Ph.D. DISSERTATION

Cryptographic Primitives for Privacy-Preserving Machine Learning: Approximate Homomorphic Encryption and Code-Based Cryptography

정보 보호 기계 학습의 암호학 기반 기술: 근사 동형 암호와 부호 기반 암호

BY

LEE YONGWOO

Feburary 2021

DEPARTMENT OF ELECTRICAL AND
COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Ph.D. DISSERTATION

Cryptographic Primitives for Privacy-Preserving Machine Learning: Approximate Homomorphic Encryption and Code-Based Cryptography

정보 보호 기계 학습의 암호학 기반 기술: 근사 동형 암호와 부호 기반 암호

BY

LEE YONGWOO

Feburary 2021

DEPARTMENT OF ELECTRICAL AND
COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Cryptographic Primitives for Privacy-Preserving Machine Learning: Approximate Homomorphic Encryption and Code-Based Cryptography

정보 보호 기계 학습의 암호학 기반 기술: 근사 동형 암호와 부호 기반 암호

지도교수 노 종 선

이 논문을 공학박사 학위논문으로 제출함

2021년 2월

서울대학교 대학원

전기 · 정보 공학부

이 용 우

이용우의 공학박사 학위 논문을 인준함

2021년 2월

위 원 장:	이 정 우 [illegible]
부위원장:	노 종 선 [illegible]
위 원:	최 완 [illegible]
위 원:	김 상 훈 [illegible]
위 원:	김 영 식 [illegible]

Abstract

In this dissertation, three main contributions are given as; i) a protocol of privacy-preserving machine learning using network resources, ii) the development of approximate homomorphic encryption that achieves less error and high-precision bootstrapping algorithm without compromising performance and security, iii) the cryptanalysis and the modification of code-based cryptosystems: cryptanalysis on IKKR cryptosystem and modification of the pqsigRM, a digital signature scheme proposed to the post-quantum cryptography (PQC) standardization of National Institute of Standards and Technology (NIST).

The recent development of machine learning, cloud computing, and blockchain raises a new privacy problem; how can one outsource computation on confidential data? Moreover, as research on quantum computers shows success, the need for PQC is also emerging. Multi-party computation (MPC) is the cryptographic protocol that makes computation on data without revealing it. Since MPC is designed based on homomorphic encryption (HE) and PQC, research on designing efficient and safe HE and PQC is actively being conducted.

First, I propose a protocol for privacy-preserving machine learning (PPML) that replaces bootstrapping of homomorphic encryption with network resources. In general, the HE ciphertext has a limited depth of circuit that can be calculated, called the level of a ciphertext. We call bootstrapping restoring the level of ciphertext that has exhausted its level through a method such as homomorphic decryption. Bootstrapping of homomorphic encryption is, in general, very expensive in time and space. However, when deep computations like deep learning are performed, it is required to do bootstrapping. In this protocol, both the client's message and servers' intermediate values are kept secure, while the client's computation and communication complexity are light.

Second, I propose an improved bootstrapping algorithm for the CKKS scheme and a method to reduce the error by homomorphic operations in the CKKS scheme. The Cheon-Kim-Kim-Song (CKKS) scheme (Asiacrypt '17) is one of the highlighted fully homomorphic encryption (FHE) schemes as it is efficient to deal with encrypted real numbers, which are the usual data type for many applications such as machine learning. However, the precision drop due to the error growth is a drawback of the CKKS scheme for data processing. I propose a method to achieve high-precision approximate FHE using the following two methods. First, I apply the signal-to-noise ratio (SNR) concept and propose methods to maximize SNR by reordering homomorphic operations in the CKKS scheme. For that, the error variance is minimized instead of the upper bound of error when we deal with the encrypted data. Second, from the same perspective of minimizing error variance, I propose a new method to find the approximate polynomials for the CKKS scheme. The approximation method is especially applied to the CKKS scheme's bootstrapping, where we achieve bootstrapping with smaller error variance compared to the prior arts. In addition to the above variance-minimizing method, I cast the problem of finding an approximate polynomial for a modulus reduction into an L2-norm minimization problem. As a result, I find an approximate polynomial for the modulus reduction without using the sine function, which is the upper bound for the polynomial approximation of the modulus reduction. By using the proposed method, the constraint of $q = \mathcal{O}(m^{3/2})$ is relaxed as $\mathcal{O}(m)$, and thus the level loss in bootstrapping can be reduced. The performance improvement by the proposed methods is verified by implementation over HE libraries, that is, HEAAN and SEAL. The implementation shows that by reordering homomorphic operations and using the proposed polynomial approximation, the reliability of the CKKS scheme is improved. Therefore, the quality of services of various applications using the proposed CKKS scheme, such as PPML, can be improved without compromising performance and security.

Finally, I propose an improved code-based signature scheme and cryptanalysis of code-based cryptosystems. A novel code-based signature scheme with small parameters and an attack algorithm on recent code-based cryptosystems are presented in this dissertation. This scheme is based on a modified Reed-Muller (RM) code, which reduces the signing complexity and key size compared with existing code-based signature schemes. The proposed scheme has the advantage of the pqsigRM decoder and uses public codes that are more difficult to distinguish from random codes. I use $(U, U + V)$-codes with the high-dimensional hull to overcome the disadvantages of code-based schemes. The proposed a decoder which efficiently samples from coset elements with small Hamming weight for any given syndrome. The proposed signature scheme resists various known attacks on RM code-based cryptography. For 128 bits of classical security, the signature size is 4096 bits, and the public key size is less than 1 MB. Recently, Ivanov, Kabatiansky, Krouk, and Rumenko (IKKR) proposed three new variants of the McEliece cryptosystem (CBCrypto 2020, affiliated with Eurocrypt 2020). This dissertation shows that one of the IKKR cryptosystems is equal to the McEliece cryptosystem. Furthermore, a polynomial-time attack algorithm for the other two IKKR cryptosystems is proposed. The proposed attack algorithm utilizes the linearity of IKKR cryptosystems. Also, an implementation of the IKKR cryptosystems and the proposed attack is given. The proposed attack algorithm finds the plaintext within 0.2 sec, which is faster than the elapsed time for legitimate decryption.

keywords: Approximate arithmetic, bootstrapping, Cheon-Kim-Kim-Song (CKKS) scheme, code-based cryptography, cryptanalysis, cryptography, data privacy, digital signatures, error correction codes, fully homomorphic encryption (FHE), lattice-based cryptography, McEliece cryptosystem, polynomial approximation, post-quantum cryptography (PQC), privacy-preserving machine learning (PPML), public-key cryptography, Reed-Muller (RM) codes

student number: 2017-34837

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The recent development of machine learning, cloud computing, and blockchain raises a new privacy problem; how can one write a smart contract on a public blockchain or outsource computation such as machine learning for confidential data? The need for cryptographic primitives for such scenarios has been exploded, and there have been extensive studies. Multi-party computation (MPC) is an encryption protocol that allows various users to collaborate to perform computations without revealing confidential data while hostile users exist. Privacy-preserving machine learning (PPML) is the most promising of these and is very useful in many applications, such as healthcare and finance, to perform machine learning algorithms on confidential data. MPC is built upon various cryptographic primitives, such as homomorphic encryption (HE) and public-key cryptography [2].

