scaling of the ring. The RLWE problem is informally described by transforming samples $(a, b = a\cdot s' + e') \in \mathcal{R}_q \times \mathcal{R}_q^\vee$ into $(a, b = a\cdot s + e) \in \mathcal{R}_q \times \mathcal{R}_q$ where $s = s'\cdot N \in \mathcal{R}$ and $e = e'\cdot N \in \mathcal{R}$, so that the coefficients of e can be sampled independently from the discrete Gaussian distribution. Another advantage of power-of-two degree cyclotomic rings is the efficient rounding operation $\lfloor\cdot\rceil_{\mathcal{R}^\vee}$ in dual fractional ideal $\mathcal{R}^\vee$. Since the columns of matrix CRT_M defined in Section 2.2 are mutually orthogonal, the encoding of plaintext can be efficiently done by rounding coefficients to the nearest integers after multiplication with the matrix CRT_M^{-1}.

3.3.1 Description

We adopt the notation of some distributions on from [GHS12b]. For a real $\sigma > 0$, $\mathcal{DG}(\sigma^2)$ samples a vector in $\mathbb{Z}^N$ by drawing its coefficient independently from the discrete Gaussian distribution of variance σ^2. For an positive integer h, $\mathcal{HWT}(h)$ is the set of signed binary vectors in $\{0, \pm 1\}^N$ whose Hamming weight is exactly h. For a real $0 \leqslant \rho \leqslant 1$, the distribution $\mathcal{ZO}(\rho)$ draws each entry in the vector from $\{0, \pm 1\}^N$, with probability $\rho/2$ for each of -1 and $+1$, and probability being zero $1-\rho$.

- $\mathsf{KeyGen}(1^\lambda, q_0, p, L)$.

- Let $q_L = p^L \cdot q_0$. Choose a power-of-two M, an integer h, an integer P and a real value $\sigma = \sigma(\lambda, q_L)$ appropriately for the security of RLWE that will be described later.
- Sample $s \leftarrow \mathcal{HWT}(h)$, $a \leftarrow \mathcal{R}_{q_L}$ and $e \leftarrow \mathcal{DG}(\sigma^2)$. Set the secret key as $sk \leftarrow (1, s)$ and the public key as $pk \leftarrow (b, a) \in \mathcal{R}_{q_L}^2$ where $b \leftarrow -as + e \pmod{q_L}$.
- Sample $a' \leftarrow \mathcal{R}_{P \cdot q_L}$ and $e' \leftarrow \mathcal{DG}(\sigma^2)$. Set the evaluation key as $evk \leftarrow (b', a') \in \mathcal{R}_{P \cdot q_L}^2$ where $b' \leftarrow -a's + e' + Ps^2 \pmod{P \cdot q_L}$.

- $\mathsf{Enc}_{pk}(m)$. For a plaintext polynomial $m \in \mathcal{R}$, sample $v \leftarrow \mathcal{ZO}(0.5)$ and $e_0, e_1 \leftarrow \mathcal{DG}(\sigma^2)$. Output $v \cdot pk + (m + e_0, e_1) \pmod{q_L}$.
- $\mathsf{Dec}_{sk}(\mathbf{c})$. For $\mathbf{c} = (b, a) \in \mathcal{R}_{q_\ell}^2$, output $b + a \cdot s \pmod{q_\ell}$.
- $\mathsf{Add}(\mathbf{c}_1, \mathbf{c}_2)$. For $\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{R}_{q_\ell}^2$, output $\mathbf{c}_{\mathsf{add}} \leftarrow \mathbf{c}_1 + \mathbf{c}_2 \pmod{q_\ell}$.
- $\mathsf{Mult}_{evk}(\mathbf{c}_1, \mathbf{c}_2)$. For $\mathbf{c}_1 = (b_1, a_1), \mathbf{c}_2 = (b_2, a_2) \in \mathcal{R}_{q_\ell}^2$, let $(d_0, d_1, d_2) = (b_1 b_2, a_1 b_2 + a_2 b_1, a_1 a_2) \pmod{q_\ell}$. Output $\mathbf{c}_{\mathsf{mult}} \leftarrow (d_0, d_1) + \lfloor P^{-1} \cdot d_2 \cdot evk \rceil \pmod{q_\ell}$.
- $\mathsf{RS}_{\ell \to \ell'}(\mathbf{c})$. For a ciphertext $\mathbf{c} = (b, a) \in \mathcal{R}_{q_\ell}^2$ at level ℓ, output the ciphertext $\mathbf{c}' \leftarrow \left(\lfloor \frac{q_{\ell'}}{q_\ell} b \rceil, \lfloor \frac{q_{\ell'}}{q_\ell} a \rceil\right)$ in $\mathcal{R}_{q_{\ell'}}^2$, i.e., $\mathbf{c}'$ is obtained by scaling the coefficients of a, b and rounding to the closest integers. We will omit the subscript $\ell \to \ell'$ when $\ell' = \ell - 1$.

The last rescaling process RS shows the intrinsic characteristic of our scheme. Technically this procedure is similar to the modulus-switching algorithm of [BGV12], but it has a completely different role in our construction. The rescaling algorithm divides a plaintext by an integer to remove some inaccurate LSBs as a rounding step in usual approximate computations using floating-point numbers or scientific notation. The magnitude of messages can be maintained almost the same during homomorphic evaluation, and thus the required size of the largest ciphertext modulus grows linearly with the depth of the circuit being evaluated.

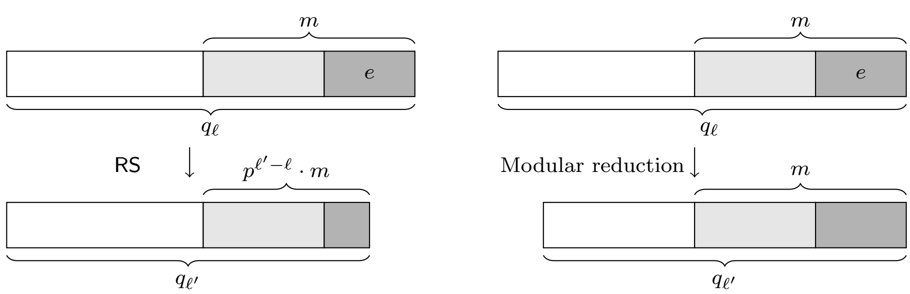
3.3.2 Security

The security of our HE scheme relies on the hardness of RLWE problem. Specifically, the distributions of encryption key pk and evaluation key evk are computationally indistinguishable from the uniform distribution over $\mathcal{R}_{P\cdot q_L}^2$ and $\mathcal{R}_{q_L}^2$, respectively, under the hardness of RLWE problem with parameter $(N, q_L, \sigma, \mathcal{HWT}(h))$. In addition, our scheme is IND-CPA secure when the RLWE problem of parameter $(N, q_L, \sigma, \mathcal{ZO}(0.5))$ is hard since the distribution of zero encryption is not distinguishable from the uniform distribution on $\mathcal{R}_{q_L}^2$.

3.3.3 Modulus Reduction

When given encryptions $\mathbf{c}, \mathbf{c}'$ of m, m' belong to the different levels ℓ and $\ell' < \ell$, we should bring a ciphertext $\mathbf{c}$ at a larger level ℓ to the smaller level ℓ' before homomorphic operation. There are two candidates: simple modulus reduction and the rescaling procedure. It should be chosen very carefully by considering the scale of messages because the simple modulus reduction $\mathbf{c} \mapsto \mathbf{c} \pmod{q_{\ell'}}$ preserves the plaintext while RS procedure changes the plaintext from m to $\frac{q_{\ell'}}{q_\ell}m$ as in Fig.3.1. Throughout this paper, we perform simple modulus reduction to the smaller modulus before computation on ciphertexts at different levels unless stated otherwise.

Figure 3.1: Rescaling and simple modulus reduction



3.4 Key Switching Technique

In this section, we suggest some useful functionalities supported in our scheme. We start with a key-switching operation on a normal ciphertext consisting of two ring elements. The purpose of a key-switching process is to transformation a ciphertext under a secret s' into an encryption of the same message with respect to the original secret key sk. The following procedure $\mathsf{KSGen}_{sk}(s')$ generates a switching key swk for this functionality.

- $\underline{\mathsf{KSGen}_{sk}(s')}$. For $s' \in \mathcal{R}$, sample $a \leftarrow \mathcal{R}_{P\cdot q_L}$ and $e \leftarrow \mathcal{DG}(\sigma^2)$. Output the switching key as $swk \leftarrow (b, a) \in \mathcal{R}^2_{P\cdot q_L}$ where $b \leftarrow -as + e + Ps' \pmod{P \cdot q_L}$.
- $\underline{\mathsf{KS}_{swk}(\mathbf{c})}$. Output the ciphertext $\mathbf{c}_{\mathsf{ks}} \leftarrow (c_0, 0) + \lfloor P^{-1} \cdot c_1 \cdot swk \rceil \pmod{q_\ell}$.

The correctness of key-switching procedure and noise estimation will be done in the next section. In scheme description, the evaluation key evk for multiplication can be understood as a switching key from $s' = s^2$ to s.

Let $\kappa_k : m(X) \mapsto m(X^k) \pmod{\Phi_M(X)}$ be a mapping defined on the set $\mathcal{P}$ for an integer k co-prime with M. Given a ciphertext $\mathbf{c}$ of a message m, we denote $\kappa_k(\mathbf{c})$ the ciphertext obtained by applying κ_k to the entries of $\mathbf{c}$. Then $\kappa_k(\mathbf{c})$ is a valid encryption of $\kappa_k(m)$ with the secret $\kappa_k(s)$. The key-switching technique can be applied to the ciphertext $\kappa_k(\mathbf{c})$ in order to get an encryption of the same message $\kappa_k(m)$ with respect to the original secret key sk.

3.4.1 Rotation

Suppose that $\mathbf{c}$ is an encryption of a message $m(X)$ with the corresponding plaintext vector $\mathbf{z} = (z_j)_{0 \leqslant j < \frac{N}{2}} \in \mathbb{C}^{N/2}$. For any $0 \leqslant i, j < N/2$, there is a mapping κ_k which sends an element in the slot of index i to an element in the slot of index j. Let us define $k = 5^{i-j} \pmod{M}$ and $\tilde{m} = \kappa_k(m)$. Then we have

$$\tilde{z}_j = \tilde{m}(\zeta_j) = \tilde{m}(\zeta^{5^j}) = m(\zeta^{k \cdot 5^j}) = m(\zeta_i) = z_i,$$

so the j-th slot of $\tilde{m}$ and the i-th slot of m have the same value. In general, we may get a ciphertext $\kappa_{5^r}(\mathbf{c})$ which corresponds a left-rotation of the plaintext vector by r slots. Below we describe the rotation procedure including the key-switching operation.

- Generate the rotation key $rk_r \leftarrow \mathsf{KSGen}_{sk}(\kappa_{5^r}(s))$.
- $\underline{\mathsf{Rot}_{rk_r}(\mathbf{c}; r)}$. Output the ciphertext $\mathsf{KS}_{rk_r}(\kappa_{5^r}(\mathbf{c}))$.

3.4.2 Conjugation

Similarly, we see that $\kappa_{-1}(\mathbf{c})$ is a valid encryption of a plaintext vector $\bar{\mathbf{z}} = (\overline{z_j})_{0 \leqslant j < N/2}$ with the secret key $\kappa_{-1}(s)$ if $\mathbf{c}$ is an encryption of $\mathbf{z}$. It follows from that fact that

$$\overline{z_j} = \overline{m(\zeta_j)} = m(\overline{\zeta_j}) = m(\zeta_j^{-1}).$$

In short, a homomorphic evaluation of the conjugation operation over plaintext slots consists of two procedures:

- Generate the conjugation key $ck \leftarrow \mathsf{KSGen}_{sk}(\kappa_{-1}(s))$.
- $\underline{\mathsf{Conj}_{ck}(\mathbf{c})}$. Output the ciphertext $\mathsf{KS}_{ck}(\kappa_{-1}(\mathbf{c}))$.

3.5 Correctness and Analysis

3.5.1 Distributions

We now give some noise estimations of suggested operations and algorithms. We follow the heuristic approach in [GHS12b, CS16]. Assume that a polynomial $a(X) \in \mathcal{P}$ is sampled from one of $\mathcal{HWT}(h)$, $\mathcal{ZO}(\rho)$, and the uniform distribution over $\mathcal{R}_q$, so its nonzero entries are independently and identically distributed. Since $a(\zeta)$ is the inner product of coefficient vector of a and the fixed vector $(1, \zeta, \ldots, \zeta^{N-1})$ of Euclidean norm $\sqrt{N}$, the random variable $a(\zeta)$ has variance $V = \sigma^2 \cdot N$, where σ^2 is the variance of each coefficient of a. Hence $a(\zeta)$ has the variances $V_G = \sigma^2 \cdot N$, $V_Z = \rho \cdot N$ and $V_U = q^2 \cdot N/12$, when a is sampled from $\mathcal{DG}(\sigma^2)$, $\mathcal{ZO}(\rho)$ or the uniform distribution over $\mathcal{R}_q$, respectively. In addition, it has the variance $V_H = h$ when $a(X)$ is chosen from $\mathcal{HWT}(h)$. Moreover, we can assume that $a(\zeta)$ is distributed similarly to a Gaussian random variable over complex plane since it is a sum of many independent and identically distributed random variables. Every evaluations at root of unity ζ^j share the same variance, so we will use 6σ as a high-probability bound on the canonical embedding norm of a when each coefficient has a variance σ^2. For a multiplication of two independent random variables close to Gaussian distributions with variances σ_1^2 and σ_2^2, we will use a high-probability bound $16\sigma_1\sigma_2$.

3.5.2 Noise Estimation

The native plaintext space of `HEAAN` can be understood as the set of polynomials $m(X)$ in $\mathbb{Z}[X]/(\Phi_M(X))$ such that $\|m\|_\infty^{\mathsf{can}} \ll q$. For convenience, we allow any polynomial with real coefficients modulo the cyclotomic polynomial as a plaintext polynomial. So a ciphertext $\mathbf{c} = (c_0, c_1) \in \mathcal{R}_{q_\ell}^2$ at level ℓ will be called an encryption of $m(X) \in \mathcal{P}$ with an error bound B if it satisfies $\langle \mathbf{c}, sk \rangle = m + e \pmod{q_\ell}$ for some polynomial $e(X) \in \mathcal{P}$ satisfying $\|e\|_\infty^{\mathsf{can}} < B$.

Lemma 3.5.1 (Encoding). *Let $m \leftarrow \mathsf{Ecd}(\mathbf{z}; \Delta)$ for $\mathbf{z} \in \mathbb{Z}[i]^{N/2}$ and $\Delta > 0$. Then $m = \Delta \cdot \tau^{-1}(\mathbf{z}) + e$ for some $e \in \mathcal{P}$ satisfying $\|e\|_\infty^{\mathsf{can}} \leqslant N/2$.*

Proof. It is directly obtained from the definition of Ecd that $m = \Delta \cdot \tau^{-1}(\mathbf{z}) + e$ for a rounding error e. Its inifinite norm is bounded by $1/2$, so we get desired bound from the inequality $\|e\|_\infty^{\mathsf{can}} \leqslant N \cdot \|e\|_\infty$. □

Lemma 3.5.2 (Rescaling). *Let $\mathbf{c}' \leftarrow \mathsf{RS}_{\ell \to \ell'}(\mathbf{c})$ for a ciphertext $\mathbf{c} \in \mathcal{R}_{q_\ell}^2$. Then $\langle \mathbf{c}', sk \rangle = \frac{q_{\ell'}}{q_\ell} \langle \mathbf{c}, sk \rangle + e \pmod{q_{\ell'}}$ for some $e \in \mathcal{P}$ satisfying $\|e\|_\infty^{\mathsf{can}} < B_{\mathsf{rs}}$ for $B_{\mathsf{rs}} = \sqrt{N/3} \cdot (3 + 8\sqrt{h})$.*

Proof. The output ciphertext $\mathbf{c}' \leftarrow \lfloor \frac{q_{\ell'}}{q_\ell}\mathbf{c} \rceil$ satisfies $\langle \mathbf{c}', sk\rangle = \frac{q_{\ell'}}{q_\ell}\langle \mathbf{c}, sk\rangle + e_{\mathsf{rs}} \pmod{q_{\ell'}}$ for the rounding error vector $\mathbf{e} = (e_0, e_1) = \mathbf{c}' - \frac{q_{\ell'}}{q_\ell}\mathbf{c}$ and the error polynomial $e_{\mathsf{rs}} = \langle \mathbf{e}, sk\rangle = e_0 + e_1 \cdot s$.