The cryptographic primitives should resist attacks using quantum computers as well as classical computers. However, polynomial-time quantum algorithms for prime factorization and the discrete logarithm problem have been proposed [3]. Hence, classical public-key algorithms such as RSA and elliptic curve cryptosystems will no longer be used after the advent of quantum computing. It is PQC that the public-key cryptography resists attacks using quantum computers. The lattice-based and code-based cryptography is the most promising candidate of PQC. Hence, the cryptographic primitives in this dissertation are based on hard problems of lattice and coding theory.

In this dissertation, three main contributions are given as; i) a protocol of privacy-preserving machine learning using network resources, ii) the development of approximate homomorphic encryption that achieves less error and high-precision bootstrapping algorithm without compromising performance and security, and iii) the cryptanalysis and the modification of code-based cryptosystems: cryptanalysis on Ivanov-Kabatiansky-Krouk-Rumenko (IKKR) cryptosystem and modification of the pqsigRM. The pqsigRM is a digital signature scheme proposed for the post-quantum cryptography (PQC) standardization of the NIST.

First, I propose an efficient protocol that dismisses the most time-consuming operation of PPML using HE, bootstrapping, by using communication resources. Homomorphic encryption enables the computation of encrypted data, but one of its drawbacks is computational complexity. Usually, the complexity of basic operations of homomorphic encryption is $O(l^2)$, where l is the level of a ciphertext, which is the maximum depth of operation that can be performed using the ciphertext. However, for the accuracy of machine learning, depth is important, which inevitably increases the amount of computation. The bootstrapping initializes the level of the ciphertext; the complexity of the homomorphic operation can be maintained at any depth. However, bootstrapping is an expensive operation. In this dissertation, I propose a method of PPML without bootstrapping using communication resources. In other words, it replaces bootstrapping with a procedure that induces a small amount of communication between participants.

Next, I propose methods of improving the approximate HE using variance-minimizing and convex optimizations. The Cheon-Kim-Kim-Song (CKKS) scheme is one of the highlighted fully homomorphic encryption (FHE) schemes as it is efficient to deal with encrypted complex(real) numbers, which are the usual data type for many applications such as machine learning [4]. In this dissertation, I propose a generally applicable method to achieve high-precision approximate FHE using the

following two techniques. First, I apply the concept of SNR and propose a method of maximizing SNR of encrypted data by reordering homomorphic operations in the CKKS scheme. For that, the variance of error of encrypted data is minimized instead of the upper bound of error when we deal with the ciphertext. Second, from the same perspective of minimizing error variance, I propose a new method of finding the approximate polynomials for the CKKS scheme. The approximation method is especially applied to the bootstrapping of the CKKS scheme, where I achieve a smaller error variance in the bootstrapping compared to the prior arts. The performance improvement of the proposed methods for the CKKS scheme is verified by implementation over HE libraries, HEAAN, and SEAL. The implementation results show that by reordering homomorphic operations and using the proposed polynomial approximation, the message precision of the CKKS scheme is improved. Specifically, the proposed method uses only depth 8, although the bootstrapping error for the CKKS method is less than the error obtained using depth 11 of the previous method. I also suggest a loose lower bound for bootstrapping error in the CKKS scheme and show that the error by the proposed method is only 2.8 bits on average larger than the lower bound. Therefore, the quality of services of various applications using the proposed CKKS scheme, such as privacy-preserving machine learning, can be improved without compromising performance and security.

Finally, I propose an efficient code-based signature scheme and cryptanalysis of code-based cryptosystems. Especially, the pqsigRM, a first-round candidate of PQC standardization by NIST and its modification, are included. By using the proposed modified RM codes and their decoding, one can find a small-Hamming-weight error vector for any given received vector. Hence, it reduces the required iteration in code-based signature schemes, such as the signature scheme proposed by CFS. The proposed signature scheme has a small parameter size. In addition, I propose here that one of the IKKR cryptosystems is equivalent to the McEliece cryptosystem and cryptanalysis for the other two. The implementation results show that the proposed attack

algorithm is efficient so that it performs faster than the legitimate decryption.

1.1 Homomorphic Encryption and Privacy-Preserving Machine Learning

Homomorphic encryption enables outsourcing arbitrary computation over encrypted data, and this privacy-preserving property is attractive for outsourcing of machine learning, called machine learning as a service. After Gentry's blueprint [2], it has been widely studied and several HE schemes have been proposed [4–12]. As HE can handle encrypted data without decryption, it is suitable for data-rich applications that require privacy. Particularly, since Cheon et al. proposed a HE scheme for complex numbers [4], called the CKKS scheme, the utilization of HE in deep learning methods has become easier for privacy-preserving applications [13–18].

Another important observation by Gentry is that encryption contains noise and the noise level grows as operations are performed on the ciphertext. It is necessary to deal with noise to avoid overwhelming the data, and there are two types of HE schemes for this purpose. The first is somewhat homomorphic encryption (SHE), in which the ciphertext size and computation increase at least linearly with the depth of the circuit. SHE is an appropriate choice for low-depth circuits; however, it has a scaling problem. The other method is an FHE. Gentry proposed the bootstrapping technique to refresh the noise, and thus, the parameter size and computation could be fixed regardless of circuit depth. However, in general, the bootstrapping of FHE schemes requires a considerable amount of computation.

The leveled HE (LHE) has a limitation of the depth of the circuit that can be performed, that is, the level of ciphertext. A larger parameter should be used to increase the level of a ciphertext. In order to perform a circuit that is deeper than the level of ciphertext, the bootstrapping should be done. Thus, the deep neural network requires significant computational time due to the bootstrapping or a large parameter in order to avoid bootstrapping. So far, privacy-preserving machine learning has used the limited depth of the neural network.

In this dissertation, a novel method to evaluate deep neural networks over encrypted data without bootstrapping. The bootstrapping is replaced by communication resources; the data of the sender is protected by homomorphic encryption, and the intermediate values are protected by information-theoretic secrecy. Unlike the hybrid methods using MPC and HE, this method optimizes the computation and communication of the client. I provide the success probability of the proposed method with the CKKS scheme as it is not straightforward and show that a protocol with less level loss and less error is achieved with a negligible failure probability. Besides, it is required to use the sparse packing method in the CKKS scheme when bootstrapping is required as the bootstrapping error is proportional to the square of slot size. With the proposed method, the full slots can be utilized without additional error growth, which results in a significant reduction in computation time and communication of both server and client in the amortized manner. Compared to the hybrid methods, the proposed technique enables the simpler structure of the client, which might reduce the size of hardware while securing the structure of the model, which is an asset of the service provider.

1.2 High-Precision CKKS Scheme and Its Bootstrapping

The CKKS scheme is an approximated homomorphic encryption scheme [4] using ring-learning with error (RLWE). The CKKS scheme [4] is one of the highlighted FHE schemes as it is efficient to deal with real (or complex) numbers, which is the usual data type for many applications such as deep learning and regression. When we deal with arbitrary precision real numbers using other FHE schemes such as (Brakerski)-Fan-Vercauteren ((B)FV) [7, 8, 11] and Brakerski-Gentry-Vaikuntanathan (BGV) [9] schemes, the size of ciphertext has an exponential growth rate according to the level, where the level of ciphertext is defined by the maximum depth of operation that can be homomorphically evaluated without bootstrapping. However, the ciphertext size has a polynomial growth rate according to the level in the CKKS scheme.

Bootstrapping for the CKKS scheme was first proposed by Cheon et al. [19]. Subsequently, several studies have been conducted to improve bootstrapping for CKKS schemes [20–22], and they commonly perform modulus reduction homomorphically by approximating it to a scaled sine function. The CKKS scheme is promising and used widely; however, the improvement of bootstrapping is crucial as most machine learning methods require operations of significant depth.