We may assume that each coefficient of e_0 and e_1 in the rounding error vector is computationally indistinguishable from the random variable in the interval $\frac{q_{\ell'}}{q_\ell}\mathbb{Z}_{q_\ell/q_{\ell'}}$ with variance $\approx 1/12$. Hence, the magnitude of scale error polynomial is bounded by $\|e_{\mathsf{rs}}\|_\infty^{\mathsf{can}} \leqslant \|e_0\|_\infty^{\mathsf{can}} + \|e_1 \cdot s\|_\infty^{\mathsf{can}} \leqslant 6\sqrt{N/12} + 16\sqrt{hN/12}$, as desired. □ □

Lemma 3.5.3 (Key-switching)**.** *Let $\mathbf{c} \in \mathcal{R}_q^2$ be a ciphertext with respect to a secret $sk' = (1, s')$ and let $swk \leftarrow \mathsf{KSGen}_{sk}(s')$. Then $\mathbf{c}_{\mathsf{ks}} \leftarrow \mathsf{KS}_{swk}(\mathbf{c})$ satisfies $\langle \mathbf{c}_{\mathsf{ks}}, sk\rangle = \langle \mathbf{c}, sk'\rangle + e_{\mathsf{ks}} \pmod{q}$ for some $e_{\mathsf{ks}} \in \mathcal{R}$ with $\|e_{\mathsf{ks}}\|_\infty^{\mathsf{can}} < P^{-1} \cdot q \cdot B_{\mathsf{ks}} + B_{\mathsf{rs}}$.*

Proof. Let e' be the error of swk satisfying $\langle swk, sk\rangle = P\cdot s' + e' \pmod{P\cdot q_L}$. Let $\mathbf{c} = (b, a)$, then $\langle a \cdot swk, sk\rangle = a \cdot P \cdot s' + c_1 \cdot e' \pmod{P \cdot q_\ell}$. From the definition, the output ciphertext satisfies $\langle \mathbf{c}_{\mathsf{ks}}, sk\rangle = b + a \cdot s' + P^{-1} \cdot a \cdot e' + e_{\mathsf{rs}} \pmod{q_\ell} = \langle \mathbf{c}', sk'\rangle + P^{-1} \cdot a \cdot e' + e_{\mathsf{rs}}$ for some rescaling error e_{rs}. Hence a key-switching error $e_{\mathsf{ks}} = P^{-1} \cdot e' + e_{\mathsf{rs}}$ is bounded by

$$\|e_{\mathsf{ks}}\|_\infty^{\mathsf{can}} \leqslant P^{-1} \cdot \|e'\|_\infty^{\mathsf{can}} + \|e_{\mathsf{rs}}\|_\infty^{\mathsf{can}} \leqslant P^{-1} \cdot q \cdot B_{\mathsf{ks}} + B_{\mathsf{rs}},$$

as desired. □

Lemma 3.5.4 (Encryption)**.** *Let $\mathbf{c} \leftarrow \mathsf{Enc}_{pk}(m)$ be an encryption of $m \in \mathcal{R}$. Then $\langle \mathbf{c}, sk\rangle = m + e \pmod{q_L}$ for some $e \in \mathcal{R}$ satisfying $\|e\|_\infty^{\mathsf{can}} < B_{\mathsf{enc}}$ for $B_{\mathsf{enc}} = 8\sqrt{2}\sigma N + 6\sigma\sqrt{N} + 16\sigma\sqrt{hN}$.*

Proof. We choose $v \leftarrow \mathcal{ZO}(0.5)^2$ and $e_0, e_1 \leftarrow \mathcal{DG}(\sigma^2)$, then set $\mathbf{c} \leftarrow v \cdot pk + (m + e_0, e_1)$. The bound B_{enc} of encryption noise is computed by the following inequality:

$$\begin{aligned} \|\langle \mathbf{c}, sk\rangle - m \pmod{q_L}\|_\infty^{\mathsf{can}} &= \|v \cdot e + e_0 + e_1 \cdot s\|_\infty^{\mathsf{can}} \\ &\leqslant \|v \cdot e\|_\infty^{\mathsf{can}} + \|e_0\|_\infty^{\mathsf{can}} + \|e_1 \cdot s\|_\infty^{\mathsf{can}} \\ &\leqslant 8\sqrt{2} \cdot \sigma N + 6\sigma\sqrt{N} + 16\sigma\sqrt{hN}, \end{aligned}$$

as desired. □

Lemma 3.5.5 (Multiplication). *Let* $\mathbf{c}_{\mathsf{mult}} \leftarrow \mathit{Mult}_{evk}(\mathbf{c}_1, \mathbf{c}_2)$ *for two ciphertexts* $\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{R}_{q_\ell}^2$. *Then* $\langle \mathbf{c}_{\mathsf{mult}}, sk \rangle = \langle \mathbf{c}_1, sk \rangle \cdot \langle \mathbf{c}_2, sk \rangle + e_{\mathsf{mult}} \pmod{q_\ell}$ *for some* $e \in \mathcal{R}$ *satisfying* $\|e_{\mathsf{mult}}\|_\infty^{can} < B_{\mathsf{mult}}(\ell)$ *for* $B_{\mathsf{ks}} = 8\sigma N/\sqrt{3}$ *and* $B_{\mathsf{mult}}(\ell) = P^{-1} \cdot q_\ell \cdot B_{\mathsf{ks}} + B_{\mathsf{rs}}$.

Proof. Let $\mathbf{c}_i = (b_i, a_i)$ for $i = 1, 2$, and let $(d_0, d_1, d_2) = (b_1 b_2, a_1 b_2 + a_2 b_1, a_1 a_2)$. This vector can be viewed as a level-ℓ encryption of $m_1 m_2$ with respect to the secret vector $(1, s, s^2)$. It follows from Lemma 3.5.2 that the ciphertext $\mathbf{c}_{\mathsf{mult}} \leftarrow (d_0, d_1) + \lfloor P^{-1} \cdot d_2 \cdot evk \rceil \pmod{q_\ell}$ satisfies $\langle \mathbf{c}_{\mathsf{mult}}, sk \rangle = \langle \mathbf{c}_1, sk \rangle \cdot \langle \mathbf{c}_2, sk \rangle + e_{\mathsf{ks}} + e_{\mathsf{rs}}$ for a key switching error $e_{\mathsf{ks}} = P^{-1} \cdot d_2 \cdot e'$ and a rounding error e_{rs}. We may assume that d_2 behaves as a uniform random variable on $\mathcal{R}_{q_\ell}$, so we have a bound $P\|e_{\mathsf{ks}}\|_\infty^{\mathsf{can}} \leqslant 16\sqrt{N q_\ell^2/12}\sqrt{N\sigma^2} = 8N \cdot \sigma q_\ell/\sqrt{3} = B_{\mathsf{ks}} \cdot q_\ell$. Therefore, a multiplication error is bounded by

$$\|e_{\mathsf{mult}}\|_\infty^{\mathsf{can}} \leqslant P^{-1} \cdot q_\ell \cdot B_{\mathsf{ks}} + B_{\mathsf{rs}},$$

as desired. □

3.5.3 Tagged Information

A homomorphic operation has an effect on the size of plaintext and the growth of message and noise. Each ciphertext will be tagged with bounds of a message and an error in order to dynamically manage their magnitudes. Hence, a full ciphertext will be of the form $(\mathbf{c}, \ell, \nu, B)$ for a ciphertext vector $\mathbf{c} \in \mathcal{R}_{q_\ell}^2$, a level $0 \leqslant \ell \leqslant L$, an upper bound $\nu \in \mathbb{R}^+$ of message and an upper bound $B \in \mathbb{R}^+$ of noise. Table 3.1 shows the full description of our scheme and homomorphic operations for ciphertexts with tagged information.

3.5.4 Relative Error

The decryption result of a ciphertext is an approximate value of plaintext, so the noise growth from homomorphic operations may cause some negative

Table 3.1: Description of our scheme

$$\begin{array}{|rl|}\hline \mathsf{Enc}_{pk}: & m \mapsto (\mathbf{c}, L, \nu, B_{\mathsf{enc}}) \text{ for some } \nu \geqslant \|m\|_\infty^{\mathsf{can}} \\ \mathsf{Dec}_{sk}: & (\mathbf{c}, \ell, \nu, B) \mapsto \langle \mathbf{c}, sk \rangle \pmod{q_\ell} \\ \mathsf{RS}_{\ell\to\ell'}: & (\mathbf{c}, \ell, \nu, B) \mapsto (\mathbf{c}', \ell', p^{\ell'-\ell}\cdot\nu, p^{\ell'-\ell}\cdot B + B_{\mathsf{rs}}) \\ \mathsf{Add}: & ((\mathbf{c}_1, \ell, \nu_1, B_1), (\mathbf{c}_2, \ell, \nu_2, B_2)) \mapsto (\mathbf{c}_{\mathsf{add}}, \ell, \nu_1+\nu_2, B_1+B_2) \\ \mathsf{Mult}_{evk}: & ((\mathbf{c}_1, \ell, \nu_1, B_1), (\mathbf{c}_2, \ell, \nu_2, B_2)) \\ & \quad \mapsto (\mathbf{c}_{\mathsf{mult}}, \ell, \nu_1\nu_2, \nu_1 B_2 + \nu_2 B_1 + B_1 B_2 + B_{\mathsf{mult}}(\ell)) \\ \hline \end{array}$$

effect such as loss of significance. Hence it needs to dynamically manage the bound of noise of ciphertexts for a correct understanding of the outputs. A full ciphertext $(\mathbf{c}, \ell, \nu, B)$ contains upper bounds of plaintext and noise, but sometimes it is convenient to consider the relative error defined by $\beta = B/\nu$.

For example, it is easy to see that the addition of ciphertexts with relative errors $\beta_i = B_i/\nu_i$ produces a ciphertext with a relative error bounded by $\max_i\{\beta_i\}$. In other case, if we multiply two ciphertexts $(\mathbf{c}_1, \ell, \nu_1, B_1), (\mathbf{c}_2, \ell, \nu_2, B_2)$ and scale down to a lower level ℓ' (as floating-point multiplication does), it produces a ciphertext at level ℓ' with a relative error

$$w' = \beta_1 + \beta_2 + \beta_1\beta_2 + \frac{B_{\mathsf{mult}}(\ell) + p^{\ell-\ell'}\cdot B_{\mathsf{rs}}}{\nu_1\nu_2}$$

from Lemmas 3.5.2 and 3.5.5. This relative error is about $\beta_1 + \beta_2$ similar to the case of unencrypted floating-point multiplication under an appropriate choice of parameter and level.

Chapter 4

Evaluation of Circuits

In this section, we describe some algorithms for evaluating some circuits commonly used in practical applications and analyze error growth of an output ciphertext based on our concrete construction. We start with the homomorphic evaluations of typical circuits such as addition and multiplication by constants, monomial, and polynomial. These can be extended to approximate series for analytic functions such as multiplicative inverse and exponential function. The required parameters and precision of results will be also analyzed together.

For the convenience of analysis, we will assume that the term $\beta_1\beta_2 + (B_{\mathsf{mult}}(\ell) + p^{\ell-\ell'} \cdot B_{\mathsf{rs}})/(\nu_1\nu_2)$ is always bounded by a fixed constant β_*, so the relative error of ciphertext $\mathbf{c}' \leftarrow \mathsf{RS}_{\ell\to\ell'}(\mathsf{Mult}(\mathbf{c}_1, \mathbf{c}_2))$ satisfies the inequality $\beta' \leqslant \beta_1 + \beta_2 + \beta_*$. We will discuss about the choice of β_* and check the validity of this assumption at the end of Section 4.1.

4.1 Polynomial Functions

The goal of this subsection is to suggest an algorithm for evaluating an arbitrary polynomial, and analyze its complexity and precision of output ciphertext. We start with the constant addition and multiplication functions $f(x) = x + a$ and $f(x) = ax$ for a constant $a \in \mathcal{R}$.

Lemma 4.1.1 (Addition/Multiplication by Constant)**.** *Let $(\mathbf{c}, \ell, \nu, B)$ be an encryption of $m \in \mathcal{P}$. For a constant $a \in \mathcal{R}$, let $\mathbf{c}_{\mathsf{a}} \leftarrow \mathbf{c} + (a, 0) \pmod{q_\ell}$ and $\mathbf{c}_{\mathsf{m}} \leftarrow a \cdot \mathbf{c} \pmod{q_\ell}$. Then $(\mathbf{c}_{\mathsf{a}}, \ell, \nu + \|a\|_\infty^{\mathsf{can}}, B)$ and $(\mathbf{c}_{\mathsf{m}}, \ell, \|a\|_\infty^{\mathsf{can}} \cdot \nu, \|a\|_\infty^{\mathsf{can}} \cdot B)$ are valid encryptions of $m + a$ and am, respectively.*

Proof. There is a polynomial $e \in \mathcal{P}$ such that $\langle \mathbf{c}, sk \rangle = m + e \pmod{q_\ell}$ and $\|e\|_\infty^{\mathsf{can}} \leqslant B$. It is obvious that $\langle \mathbf{c}_{\mathsf{a}}, sk \rangle = a + \langle \mathbf{c}, sk \rangle = (a + m) + e \pmod{q_\ell}$. We also have $\langle \mathbf{c}_{\mathsf{m}}, sk \rangle = a \cdot (m + e) = am + ae \pmod{q_\ell}$ and $\|a \cdot e\|_\infty^{\mathsf{can}} \leqslant \|a\|_\infty^{\mathsf{can}} \cdot B$. □

Now we describe an algorithm to evaluate the power polynomial x^d for a power of two integer d. For simplicity, we assume that the bound ν of message m is equal to the base p.

Algorithm 1 Power polynomial $f(x) = x^d$ of a power-of-two degree

```
1: procedure POWER(c ∈ R^2_{q_ℓ}, d = 2^r)
2:     c_0 ← c
3:     for j = 1 to r do
4:         c_j ← RS(Mult(c_{j−1}, c_{j−1}))
5:     end for
6:     return c_r
7: end procedure
```

For an input polynomial $m \in \mathcal{P}$ of size $\|m\|_\infty^{\mathsf{can}} \leqslant p$, Algorithm 1 repeatedly performs the rescaling procedure after each of squaring step to maintain the size of message, thus the output of Algorithm 1 is an encryption of the scaled value $p \cdot f(m/p) = m^d/p^{d-1}$. The following lemma explains the correctness of Algorithm 1 and gives the relative error of the output ciphertext.

Lemma 4.1.2. *Let $(\mathbf{c}, \ell, p, \beta_0 \cdot p)$ be an encryption of $m \in \mathcal{P}$ and d be a power-of-two integer. Then Algorithm 1 outputs a valid encryption $(\mathbf{c}_r, \ell - r, p, \beta_d \cdot p)$ of m^d/p^{d-1} for some real number $\beta_d \leqslant d \cdot \beta_0 + (d-1) \cdot \beta_*$.*

Proof. We argue by induction on j. It is easy to see that $(\mathbf{c}_j, \ell - j, p, \beta_{2^j} \cdot p)$ is an encryption of $m^{2^j}/p^{2^j - 1}$ for some real number $\beta_{2^j} \leqslant 2 \cdot \beta_{2^{j-1}} + \beta_*$. After

r iterations, it produces an encryption $(\mathbf{c}_r, \ell - r, p, \beta_{2^r} \cdot p)$ of m^{2^r}/p^{2^r-1} for some β_{2^r} such that $\beta_{2^r} \leqslant 2^r \cdot \beta_0 + (2^r - 1) \cdot \beta_*$. $\square$

Algorithm 1 can be extended to an algorithm which evaluates an arbitrary polynomial. Similar to the previous case, this extended algorithm outputs an encryption of the scaled value $p \cdot f(m/p) = m^d/p^{d-1}$.

Lemma 4.1.3. *Let $(\mathbf{c}, \ell, p, B)$ be an encryption of $m \in \mathcal{P}$ and let d be a positive integer. Then one can compute a valid encryption $(\mathbf{c}', \ell - \lceil \log d \rceil, p, \beta_d \cdot p)$ of m^d/p^{d-1} for some real number $\beta_d \leqslant d \cdot \beta_0 + (d-1) \cdot \beta_*$.*

Lemma 4.1.4 (Polynomial)**.** *Let $f(x) = \sum_{j=0}^{d} a_j x^j$ be a nonzero polynomial of coefficients a_j in $\mathcal{R}$ and of degree d. Let $(\mathbf{c}, \ell, p, w_0 \cdot p)$ be an encryption of $m\mathcal{P}$. Then one can compute a valid encryption $(\mathbf{c}', \ell - \lceil \log d \rceil, M_f, \beta_d \cdot M_f)$ of $p \cdot f(m/p)$ for $M_f = p \cdot \sum_{j=0}^{d} \|a_j\|_\infty^{\mathsf{can}}$ and for some real number $\beta_d \leqslant d \cdot \beta_0 + (d-1) \cdot \beta_*$.*

If the relative error of input ciphertext satisfies $\beta_0 \leqslant \beta_*$, the relative error of the resulting ciphertext is bounded by $\beta_d \leqslant d \cdot \beta_0 + (d-1) \cdot \beta_* \leqslant 2d \cdot \beta_0$. Hence, the precision loss is bounded by $(\log d + 1)$ bits, which is comparable to loss of significance occurring in unencrypted numerical computations. The evaluation of polynomial of degree d can be done in d homomorphic multiplications between ciphertext of depth $r = \lceil \log d \rceil$ by computing the encryptions of $m, m^2/p, \ldots, m^d/p^{d-1}$ simultaneously. We may apply the Paterson-Stockmeyer algorithm [PS73] to the evaluation procedure. Then a degree d polynomial can be evaluated using $\mathcal{O}(\sqrt{d})$ multiplications, which gives a similar upper bound on relative error as the naive approach.

Let us return to the assumption $\beta_1\beta_2 + (B_{\mathsf{mult}}(\ell) + p^{\ell-\ell'} \cdot B_{\mathsf{rs}})/(\nu_1\nu_2) \leqslant \beta_*$. We will choose β_* as an upper bound of relative errors of fresh ciphertexts in our scheme. After evaluation of circuits of depth less than $(L-1)$, the resulting ciphertext will have a relative error less than $2^L \cdot \beta_*$. It means that the first term $\beta_1\beta_2$ will be bounded by $2^{L+1} \cdot \beta_*^2$ after evaluation. The condition $2^{L+1} \cdot \beta_*^2 \leqslant \frac{1}{2}\beta_*$, or equivalently $\beta_* \leqslant 2^{-L-2}$, seems to be natural; otherwise the relative error becomes $2^{L+1} \cdot \beta_* \geqslant 2^{-1}$ after evaluation, so the decryption result will have almost no information. Thus we have $\beta_1 + \beta_2 \leqslant \frac{1}{2}\beta_*$. The second term is equal to $(p^{\ell'-\ell} \cdot B_{\mathsf{mult}}(\ell) + B_{\mathsf{rs}})/\nu'$ where $\nu' = p^{\ell'-\ell} \cdot \nu_1\nu_2$ is the message bound of new ciphertext obtained by rescaling after multiplication. The numerator is asymptotically bounded by $p^{\ell'-\ell} \cdot B_{\mathsf{mult}}(\ell) + B_{\mathsf{rs}} = \mathcal{O}(N)$. If the message bound always satisfies $\nu' \geqslant p$ as in our algorithms, the second term is $(B_{\mathsf{mult}}(\ell) + p^{\ell-\ell'} \cdot B_{\mathsf{rs}})/(\nu_1\nu_2) = \mathcal{O}(p^{-1} \cdot N)$ which is smaller than a half of relative error of fresh ciphertext because $\beta_* \geqslant p^{-1} \cdot B_{\mathsf{enc}} = \Omega(p^{-1} \cdot \sigma N)$.

4.2 Approximate Polynomials and Multiplicative Inverse

We now homomorphically evaluate analytic functions $f(x)$ using their Taylor decomposition $f(x) = T_d(x) + R_d(x)$ for $T_d(x) = \sum_{j=0}^{d} \frac{f^{(j)}(0)}{j!} x^j$ and $R_d(x) = f(x) - T_d(x)$. Lemma 4.1.4 can be utilized to evaluate the rounded polynomial of scaled Taylor expansion $\lfloor p^u \cdot T_d \rceil(x)$ of $f(x)$ for some non-negative integers u and d, which outputs an approximate value of $p^{u+1} \cdot f(m/p)$. The bound of error is obtained by aggregating the error occurring during evaluation, the rounding error and the error of the remainder term $p^{u+1} \cdot R_d(m/p)$. In the case of RLWE-based constructions, we should consider the corresponding plaintext vector $\tau(m) = (z_j)_{0 \leqslant j < N/2}$ and convergence of series in each slot.

As an example, the exponential function $f(x) = \exp(x)$ has the Taylor polynomial $T_d(x) = \sum_{j=0}^{d} \frac{1}{j!} x^j$ and the remaining term is bounded by $|R_d(x)| \leqslant \frac{e}{(d+1)!}$ when $|x| \leqslant 1$. Assume that we are given an encryption $(\mathbf{c}, \ell, p, \beta_0 \cdot p)$ of m. With the input ciphertext $\mathbf{c}$ and the polynomial $\lfloor p^u \cdot T_d \rceil(x)$, one can compute an encryption of $p^{u+1} \cdot T_d(m/p)$. We see that an error of the resulting ciphertext is bounded by

$$dp + p^{u+1} \cdot \sum_{j=1}^{d} \frac{1}{j!}(j \cdot \beta_0 + (j-1)\beta_*) \leqslant dp + p^{u+1} \cdot (e\beta_0 + \beta_*).$$

If we write $\exp(m/p) := \tau^{-1}(\exp(z_j/p))_{0 \leqslant j < N/2})$, the output ciphertext can be also viewed as an encryption of $p^{u+1}\cdot\exp(m/p)$ of the form $(\mathbf{c}',\ell-\lceil\log d\rceil,\nu',B')$ for $\nu'=p^{u+1}\cdot e$ and $B'=dp+p^{u+1}\cdot(e\beta_0+\beta_*+\frac{e}{(d+1)!})$, and its relative error is bounded by $\beta'\leqslant(\beta_0+\beta_*\cdot e^{-1})+(p^{-u}\cdot d\cdot e^{-1}+\frac{1}{(d+1)!})$. If $\beta_0\geqslant\beta_*$, then we may take integers d and u satisfying $(d+1)!\geqslant 4\beta_0^{-1}$ and $p^u\geqslant 2\beta_0^{-1}\cdot d$ to make the relative error less than $2\beta_0$. In this case, the precision loss during evaluation of exponential function is less than one bit.

In the case of multiplicative inverse, we adopt an algorithm described in [cDSM15] to get a better complexity. Assuming that a complex number x satisfies $|\hat{x}|\leqslant 1/2$ for $\hat{x}=1-x$, we get

$$x(1+\hat{x})(1+\hat{x}^2)(1+\hat{x}^{2^2})\cdots(1+\hat{x}^{2^{r-1}})=1-\hat{x}^{2^r}. \tag{4.2.1}$$

Note that $|\hat{x}^{2^r}|\leqslant 2^{-2^r}$, and it converges to one as r goes to infinity. Hence, $\prod_{j=0}^{r-1}(1+\hat{x}^{2^j})=x^{-1}(1-\hat{x}^{2^r})$ can be considered as an approximate multiplicative inverse of x with 2^r bits of precision.

For homomorphic evaluation, we change a scale and assume that a complex number z_j satisfies $|\hat{z_j}|\leqslant p/2$ for $\hat{z_j}=p-z_j$. The standard approach starts by normalizing those numbers to be in the unit interval by setting $x=z_j/p$. Since we cannot multiply fractions over encrypted data, the precision point should move to the left for each term of (4.2.1). That is, we multiply both sides of the equation (4.2.1) by p^{2^r} and then it yields

$$z_j(p+\hat{z_j})(p^{2^1}+\hat{z_j}^{2^1})(p^{2^2}+\hat{z_j}^{2^2})\cdots(p^{2^{r-1}}+\hat{z_j}^{2^{r-1}})=p^{2^r}-\hat{z_j}^{2^r}.$$

Therefore, the product $p^{-2^r}\cdot\prod_{i=0}^{r-1}(p^{2^i}+\hat{z_j}^{2^i})$ can be seen as the approximate inverse of z_j with 2^r bits of precision. Let $\hat{\mathbf{z}}=(\hat{z_j})_{0\leqslant j<N/2}$ and $\mathbf{z}^{-1}=(z_j^{-1})_{0\leqslant j<N/2}$. Algorithm 2 takes an encryption of $\hat{m}=\tau^{-1}(\hat{\mathbf{z}})$ as an input and outputs an encryption of its scaled multiplicative inverse $p^2\cdot\tau^{-1}(\mathbf{z}^{-1})$ by evaluating the polynomial $\prod_{j=0}^{r-1}(p^{2^j}+\hat{m}^{2^j})$. The precision of the resulting ciphertext and the optimal iterations number r will be analyzed in the following lemma.

Lemma 4.2.1 (Multiplicative Inverse)**.** *Let* $(\mathbf{c},\ell,p/2,B_0=\beta_0\cdot p/2)$ *be an*

Algorithm 2 Inverse function $f(x) = x^{-1}$

procedure INVERSE($\mathbf{c} \in \mathcal{R}_{q_\ell}^2, r$)
    $\mathbf{p} \leftarrow (p, 0)$
    $\mathbf{c}_0 \leftarrow \mathbf{c}$
    $\mathbf{v}_1 \leftarrow \mathbf{p} + \mathbf{c}_0 \pmod{q_{\ell-1}}$
    for $j = 1$ to $r - 1$ **do**
        $\mathbf{c}_j \leftarrow \mathsf{RS}(\mathsf{Mult}(\mathbf{c}_{j-1}, \mathbf{c}_{j-1}))$
        $\mathbf{v}_{j+1} \leftarrow \mathsf{RS}(\mathsf{Mult}(\mathbf{v}_j, \mathbf{p} + \mathbf{c}_j))$
    end for
    return $\mathbf{v}_r$
end procedure

encryption of $\hat{m} \in \mathcal{R}$ and let $m = p - \hat{m}$. Then Algorithm 2 outputs a valid encryption $(\mathbf{v}_r, \ell - r, 2p, \beta \cdot 2p)$ of $m' = p \cdot \prod_{i=0}^{r-1}(1 + (\hat{m}/p)^{2^i})$ for some $\beta \leqslant \beta_0 + r\beta_*$.

Proof. From Lemma 4.1.1, $(\mathbf{v}_1, \ell - 1, 3p/2, B_0)$ is a valid encryption of $p + \hat{m}$ and its relative error is $\beta'_1 = \beta_0/3$. It also follows from Lemma 4.1.2 that $(\mathbf{c}_j, \ell - j, 2^{-2^j} \cdot p, \beta_j \cdot 2^{-2^j} \cdot p)$ is a valid encryption of $\hat{m}^{2^j}/p^{2^j-1}$ for some real number $\beta_j \leqslant 2^j \cdot (\beta_0 + \beta_*)$, and so $(\mathbf{p} + \mathbf{c}_j, \ell - j, (1 + 2^{-2^j})p, \beta_j \cdot 2^{-2^j} \cdot p)$ is a valid encryption of $p + \hat{m}^{2^j}/p^{2^j-1} = (p^{2^j} + \hat{m}^{2^j})/p^{2^j-1}$ with a relative error $\beta'_j \leqslant \beta_j/(2^{2^j} + 1) \leqslant 2^j \cdot (\beta_0 + \beta_*)/(2^{2^j} + 1)$, respectively.

Using the induction on j, we can show that

$$\left(\mathbf{v}_j, \ell - j, p \cdot \prod_{i=0}^{j-1}(1 + 2^{-2^i}), \beta''_j \cdot p \cdot \prod_{i=0}^{j-1}(1 + 2^{-2^i})\right)$$

is a valid encryption of $\prod_{i=0}^{j-1}(p^{2^i} + \hat{m}^{2^i})/p^{2^j-2} = p \cdot \prod_{i=0}^{j-1}(1 + (\hat{m}/p)^{2^i})$ with a relative error $\beta''_j \leqslant \sum_{i=0}^{j-1} \beta'_i + (j-1) \cdot \beta_*$. Note that the message is bounded by $p \cdot \prod_{i=0}^{j-1}(1 + 2^{-2^i}) = (2p) \cdot (1 - 2^{-2^j}) < 2p$ and the relative error satisfies

$$\beta''_j \leqslant \left(\sum_{i=0}^{j-1} \frac{2^i}{2^{2^i} + 1}\right) \cdot (\beta_0 + \beta_*) + (j-1) \cdot \beta_* \leqslant \beta_0 + j \cdot \beta_*$$

from the fact that $\sum_{i=0}^{\infty} \frac{2^i}{2^{2^i}+1} = 1$. Therefore, the output $\mathbf{v}_r$ of Algorithm 2 represents a valid encryption $(\mathbf{v}_r, \ell - r, 2p, \beta \cdot 2p)$ of $m' = p \cdot \prod_{i=0}^{r-1}(1 + (\hat{m}/p)^{2^i})$ for some $\beta \leqslant \beta_0 + r \cdot \beta_*$. □ □

Let $m^{-1}(X) := \tau^{-1}(\mathbf{z}^{-1})$ be the polynomial in $\mathcal{R}$ corresponding to $\mathbf{z}^{-1}$. The output ciphertext $(\mathbf{v}_r, \ell - r, 2p, \beta \cdot 2p)$ of the previous lemma can be also viewed as an encryption of $p^2 \cdot m^{-1}$. The error bound is increased by the convergence error $\|p^2 \cdot m^{-1} - m'\|_\infty^{\mathsf{can}} = \|p^2 \cdot m^{-1} \cdot (\hat{m}/p)^{2^r}\|_\infty^{\mathsf{can}} \leqslant 2^{-2^r} \cdot 2p$. Therefore, the ciphertext $(\mathbf{v}_r, \ell - r, 2p, (\beta + 2^{-2^r}) \cdot 2p)$ is a valid encryption of m' and its relative error is $\beta + 2^{-2^r} \leqslant \beta_0 + r\beta_* + 2^{-2^r}$, which is minimized when $r\beta_* \approx 2^{-2^r}$. Namely, $r = \lceil \log\log \beta_*^{-1} \rceil$ yields the inequality $\beta_0 + r\beta_* + 2^{-2^r} \leqslant \beta_0 + 2r\beta_* = \beta_0 + 2\lceil \log\log \beta_*^{-1} \rceil \cdot \beta_*$. Thus the precision loss during evaluation of multiplicative inverse is less than one bit if $2\lceil \log\log \beta_*^{-1} \rceil \cdot \beta_* \leqslant \beta_0$.

The optimal iterations number r can be changed upon more/less information about the magnitude of $\hat{m}$. Assume that we have an encryption of message $\hat{m}$ whose size is bounded by $\|\hat{m}\|_\infty^{\mathsf{can}} \leqslant \epsilon p$ for some $0 < \epsilon < 1$. By applying Lemma 4.2.1, we can compute an encryption of $p \cdot \prod_{i=0}^{r-1}(1 + (\hat{m}/p)^{2^i}) = (p^2 \cdot m^{-1}) \cdot (1 - (\hat{m}/p)^{2^r})$ with a relative error $\beta \leqslant \beta_0 + r\beta_*$, which is an approximate value of $p^2 \cdot m^{-1}$ with an error bounded by $\epsilon^{2^r} \cdot 2p$. Then the optimal iterations number is $r \approx \log\log \beta_*^{-1} - \log\log \epsilon^{-1}$ and the relative error becomes $\beta \leqslant \beta_0 + 2\lceil (\log\log \beta_*^{-1} - \log\log \epsilon^{-1}) \rceil \cdot \beta_*$ when $r = \lceil (\log\log \beta_*^{-1} - \log\log \epsilon^{-1}) \rceil$.

4.3 Fast Fourier Transform

Let d be a power of two integer and consider the complex primitive d-th root of unity $\xi = \exp(2\pi i/d)$. For a complex vector $\mathbf{u} = (u_0, \ldots, u_{d-1})$, its discrete Fourier transform (DFT) is defined by the vector $\mathbf{v} = (v_0, \ldots, v_{d-1}) \leftarrow \mathsf{DFT}(\mathbf{u})$ where $v_k = \sum_{j=0}^{d-1} \xi^{jk} \cdot u_j$ for $k = 0, \ldots, d-1$. The DFT has a numerous applications in mathematics and engineering such as signal processing technology. The basic idea is to send the data to Fourier space, carry out Hadamard operations and bring back the computation result to a original domain via the inverse DFT. We denote by $W_d(z) = (z^{j \cdot k})_{0 \leqslant j,k < d}$ the Vandermonde matrix generated by $\{z^k : 0 \leqslant k < d\}$. The DFT of $\mathbf{u}$ can be evaluated by the matrix multiplication $\mathsf{DFT}(\mathbf{u}) = W_d(\xi) \cdot \mathbf{u}$, but the complexity of DFT can be reduced down to $\mathcal{O}(d \log d)$ using FFT algorithm by representing the DFT matrix $W_d(\xi)$ as a product of sparse matrices.

Recently, Costache et al. [CSV16] suggested an encoding method which sends the complex d-th root of unity to the monomial $Y = X^{M/d}$ over cyclotomic ring $\mathcal{R} = \mathbb{Z}[X]/(\Phi_M(X))$ for cryptosystem. Then homomorphic evaluation of DFT is simply represented as a multiplication of the matrix $W_d(Y)$ to a vector of ciphertexts over polynomial ring.

On the other hand, our RLWE-based HE scheme can take advantage of batch technique as described in Section 3.2. In the slot of index $0 \leqslant k < N/2$, the monomial $Y = X^{M/d}$ and matrix $W_d(Y)$ are converted into ξ^k and the DFT matrix $W_d(\xi^k)$, respectively, depending on primitive root of unity ξ^k. However, our batching scheme is still meaningful because the evaluation result of whole pipeline consisting of DFT, Hadamard operations, and inverse DFT is independent of index k, even though $W_d(Y)$ corresponds to the DFT matrices generated by different primitive d-th roots of unity.

It follows from the property of ordinary FFT algorithm that if $(\mathbf{c}_i, \ell, \nu, B)$ is an encryption of u_i for $i = 0, \ldots, d-1$ and $\mathbf{v} = (v_0, \ldots, v_{d-1}) \leftarrow W_d(Y) \cdot \mathbf{u}$, then the output of FFT algorithm using $X^{M/d}$ instead of ξ forms valid encryptions $(\mathbf{c}'_i, \ell, \sqrt{d} \cdot \nu, \sqrt{d} \cdot B)$. Note that the precision of input ciphertexts is preserved as B/ν. Our FFT algorithm takes a similar time with [CSV16] in the same parameter setting, but the amortized time is much smaller thanks to our own plaintext packing technique. In the evaluation of whole pipeline DFT-Hadamard multiplication-inverse DFT, one may scale down the transformed ciphertexts by $\sqrt{d}$ before Hadamard operations to maintain the magnitude of messages and reduce the required levels for whole pipeline.

The fast polynomial multiplication using the FFT algorithm is a typical example that computes the exact value using approximate arithmetic. In particular for the case of integral polynomials, the exact multiplication can be recovered from its approximate value since we know that their multiplication is also an integral polynomial. Likewise, when the output of a circuit has a specific format or property, it is possible to get the exact computation result from its sufficiently close approximation.

4.4 Implementation

In this section we describe how to select parameters for evaluating arithmetic circuits described in this section. We also provide implementation results with concrete parameters. Our implementation is based on the NTL C++ library running over GMP. Every experimentation was performed on a machine with an Intel Core i5 running at 2.9 GHz processor using a parameter set with 80-bit security level.

We need to set the ring dimension N that satisfies the security condition $N \geqslant \frac{\lambda+110}{7.2}\log(P \cdot q_L)$ to get λ-bit security level. [LP11, GHS12b] We note that $P \cdot q_L$ is the largest modulus to generate evaluation key and it suffices to assume that P is approximately equal to q_L. In our implementation, we used the Gaussian distribution of standard deviation $\sigma = 3.2$ to sample error polynomials, and set $h = 64$ as the number of nonzero coefficients in a secret key $s(X)$.

Evaluation of Typical Circuits. In Table 4.1, we present the parameter setting and performance results for computing a power of a ciphertext, the multiplicative inverse of a ciphertext and exponential function. The average running times are only for ciphertext operations, excluding encryption and decryption procedures. As described in Section 3.2, each ciphertext can hold $N/2$ plaintext slots and one can perform the computation in parallel in each slot. Here the amortized running time means a relative time per slot.

The homomorphic evaluation of the circuit x^{1024} with an input of 36-bit precision is hard to be implemented in practice over previous methods. Meanwhile, our scheme can compute this circuit simultaneously over 2^{14} slots in about 7.46 seconds, yielding an amortized rate of 0.43 milliseconds per slot. Computation of the multiplicative inverse is done by evaluating the polynomial up to degree 8 as described in Algorithm 2. It gives an amortized time per slots of about 0.11 milliseconds. In the case of exponential function, we used terms in its Taylor expansion up to degree 8 and it results in an amortized time per slots of 0.16 milliseconds.