Homomorphic evaluation of the modulus reduction is the key part of the bootstrapping of the CKKS scheme. As only addition and multiplication can be evaluated homomorphically, and modulus reduction cannot be represented by addition and multiplication, a polynomial approximation for modulus reduction is required.

In most bootstrapping methods studied so far, the scaled sine function (or shifted to the cosine function) is deemed to approximate the modulus reduction [19–21]. This is because the sine function is a periodic function that is close to a first-order polynomial near the origin, and polynomial approximations to trigonometric functions have been studied a lot. Thus, a polynomial approximation for the scaled sine function is used to evaluate the modulus reduction homomorphically. In [19], the sine function was approximated by Taylor expansion of an exponential function using $e^{i\theta} = \cos\theta + i\sin\theta$ and the double angle formula $e^{i2\theta} = (e^{i\theta})^2$. The Chebyshev interpolation method improves the polynomial approximation of the sine function [21]. Based on the fact that the size of a message is significantly less than the ciphertext modulus, better nodes for Chebyshev interpolation was selected, and the approximation was refined [20].

1.2.1 Near-Optimal Bootstrapping of the CKKS Scheme Using Least Squares Method

In this dissertation, instead of approximating the sine function, I propose to cast the problem of finding approximate polynomials for a modulus reduction into the L2-norm minimization problem for which an optimal solution can be directly computed. Thus, the fundamental error caused by the use of trigonometric functions can be conquered. An approximation by the minimax polynomial for the modulus reduction is desirable; however, the shape of the modulus reduction function makes it difficult to find the minimax polynomial. Thus, instead, I propose a discretized optimization method that can be solved efficiently with a unique solution. Through the solution of the modified discretized problem, I can reduce the degree of the approximate polynomial for the modulus reduction while achieving a low margin of error. Consequently, operations required for the homomorphic modulus reduction are reduced compared with the best-known method [20], where the double angle formula is excluded.

When conventional methods are used, the sine function dominates the approximation error; in other words, the approximation error cannot be less than the difference between the sine function and modulus reduction. Therefore, the message size is limited to $m < q^{2/3}$, and thus plaintext precision is also limited, where q denotes a value of the ciphertext modulus. However, the proposed method does not use the sine function, and thus I can obtain a precise approximate polynomial or utilize a message that is larger in size. For example, when $m/q < 2^{-10}$, the proposed method finds an approximate polynomial with a maximum error of less than 2^{-40} with only a circuit depth of 7, whereas the best-known modified Chebyshev interpolation method cannot because the error saturates to 2^{-27}. Therefore, the proposed method is essential for applications that require precise calculations. Moreover, accurate approximate polynomials for modulus reductions of larger messages can be found. For example, I achieve 2^{-20} error for $m/q \approx 2^{-6}$ with only a depth of 7, whereas conventional methods cannot be used with the message $m/q \approx 2^{-6}$ because the error saturates to 2^{-15}.

This means that a user can handle a large, accurate number, and the selection of parameters for the CKKS scheme can be expanded using the proposed method. Thus, the proposed method using the L2-norm minimization makes it possible to take a trade-off between the computational complexity (the degree of approximate polynomial) and the approximation error for the CKKS scheme. By using the proposed method, the constraint of $q = \mathcal{O}(m^{3/2})$ is relaxed as $\mathcal{O}(m)$, and thus the level loss in bootstrapping can be reduced.

1.2.2 Variance-Minimizing and Optimal Bootstrapping of the CKKS Scheme

The CKKS scheme provides the trade-off between the efficiency and precision of messages, where messages in the CKKS scheme contain errors, and the errors are accumulated during homomorphic operations. To our best knowledge, research to the date has provided high-probability upper bounds for errors in encrypted data [4, 19, 23]. As the processing of messages in the CKKS scheme proceeds, the upper bound of errors in encrypted data is increased, and thus it becomes a loose and useless bound.

In previous studies on the error of the CKKS scheme, the probabilistic concept has been used to some extent. Error control in the CKKS scheme so far provided the high-probability upper bounds of error [4, 19] or average precision of message [21]. The high-probability upper bounds are derived from the distribution of error, and the average precision of the message is about the average error, which is a probabilistic term. The CKKS scheme is considered as an erroneous channel, and thus methodologies from communication theory can be adopted, which are the power ratio of a signal (message) and errors. The signal power can be controlled by the scaling factor of the message, and I show how to minimize the noise power during approximate homomorphic operations in the CKKS scheme. Since the errors in the CKKS system are additive, the central limit theorem can be used to treat the error as a Gaussian distribution. Therefore, it is better to control the variance of errors rather than the high probability upper bound of errors and keep them as tagged information for the ciphertext.

Since a drawback of the CKKS scheme is that errors are accumulated, many studies have been conducted to reduce errors. Recently, Kim et al. proposed a new method to reduce errors in encrypted data of the CKKS scheme and its residue number system

(RNS) variants using lazy rescaling and different scaling factors at each level [23]. Although the error was reduced in their paper, the high-probability upper bound was still used as a measure of error. Especially, error amplification during the bootstrapping in the CKKS scheme has been studied in a lot of research. After the first bootstrapping method was proposed in [19], the Chebyshev interpolation method has been applied to the homomorphic evaluation of modulus reduction [20], a method for direct approximation was proposed in [24], and the algorithm for finding minimax approximate polynomial and inverse sine method was proposed in [25].

In this dissertation, I propose a method of managing the variance of errors to maximize the signal-to-noise ratio (SNR) of the messages in the CKKS scheme rather than minimizing the high-probability upper bounds. First, to minimize the error variance of the message, a criterion for optimizing the order of homomorphic operations is proposed. In the proposed method, the error variance of the CKKS scheme is treated as a value to be controlled rather than the upper bound of error. This method can improve the stability of various applications that use approximate homomorphic encryption by reordering homomorphic operations, and it can also improve the accuracy of the resultant message. The second contribution is the optimization of the approximate polynomials in terms of the error variance of the message for the CKKS scheme. The method is the first method to find the optimal approximation polynomial that minimizes not only the approximation error but also the error in polynomial basis that is amplified by coefficients. I improve the bootstrapping algorithm of the CKKS scheme using the proposed polynomial approximation method, where bootstrapping is implemented with smaller errors and less depth consumption. It is shown in this dissertation that the proposed method reduces the magnitude of bootstrapping error compared to the previous work [25]. Moreover, the proposed method resolves the problem that the approximate polynomials have large coefficients, which could only be solved by using the double angle formula in the previous work. However, the proposed method makes it possible to use a direct approximation for the modulus reduction. The comparison with the previous methods shows that by using the proposed method, I can improve the message precision after bootstrapping while reducing the level consumption for bootstrapping. Specifically, the proposed method uses only depth 8, although the bootstrapping error for the CKKS method is less than the error obtained using depth 11 of the previous method.

1.3 Efficient Code-Based Signature Scheme and Cryptanalysis of the Ivanov-Kabatiansky-Krouk-Rumenko Cryptosystems

Recently, code-based cryptographic algorithms have been extensively studied in PQC. Code-based cryptography is based on the syndrome decoding problem and its variants. The syndrome decoding problem is to find a vector $\boldsymbol{e}$ satisfying $\mathbf{H}\boldsymbol{e}^T = \boldsymbol{s}^T$ and $\mathrm{wt}(\boldsymbol{e}) \leqslant w$, where $\mathbf{H}$ is a parity check matrix of a random (n, k) code, $\boldsymbol{s}$ is a random syndrome vector, w is a small value, and $\mathrm{wt}(\boldsymbol{e})$ denotes the Hamming weight of a vector $\boldsymbol{e}$. The code-based cryptosystems and signature schemes are based on the hardness of the decoding problem [26].

Berlekamp and McEliece first proved the hardness of the syndrome decoding problem [26] and McEliece proposed a cryptosystem based on Goppa codes [27]. Thus, an adversary has to solve the decoding problem or distinguish a permuted Goppa code, while the legitimate users can still properly decode. Although lots of variants of the McEliece cryptosystem have been proposed using different codes, for some of them, key distinguishing attacks have been discovered in [28–30].

After McEliece first introduced a code-based cryptosystem (called the McEliece cryptosystem) [27], many variants of it have been proposed. There are several code-based public-key encryptions and key-establishment algorithms in the second round of the PQC standardization by NIST [31]. In addition to the code-based public-key encryption schemes, code-based signature schemes are also proposed [32]– [33] and

[34]. I present a novel code-based signature scheme called modified pqsigRM. Also, I provide a polynomial-time attack algorithm for one of the IKKR cryptosystems that is a recently proposed code-based cryptosystem. Moreover, it is shown that the other two IKKR cryptosystems are equivalent to the McEliece cryptosystem, so it is not an improvement of the McEliece cryptosystem.

1.3.1 Modified pqsigRM: An Efficient Code-Based Signature Scheme

The modified pqsigRM is based on a modified Reed–Muller (RM) code [34–36], which reduces the signing complexity and key size compared with existing code-based signature schemes. In fact, it strengthens pqsigRM submitted to NIST for post-quantum cryptography standardization [37]. The proposed scheme has the advantage of the pqsigRM decoder and uses public codes that are more difficult to distinguish from random codes. I use $(U, U+V)$-codes with the high-dimensional hull to overcome the disadvantages of code-based schemes. The proposed decoder samples from coset elements with small Hamming weight for any given syndrome and efficiently finds such an element. With the modified RM code, the proposed signature scheme resists various known attacks on RM-code-based cryptography. For 128 bits of classical security, the signature size is 4096 bits, and the public key size is less than 1 MB.

Courtois, Finiasz, and Sendrier proposed the CFS signature scheme [38], which is a code-based signature scheme using a full-domain hash (FDH) approach. In this scheme, $t!$ hashes and decoding are required on average to sign a message when an (n, k) Goppa code with error correction capability t is used. It is proposed to use high-rate Goppa codes, which have relatively small error correction capability $t = \frac{n-k}{\log n}$, to reduce the signing time. Therefore, it has a large signing complexity and certain drawbacks in terms of parameter scaling. Moreover, it has been shown in [29] that high-rate Goppa codes can be distinguished from random codes. This falsifies the assumption of existential unforgeability under a chosen message attack (EUF-CMA) security proof in [39], which is based on the indistinguishability of Goppa codes. Although Morozov et al. proved the strong EUF-CMA security of the CFS signature scheme without the indistinguishability of Goppa codes [40], the large key size and expensive signing remain as drawbacks.

There are several variants of the CFS signature scheme, such as signature schemes using LDGM codes [41] and block wise-triangular secret key [42]. To find a signature with a small Hamming weight, the scheme in [41] uses a sparse coset element added to a codeword with a small Hamming weight. Even though this is efficient and has a small key size, an attack algorithm was presented in [43]. An attack algorithm for the signature scheme using a blockwise-triangular secret key was also proposed [44].

The Kabatianskii-Krouk-Smeets (KKS) signature scheme [45] and its variants [46, 47] take a different approach than CFS signature scheme. However, owing to the attack proposed in [48], these are considered (at best) to be one-time signature schemes. Moreover, from the attacks in [49], it is known that the parameters in the KKS scheme and its variants should be carefully chosen.

SURF is a variant of CFS signature scheme using $(U, U + V)$-codes [50]. SURF uses $(n, k_U + k_V)$ binary codes defined by $\{(\boldsymbol{u}|\boldsymbol{u} + \boldsymbol{v})|\boldsymbol{u} \in U, \boldsymbol{v} \in V\}$, where U and V are $(n/2, k_U)$ and $(n/2, k_V)$ random binary codes, respectively. A variant of the Prange decoder is applied to SURF to find an error vector with a small Hamming weight. The security of SURF is based on the decoding-one-out-of-many (DOOM) problem, in which a solution for the syndrome decoding problem is sought in the presence of several syndromes. Unfortunately, as it has been demonstrated that the hull of any $(U, U + V)$-code is highly probable to be a two-repetition code when U and V are random binary codes [50], the hull of the public key can be used for key attacks on SURF. In the recently proposed signature scheme, Wave [32], the generalized ternary $(U, U + V)$-codes are used instead of binary codes as they efficiently resist the hull attack in [50]. Moreover, finding errors with large Hamming weight for the given syndrome allows small parameters. A tighter security reduction using rejection sampling and preimage samplable functions [51] was proved in [32].

In this dissertation, a new code-based signature scheme using binary codes with a $(U, U+V)$-code as its subcode is proposed. For two linear codes $\mathcal{C}_1$ and $\mathcal{C}_2$, $\mathcal{C}_2$ is called a subcode of $\mathcal{C}_1$ if all codewords in $\mathcal{C}_2$ are in $\mathcal{C}_1$. The subcode used in the proposed signature scheme is a binary $(U, U+V)$-code, where U and V are obtained by modifying the RM codes. I design V and $U^\perp$ to have a sufficient number of common codewords, where $U^\perp$ denotes the dual code of U. By the relationships between U and V, it is shown that the proposed signature scheme resists the attack for $(U, U+V)$-codes in [50]. Further, an efficient and randomized decoding algorithm is proposed. This algorithm makes it possible to reduce the key size and signature length. As the codes in the proposed signature scheme are a modification of RM codes, the decoding algorithm makes use of the recursive structure. The proposed signature scheme is an improvement of pqsigRM [37] submitted to NIST for PQC standardization, and it resolves the weaknesses of early versions of pqsigRM by modifying the public code. Moreover, I ensure the distinguishability of the public code of the proposed signature scheme.

1.3.2 Ivanov-Kabatiansky-Krouk-Rumenko Cryptosystems and Its Equality

Recently, Ivanov, Kabatiansky, Krouk, and Rumenko proposed new variants of the McEliece cryptosystem at CBCrypto 2020, affiliated with Eurocrypt 2020 [52]. The IKKR cryptosystems use structured error vectors with arbitrary Hamming weight rather than a random error vector with Hamming weight less than or equal to t. The goal of IKKR cryptosystems is to make information set decoding harder by adding an error vector with a larger Hamming weight. There are three algorithms proposed in [52], which are *prototype*, *modified version of prototype*, and *upgraded IKKR cryptosystems*.

In this dissertation, I propose cryptanalysis of the modified version of the prototype and the upgraded IKKR cryptosystems. The linearity of encryption of the IKKR cryptosystems is used for the proposed cryptanalysis. In other words, I construct a system of linear equations to find plaintext with public-key and the corresponding ciphertext. By solving the system of linear equations, a polynomial-time attack algorithm is performed against the IKKR cryptosystems. I also prove that the prototype IKKR cryptosystem is equal to the McEliece cryptosystem. Besides, I present a proof-of-concept implementation of the IKKR cryptosystems and the proposed attack algorithm. It turns out that the proposed attack finds the plaintext corresponding to a given ciphertext within 0.2s in a desktop computer, which is even faster than the proposed legitimate decryption.