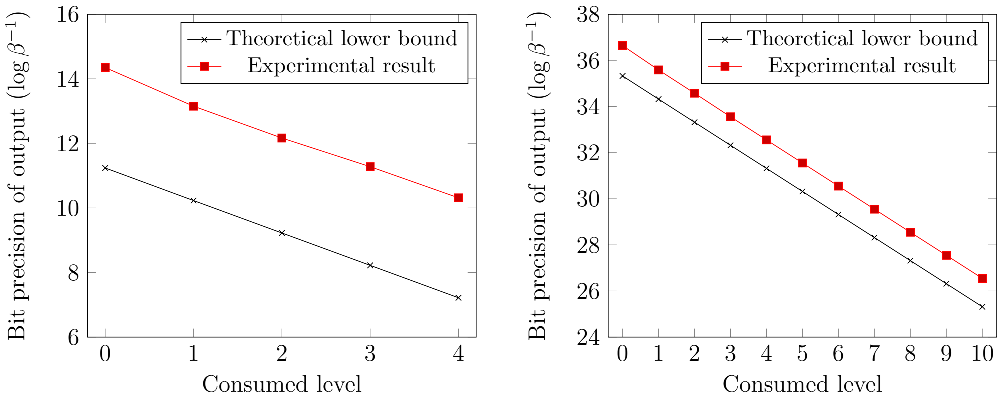
Table 4.1: Implementation results for homomorphic evaluation of typical circuits

Function	N	$\log q$	$\log p$	Consumed levels	Input precision	Total time	Amortized time
x^{16}	2^{13}	155	30	4	14 bits	0.31s	0.07ms
x^{-1}						0.45s	0.11ms
$\exp(x)$						0.65s	0.16ms
x^{1024}	2^{15}	620	56	10	36 bits	7.46s	0.43ms

Significance Loss. In Section 4.1, we analyzed the theoretical upper bounds on the growth of relative errors during evaluations. We can see from experimental result that initial precision is about 4 bits greater than theoretic bound of precision since we multiply 16 to the variance of encryption error to get a high probability bound. In Fig.4.1, we depict bit precisions of output ciphertexts during the evaluation of homomorphic multiplications (e.g. x^{16} for the left figure and x^{1024} for the right figure). We can actually check that both theoretic bound and experimental result of precision loss during homomorphic multiplications is less than 4.1 (or resp. 10.1) when the depth of the circuit is 4 (or resp. 10).

Logistic Function. Let us consider the logistic function $f(x) = (1 + \exp(-x))^{-1}$, which is widely used in statistics, neural networks and machine learning as a probability function. For example, logistic regression is used for a prediction of the likelihood to have a heart attack in an unspecified period for men, as indicated in [BLN14]. It was also used as a predictive equation to screen for diabetes, as described in [TH02]. This function can be

Figure 4.1: The variation of bit precision of ciphertexts when $(f(x), N, \log p, \log q) = (x^{16}, 2^{13}, 30, 155)$ and $(x^{1024}, 2^{15}, 56, 620)$.



approximated by its Taylor series

$$f(x) = \frac{1}{2} + \frac{1}{4}x - \frac{1}{48}x^3 + \frac{1}{480}x^5 - \frac{17}{80640}x^7 + \frac{31}{1451520}x^9 + \mathcal{O}(x^{11}).$$

In [BLN14, CJLL17], every real number is scaled by a predetermined factor to transform it as a binary polynomial before computation. The plaintext modulus t should be set large enough so that no reduction modulo t occurs in the plaintext space. The required bit size of plaintext modulus exponentially increases on the depth of the circuit, which strictly limits the performance of evaluation. On the other hand, the rescaling procedure in our scheme has the advantage that it significantly reduces the size of parameters (e.g. $(\log p, \log q) = (30, 155)$).

The parallelized computation for logistic function is especially important in real world applications such as statistic analysis using multiple data. In previous approaches, each slot of plaintext space should represent a larger degree than encoded polynomials so they could support only a few numbers of slots. On the other hand, we provide a parallelization method with an amortization amount independent from target circuit and get a better amortized time of evaluation.

Table 4.2: Comparison of implementation results for homomorphic evaluation of logistic function

Method	N	$\log q$	Polynomial degree	Amortization amount	Total time	Amortized time
[BLN14]	2^{14}	512	7	-	$> 30s$	-
[CJLL17]	17430	370	7	-	$1.8s$	-
Ours	2^{13}	155	7	2^{12}	0.54s	0.13ms
	2^{14}	185	9	2^{13}	0.78s	0.09ms

Discrete Fourier Transform. With the parameters $(N, \log p) = (2^{13}, 50)$, we encrypt coefficients of polynomials and homomorphically evaluate the standard processing (FFT-Hadamard product of two vectors-inverse FFT) in 73 minutes (amortized 1.06 seconds per slot) when $d = 2^{13}$. We could reduce down the evaluation time to 22 minutes (amortized 0.34 seconds per slot) by adapting the multi-threading method on a machine with six Intel Xeon E5 2.7GHz processors with 64 GB RAM, compared to 17 minutes of the previous work [CSV16] on a machine with four Intel Core i7 2.9 GHz processors with 16 GB RAM. Since the rescaling procedure of transformed ciphertexts enables us to efficiently carry out higher degree Hadamard operations in Fourier space, the gap of parameter and running time between our scheme and previous methods grows very quickly as degree N and the depth of Hadamard operation increase. For instance, we also homomorphically evaluate the product of 8 polynomials, using pipeline consisting of FFT-Hadamard product of eight vectors-inverse FFT with parameters $(N, \log q) = (2^{14}, 250)$ in amortized time of 1.76 seconds.

Table 4.3: Comparison of implementation results for homomorphic evaluation of a full FFT pipeline

Method	d	N	$\log q$	Degree of Hadamard operation	Amortization amount	Total time	Amortized time
[CSV16]	2^4	2^{13}	150	2	-	0.46s	-
	2^{13}	2^{14}	192	2	-	17min	-
Ours	2^4	2^{13}	120	2	2^{12}	0.88s	0.21ms
	2^{13}	2^{13}	130	2	2^{12}	22min	0.34s
	2^{13}	2^{14}	250	8	2^{13}	4h	1.76s

Chapter 5

Bootstrapping

The bootstrapping procedure of the existing HE schemes can be understood as a homomorphic evaluation of decryption circuit. For example, the (Ring) LWE-based HE schemes have a common decryption structure $\langle \mathbf{c}, sk \rangle$. The BGV type schemes [BGV12, GHS12b] support an addition and multiplication with the reduction to a plaintext modulus, and they have a decryption circuit of the form $m = [[\langle \mathbf{c}, sk \rangle]_q]_t$ for the plaintext modulus t. To homomorphically evaluate the decryption circuit in a larger ciphertext modulus, they choose a temporary plaintext modulus close to q to simplify the modular reduction operation and represent the decryption formula as a lower degree polynomial over the plaintext space [GHS12a, HS15].

In the case of our HE scheme, it does not support any modulus reduction operation and this makes the bootstrapping much harder. To homomorphically evaluate the decryption procedure $[\langle \mathbf{c}, sk \rangle]_q$, we need to represent even the modulus reduction step $[\cdot]_q$ as a polynomial over the integers. For example, one of the naive approach for expression of a modular reduction is to use the polynomial interpolation of the modulus operation over the domain of $z = \langle \mathbf{c}, sk \rangle$, but it is a limiting factor for practical implementation due to its depth and complexity of evaluation.

This paper presents a methodology to refresh a ciphertext of the `HEAAN` scheme and allows the evaluation of an arbitrary circuit. We take advantage of its intrinsic characteristic - *approximate* computations on encrypted data. Since the decryption structure is already contains some error following the significant figures of a plaintext, the goal of bootstrapping is to evaluate the decryption formula approximately and compute an encryption of the original message in a large ciphertext modulus.

5.1 Decryption Formula over the Integers

The goal of this section is to suggest a method to evaluate the decryption formula of `HEAAN` scheme for bootstrapping. Its decryption formula consists of two parts − inner product $Z = \langle \mathbf{c}, sk \rangle$ over the integers (or integral polynomials) and modulus reduction $m = Z \pmod q$. Our first observation is that `HEAAN` can support an arithmetic operations over a characteristic zero space such as $\mathbb{Z}$ and $\mathbb{C}$, and so we need to evaluate the decryption formula homomorphically using arithmetic operations on a characteristic zero space.

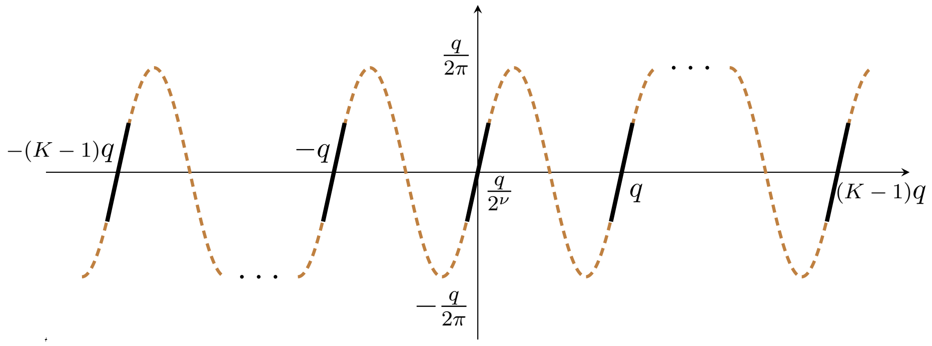
The main difficulty of the evaluation is that reduction of the result modulo q is not represented as a polynomial of a small degree. A naive approach such as polynomial interpolation method gives a huge degree polynomial, resulting in very large parameter size and high computational cost for bootstrapping process. Instead, we reduce the required circuit depth by allowing some error in polynomial approximation of decryption formula and taking advantage of approximate arithmetic.

5.1.1 Approximation to a Trigonometric Function

Let $\mathbf{c}$ be a ciphertext relative to secret key sk and modulus q. Since sk is sampled from a small distribution, the size of its decryption structure $Z = \langle \mathbf{c}, sk \rangle$ is bounded by Kq for a fixed constant K depending only on the HE parameter. Then we can say that a decryption formula of `HEAAN` is defined on the set $\mathbb{Z} \cap (-Kq, Kq)$ and it maps an arbitrary integer $z \in \mathbb{Z} \cap (-Kq, Kq)$ to $[z]_q$.

For the efficiency of approximation, we first assume that a message m of

Figure 5.1: Modular reduction and scaled sine functions



an input ciphertext is still much smaller than the ciphertext modulus q, so that $Z = qI + m$ for some I and m such that $\|I\|_\infty < K$ and $\|m\|_\infty \ll q$. Then the modulus reduction circuit $F(z) = [z]_q$ on a restricted domain becomes a *piecewise linear* function (see Fig.5.1). We point out that this piecewise linear function is periodic and identical near zero, so it looks like a part of the scaled sine function

$$G(z) = \frac{q}{2\pi} \sin\left(\frac{2\pi}{q} z\right).$$

Note that it is a good approximation of the piecewise linear function when an input value is nearby a multiple of q. Specifically, the maximum error between $F(z)$ and $G(z)$ is bounded by

$$|F(z) - G(z)| = \frac{q}{2\pi}\left|\frac{2\pi m}{q} - \sin\left(\frac{2\pi m}{q}\right)\right| \leqslant \frac{q}{2\pi}\cdot\frac{1}{3!}\left(\frac{2\pi}{q}\cdot\frac{q}{2^\nu}\right)^3 < \frac{q}{2^{3\nu-3}}$$

when a message $m = [z]_q$ is bounded by $|m| < q/2^\nu$ for some $\nu \geqslant 2$. In the RLWE-based construction of HEAAN, small *complex* errors are added to plaintext slots during encryption, evaluation, rescaling and slot permutation. Therefore, we have one constraint that a decryption formula should be tolerant of small complex errors. This approximation has an advantage, in that a complex error does not blow up by a scaled trigonometric function since it is analytic and $|G'(z)| \leqslant 1$ on a complex plane. Namely, it satisfies that

$|G(z+e)-G(z)| \leqslant |e|$ for arbitrary z and e in $\mathbb{C}$.

5.1.2 Evaluation Strategy

As discussed before, the scaled sine function $G(z)$ might be considered a good approximation of the decryption formula. Now our goal is to evaluate a trigonometric function efficiently using a HE for approximate arithmetic. A Taylor polynomial

$$T(z) = \frac{q}{2\pi}\sum_{j=0}^{n-1}\frac{(-1)^j}{(2j+1)!}\left(\frac{2\pi}{q}\cdot z\right)^{2j+1}$$

of $G(z)$ might be a candidate approximation since the size of error converges to zero very rapidly as n grows, i.e., the maximum error between $G(z)$ and $T(z)$ is bounded by

$$|G(z)-T(z)| \leqslant \frac{q}{2\pi}\cdot\frac{1}{(2n+1)!}\,(2\pi K)^{2n+1} \leqslant \frac{q}{\sqrt{8\pi^3(2n+1)}}\left(\frac{2\pi e\cdot K}{2n+1}\right)^{2n+1}$$

from the Stirling's formula, when $|z| < 2\pi K$. This bound becomes small enough when the degree of a Taylor polynomial is $\Omega(K)$.

Even though a Taylor polynomial approach achieves a good precision of approximation, there are some problems in practical implementation of evaluation phase. Since the factorial function grows asymptotically faster than exponential functions, it is difficult to make an evaluation strategy based on homomorphic operations and rescaling procedure for a Taylor polynomial. The main bottleneck is computational cost, that is, the complexity of evaluation increases exponentially with a depth of a circuit, e.g. $\mathcal{O}(\sqrt{n})$ using the Paterson-Stockmeyer algorithm [PS73] for the evaluation function of degree n.

To represent the decryption formula in a simple form and achieve higher speed of the evaluation, we use the identities

$$\begin{cases} \cos 2\theta = \cos^2\theta - \sin^2\theta \\ \sin 2\theta = 2\cos\theta\sin\theta. \end{cases}$$

From approximate values of trigonometric functions in a restricted domain, we extend to find good approximations on a wider (doubled) range.

Lemma 5.1.1 (Iteration Error)**.** *Let (α, w) be an approximation of trigonometric functions $(\cos\theta, \sin\theta)$ with an error $\mathbf{e}$ of size $\epsilon = \|\mathbf{e}\|_2$. Let $\alpha' \leftarrow \alpha^2 - \beta^2$ and $w' \leftarrow 2\alpha\beta$. Then $(\alpha', \beta') = (\cos 2\theta, \sin 2\theta) + \mathbf{e}'$ for some error $\mathbf{e}'$ such that $\|\mathbf{e}'\|_2 \leqslant 2\epsilon + \epsilon^2$.*

Proof. Let $e_1 = \alpha - \cos\theta$ and $e_2 = \beta - \sin\theta$. Then we get $\alpha' - \cos 2\theta = 2e_1\cos\theta - 2e_2\sin\theta + e_1^2 - e_2^2$ and $\beta' - \sin 2\theta = 2(e_1\sin\theta + e_2\cos\theta + e_1e_2)$ from the definition. A bound for the size of $\mathbf{e}'$ is obtained by

$$\begin{aligned} \|\mathbf{e}'\|_2^2 &= 4\left((e_1\cos\theta - e_2\sin\theta)^2 + (e_1\sin\theta + e_2\cos\theta)^2\right) + (e_1^2 - e_2^2)^2 + 4(e_1e_2)^2 \\ &\quad + 4(e_1\cos\theta - e_2\sin\theta)(e_1^2 - e_2^2) + 4(e_1\sin\theta + e_2\cos\theta)(2e_1e_2) \\ &= 4\epsilon^2 + \epsilon^4 + 4\epsilon^2(e_1\sin\theta + e_2\cos\theta) \leqslant 4\epsilon^2 + 4\epsilon^3 + \epsilon^4 = (2\epsilon + \epsilon^2)^2, \end{aligned}$$

from the Cauchy-Schwarz inequality. □

We start from high-precision approximations of trigonometric functions within a small range and repeat an evaluation of the above identities recursively to get an approximation of the scaled sine function in the desirable domain. The homomorphic evaluation of the scaled sine function consists of the following steps. Note that we multiply a scale factor Δ to prevent the precision loss during iterations. Division by a constant Δ will be performed by the rescaling process of `HEAAN`.

1. A value z is given such that the size is bounded by Kq.
2. Compute polynomials $C_0(z) \approx \Delta\cdot\cos\left(\frac{2\pi}{q}\cdot\frac{z}{2^t}\right)$ and $S_0(z) \approx \Delta\cdot\sin\left(\frac{2\pi}{q}\cdot\frac{z}{2^t}\right)$ using the Taylor series of a small degree d_0.

3. For $j = 0, 1, \ldots, t-1$, repeat the computations as $C_{j+1}(z) = \Delta^{-1} \cdot (C_j(z)^2 - S_j(z)^2)$ and $S_{j+1}(z) = \Delta^{-1} \cdot (2S_j(z)C_j(z))$.

4. Return $S_t(z)$.

Since the input $\frac{2\pi}{q} \cdot \frac{z}{2^t}$ is contained in the small interval $\left(-\frac{\pi K}{2^{t-1}}, \frac{\pi K}{2^{t-1}}\right)$, even the Taylor polynomials of a small degree d_0 can provide an enough accuracy. The output $S_t(z)$ is a polynomial of degree $d_t = d_0 \cdot 2^t$ and it is an approximation of the scaled trigonometric function $\Delta \cdot \sin\left(\frac{2\pi}{q}z\right) = \frac{2\pi \cdot \Delta}{q} G(z)$ over a wide interval $\frac{2\pi}{q} z \in (-2\pi K, 2\pi K)$. The main advantage of this method is that the complexity of whole evaluation grows linearly with the depth of the decryption formula.

Let d_0 be an initial degree for $C_0(z)$ and $S_0(z)$. Then the ℓ_2-norm of initial error $\Delta^{-1} \cdot (C_0(z), S_0(z)) - \left(\cos\left(\frac{2\pi}{q} \cdot \frac{z}{2^t}\right), \sin\left(\frac{2\pi}{q} \cdot \frac{z}{2^t}\right)\right)$ is bounded by

$$\frac{1}{(d_0+1)!}\left(\frac{2\pi}{q} \cdot \frac{Kq}{2^t}\right)^{d_0+1} + \frac{1}{(d_0+2)!}\left(\frac{2\pi}{q} \cdot \frac{Kq}{2^t}\right)^{d_0+2} \approx \frac{1}{(d_0+1)!}\left(\frac{\pi K}{2^{t-1}}\right)^{d_0+1}$$

from the Taylor remainder theorem. This bound is doubled after each iteration from Lemma 5.1.1. Therefore, we get a bound for an approximation as follows:

$$\begin{aligned} \left| S_t(z) - \Delta \cdot \sin\left(\frac{2\pi}{q} z\right) \right| &\leqslant \frac{\Delta \cdot 2^t}{(d_0+1)!}\left(\frac{\pi K}{2^{t-1}}\right)^{d_0+1} \\ &\leqslant \frac{\Delta \cdot 2^t}{\sqrt{2\pi(d_0+1)}}\left(\frac{\pi e \cdot K}{2^{t-1}(d_0+1)}\right)^{d_0+1} \end{aligned}$$

from the Stirling's formula.