1.4 Organization of the Dissertation

The remainder of the dissertation is organized as follows. In Chapter 2, the preliminaries for PPML, HE, and code-based cryptography are given. I propose a protocol for PPML such that the bootstrapping is replaced by a communication network resource in Chapter 3 and thus, an accurate PPML is possible with less error and higher throughput. The method of minimizing error variance in approximate HE is given in Chapter 4. It is also proposed the theoretical bound-achieving bootstrapping algorithm in the same chapter. I show the efficient code-based signature scheme in Chapter5 as well as the cryptanalysis and equivalence of the IKKR cryptosystem. Finally, I conclude in Chapter 6 with remarks.

Chapter 2

Preliminaries

2.1 Basic Notation

Vectors are denoted in boldface such as $\boldsymbol{x}$ and every vector is a column vector. Matrices are denoted by boldfaced capital letters, for example, $\mathbf{A}$. I denote the inner product of two vectors by $\langle \cdot, \cdot \rangle$ or simply $\cdot$. Let $\boldsymbol{u} \times \boldsymbol{v}$ denote the component-wise multiplication of two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$. Matrix multiplication is denoted by $\cdot$ or can be omitted when it is unnecessary. $x \leftarrow \mathcal{D}$ denotes the sampling x according to a distribution $\mathcal{D}$. When a set is used instead of distribution, it means that x is sampled uniformly at random among the set elements. Random variables are denoted by capital letters such as X. $E[X]$ and $Var[X]$ denote the mean and variance of random variable X, respectively. Some capital letters may represent something other than a random variable such as a constant, but this is context-sensitive. Lp-norm of a vector is denoted by $\|\boldsymbol{x}\|_p = (\sum_i \boldsymbol{x}[i]^p)^{-p}$, where $\boldsymbol{x}[i]$ denotes the i-th element of vector $\boldsymbol{x}$.

$(\boldsymbol{x}_0|\boldsymbol{x}_1)$ denotes the concatenation of two vectors $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$. For example, $h(\boldsymbol{m}|r)$ means the hash function h with input $(\boldsymbol{m}|r)$, where $(\boldsymbol{m}|r)$ represents the concatenation of binary representation of vector $\boldsymbol{m}$ and a random value r. $\boldsymbol{x}^\sigma$ denotes that a vector $\boldsymbol{x}$ is permuted by a permutation σ, for example, $\boldsymbol{x}^\sigma = (x_1, x_3, x_2, x_0)$, where $\boldsymbol{x} = (x_0, x_1, x_2, x_3)$ and $\sigma = (1, 3, 2, 0)$.

$\mathbf{A}_{\mathcal{J}}$ denotes the matrix which contains only columns with indices in $\mathcal{J}$ of $\mathbf{A}$, where $\mathcal{J}$ is a set of indices. The notation $[\mathbf{A}|\mathbf{B}]$ refers the augmented matrix given the matrices $\mathbf{A}$ and $\mathbf{B}$.

2.2 Privacy-Preserving Machine Learning and Security Terms

2.2.1 Privacy-Preserving Machine Learning and Security Terms

Privacy-preserving machine learning is the protocol composed of two participants: sender and receiver. The receiver has restricted computation ability, and the sender has powerful computation ability and a trained model. Another name for privacy-preserving machine learning is secure machine learning as a service (MLaaS). Unlike ordinary machine learning as a service, a security issue is considered in the privacy-preserving machine learning, which means that the input of the receiver is kept secure in the procedure.

Privacy-preserving is a sort of secure two-party computation (2PC). 2PC is firstly proposed by Yao [53], Goldreich, Micali, and Wigderson [54]. Protocols for 2PC allow two parties to compute any function f of their private inputs x and θ without revealing anything more than the output $f(x, \theta)$ of the function. When there are two or more than two parties, it is called secure MPC. In this section, I introduce some requirements that correspond to 2PC.

Non-Interactivity

Each player sends just a single message in 2PC in a setting with a non-interactivity requirement. The first player, the receiver, computes some message m_1 based on his input x and sends m_1 to the second player, the sender. The sender then computes a response m_2 based on its input θ and the message m_1, and then sends it back to the receiver. After receiving m_2, the receiver can finally compute and output $f(x, \theta)$. In a setting with a non-interactivity requirement, it is required that only the receiver obtains the output $f(x, \theta)$; otherwise, the sender can choose arbitrary θ of his choice, and then obtain $f(x, \theta)$ as many as possible. This leaks information of x.

Succinctness

Protocols for a succinct non-interactive secure 2PC allows both the communication complexity and receiver running time of an honest receiver is essentially independent of the running time of f. Gentry's breakthrough result on FHE yields a succinct non-interactive secure 2PC [2]. Assuming an FHE scenario, the receiver encrypts x by FHE and his own key, then the ciphertext $\mathsf{Enc}(x)$ and the public key pk are sent to the sender. The sender evaluates f homomorphically using his own input θ. During the homomorphic encryption, the sender cannot obtain information of x. After the homomorphic evaluation, $\mathsf{Enc}(f(x, y))$ is given to the sender; finally, the sender sends it to the receiver. As only the receiver can decrypt, the receiver can figure out $f(x, y)$. It is noted here that the communication complexity is two ciphertexts, and all the computation required to the receiver is one encryption and one decryption; therefore, it is succinct.

2.2.2 Privacy-Preserving Machine Learning

The capability of HE performing multiplication and addition enables secure evaluation of machine learning whose core operations are generalized matrix multiplications, such as convolution. It is preferable to use a pre-trained network without modification as access to training data is not always guaranteed, and it is advantageous to utilize many of the fruitful results of machine learning in plaintext. However, the non-polynomial operations, such as activation and pooling, are not supported by HE schemes. Thus, it is required to re-design a HE-friendly network or perform non-polynomial operations with approximation polynomials.

HE-Friendly Neural Networks

In order to use HE efficiently, the network is modified so that it fits HE algorithms. For example, the comparison operation is quite tricky in HE, and thus max pooling is replaced by mean pooling [15]. The ReLU function is replaced by several polynomial activation functions [16]. The depth of the neural network can also be adjusted so that it does not require a large ciphertext [55].

Pre-Trained Neural Networks

When we use the HE-friendly neural network, the network should be re-trained. Hence, it is required to full access to the training data, which is quite expensive. Moreover, there is some restriction on activation, pooling, and depth, the performance of the algorithm drops.

In contrast, the use of pre-trained networks does not require full access to the training data nor accuracy drop. HE schemes do not support modern activation functions and pooling in general, and thus there exist two approaches: approximation with i) FHE method and ii) the hybrid method with MPC. In the FHE approach, the non-polynomial layers are replaced by approximate polynomials, and in the hybrid approach, the non-polynomial layers are performed by MPC protocols.

2.3 The CKKS Scheme and Its Bootstrapping

2.3.1 The CKKS Scheme

This section briefly introduces the CKKS scheme [4] and its RNS variant, the RNS-CKKS scheme [20, 56]. For a positive integer M, let $\Phi_M(X)$ be the M-th cyclotomic polynomial of degree N, where M is a power of two, $M = 2N$, and $\Phi_M(X) = X^N + 1$. Let $\mathcal{R} = \mathbb{Z}/\left\langle \Phi_M(X) \right\rangle$ be the ring of integers of a number field $\mathcal{S} = \mathbb{Q}/\left\langle \Phi_M(X) \right\rangle$, where $\mathbb{Q}$ is the set of rational numbers and I write $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$.