Complexity. We can evaluate the Taylor polynomials of sine and cosine functions of degree d_0 in $2d_0$ homomorphic multiplications. Each of recursive step requires two multiplications, except the last step, which only needs to compute the sine part. Hence the number of homomorphic multiplications for whole evaluation is $2t - 1 + 2\log d_0$ for some constant $d_0 = \mathcal{O}(1)$.

5.2 Bootstrapping for HEAAN

5.2.1 Linear Transformations on Packed Ciphertexts

We first explain a method to homomorphically evaluate the linear transformations over the vector of plaintext slots. We make use of the rotation and conjugation operations as described below.

In general, an arbitrary linear transformation over the complex space $\mathbb{C}^{N/2}$ of plaintext slots can be represented as $\mathbf{z} \mapsto A \cdot \mathbf{z} + B \cdot \overline{\mathbf{z}}$ for some complex matrices $A, B \in \mathbb{C}^{N/2 \times N/2}$. It can be best done by handling the matrix in a diagonal order and making use of SIMD computation. Specifically, let $\mathbf{u}_j = (A_{0,j}, A_{1,j+1}, \ldots, A_{\frac{N}{2}-j-1,\frac{N}{2}-1}, A_{\frac{N}{2}-j,0}, \ldots, A_{\frac{N}{2}-1,j-1}) \in \mathbb{C}^{N/2}$ denote the shifted diagonal vector of A for $0 \leqslant j < N/2$. Then we have

$$A \cdot \mathbf{z} = \sum_{0 \leqslant j < N/2} (\mathbf{u}_j \odot \rho(\mathbf{z}; j)) \tag{5.2.1}$$

where $\rho(\mathbf{z}; j) = (z_j, \ldots, z_{\frac{N}{2}-1}, z_0, \ldots, z_{j-1})$ is the shifted vector of $\mathbf{z}$ by j positions and $\odot$ denotes the Hadamard (component-wise) multiplication between vectors.

Therefore, the matrix multiplication $A \cdot \mathbf{z}$ is expressed as combination of rotations and scalar multiplications. The vector rotation $\rho(\mathbf{z}; j)$ can be homomorphically evaluated by $\mathsf{Rot}_{rk}(\mathbf{c}; j)$ while the Hadamard (component wise) scalar multiplication is done by multiplying the polynomials $\tau^{-1}(\mathbf{u}_j)$. See Algorithm 3 for an explicit description of the homomorphic matrix multiplication. Similarly, the second term $B \cdot \overline{\mathbf{z}}$ is obtained by taking slot-wise conjugation $\mathsf{Conj}_{ck}(\cdot)$ and multiplying matrix B.

For a precise evaluation of a matrix operation, we might multiply a scale factor $\Delta \geqslant 1$ to the polynomial $\tau^{-1}(\mathbf{u}_j)$ before rounding, so that we can reduce the relative size of the rounding error and maintain the precision of the resulting plaintext. Since the coefficients of a rounding error is bounded by $1/2$, its size relative to the canonical embedding norm is bounded by $N/2$. Then the ciphertext $\mathbf{c}_j$ will be an encryption of $\Delta \cdot \mathbf{u}_j \odot \rho(\mathbf{z}; j)$ with an error

Algorithm 3 Matrix Multiplication

procedure MatMult($\mathbf{c} \in \mathcal{R}_q^2, A \in \mathbb{C}^{N/2 \times N/2}$)
  $\mathbf{c}' \leftarrow \lfloor \tau^{-1}(\mathbf{u}_0) \rceil \cdot \mathbf{c} \pmod q$
  for $j = 1$ to $N/2 - 1$ **do**
    $\mathbf{c}_j \leftarrow \lfloor \tau^{-1}(\mathbf{u}_j) \rceil \cdot \mathsf{Rot}_{rk_j}(\mathbf{c}; j) \pmod q$
    $\mathbf{c}' \leftarrow \mathsf{Add}(\mathbf{c}', \mathbf{c}_j) \pmod q$
  end for
  return $\mathbf{c}'$
end procedure

bounded by $\Delta \cdot B_{\mathsf{rs}} + (N/2) \cdot \|\mathbf{z}\|_\infty$. After aggregating the ciphertexts $\mathbf{c}_j$ and applying a rescaling procedure by a scalar Δ, we get a desired ciphertext with an error bounded by $(N/2) \cdot B_{\mathsf{rs}} + \Delta^{-1} \cdot (N/2)^2 \cdot \|\mathbf{z}\|_\infty$.

There is an optimization technique that reduces the size of rescaling error. If we carry out all the homomorphic computations in a large modulus $P \cdot q$ and take only the resulting ciphertext back to the ordinary modulus, then we will have only one rescaling error in the output ciphertext. Hence the final error of matrix multiplication can be bounded by $B_{\mathsf{rs}} + \Delta^{-1} \cdot (N/2)^2 \cdot \|\mathbf{z}\|_\infty$.

5.2.2 An Overview of Recryption Procedure

This section gives a high level structure of the bootstrapping process for the `HEAAN` scheme. We employ the ciphertext packing method and combine it with our efficient evaluation strategy to achieve a better performance in terms of memory and computation cost.

Let $\mathbf{c}$ be an input ciphertext of bootstrapping procedure with a ciphertext modulus q satisfying $m(X) = [\langle \mathbf{c}, sk \rangle]_q$. We start with the point that its decryption structure $Z(X) = \langle \mathbf{c}, sk \rangle \pmod{\Phi_M(X)}$ over the integers is of the form $Z = qI + m$ for some small $I(X) \in \mathcal{R}$ with a bound $\|I\|_\infty < K$. Hence $\mathbf{c}$ itself can be considered as an encryption of $Z(X) = Z_0 + Z_1 X + \cdots + Z_{N-1} X^{N-1}$ in a large ciphertext modulus $Q_0 \gg q$ due to $[\langle \mathbf{c}, sk \rangle]_{Q_0} = Z(X)$. Our bootstrapping procedure aims to homomorphically evaluate the reduction modulo q using arithmetic operations over the integers, so that we can generate an encryption of the original message $m = [Z]_q$ with a ciphertext modulus larger than q.

Below we describe all the parts of recryption procedure in more details. We denote the following three steps by CoeffToSlot, Eval and SlotToCoeff, respectively. See Fig.5.2 for an illustration.

Figure 5.2: Bootstrapping process

$$\boxed{\mathbf{c} = \mathsf{Enc}(Z(X)) \pmod{q}}$$

$$\Big\downarrow \ \textsc{CoeffToSlot}$$

$$\boxed{\begin{array}{l} \tilde{\mathbf{c}}_0 = \mathsf{Enc}(Z_0, \ldots, Z_{N/2-1}) \pmod{Q_1} \\ \tilde{\mathbf{c}}_1 = \mathsf{Enc}(Z_{N/2}, \ldots, Z_{N-1}) \pmod{Q_1} \end{array}}$$

$$\Big\downarrow \ \textsc{Eval}$$

$$\boxed{\begin{array}{l} \tilde{\mathbf{c}}_0' = \mathsf{Enc}(m_0, \ldots, m_{N/2-1}) \pmod{Q_2} \\ \tilde{\mathbf{c}}_1' = \mathsf{Enc}(m_{N/2}, \ldots, m_{N-1}) \pmod{Q_2} \end{array}}$$

$$\Big\downarrow \ \textsc{SlotToCoeff}$$

$$\boxed{\mathbf{c}' = \mathsf{Enc}(m(X)) \pmod{Q_3}}$$

Putting polynomial coefficients in plaintext slots. Given the input ciphertext $\mathbf{c} \in \mathcal{R}_q^2$ with a decryption structure $Z(X) = \langle \mathbf{c}, sk \rangle$, this step converts it into ciphertexts with a modulus $Q_1 \gg q$ that contains the coefficients of $Z(X) = Z_0 + Z_1 X + \cdots + Z_{N-1} X^{N-1}$ in their plaintext slots. Let $\mathbf{z} = \tau(Z) \in \mathbb{C}^{N/2}$ be the vector of plaintext slots of ciphertext $\mathbf{c}$. Since each ciphertext can store at most $N/2$ plaintext values, we will generate two ciphertexts $\tilde{\mathbf{c}}_0, \tilde{\mathbf{c}}_1 \in \mathcal{R}_{Q_1}^2$ that encrypt the vectors $\tilde{\mathbf{z}}_0 = (Z_0, \ldots, Z_{\frac{N}{2}-1})$ and $\tilde{\mathbf{z}}_1 = (Z_{\frac{N}{2}}, \ldots, Z_{N-1})$, respectively.

We recall the linear relation between the coefficient vector of a polynomial and its corresponding vector of plaintext slots mentioned in Section 3.2. If we divide the matrix U into two square matrices

$$U_0 = \begin{bmatrix} 1 & \zeta_0 & \dots & \zeta_0^{\frac{N}{2}-1} \\ 1 & \zeta_1 & \dots & \zeta_1^{\frac{N}{2}-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \zeta_{\frac{N}{2}-1} & \dots & \zeta_{\frac{N}{2}-1}^{\frac{N}{2}-1} \end{bmatrix} \quad \text{and} \quad U_1 = \begin{bmatrix} \zeta_0^{\frac{N}{2}} & \zeta_0^{\frac{N}{2}+1} & \dots & \zeta_0^{N-1} \\ \zeta_1^{\frac{N}{2}} & \zeta_1^{\frac{N}{2}+1} & \dots & \zeta_1^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ \zeta_{\frac{N}{2}-1}^{\frac{N}{2}} & \zeta_{\frac{N}{2}-1}^{\frac{N}{2}+1} & \dots & \zeta_{\frac{N}{2}-1}^{N-1} \end{bmatrix},$$

then we get an identity $\tilde{\mathbf{z}}_k = \frac{1}{N}(\overline{U_k}^T \cdot \mathbf{z} + U_k^T \cdot \overline{\mathbf{z}})$ for $k = 0, 1$. Therefore, the ciphertexts $\tilde{\mathbf{c}}_0$ and $\tilde{\mathbf{c}}_1$ can be generated using linear transformations from a plaintext vector $\mathbf{z}$ to the vectors $\tilde{\mathbf{z}}_0$ and $\tilde{\mathbf{z}}_1$. As discussed in Section 5.2.1, our general methodology for linear transformations over the plaintext slots can be applied to an evaluation of this procedure.

Evaluation of decryption formula. The next procedure takes as inputs the resulting ciphertexts of previous step and evaluates the piecewise linear function $F(z)$, which is a decryption formula of `HEAAN`. We use our approximation method to a trigonometric function $S_t(z)$ and adapt the iterative evaluation strategy from $C_0(z)$ and $S_0(z)$ to $S_t(z)$ to reduce the required number of homomorphic multiplications.

Let $\tilde{\mathbf{c}}'_k$ be the resulting ciphertexts of the evaluation using $\tilde{\mathbf{c}}_k$ for $k = 0, 1$. Since each plaintext slot of $\tilde{\mathbf{c}}_0$ and $\tilde{\mathbf{c}}_1$ contains a coefficient $Z_j = qI_j + m_j$ for some $0 \leqslant j < N$, the output ciphertexts will encrypt the vectors $\tilde{\mathbf{z}}'_0 = (m_0, \dots, m_{\frac{N}{2}-1})$ and $\tilde{\mathbf{z}}'_1 = (m_{\frac{N}{2}}, \dots, m_{N-1})$ in the slots.

Switching back to coefficient representation. The final step is to pack all the coefficients m_j in the plaintext slots back in a single ciphertext. This process is exactly the inverse of the CoeffToSlot transformation. Suppose that we have two ciphertexts that encrypt the vectors $\tilde{\mathbf{z}}'_0 = (m_0, \dots, m_{\frac{N}{2}-1})$ and $\tilde{\mathbf{z}}'_1 = (m_{\frac{N}{2}}, \dots, m_{N-1})$. We aim to generate a ciphertext $\mathbf{c}'$ such that $\langle \mathbf{c}', sk \rangle \pmod{Q_3} \approx m(X)$ for some integer Q_3. Since the plaintext vector $\mathbf{z} = \tau(m)$ corresponding to $m(X)$ satisfies the identity $\mathbf{z} = U \cdot \mathbf{m} = U_0 \cdot \tilde{\mathbf{z}}'_0 +$

$U_1 \cdot \tilde{\mathbf{z}}_1'$, this transformation is also represented as a linear transformation from the plaintext vectors $\mathbf{z}_0$ and $\mathbf{z}_1$.

The first and final linear transformations consume only one level for scalar multiplications but require a number of slot rotations. The evaluation step evaluates a polynomial $S_t(z)$ homomorphically. This procedure consumes the most amount of levels during recryption, but the computational cost is comparably small from our recursive evaluation strategy.

5.2.3 Estimation of Noise and Complexity

In this subsection, we describe each step for recryption procedure in more detail with specific algorithms and noise estimation. We start with an analysis about an upper bound on $\|I\|_\infty$. Since each coefficient of a ciphertext $\mathbf{c} = (c_0, c_1)$ belong to $\mathbb{Z}_q$, each of coefficient of $\langle \mathbf{c}, sk \rangle$ is the sum of $(h+1)$ elements in $\mathbb{Z}_q$ which is bounded by $\frac{q}{2}(h+1)$. Hence every coefficient of $I(X) = \lfloor \frac{1}{q}\langle \mathbf{c}, sk \rangle \rceil$ is bounded by $\frac{1}{2}(h+1) \approx \frac{1}{2}\|s\|_1$. In practice, the coefficients of c_i look like a random variable over the interval $\mathbb{Z}_q$ and a coefficient of $\frac{1}{q}\langle \mathbf{c}, sk \rangle$ behaves as the sum of $(h+1)$-number of i.i.d. uniform and random variable on $(-\frac{1}{2}, \frac{1}{2})$. This heuristic assumption gives us a smaller bound for $\|I\|_\infty$.

When implementing a matrix operation for linear transformation, an additional error is added to a plaintext during the key-switching procedure such as rotation and conjugation. The key-switching error is bounded by $P^{-1} \cdot q \cdot B_{\mathsf{ks}} + B_{\mathsf{rs}}$ from Lemma 3.5.3, but the first term $P^{-1} \cdot q \cdot B_{\mathsf{ks}}$ is always very small compared to B_{rs} because q is much smaller than P. Hence we will ignore this term during analysis and set the same bound B_{rs} for key-switching and rescaling errors.

First, we perform the matrix multiplication as described in Section 5.2.1, perform the conjugation operation and apply a rescaling procedure by a scalar N. As a result, we obtain a ciphertext $\tilde{\mathbf{c}}_0 \in \mathcal{R}_{Q_1}^2$ that encrypts $(Z_0, \ldots, Z_{\frac{N}{2}-1})$ with an additional error bounded by $2(\Delta^{-1} \cdot (N/2)^2 \cdot \|Z\|_\infty + B_{\mathsf{rs}})$ from two matrix multiplications. In a similar way, we get a ciphertext $\tilde{\mathbf{c}}_1 \in \mathcal{R}_{Q_1}^2$ encrypting $(Z_{\frac{N}{2}}, \ldots, Z_{N-1})$ with the same error bound.

We now take $\tilde{\mathbf{c}}_0$ and $\tilde{\mathbf{c}}_1$ as inputs of evaluation of approximate polynomial $S_t(z)$. Each of plaintext slots contains $Z_j + e_j$ for some $0 \leqslant j < N$, such that $Z_j = qI_j + m_j$ and a small error e_j. We use our approximation polynomial $S_t(z)$ to evaluate the decryption formula $F(Z) = [Z]_q$ and generate an encryption of $\tilde{\mathbf{z}}'_0 = (m_0, \ldots, m_{\frac{N}{2}-1})$ and $\tilde{\mathbf{z}}'_1 = (m_{\frac{N}{2}}, \ldots, m_{N-1})$. The effect of an input error e_j and an polynomial approximation error of the decryption formula can be measured by

$$\begin{aligned}|F(Z_j) - \frac{q}{2\pi\Delta} S_t(Z_j + e_j)| \leqslant\ & |F(Z_j) - G(Z_j)| + |G(Z_j) - G(Z_j + e_j)| \\ & + |G(Z_j + e_j) - \frac{q}{2\pi\Delta} S_t(Z_j + e_j)|.\end{aligned}$$

The first term is bounded by $q \cdot 2^{-3\nu+3}$ from Section 5.1.1 and the second term is bounded by $|e_j| \leqslant \Delta^{-1} \cdot (N/2)^2 \cdot \|Z\|_\infty + B_{\mathsf{rs}}$ as described above. The third term is bounded by $\frac{q}{2\pi\Delta} \frac{2^t}{(d_0+1)!} \left(\frac{\pi K}{2^{t-1}}\right)^{d_0+1}$ from Subsection 5.1.2. We also have an additional error that comes from the key-switching and rescaling processes during homomorphic evaluation of $S_t(\cdot)$. An initial evaluation of $\Delta \cdot C_0(\cdot)$ and $\Delta \cdot S_0(\cdot)$ have an evaluation error of size $\leqslant B_{\mathsf{rs}}$, and it is doubled and added to B_{rs} after each iteration. Hence the evaluation error after t iterations is roughly bounded by $2^{t+1} \cdot B_{\mathsf{rs}}$. Finally the inverse linear transformation will generate a small error bounded by $\Delta^{-1} \cdot (N/2)^2 \cdot \|m\|_\infty^{\mathsf{can}} + B_{\mathsf{rs}} \leqslant \Delta^{-1} \cdot (N/2)^2 \cdot \|Z\|_\infty + B_{\mathsf{rs}}$, similar to the case of the first linear transformation step.

By combining above noise estimations, we conclude that the output ciphertext of a whole pipeline is an encryption of the original message $m(X)$ with an additional noise bounded by

$$\frac{q}{2^{3\nu-3}} + 2\left(\frac{(N/2)^2 \cdot \|Z\|_\infty}{\Delta} + B_{\mathsf{rs}}\right) + \frac{q}{2\pi} \frac{2^t}{(d_0+1)!} \left(\frac{\pi K}{2^{t-1}}\right)^{d_0+1} + \frac{q \cdot 2^{t+1}}{2\pi\Delta} \cdot B_{\mathsf{rs}}.$$

Asymptotically, this noise bound can be $\mathcal{O}(B_{\mathsf{rs}})$ when the parameters ν, Δ, d_0 and t are large enough. In the following section, we give some experimental results to show that a bootstrapping noise is small enough under an appropriate choice of parameter, so that it would not destroy the significant bits of input message.