The CKKS scheme [4] and its RNS variants [20,23,56] provide homomorphic operations on encrypted real number data with errors. This is done by canonical embedding and its inverse. Recall that canonical embedding Emb of $a(X) \in \mathbb{Q}/\langle\Phi_M(X)\rangle$ into $\mathbb{C}^N$ is the vector of the evaluation values a at the roots of $\Phi_M(X)$ and Emb^{-1} is its inverse. Let π denote a natural projection from $\mathbb{H} = \{(z_j)_{j\in\mathbb{Z}_M^*} : z_j = \overline{z_{-j}}\}$ to $\mathbb{C}^{N/2}$, where $\mathbb{Z}_M^*$ is the multiplicative group of integer modulo M. The encoding $(\mathbb{C}^{N/2} \to \mathcal{R})$ and decoding are defined as follows.

- $\mathsf{Ecd}(\boldsymbol{z}; \Delta)$: For an $(N/2)$-dimensional vector $\boldsymbol{z}$, the encoding procedure returns

$$m(X) = \mathsf{Emb}^{-1}\left(\left\lfloor \Delta \cdot \pi^{-1}(\boldsymbol{z}) \right\rceil_{\mathsf{Emb}(\mathcal{R})}\right) \in \mathcal{R},$$

  where Δ is the scaling factor and $\left\lfloor \pi^{-1}(\boldsymbol{z}) \right\rceil_{\mathsf{Emb}(\mathcal{R})}$ denotes the discretization of $\pi^{-1}(\boldsymbol{z})$ into an element of $\mathsf{Emb}(\mathcal{R})$.

- $\mathsf{Dcd}(m; \Delta)$: For an input polynomial $m(X) \in \mathcal{R}$, output a vector

$$\boldsymbol{z} = \pi(\Delta^{-1} \cdot \mathsf{Emb}(m)) \in \mathbb{C}^{N/2},$$

  where its entry of index j is given as $z_j = \Delta^{-1} \cdot m(\zeta_M^j)$ for $j \in T$, where ζ_M is the M-th root of unity and T is a multiplicative subgroup of $\mathbb{Z}_M^*$ satisfying $\mathbb{Z}_M^*/T = \{\pm 1\}$. This can be basically represented by multiplication by an $N/2 \times N$ matrix $\mathbf{U}$ whose entries are $\mathbf{U}_{ij} = \zeta_i^j$, where $\zeta_i := \zeta^{5^i}$.

The infinity norm of $\mathsf{Emb}(a)$ for $a(X) \in \mathcal{R}$ is called the canonical embedding norm of a, denoted by $\|a\|_\infty^{\mathsf{can}} = \|\mathsf{Emb}(a)\|_\infty$. Refer [4] for the property of the canonical embedding norm.

Adopting notations in [4] and [2], I define three distributions as follows. For a real number $\sigma > 0$, $\mathcal{DG}(\sigma^2)$ denotes the distribution of vectors in $\mathbb{Z}^N$, whose entries are sampled independently from the discrete Gaussian distribution of variance σ^2. $\mathcal{HWT}(h)$ is the set of signed binary vectors in $\{0, \pm 1\}^N$ with Hamming weight h and $\mathcal{ZO}(\rho)$ denotes the distribution of vectors from $\{0, \pm 1\}^N$ with probability $\rho/2$

for each of ± 1 and probability of being zero $1 - \rho$. Suppose that we have ciphertexts of level l for $0 \leqslant l \leqslant L$, where level l means the maximum number of possible multiplications before bootstrapping. For convenience, I fix a power-of-two base $p > 0$ and a power-of-two modulus q and let $q_l = q \cdot p^l$. The base integer p is usually equivalent to the scaling factor Δ.

The CKKS scheme is defined with the following key generation, encryption, decryption, and the corresponding homomorphic operations.

- $\mathsf{KeyGen}(1^\lambda)$:
  - Given the security parameter λ, we choose a power-of-two M, an integer h, an integer P, a real number σ, and a maximum ciphertext modulus Q, such that $Q \geqslant q_L$.
  - Sample the following values:
$$s \leftarrow \mathcal{HWT}(h), a \leftarrow \mathcal{R}_{q_L}, e \leftarrow \mathcal{DG}(\sigma^2).$$
  - Set the secret key and the public key as
$$\mathsf{sk} := (1, s), \mathsf{pk} := (b, a) \in \mathcal{R}_{q_L}^2,$$
    respectively, where
$$b = -as + e \pmod{q_L}.$$
- $\mathsf{KSGen}_{\mathsf{sk}}(s')$:

  Sample $a' \leftarrow \mathcal{R}_{Pq_L}$ and $e' \leftarrow \mathcal{DG}(\sigma^2)$. Output the switching key
$$\mathsf{swk} := (b', a') \in \mathcal{R}_{Pq_L}^2,$$
  where $b' = -a's + e' + Ps' \pmod{Pq_L}$.
  - Set the evaluation key as $\mathsf{evk} := \mathsf{KSGen}_{\mathsf{sk}}(s^2)$.

- $\mathsf{Enc}_{\mathsf{pk}}(m)$:

  Sample $v \leftarrow \mathcal{ZO}(0.5)$ and $e_0, e_1 \leftarrow \mathcal{DG}(\sigma^2)$.

  Output $\boldsymbol{c} = v \cdot \mathsf{pk} + (m + e_0, e_1) \pmod{q_L}$.

- $\mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{c})$:

  Output $\bar{m} = \langle \boldsymbol{c}, \mathsf{sk} \rangle$.

- $\mathsf{Add}(\boldsymbol{c}_1, \boldsymbol{c}_2)$:

  For $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{R}_{q_l}^2$, output

$$\boldsymbol{c}_{\mathsf{add}} = \boldsymbol{c}_1 + \boldsymbol{c}_2 \pmod{q_l}.$$

- $\mathsf{Mult}_{\mathsf{evk}}(\boldsymbol{c}_1, \boldsymbol{c}_2)$:

  For $\boldsymbol{c}_1 = (b_1, a_1)$ and $\boldsymbol{c}_2 = (b_2, a_2) \in \mathcal{R}_{q_l}^2$, let

$$(d_0, d_1, d_2) := (b_1 b_2, a_1 b_2 + a_2 b_1, a_1 a_2) \pmod{q_l}.$$

  Output

$$\boldsymbol{c}_{\mathsf{mult}} = (d_0, d_1) + \mathsf{KS}_{\mathsf{evk}}((0, d_2)),$$

  where $\lfloor \cdot \rceil$ denotes the rounding operation.

- $\mathsf{cAdd}(\boldsymbol{c}_1, \boldsymbol{a}; \Delta)$:

  For a $\boldsymbol{a}\mathbb{C}^{N/2}$ and a scaling factor Δ, output

$$\boldsymbol{c}_{\mathsf{cadd}} \leftarrow \boldsymbol{c} + (\mathsf{Ecd}(\boldsymbol{a}; \Delta), 0).$$

- $\mathsf{cMult}(\boldsymbol{c}_1, \boldsymbol{a}; \Delta)$:

  For a $\boldsymbol{a}\mathbb{C}^{N/2}$ and a scaling factor Δ, output

$$\boldsymbol{c}_{\mathsf{cmult}} \leftarrow \mathsf{Ecd}(\boldsymbol{a}; \Delta) \cdot \boldsymbol{c}.$$

- $\mathsf{RS}_{l\to l'}(\boldsymbol{c})$:

  For $\boldsymbol{c} \in \mathcal{R}_{q_l}^2$, output
  $$\boldsymbol{c}_{\mathsf{RS}} = \left\lfloor \frac{q_{l'}}{q_l}\boldsymbol{c} \right\rceil \pmod{q_{l'}}.$$
  The subscript is omitted when $l' = l - 1$.

- $\mathsf{KS}_{\mathsf{swk}}(\boldsymbol{c})$:

  For $\boldsymbol{c} = (c_0, c_1) \in \mathcal{R}_{q_l}^2$, output
  $$\boldsymbol{c}_{\mathsf{KS}} = (c_0, 0) + \left\lfloor P^{-1} \cdot c_1 \cdot \mathsf{swk} \right\rceil \pmod{q_l}.$$

I note that $\boldsymbol{c}_{\mathsf{mult}} = (d_0, d_1) + \mathsf{KS}_{\mathsf{evk}}(0, d_2)$. The key switching techniques are used to provide various operations such as complex conjugate and rotation.

There are computationally more efficient variants of the CKKS scheme, namely the RNS-CKKS scheme in [20] and [56], and the basic operations supported therein are similar. Hence, it is worth noting that the proposed methods in this dissertation aim for all the variants of the CKKS scheme as well as the original CKKS scheme.

2.3.2 CKKS Scheme in RNS

The RNS-CKKS scheme performs all operations in RNS. In other words, the power-of-two modulus $q_l = q \cdot p^l$ is replaced with $\prod_{i=0}^{l} p_i$, where p_i's are chosen as primes that satisfy $p_i = 1 \pmod{2N}$ to support efficient number theoretic transform (NTT). These prime numbers are also chosen such that p/p_i is in the range $(1 - 2^{\eta}, 1 + 2^{\eta})$, where η is kept small, for a scaling factor p. I note that $q_0 = p_0$ is much greater than p as the coefficients of final message should not be greater than the ciphertext modulus q_0.

The RNS-CKKS scheme differs from the original CKKS scheme in the rescaling and key switching. To take advantage of RNS, I use hybrid key switching technique proposed in [20]. First, for predefined dnum, a small integer such as 4, I define partial products $\{\tilde{q}_j\}_{0\leqslant j<\mathsf{dnum}} = \left\{\prod_{i=j\alpha}^{(j+1)\alpha-1} p_i\right\}_{0\leqslant j<\mathsf{dnum}}$, where $\alpha = (L+1)/\mathsf{dnum}$. For level l and $\mathsf{dnum}' = \lceil (l+1)/\alpha \rceil$, I define [20]

$$\mathcal{WD}_l(a) = \left(\left[a\frac{\tilde{q}_0}{q_l} \right]_{\tilde{q}_0}, \cdots, \left[a\frac{\tilde{q}_{\mathsf{dnum}'-1}}{q_l} \right]_{\tilde{q}_{\mathsf{dnum}'-1}} \right) \in \mathcal{R}^{\mathsf{dnum}'},$$

$$\mathcal{PW}_l(a) = \left(\left[a\frac{q_l}{\tilde{q}_0} \right]_{q_l}, \cdots, \left[a\frac{q_l}{\tilde{q}_{\mathsf{dnum}'-1}} \right]_{q_l} \right) \in \mathcal{R}_{q_l}^{\mathsf{dnum}'}.$$

Then, for any $(a,b) \in \mathcal{R}_{q_l}^2$, we have

$$\langle \mathcal{WD}_l(a), \mathcal{PW}_l(b) \rangle = a \cdot b \pmod{q_l}.$$

Then, the rescaling and key switching in the RNS-CKKS scheme are defined as follows:

- $\mathsf{KSGen}_{\mathsf{sk}}(s')$: For auxiliary modulus $P = \prod_{i=0}^{k} p_i' \approx \max_j \tilde{q}_j$, sample $a_k' \leftarrow \mathcal{R}_{Pq_L}$ and $e_k' \leftarrow \mathcal{DG}(\sigma^2)$. Output the switching key

  $$\begin{aligned} \mathsf{swk} :=& (\mathsf{swk}_0, \mathsf{swk}_1) \\ =& (\{b_k'\}_{k=0}^{\mathsf{dnum}'-1}, \{a_k'\}_{k=0}^{\mathsf{dnum}'-1}) \in \mathcal{R}_{Pq_L}^{2\times \mathsf{dnum}'}, \end{aligned}$$

  where $b_k' = -a_k's + e_k' + P \cdot \mathcal{PW}(s')_k \pmod{Pq_L}$.

  – Set the evaluation key as $\mathsf{evk} := \mathsf{KSGen}_{\mathsf{sk}}(s^2)$.

- $\mathsf{RS}(\boldsymbol{c})$:

  For $\boldsymbol{c} \in \mathcal{R}_{q_l}^2$, output

  $$\boldsymbol{c}_{\mathsf{RS}} = \lfloor p_l^{-1}\boldsymbol{c} \rceil \pmod{q_{l-1}}.$$

- $\mathsf{KS}_{\mathsf{swk}}(\boldsymbol{c})$:

  For $\boldsymbol{c} = (c_0, c_1) \in \mathcal{R}_{q_l}^2$ and $\mathsf{swk} := (\mathsf{swk}_0, \mathsf{swk}_1)$, output

  $$\begin{aligned} \boldsymbol{c}_{\mathsf{KS}} = (c_0 + &\left\lfloor \frac{\langle \mathcal{WD}_l(c_1), \mathsf{swk}_0 \rangle}{P} \right\rceil, \\ &\left\lfloor \frac{\langle \mathcal{WD}_l(c_1), \mathsf{swk}_1 \rangle}{P} \right\rceil) \pmod{q_l}. \end{aligned}$$

To remove the approximation error introduced by approximate rescaling, one can use different scaling factor for each level as given in [23].

I note that FullRNS-HEAAN library is ($\mathsf{dnum} = 1$)-case and SEAL is ($\mathsf{dnum} = L+1$)-case. It is also noted that the key switching method using $\mathcal{WD}$ and $\mathcal{PD}$ can also be applied to the original CKKS scheme, and thus the main differences between the original CKKS scheme and the RNS-CKKS scheme are their modulus and rescaling algorithm. However, since I use HEAAN as the library of the original CKKS scheme and SEAL as the library of the RNS-CKKS scheme, I provide a description for each.

2.3.3 Bootstrapping of the CKKS Scheme

There are several studies for bootstrapping of the CKKS scheme [19–21, 24, 25]. The bootstrapping consists of the following four steps: ModRaise, CoeffToSlot, EvalMod, and SlotToCoeff.

Modulus Raising (ModRaise)

ModRaise is the procedure to change the modulus of a ciphertext to a larger modulus. Let $\boldsymbol{c}$ be the ciphertext satisfying $m(X) = [\langle \boldsymbol{c}, \mathsf{sk}\rangle]_q$. It can be seen that $t(X) = \langle \boldsymbol{c}, \mathsf{sk}\rangle \pmod{X^N + 1}$ is of the form $t(X) = qI(X) + m(X)$ for $I(X) \in \mathcal{R}$ with a bound $\|I(X)\|_\infty < K$, where K is upper bounded by $\mathcal{O}(\sqrt{h})$. The following procedure aims to compute the remainder of the coefficients of $t(X)$ when it is divided by q, homomorphically. In other words, we homomorphically calculate the modulus reduction function, $[\cdot]_q$ for the coefficients of $t(X)$. However, as the modulus reduction is not an arithmetic operation, it should be evaluated by an approximate polynomial and thus, the crucial point of bootstrapping is to find a polynomial approximating the modulus reduction function.