Complexity of our bootstrapping. For simplicity, let $t = \log(Kq)$ and $\Delta = 2^t q$. Then the second term of error bound is less than $\frac{q}{2\pi}\frac{2^t}{(d_0+1)!}(\frac{4\pi}{q})^{d_0+1}$ and it is even less than 1 if d_0 is larger than 3. The third term is about $\frac{1}{\pi}B_{\mathsf{rs}}$, which is less than B_{rs}. Finally we can ignore the last term because the Δ is much larger than $\|Z\|_\infty$ and N^2.

- CoeffToSlot/SlotToCoeff: N rotations with N scalar multiplications. This part can be optimized to $\mathcal{O}(\sqrt{N})$ rotations with $\mathcal{O}(N)$ scalar multiplications using baby-giant step method mentioned in Section 6.1.

- Eval: Since we have to evaluate a trigonometric function with two output ciphertexts of CoeffToSlot step, the number of homomorphic multiplications is $4t + 4\log d_0 - 2 = \mathcal{O}(\log(Kq))$.

5.3 Implementation

In this section, we give some experimental results based on recommended parameter sets. We will not compare the result of implementation with other bootstrapping methods. In the case of other HE schemes which support exact computation, it is inefficient to recrypt a ciphertext with a large plaintext space (e.g. 8∼24 bits). Therefore, it seems less meaningful to compare performance to other bootstrapping implementation with similar parameters. Every experimentation was performed on a machine with an Intel® Xeon® CPU E5-2620 v4 at 2.10 GHz processor with the single thread setting using parameter sets of 80-bit security level.

5.3.1 Optimized Matrix Multiplication

As described in Algorithm 3, the naive method for a matrix multiplication requires $N/2-1$ rotations, but we can reduce the number of operations down to $\mathcal{O}(\sqrt{N})$. Let $n_1 = \mathcal{O}(\sqrt{N})$ and denote $n_2 = \lfloor N/2d \rceil$. It follows from [HS15] that Equation (5.2.1) can be expressed as

$$\begin{aligned} A \cdot \mathbf{z} &= \sum_{0 \leqslant j < n_2} \sum_{0 \leqslant i < n_1} (\mathbf{u}_{n_1 \cdot j + i} \odot \rho(\mathbf{z}; n_1 \cdot j + i)) \\ &= \sum_{0 \leqslant j < n_2} \rho \left(\sum_{0 \leqslant i < n_1} \rho(\mathbf{u}_{n_1 \cdot j + i}, -n_1 \cdot j) \odot \rho(\mathbf{z}; i); n_1 \cdot j \right). \end{aligned}$$

For the homomorphic evaluation of this circuit, we first compute the ciphertexts of $\rho(\mathbf{z}; k)$ for $i = 1, \ldots, n_1 - 1$. For each index i, we perform n_1 constant multiplications and aggregate the resulting ciphertexts.

In total, the matrix multiplication can be evaluated homomorphically with $(n_1 - 1) + (n_2 - 1) = \mathcal{O}(\sqrt{N})$ rotations and $n_1 \cdot n_2 = \mathcal{O}(N)$ constant multiplications.

5.3.2 Recryption with Sparsely Packed Ciphertexts

Algorithm 4 Homomorphic evaluation of the subtotal procedure

```
1: procedure SubTotal(c ∈ R_q^2, d = 2^r, ℓ = N/2d)
2:     c' ← c (mod q)
3:     for j = 0 to r − 1 do
4:         c_j ← Rot_{rk_j}(c'; ℓ · 2^j) (mod q)
5:         c' ← Add(c', c_j) (mod q)
6:     end for
7:     return c'
8: end procedure
```

If we use "sparsely-packed" ciphertexts, then we can reduce the complexity of CoeffToSlot procedure. Our first observation is that the only coefficients $(m_0, m_{\frac{N}{2\ell}}, ..., m_{N-\frac{N}{2\ell}})$ are nonzero if a message $m(X)$ is an encoding of a vector with $\ell < N/2$ number of slots. On the other hand, Algorithm 4 takes as input an encryption $\mathsf{Enc}(m(X) + qI(X))$ with $I(X) = I_0 + I_1 X + ... + I_{N-1} X^{N-1}$, and outputs an encryption $\mathsf{Enc}(\frac{N}{2\ell}(m(X) + qI'(X)))$ where $I'(X) = I_0 + I_{\frac{N}{2\ell}} X^{\frac{N}{2\ell}} + ... + I_{N-\frac{N}{2\ell}} X^{N-\frac{N}{2\ell}}$ is obtained from $I(X)$ by erasing all the coefficients whose index is not divisible by $\frac{N}{2\ell}$.

In CoeffToSlot step, we only need to compute an encryption of $\mathbf{z}' = (m_0 + qI_0, m_{\frac{N}{2\ell}} + qI_{\frac{N}{2\ell}} \ldots, m_{N-\frac{N}{2\ell}} + qI_{N-\frac{N}{2\ell}})$, instead of computing two ciphertexts as before. Similar to Section 3.2, $\mathbf{z}'$ can be computed by $\mathbf{z}' = \frac{1}{N}(\overline{U'}^T \cdot \textsc{SubTotal}(\mathbf{z}) + U'^T \cdot \overline{\textsc{SubTotal}(\mathbf{z})})$ where

$$U' = \begin{bmatrix} 1 & \zeta_0^{\frac{N}{2\ell}} & \zeta_0^{\frac{2N}{2\ell}} & \cdots & \zeta_0^{N-\frac{N}{2\ell}} \\ 1 & \zeta_{\frac{N}{4\ell}}^{\frac{N}{2\ell}} & \zeta_{\frac{N}{4\ell}}^{\frac{2N}{2\ell}} & \cdots & \zeta_{\frac{N}{4\ell}}^{N-\frac{N}{2\ell}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \zeta_{\frac{N}{2}-\frac{N}{4\ell}}^{\frac{N}{2\ell}} & \zeta_{\frac{N}{2}-\frac{N}{4\ell}}^{\frac{2N}{2\ell}} & \cdots & \zeta_{\frac{N}{2}-\frac{N}{4\ell}}^{N-\frac{N}{2\ell}} \end{bmatrix}.$$

This optimization can reduce the complexity of CoeffToSlot step, from two matrix multiplications of size $N/2 \times N/2$ down to only one of size $2\ell \times 2\ell$. Finally, the SlotToCoeff step can be computed using the same method as the first one.

5.3.3 Experimental Results

In our experiment, we processed for a number of slots from 1 to 2^7. Table 5.1, Figure 5.3 and Figure 5.4 show our result of bootstrapping. In Table 5.1, the first row gives the ring dimension of RLWE setting and we only used power of two integers. The second row gives the largest ciphertext modulus which is used in bootstrapping. The third row gives the bit precision of input message and the fourth row gives the ciphertext modulus of the input for bootstrapping. The other rows show the specific parameters from Section 5.2.3 and timing results for bootstrapping. The before and after levels are computed by dividing the largest and result ciphertext modulus by the bit size of message block, which is $(\log q - \nu)$. It follows from that fact that the size of the modulus for rescaling operation is the same with the message block size in HEAAN.

In Figure 5.3, we show the bootstrapping timings for various numbers of

Table 5.1: Experimental result with a single integral message

Params	Set-I	Set-II	Set-III	Set-IV
N	32768	32768	65536	65536
$\log Q$	620	620	1240	1240
Message	8 bits	12 bits	16 bits	24 bits
$\log q$	29	37	41	54
ν	6	10	10	15
d_0	7	7	7	7
t	6	7	7	9
Before/After levels	26/11	22/5	40/23	31/11
CoeffToSlot	9.1 sec	9.2 sec	49.1 sec	49 sec
Eval	20.1 sec	20.9 sec	110.9 sec	120.1 sec
Total	29.2 sec	30.1 sec	160 sec	169.1 sec

slots (each slot contains one complex number). In Figure 5.4, we show the amortized timings, which mean relative times per slot. We can notice that the amortized time becomes lower when the number of slots is increasing.

Figure 5.3: Timings for bootstrapping with the variation of number of slots (bootstrapping for complex numbers)

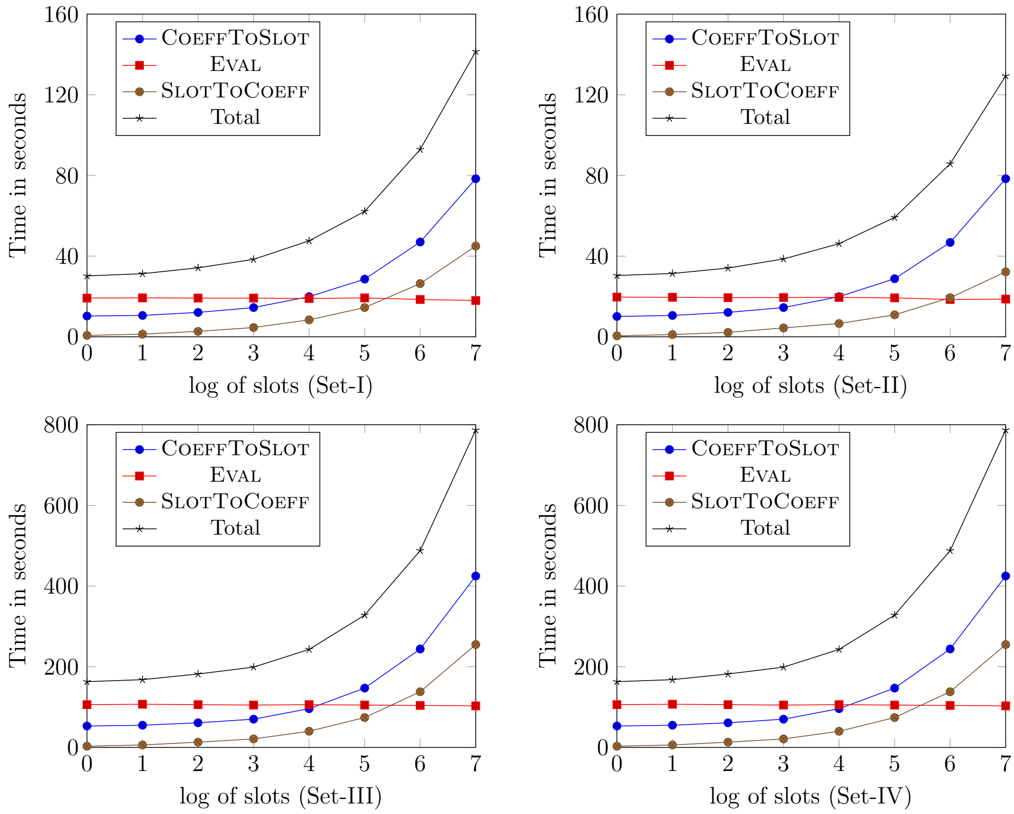
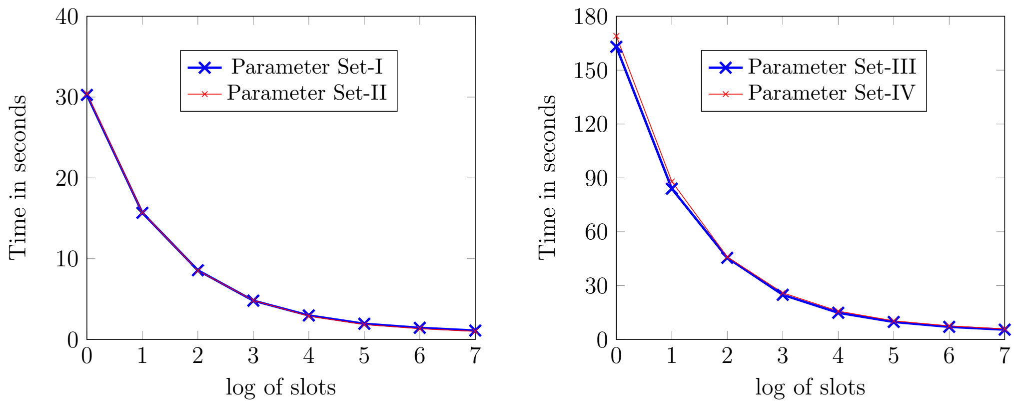


Figure 5.4: Amortized timings for bootstrapping with the variation of number of slots (bootstrapping for complex numbers)



Chapter 6

Privacy-Preserving Logistic Regression

Learning a model without accessing raw data has been an intriguing idea to security and machine learning researchers for years. In an ideal setting, we want to encrypt sensitive data to store them on a commercial cloud and run analysis without ever decrypting the data to preserve the privacy. Homomorphic encryption technique is a perfect match for secure data outsourcing but it is a very challenging task to support real-world machine learning tasks. Existing framework can only handle simplified cases with low-degree polynomials such as linear means classifier and linear discriminative analysis. But there was no practical support to the mainstream learning models. We present the first homomorphically encrypted logistic regression model based on the critical observation that a precision loss of classification models is sufficiently small so that the decision plan stays still. We innovated on: (1) a novel homomorphic encryption scheme optimized for real numbers computation, (2) the least squares approximation of the logistic function for accuracy and efficiency (i.e., reduce computation cost), and (3) new packing and parallelization techniques. Using real world datasets, we demonstrated the feasibility of our model in speed and memory consumption.

Related Works. Graepel et al. throw lights on machine learning with homomorphically encrypted data [GLN12]. The paper discussed scenarios that are appropriate and inappropriate to exercise machine learning with HE techniques. Authors provided two examples: linear means classifier and linear discriminative analysis, which can be achieved by using low-degree polynomials in HE. However, these simple parametric models do not handle complex datasets well and they are not representing the mainstream machine learning technologies used in biomedical research [DOM02, MWW+17]. Additional work was carried out by [BLN14] to demonstrate the feasibility of making a prediction on encrypted medical data in the Microsoft's Azure cloud. But instead of learning from the data, this model only makes prediction using learned logistic regression models in a privacy-preserving manner. Similarly, a more recent work called Cryptonets [GBDL+16] applied neural networks to encrypted data only for the evaluation purpose. To the best of our knowledge, no existing method is capable of carrying out regular machine learning tasks with encrypted data in a real-world setting. There are several prominent challenges related to scalability and efficiency. Traditional methods cannot handle many iterations of multiplications, which lead to a deep circuit and an exponential growth on the computational cost and storage size of the ciphertext. On the other hand, it is a non-trivial task to approximate some critical functions in machine learning models using only low-degree polynomials. Naive approximation might lead to big errors and make the solutions intractable. Our framework proposes novel methods to handle these challenges and makes it possible to learn a logistic regression model on encrypted data based completely on homomorphic encryption.

6.1 Background

Logistic regression is a widely used learning model in biomedicine [DOM02]. Data for supervised learning consists of pairs $(\mathbf{x}_i, y_i)$ of a vector of co-variates $\mathbf{x}_i = (x_{i1}, \cdots, x_{id}) \in \mathbb{R}^d$ and a class label y_i for $i = 1, \cdots, n$. We assume that $y_i \in \{\pm 1\}$ for binary classification (1 for positive and -1 for negative). Mathematically, it estimates a *multiple linear regression* function defined as

$$\log\left(\frac{\Pr[y_i = 1|\mathbf{x}_i]}{1 - \Pr[y_i = 1|\mathbf{x}_i]}\right) = w_0 + \sum_{j=1}^{d} w_j x_{ij}$$

where $\mathbf{w} = (w_0, w_1, w_2, \cdots, w_d)$ is the model parameter to be estimated. It can be alternatively represented as $\Pr[y_i = 1|\mathbf{x}_i] = \sigma((1, \mathbf{x}_i)^T \mathbf{w})$ for the sigmoid function $\sigma(x) = \frac{1}{1+\exp(-x)}$. Training methods of logistic regression aim to find an optimal $\mathbf{w}$ which maximizes the likelihood estimator

$$\prod_{i=1}^{n} \Pr(y_i|\mathbf{x}_i) = \prod_{i=1}^{n} \frac{1}{1 + \exp(-y_i(1, \mathbf{x}_i)^T \mathbf{w})},$$

or equivalently, minimizes the cost (loss) log-likelihood function

$$\frac{1}{n}\sum_{i=1}^{n} \log(1 + \exp(-\mathbf{z}_i^T \mathbf{w}))$$

for $\mathbf{z}_i = y_i \cdot (1, \mathbf{x}_i) \in \mathbb{R}^{d+1}$.

6.2 Privacy-Preserving Logistic Regression

Unlike linear regression, logistic regression does not have a closed formula solution in most cases. As a result, we need to use nonlinear optimization methods to find the maximum likelihood estimators of the regression parameters. Newton Raphson [Ypm95] and the gradient descent [Bot10] are the most commonly used methods for training. Since the Newton's method involves matrix inversion and HE schemes do not naturally support division or matrix inversion, it is difficult to evaluate Newton's method with HE schemes. On the other hand, gradient descent does not require the division operation, and therefore is a better candidate for homomorphically encrypted logistic regression. Thus we choose gradient descent algorithm as the training method for logistic regression.

Let $(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \{\pm 1\}$ be the supervised learning samples for $i = 1, \cdots, n$. If we write $\mathbf{z}_i = y_i \cdot (1, \mathbf{x}_i) \in \mathbb{R}^{d+1}$, then the cost function for logistic regression is defined by $\frac{1}{n}\sum_{i=1}^{n} \log(1 + \exp(-\mathbf{z}_i^T \mathbf{w}))$. Its gradient with respect to $\mathbf{w}$ is computed by $-\frac{1}{n}\sum_{i=1}^{n} \sigma(-\mathbf{z}_i^T \mathbf{w}) \cdot \mathbf{z}_i$. To find a local minimum point, the gradient descent method updates the regression parameters using the equation

$$\mathbf{w} \leftarrow \mathbf{w} + \frac{\alpha}{n} \sum_{i=1}^{n} \sigma(-\mathbf{z}_i^T \mathbf{w}) \cdot \mathbf{z}_i,$$

where α is the learning rate.

6.2.1 Polynomial Approximation

Although the gradient descent method seems better suited than other training methods for the homomorphic evaluation, some technical problems remain for implementation. In the above update formula, the sigmoid function is the biggest obstacles since the existing HE schemes only allow evaluation of polynomial functions. Hence the Taylor polynomials $T_d(x) = \sum_{k=0}^{d} \frac{f^{(k)}(0)}{k!} x^k$ have been commonly used for approximation of the sigmoid function ([BLN14, MZ17]):

$$\sigma(x) = \frac{1}{2} + \frac{1}{4}x - \frac{1}{48}x^3 + \frac{1}{480}x^5 - \frac{17}{80640}x^7 + \frac{31}{1451520}x^9 + \mathcal{O}(x^{11}).$$

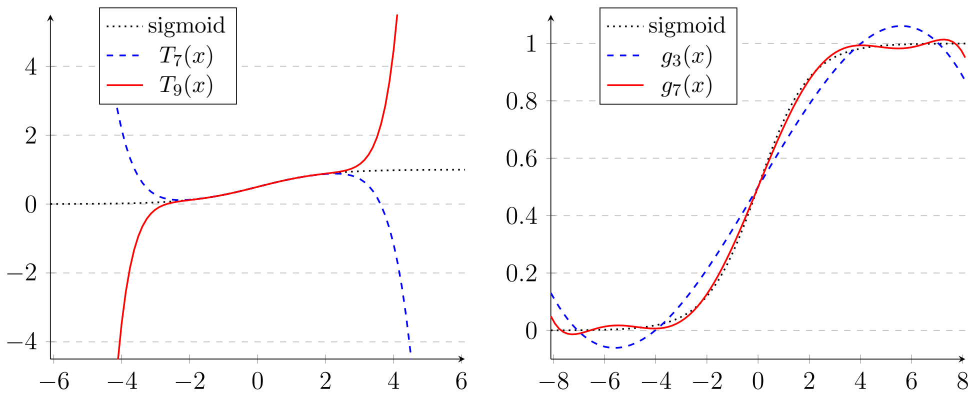
We observed the input values $(-\mathbf{z}_i^T \mathbf{w})$ of sigmoid function during iterations on real-world datasets and concluded that the Taylor polynomial $T_9(x)$ of degree 9 is still not enough to obtain the desired accuracy (see Fig 6.1). The size of error grows rapidly as $|x|$ increases, for instance, we have $T_9(4) \approx 4.44$, $T_9(6) \approx 31.23$, and $T_9(8) \approx 138.12$. In addition, we have to use a higher degree Taylor polynomial to guarantee the accuracy of regression, but it requires too large depth and many homomorphic multiplications to be practically implemented.

In short, the Taylor polynomial is not a good candidate for approximation because it is a *local* approximation near a certain point. Instead we adapt a *global* approximation method which minimizes the *mean squared error*. For an integrable function f, its mean square over an interval I is defined by

$$\frac{1}{|I|}\int_I f(x)^2 dx$$

where $|I|$ denotes the length of I. The least squares method aims to find a polynomial $g(x)$ of degree d which minimizes the mean squared error $\frac{1}{|I|}\int_I (g(x)-f(x))^2 dx$. The least squares approximation has a closed formula and it can be efficiently calculated using linear algebra.

Figure 6.1: Taylor polynomials (left) and Least squares (right)



In our implementation, we use the degree 3 and 7 least squares approximations of sigmoid function over the interval $[-8, 8]$ which contains all of the input values $(\mathbf{z}_i^T\mathbf{w})$ during iterations. The least square polynomials are computed as $g_3(x) = 0.5 + a_1(x/8) + a_3(x/8)^3$ and $g_7(x) = 0.5 + b_1(x/8) + b_3(x/8)^3 + b_5(x/8)^5 + b_7(x/8)^7$ for the coefficients vector $(a_1, a_3) \approx (1.20096, -0.81562)$ and $(b_1, b_3, b_5, b_7) \approx (1.73496, -4.19407, 5.43402, -2.50739)$. The degree 3 least squares approximation requires a smaller depth for evaluation while the degree 7 polynomial has a better precision (see Fig 6.1).

6.2.2 Homomorphic Evaluation of GD Algorithm

We will describe how to encode data and explain how to analyze logistic regression on encrypted data. Our idea is to use batching with n slots and perform n evaluations in parallel, where n is the number of training data samples.

We start with a useful aggregation operation across plaintext slots. Specifically, given a ciphertext representing plaintext vector $(m_1, \cdots, m_\ell)$, we introduce an algorithm which generates a ciphertext representing $\sum_{i=1}^{\ell} m_i$ in each plaintext slot [CKK15, CKK16]. Assume that ℓ is chosen as a power of two integer. The cyclic rotation by one produces a ciphertext encrypting the plaintext vector $(m_2, \cdots, m_\ell, m_1)$. Then an encryption of the vector $(m_1 + m_2, m_2 + m_3, \cdots, m_\ell + m_1)$ is obtained by adding the original ciphertext. We repeatedly apply this method $\log \ell - 1$ times with a rotation by a power of two which generates a desired ciphertext, that is, every plaintext slot contains the same value $\sum_{i=1}^{\ell} m_i$. The total procedure is described in Algorithm 5.

Algorithm 5 $\mathsf{AllSum}(\mathsf{ct})$

Input: Ciphertext ct encrypting plaintext vector $(m_1, \cdots, m_\ell)$
Output: Ciphertext encrypting $\sum_{i=1}^{\ell} m_i$ in each plaintext slot

```
for i = 0, 1, ..., log ℓ − 1 do
    Compute ct ← Add(ct, Rot(ct; 2^i))
end for
return ct
```

Let us now assume that we are given n training data samples $\mathbf{z}_i$ with $(d+1)$ features. As mentioned before, our goal is to securely evaluate the following arithmetic circuit:

$$\mathbf{w} \leftarrow \mathbf{w} + \frac{\alpha}{n} \sum_{i=1}^{n} g(-\mathbf{z}_i^T \mathbf{w}) \cdot \mathbf{z}_i, \tag{6.2.1}$$

where $g(x)$ denotes the approximate polynomial of sigmoid function chosen in the previous subsection. We set the initial $\mathbf{w}$ as the zero vector for simplicity.

For a fixed integer p, all the elements are scaled by p and then converted into the nearest integers for quantization. The client first receives the ci-

Algorithm 6 Procedure of secure logistic regression algorithm

Input: Ciphertexts $\{\mathsf{ct.z}_j\}_{0 \leqslant j \leqslant d}$, a polynomial $g(x)$, and number of iterations $\mathsf{IterNum}$

for $j = 0, 1, \ldots, d$ **do**
  $\mathsf{ct.beta}_j \leftarrow \mathbf{0}$
end for
for $i = 1, 2, \ldots, \mathsf{IterNum}$ **do**
  $\mathsf{ct.ip} \leftarrow \mathsf{RS}(\sum_{j=0}^{d} \mathsf{Mult}(\mathsf{ct.beta}_j, \mathsf{ct.z}_j), p)$
  $\mathsf{ct.g} \leftarrow \mathsf{PolyEval}(-\mathsf{ct.ip}, \lfloor p \cdot g(x) \rceil)$
  for $j = 0, 1, \ldots, d$ **do**
    $\mathsf{ct.grad}_j \leftarrow \mathsf{RS}(\mathsf{Mult}(\mathsf{ct.g}, \mathsf{ct.z}_j), p)$
    $\mathsf{ct.grad}_j \leftarrow \mathsf{RS}(\mathsf{AllSum}(\mathsf{ct.grad}_j), \frac{n}{\alpha})$
    $\mathsf{ct.beta}_j \leftarrow \mathsf{ct.beta}_j + \mathsf{ct.grad}_j$
  end for
end for
return $(\mathsf{ct.beta}_0, \ldots, \mathsf{ct.beta}_d)$.

phertexts encrypting $p \cdot \mathbf{z}_i$'s from n users, and then compromises them to obtain $(d+1)$ ciphertexts $\mathsf{ct.z}_j$ for $0 \leqslant j \leqslant d$ which encrypts the vector $p \cdot (z_{1j}, \cdots, z_{nj})$ of j-th attributes using batching technique. If n is not a power of two, the plaintext slots are zero-padded so that the number of slots divides $N/2$. Finally, these resulting ciphertexts $\mathsf{ct.z}_0, \cdots, \mathsf{ct.z}_d$ are sent to the the server for the evaluation of gradient descent.

The public server generates the trivial ciphertexts $\mathsf{ct.beta}_0, \cdots, \mathsf{ct.beta}_d$ as zero polynomials in $\mathcal{R}_q$. At each iteration, it performs a homomorphic multiplication of ciphertexts $\mathsf{ct.beta}_j$ and $\mathsf{ct.z}_j$, and outputs a ciphertext encrypting the plaintext vector $p^2(z_{1j}w_j, \ldots, z_{nj}w_j)$ for $0 \leqslant j \leqslant d$. Then it aggregates the ciphertexts and performs the rescaling operation with p to manipulate the size of plaintext, returning a ciphertext $\mathsf{ct.ip}$ which represents a plaintext vector approximate to $p(\sum_{j=0}^{d} z_{1j}w_j, \ldots, \sum_{j=0}^{d} z_{nj}w_j) = p(\mathbf{z}_1^T\mathbf{w}, \ldots, \mathbf{z}_n^T\mathbf{w})$.

For the evaluation of the least squares polynomial $g(x)$ at $(-\mathbf{z}_i^T\mathbf{w})$, we adapt the polynomial evaluation algorithm, denoted by $\mathsf{PolyEval}(\cdot)$, suggested in [CKKS17] (see Theorem 1 for more detail). Each of coefficients should be scaled by p to transform into an integral polynomial. The output ciphertext $\mathsf{ct.g}$ contains $p \cdot g(-\mathbf{z}_i^T \mathbf{w})$ in the i-th slot. Finally the server performs the multiplication with $\mathsf{ct.g}$ and $\mathsf{ct.z}_j$, the AllSum procedure, and rescaling by $\frac{n}{\alpha}$ to compute ciphertexts $\mathsf{ct.grad}_0, \ldots, \mathsf{ct.grad}_d$ corresponding to entries of gradient vector multiplied by the learning rate. Then it only needs to perform the addition with the gradient vector over encryption, which yields a ciphertext $\mathsf{ct.beta}_j$ encrypting the approximation value to (6.2.1) with scaled by p in each plaintext slot. Our secure logistic regression algorithm is described in Algorithm 6.

Algorithm 6 reduces the ciphertext modulus by $(\lceil \log \deg g \rceil + 3) \log p + \log \frac{n}{\alpha}$ bits to update the encryption of $\mathbf{w}$ during each iteration. For more efficient implementation, we add some techniques to reduce the number of bits consumed by evaluation procedure. We express the evaluation circuit as follows:

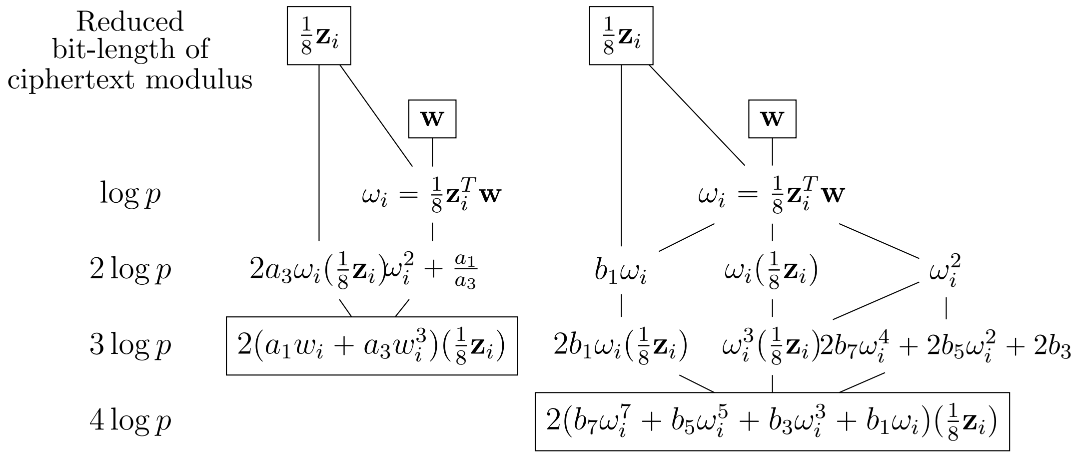
$$\mathbf{w} \leftarrow \mathbf{w} + \frac{4\alpha}{n} \sum_{i=1}^{n} \frac{\mathbf{z}_i}{8} - \frac{4\alpha}{n} \sum_{i=1}^{n} \left(2g(\mathbf{z}_i^T \mathbf{w}) - 1\right) \cdot \frac{\mathbf{z}_i}{8}. \qquad (6.2.2)$$

Note that the polynomial $2g(x)-1$ can be understood as a polynomial of $(x/8)$ with odd degree terms and similar size of coefficients: $2g_3(x)-1 = 2a_1(x/8)+2a_3(x/8)^3$ and $2g_7(x) - 1 = 2b_1(x/8) + 2b_3(x/8)^3 + 2b_5(x/8)^5 + 2b_7(x/8)^7$ for $(a_1, a_3) \approx (1.20096, -0.81562)$ and $(b_1, b_3, b_5, b_7) \approx (1.73496, -4.19407, 5.43402, -2.50739)$.

If the client generates encryptions of $p(\frac{1}{8}\mathbf{z}_i)$ instead of $p\mathbf{z}_i$, the required bit length of ciphertext modulus per iteration is decreased. On the other hand, the server uses a pre-computation step to reduce the complexity of update equation: it performs the AllSum procedure and applies the rescaling operation with the scale factor of $\frac{n}{4\alpha}$ on $\mathsf{ct.z}_j$ for $0 \leqslant j \leqslant d$. As a result, we obtain a ciphertext $\mathsf{ct.sum}_j$ which encrypts an approximate value of $\frac{4\alpha p}{n} \sum_{i=1}^{n} \frac{z_{ij}}{8}$ in each plaintext slot. These ciphertexts will be stored during evaluation and used for update of the j-th component of weight vector $\mathbf{w}$. In particular, the ciphertexts $\mathsf{ct.beta}_0, \cdots, \mathsf{ct.beta}_d$ corresponding to the entries $\mathbf{w}$ becomes $\mathsf{ct.sum}_0, \cdots, \mathsf{ct.sum}_d$ at the first iteration.

Figure 6.2 shows how to evaluate the arithmetic circuit $(2g(\mathbf{z}_i^T\mathbf{w}) - 1) \cdot (\frac{1}{8}\mathbf{z}_i)$ when $g(x) = g_3(x)$ or $g(x) = g_7(x)$. We take encryptions of $p\mathbf{w}$ and $\frac{p}{8}\mathbf{z}_i$ as input of algorithm to minimize the number of required multiplication and depth. Consequently, the proposed method reduces the ciphertext modulus by $3\log p + \log(\frac{n}{4\alpha})$ bits or $4\log p + \log(\frac{n}{4\alpha})$ bits when $g(x) = g_3(x)$ or $g(x) = g_7(x)$, respectively.

Figure 6.2: Evaluation procedure of least squares approximations $(2g(\mathbf{z}_i^T\mathbf{w}) - 1) \cdot (\frac{1}{8}\mathbf{z}_i)$ when $g(x) = g_3(x)$ (left) or $g(x) = g_7(x)$ (right)



6.3 Implementation

In this section, we explain how to set the parameters and present our implementation results using the proposed techniques.

6.3.1 Parameter Setting

It follows from Section 6.2.2 that a lower-bound on the bit length of fresh ciphertext modulus ($\log q$) is as follows:

$$\begin{cases} \log \frac{n}{4\alpha} + (\mathsf{IterNum} - 1)(\log \frac{n}{4\alpha} + 3\log p) + \log q_0 & \text{when } g = g_3, \\ \log \frac{n}{4\alpha} + (\mathsf{IterNum} - 1)(\log \frac{n}{4\alpha} + 4\log p) + \log q_0 & \text{when } g = g_7, \end{cases}$$

where $\mathsf{IterNum}$ is the number of iterations of gradient descent algorithm and q_0 is the output ciphertext modulus. The output ciphertext represents the desired vector $\mathbf{w}$ scaled by p, which means that $\log q_0$ should be larger than $\log p$.

The security of the underlying homomorphic encryption scheme relies on the hardness of the RLWE assumption. We derive a lower-bound on the ring dimension as

$$N \geqslant \frac{\lambda + 110}{7.2} \cdot \log Q \tag{6.3.3}$$

to get λ-bit security level from the security analysis of [GHS12b] where Q denotes the largest modulus of given RLWE samples. In other words, we will take the smallest power of two integer N satisfying the inequality (6.3.3).

6.3.2 Technical Details

Experimentation environment. All the experiments were performed on an Intel Xeon running at 2.3 GHz processor with 16-cores, which is a standard AWS EC2 instance. In our implementation, we used a variant of our library based on NTL library [S$^+$01]. Our implementation is publicly available on github [HEL17].

Datasets. We develop our approximation algorithm using the Myocardial Infarction dataset from Edinburgh [KFMH96]. The others were obtained from Low Birth Weight Study, Nhanes III, Prostate Cancer Study, and Umaru Impact Study datasets [lbw17, nha17, pcs17, uis17]. All these datasets have a single binary outcome variable, which can be readily used to train binary classifiers like logistic regression. Table 1 illustrates the datasets with the number of observations (rows) and the number of features (columns). We split the original datasets into training and testing sets; 90% of the data were chosen randomly for learning (with the learning rate $\alpha \approx 1$) and 10% were used to test the trained models.

Table 6.1: Description of datasets in our experiment

data	# of observations	# of features
Edinburgh	1253	10
lbw	189	10
nhanes3	15649	16
pcs	379	10
uis	575	9

Parameters and Timings for the HE Scheme. Each coefficient of the secret key is chosen at random from $\{0, \pm 1\}$ and we set the number of nonzero coefficients in the key at $h = 64$. We use the standard deviation $\sigma = 3.2$ for discrete Gaussian distribution to sample random error polynomials. We assume that all the inputs have $\log p = 28$ bits of precision and set the bit length of the output ciphertext modulus as $\log q_0 = \log p + 10$. As discussed before, when evaluating the gradient descent algorithm with $g(x) = g_7(x)$, a ciphertext modulus is reduced more than $g(x) = g_3(x)$ at each iteration. So we set the number of iterations as $\mathsf{IterNum} = 25$ (resp. $\mathsf{IterNum} = 20$) when $g(x) = g_3(x)$ (resp. $g(x) = g_7(x)$) to take the ciphertext modulus of similar size. We could actually obtain the approximate bit length of fresh ciphertexts modulus ($\log q$) around 2179 to 2386. We took the ring dimension $N = 2^{17}$ to ensure 80-bit security. For this setting, the public key and a freshly encrypted ciphertext have two ring elements in $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$ so the size is $2N \log q \approx 75$ MB. The key generation takes about 56$\sim$58 seconds and the encryption takes about 1.1$\sim$1.3 seconds. We summarized the parameter setting in Table 6.2.

Table 6.2: Parameter setting in the implementation

λ	N	$\log p$	$\log q_0$	$\log q$
80	2^{17}	28	38	2179$\sim$2386

We can converge to the optimum within a small number of iterations (20$\sim$25), which makes it very promising to train a homomorphic encrypted

Table 6.3: Experimental results of our secure logistic regression algorithm

Dataset	HE-based LR								Unencrypted LR		MSE	NMSE
	deg g	log q	Enc	Eval	Dec	Storage	Accuracy	AUC	Accuracy	AUC		
Edinburgh	3	2254	12s	114min	6.3s	0.68GB	88.65%	0.967	90.83%	0.972	0.0261	0.0352
	7	2326	12s	114min	6.0s	0.71GB	89.96%	0.968	90.39%	0.968	0.0007	0.0013
lbw	3	2179	10s	99min	4.9s	0.66GB	75.41%	0.764	75.41%	0.623	0.0097	0.0697
	7	2266	11s	86min	4.5s	0.69GB	78.69%	0.768	77.05%	0.763	0.0005	0.0046
nhanes3	3	2329	21s	235min	12s	1.1GB	78.95%	0.779	79.02%	0.793	0.0038	0.0285
	7	2386	21s	208min	13s	1.2GB	79.29%	0.780	79.25%	0.779	0.0002	0.0018
pcs	3	2204	11s	103min	4.4s	0.67GB	72.36%	0.834	73.17%	0.842	0.0116	0.0821
	7	2286	11s	97min	4.5s	0.69GB	69.92%	0.837	69.92%	0.840	0.0004	0.0031
uis	3	2229	10s	104min	5.1s	0.61GB	77.78%	0.765	77.78%	0.764	0.0073	0.1534
	7	2306	11s	96min	4.3s	0.63GB	77.78%	0.768	77.78%	0.768	0.0003	0.0078

logistic regression model and mitigate the privacy concerns. We evaluated our models performance based on running time (encryption, evaluation, decryption), storage (encrypted dataset size), and discrimination in Table 6.3. For discrimination, we used the decrypted model parameter $\mathbf{w}$ and calculated the accuracy (%) which is the percentage of the correct predictions on the testing dataset. In addition, we used a popular metric *Area Under the ROC Curve* (AUC) to measure the model's classification performance.

Our implementation shows that the evaluation of gradient descent algorithm with the degree 7 least squares polynomial yields better accuracy and AUC than degree 3. It is quite close to the unencrypted result of logistic regression using the sigmoid function with the same number of iterations; for example, on the training model of Edinburgh dataset, we could obtain the model parameter $\mathbf{w} = (-1.74928, 0.0988924, 0.203933, 0.333984, 1.32132, 0.385157, 0.913334, 0.235844, 0.258459, -0.118332)$. As shown in Table 6.3, it can reach 89.96% accuracy and 0.968 AUC on the testing dataset. When using the sigmoid function on the same training dataset, the model parameter $\mathbf{w}$ is $(-1.6722, 0.0993009, 0.199344, 0.326173, 1.29257, 0.380557, 0.897107, 0.24128, 0.258232, -0.104058)$ which gives 90.39% accuracy and 0.968 AUC. For a more accurate comparison of the model parameters between our encrypted approach and unencrypted logistic regression, we used the mean squared error (MSE) which measures the average of the squares of the errors. We could also normalize it by the the average of the squares of the model parameter, called a *normalized mean squared error* (NMSE). As shown in Table 3, these values of degree 7 are closer to zero which inspires us that the polynomial approximation of that degree is pretty accurate for logistic regression.

Chapter 7

Conclusions

In this paper we constructed a new homomorphic encryption scheme for approximate arithmetic. We opened a new paradigm in this area by considering an error of RLWE problem as a part of computational error. The main advantage of our scheme comes from the use of rescaling procedure and a new packing method of multiple complex numbers. We also gave a method to make our scheme bootstrappable and refresh a ciphertext in a low level. We proved the effiency of our scheme by implementing a library and evaluating some typical circuits, and applying it to the privacy-perserving logistic regression.

Improving the evaluation of some circuits and applying our scheme to other applications would be a next goal. For example, our team submitted an improved solution for logistic regression and got the best award in iDASH 2017 Security & Privacy workshop.* Our team has a plan to make a better and more general control method for encrypted cyber-physical system. We also aim to optimize our homomorphic encryption library `HEAAN`. It would be great if we could use CRT+NTT polynomial representation method in `HEAAN` to reduce the complexity of homomorphic operations.

*http://www.humangenomeprivacy.org/2017/

Bibliography

[AN16] Seiko Arita and Shota Nakasato. Fully homomorphic encryption for point numbers. Cryptology ePrint Archive, Report 2016/402, 2016. http://eprint.iacr.org/.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *Proc. of ITCS*, pages 309–325. ACM, 2012.

[BLLN13] Joppe W Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In *Cryptography and Coding*, pages 45–64. Springer, 2013.

[BLN14] Joppe W Bos, Kristin Lauter, and Michael Naehrig. Private predictive analysis on encrypted medical data. *Journal of biomedical informatics*, 50:234–243, 2014.

[Bot10] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Advances in Cryptology–CRYPTO 2012*, pages 868–886. Springer, 2012.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *Proceedings* *of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, FOCS'11, pages 97–106. IEEE Computer Society, 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from Ring-LWE and security for key dependent messages. In *Advances in Cryptology–CRYPTO 2011*, pages 505–524. Springer, 2011.

[cDSM15] Gizem S. Çetin, Yarkın Doröz, Berk Sunar, and William J. Martin. An investigation of complex operations with word-size homomorphic encryption. Cryptology ePrint Archive, Report 2015/1195, 2015. http://eprint.iacr.org/2015/1195.

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *Advances in Cryptology–ASIACRYPT 2016*, pages 3–33. Springer, 2016.

[CHK+17] Jung Hee Cheon, Koohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate number homomorphic encryption. preprint, 2017.

[CJLL17] Jung Hee Cheon, Jinhyuck Jung, Joohee Lee, and Keewoo Lee. Privacy-preserving computations of predictive medical models with minimax approximation and non-adjacent form. Proceedings of WAHC 2017, 2017.

[CKK15] Jung Hee Cheon, Miran Kim, and Myungsun Kim. Search-and-compute on encrypted data. In *International Conference on Financial Cryptography and Data Security*, pages 142–159. Springer, 2015.

[CKK16] Jung Hee Cheon, Miran Kim, and Myungsun Kim. Optimized search-and-compute circuits and their application to query evaluation on encrypted data. *IEEE Transactions on Information Forensics and Security*, 11(1):188–199, 2016.

[CKKS16] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Implementation of HEA-AN, 2016. `https://github.com/kimandrik/HEAAN`.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 409–437. Springer, 2017.

[CKL15] Jung Hee Cheon, Miran Kim, and Kristin Lauter. Homomorphic computation of edit distance. In *International Conference on Financial Cryptography and Data Security*, pages 194–212. Springer, 2015.

[CLT14] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In *Public-Key Cryptography–PKC 2014*, pages 311–328. Springer, 2014.

[CS15] Jung Hee Cheon and Damien Stehlé. Fully homomophic encryption over the integers revisited. In *Advances in Cryptology–EUROCRYPT 2015*, pages 513–536. Springer, 2015.

[CS16] Ana Costache and Nigel P Smart. Which ring based somewhat homomorphic encryption scheme is best? In *Cryptographers' Track at the RSA Conference*, pages 325–340. Springer, 2016.

[CSV16] Anamaria Costache, Nigel P. Smart, and Srinivas Vivek. Faster homomorphic evaluation of discrete fourier transforms. Cryptology ePrint Archive, Report 2016/1019, 2016. `http://eprint.iacr.org/2016/1019`.

[CSVW16] Anamaria Costache, Nigel P. Smart, Srinivas Vivek, and Adrian Waller. Fixed point arithmetic in SHE schemes. Cryptology ePrint Archive, Report 2016/250, 2016. http://eprint.iacr.org/2016/250.

[DGBL+17] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Manual for using homomorphic encryption for bioinformatics. *Proceedings of the IEEE*, 105(3):552–567, 2017.

[DGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology–EUROCRYPT 2010*, pages 24–43. Springer, 2010.

[DHS16] Yarkın Doröz, Yin Hu, and Berk Sunar. Homomorphic AES evaluation using the modified LTV scheme. *Designs, Codes and Cryptography*, 80(2):333–358, 2016.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology–EUROCRYPT 2015*, pages 617–640. Springer, 2015.

[DOM02] Stephan Dreiseitl and Lucila Ohno-Machado. Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics*, 35(5):352–359, 2002.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology–CRYPTO 2012*, pages 643–662. Springer, 2012.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.

[GBDL+16] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pages 201–210, 2016.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `http://crypto.stanford.edu/craig`.

[GHS12a] Craig Gentry, Shai Halevi, and Nigel P. Smart. Better bootstrapping in fully homomorphic encryption. In *Public Key Cryptography–PKC 2012*, pages 1–16. Springer, 2012.

[GHS12b] Craig Gentry, Shai Halevi, and Nigel P Smart. Homomorphic evaluation of the AES circuit. In *Advances in Cryptology–CRYPTO 2012*, pages 850–867. Springer, 2012.

[GLN12] Thore Graepel, Kristin Lauter, and Michael Naehrig. ML confidential: Machine learning on encrypted data. In *International Conference on Information Security and Cryptology*, pages 1–21. Springer, 2012.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013*, pages 75–92. Springer, 2013.

[HEL17] HELR. Privacy-preserving logistic regression based on homomorphic encryption. `https://github.com/K-miran/HELR`, 2017.

[HS13] Shai Halevi and Victor Shoup. Design and implementation of a homomorphic-encryption library. *IBM Research (Manuscript)*, 2013.

[HS15] Shai Halevi and Victor Shoup. Bootstrapping for HElib. In *Advances in Cryptology–EUROCRYPT 2015*, pages 641–670. Springer, 2015.

[JA16] Angela Jäschke and Frederik Armknecht. Accelerating homomorphic computations on rational numbers. In *International Conference on Applied Cryptography and Network Security*, pages 405–423. Springer, 2016.

[JLW+13] Wenchao Jiang, Pinghao Li, Shuang Wang, Yuan Wu, Meng Xue, Lucila Ohno-Machado, and Xiaoqian Jiang. WebGLORE: a web service for Grid LOgistic regression. *Bioinformatics*, 29(24):3238–3240, 2013.

[KFMH96] RL Kennedy, HS Fraser, LN McStay, and RF Harrison. Early diagnosis of acute myocardial infarction using clinical and electrocardiographic data at presentation: derivation and evaluation of logistic regression models. *European heart journal*, 17(8):1181–1191, 1996.

[KLS+16] Junsoo Kim, Chanhwa Lee, Hyungbo Shim, Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Encrypting controller using fully homomorphic encryption for security of cyber-physical systems. *IFAC-PapersOnLine*, 49(22):175–180, 2016.

[KSC17] Miran Kim, Yongsoo Song, and Jung Hee Cheon. Secure searching of biomarkers through hybrid homomorphic encryption scheme. *BMC medical genomics*, 10(2):42, 2017.

[KSW+17] Miran Kim, Yongsoo Song, Shuang Wang, Yuhou Xia, and Xiaoqian Jiang. Privacy-preserving logistic regression based on homomorphic encryption. preprint, 2017.

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012*, pages 1219–1234. ACM, 2012.

[lbw17] lbw. Low birth weight study data. `https://rdrr.io/rforge/LogisticDx/man/lbw.html`, 2017.

[LLAN14] Kristin Lauter, Adriana López-Alt, and Michael Naehrig. Private computation on encrypted genomic data. In *International Conference on Cryptology and Information Security in Latin America*, pages 3–27. Springer, 2014.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In *Topics in Cryptology–CT-RSA 2011*, pages 319–339. Springer, 2011.

[LP16] Kim Laine and Rachel Player. Simple encrypted arithmetic library-seal (v2. 0). Technical report, 2016.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Advances in Cryptology–EUROCRYPT 2010*, pages 1–23, 2010.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-lwe cryptography. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 35–54. Springer, 2013.

[Mic16] Microsoft. Hosting and cloud study 2016. `http://download.microsoft.com/download/9/4/1/`

941D4C3D-5093-4468-BE11-FB18E4FBD199/Cloud%20and%20Hosting%20Trends%20-The%20Digital%20Revolution,%20Powered%20by%20Cloud.pdf, 2016.

[MWW+17] Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in Bioinformatics*, page bbx044, 2017.

[MZ17] Payman Mohassel and Yupeng Zhang. SecureML: A system for scalable privacy-preserving machine learning. Cryptology ePrint Archive, Report 2017/396, 2017. http://eprint.iacr.org/2017/396.

[nha17] nhanes3. Nhanes iii data. https://rdrr.io/rforge/LogisticDx/man/nhanes3.html, 2017.

[NLV11] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 113–124. ACM, 2011.

[pcs17] pcs. Prostate cancer study data. https://rdrr.io/rforge/LogisticDx/man/pcs.html, 2017.

[PS73] Michael S Paterson and Larry J Stockmeyer. On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM Journal on Computing*, 2(1):60–66, 1973.

[Rou17] David Rousseau. Biomedical research: Changing the common rule by david rousseau — ammon & rousseau translations. https://www.ammon-rousseau.com/changing-the-rules-by-david-rousseau/, 2017.

[S+01] Victor Shoup et al. Ntl: A library for doing number theory, 2001.

[SV10] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public-Key Cryptography–PKC 2010*, pages 420–443. Springer, 2010.

[SV14] Nigel P Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *Designs, codes and cryptography*, 71(1):57–81, 2014.

[TH02] Bahman P Tabaei and William H Herman. A multivariate logistic regression equation to screen for diabetes development and validation. *Diabetes Care*, 25(11):1999–2003, 2002.

[TJ12] John J Trinckes and Jr. *The Definitive Guide to Complying with the HIPAA/HITECH Privacy and Security Rules*. CRC Press, 3 December 2012.

[uis17] uis. Umaru impact study data. `https://rdrr.io/rforge/LogisticDx/man/uis.html`, 2017.

[WJKOM12] Yuan Wu, Xiaoqian Jiang, Jihoon Kim, and Lucila Ohno-Machado. Grid Binary LOgistic REgression (GLORE): building shared models without sharing data. *Journal of the American Medical Informatics Association*, 19(5):758–764, 2012.

[WJW+13] Shuang Wang, Xiaoqian Jiang, Yuan Wu, Lijuan Cui, Samuel Cheng, and Lucila Ohno-Machado. Expectation Propagation LOgistic REgression (EXPLORER): distributed privacy-preserving online model learning. *Journal of biomedical informatics*, 46(3):480–496, 2013.

[WZD+16] Shuang Wang, Yuchen Zhang, Wenrui Dai, Kristin Lauter, Miran Kim, Yuzhe Tang, Hongkai Xiong, and Xiaoqian Jiang. Healer: Homomorphic computation of exact logistic regression for secure rare disease variants analysis in gwas. *Bioinformatics*, 32(2):211–218, 2016.

[Ypm95] Tjalling J Ypma. Historical development of the newton–raphson method. *SIAM review*, 37(4):531–551, 1995.

국문초록

동형 암호는 복호화없이 암호화된 데이터의 연산을 가능하게 하는 암호 체계이다. 동형암호 기술은 공용 서버 위에서의 계산량 아웃소싱에 기반한 수많은 응용분야들을 가지고 있다. 하지만, 기존의 동형암호 스킴들은 공통적으로 실수연산 등 근사계산에 부적합하다는 한계점이 존재했다.

본 논문에서는 근사계산을 위한 새로운 동형암호 설계 방법을 제시한다. 이 동형암호는 암호화된 메시지간의 덧셈, 곱셈 뿐만 아니라 메시지 크기의 조절을 위한 반올림 연산을 함께 지원한다. 주요 아이디어는 노이즈를 메시지의 유효숫자 뒤에 더하는 것으로, 이 노이즈는 본래 스킴의 안전성을 위해 삽입되지만 근사계산 과정에서 발생하는 에러의 일부로 생각하며 반올림 과정에서 그 크기가 줄어든다. 결과적으로, 본래 지수함수적 크기를 가졌던 기존 방법들과 비교해 암호문 모듈러스의 크기를 회로 깊이에 대해 선형적인 크기를 가지도록 감소시킬 수 있었다. RLWE 문제에 기반하여 스킴을 설계하는 경우 하나의 다항식을 복소수 벡터에 대응시켜 하나의 암호문이 다수의 메시지를 암호화하고 동시에 연산을 진행하는 새로운 배칭 기술을 제안하였다. 또한 고유 라이브러리를 작성하였고 이를 이용하여 위 스킴이 역원, 지수함수, 로지스틱 함수 및 이산 푸리에 변환 등 위 스킴이 동형암호를 이용한 효율적인 근사계산에 적합함을 보였다.

이 동형암호는 leveled 구조를 가지고 있어 제한된 횟수의 연산만을 지원한다는 한계점을 가지고 있었지만 본 학위 논문에서는 낮은 레벨의 암호문의 추가적인 연산을 위한 새로운 재부팅 기법을 소개한다. 재부팅 과정은 복호화 과정을 근사계산 동형암호를 이용하여 수행해야 한다는 점 때문에 모듈러 연산이 어렵다는 점이 주쟁점이다. 하지만 이 함수를 삼각함수로 근사킬 수 있다는 점과 sine 함수를 특유의 성질을 이용하여 효율적으로 계산할 수 있다는 점을 이용하여 재부팅에 소요되는 계산량을 줄일 수 있었다. 또한 다수의 메시지를 동시에 암호화할 수 있도록 RLWE 기반 스킴에서의 재부팅 기법을 제시하고 실제 구현을 통해 그 효율성을 증명하였다.

마지막으로 위 동형암호 스킴을 실제 요구되는 응용분야에 적용함으로서 그 효용성을 입증하려 하였다. 특히 근사동형암호 라이브러리를 이용하여 생체 데이터로부터 로지스틱 회귀모델을 계산하는 과정을 동형암호화 된 상태에서 수행하였다.

주요어휘: 동형암호, 근사계산, 재부팅, 로지스틱 회귀
학번: 2012-23025