Homomorphic Evaluation of Encoding (COEFFTOSLOT)

Approximate homomorphic operations are performed in plaintext slots. Thus, to deal with $t(X)$, we have to put polynomial coefficients in plaintext slots. In COEFFTOSLOT step, $\mathsf{Emb}^{-1} \circ \pi^{-1}$ is performed homomorphically using matrix multiplication [19] or FFT-like operations or a hybrid method of both [21]. Then, we have two ciphertexts encrypting $\boldsymbol{z}_0' = (t_0, \ldots, t_{\frac{N}{2}-1})$ and $\boldsymbol{z}_1' = (t_{\frac{N}{2}}, \ldots, t_{N-1})$ (or combined using imaginary, e.g., $(t_0 + i \cdot t_{\frac{N}{2}}, \ldots, t_{\frac{N}{2}-1} + i \cdot t_{N-1})$), where t_j denotes the j-th coefficient of $t(X)$.

Evaluation of the Approximate Modulus Reduction (EVALMOD)

In the EVALMOD step, an approximate evaluation of modulus reduction function of t_i's is performed. As the modulus reduction function is not represented by additions and multiplications, an approximate polynomial for this function is used, instead. For approximation, it is desirable to control the size of the message so that we can ensure $m_i \leqslant \epsilon \cdot q$ for a small ϵ, where m_i is a coefficient of the message polynomial $m(X)$. At first, Cheon et al. approximated the modulus reduction function as $\frac{q}{2\pi} \sin\left(\frac{2\pi t}{q}\right)$ and used an approximate polynomial for sine function using Taylor series expansion of exponential function in [19]. Hence there exists a fundamental error between the approximate polynomial and modulus reduction function, that is, the difference of sine function and modulus reduction function, which is upper bounded by

$$\left| m - \frac{q}{2\pi} \sin\left(2\pi \frac{m}{q}\right) \right| \leqslant \frac{q}{2\pi} \cdot \frac{1}{3!} \left(\frac{2\pi |m|}{q} \right)^3,$$

where $t(X) = qI(X) + m(X)$. Then, a Taylor series expansion and the double angle formula were adopted as the approximate polynomial of the sine function.

After that, the method of improving polynomial approximation using Chebyshev interpolation was proposed [21]. By selecting optimized nodes for Chebyshev interpolation, Han et al. significantly improved the performance of the approximation in the bootstrapping of the CKKS scheme [20]. However, in both approaches, the sine function is used, and thus there is still the fundamental approximation error. Then, a direct approximation method using a discretization of the target function and the least square method is proposed in [24]. A composition with inverse sine function is proposed in [25] to remove the fundamental approximation error between the sine function and the modulus reduction. In [25], an approximation algorithm that finds the minimax approximate polynomial, namely the modified Remez algorithm, is used.

Homomorphic Evaluation of Decoding (SLOTTOCOEFF)

SLOTTOCOEFF is the inverse operation of COEFFTOSLOT.

2.3.4 Statistical Characteristics of Modulus Reduction and Failure Probability of Bootstrapping of the CKKS Scheme

After MODRAISE, the plaintext in the ciphertext $\boldsymbol{c} = (c_0, c_1)$ is given as

$$\begin{aligned} t(X) &= q \cdot I(X) + m(X) \\ &= \langle \boldsymbol{c}, \mathsf{sk} \rangle \left(\bmod X^N + 1 \right). \end{aligned}$$

As sk is sampled from the distribution $\mathcal{HWT}(h)$, it has a small Hamming weight h. Each coefficient of a ciphertext (c_0, c_1) is an element of $\mathbb{Z}_q$ and thus, each coefficient of $\langle \boldsymbol{c}, \mathsf{sk} \rangle = c_0 + c_1 s$ is considered as a sum of $(h+1)$ elements in $\mathbb{Z}_q$. Therefore, $I(X) = \left\lfloor \frac{1}{q} \langle \boldsymbol{c}, \mathsf{sk} \rangle \right\rceil$ is upper bounded by $\frac{1}{2}(h+1)$. In practice, a heuristic assumption is used and a high-probability upper bound $K = O(\sqrt{h})$ for $\|I\|_\infty$ is used. For example, it is usual to use $h = 64$ and then it is assumed that $\|I\|_\infty < K = 12$.

As (c_0, c_1) is ciphertext, each coefficient of c_0 and c_1 can be considered as distributed uniformly at random by the RLWE assumption. Hence, each coefficient of t is the sum of $h+1$ independent uniform random variables; in other words, it follows a distribution similar to the well-known Irwin–Hall distribution. The approximate polynomial for modulus reduction is designed under the assumption that $\|I\|_\infty < K$. The high-probability upper bound K is acceptable, but it outputs a useless value when the input is not in the desired domain, and this results in the bootstrapping failure. Thus, by using the high-probability upper bound, the bootstrapping becomes efficient, but it has a certain failure probability. For example, the probability that $\|I\|_\infty \geqslant K$ is $2^{-24.06}$, when $h = 64$ and $K = 12$.

As we know the distribution of I, the probability distribution of each coefficient can be obtained. I note here that a probabilistic approach is already used in the error estimation and bootstrapping of the CKKS scheme, and thus it is reasonable to reduce the error of the CKKS scheme in a probabilistic manner. This approach can be applied in all of the homomorphic computation and polynomial approximation using the CKKS scheme.

2.4 Approximate Polynomial and Signal-to-Noise Perspective for Approximate Homomorphic Encryption

2.4.1 Chebyshev Polynomials

The Chebyshev interpolation is a well-known polynomial interpolation method that uses the Chebyshev polynomials as a basis of the interpolation polynomial. The Chebyshev polynomial of the first kind, in short, the Chebyshev polynomial is defined by the recursive relation [57]

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x). \end{aligned}$$

The Chebyshev polynomial of degree n has n distinct roots in the interval $[-1, 1]$ and all its extrema are also in $[-1, 1]$. Moreover, $\frac{1}{2^{n-1}}T_n(x)$ is the polynomial, whose maximal absolute value is minimal among monic polynomials of degree n and the absolute value is $\frac{1}{2^{n-1}}$. In addition to the above, the Chebyshev polynomial has desirable properties as a basis for an approximate polynomial.

In Chebyshev interpolation, the n-th degree polynomial $p_n(x)$ is represented as a sum of the Chebyshev polynomials of the form

$$p_n(x) = \sum_{i=0}^{n} c_i T_i(x).$$

$p_n(x)$ is an approximate polynomial for $f(x)$ by interpolating $n+1$ points $\{x_0, x_1, \ldots, x_n\}$, where

$$c_i = \frac{2}{n+1} \sum_{k=0}^{n} f(x_k) T_i(x_k).$$

Selecting points $\{x_0, x_1, \ldots, x_n\}$ is key for a good approximation.

2.4.2 Signal-to-Noise Perspective of the CKKS Scheme

In the field of communications, there has been extensive research on noisy media such as wireless communication or data storage. In this perspective, the CKKS scheme is one of the noisy media; encryption and decryption correspond to transmission and reception, respectively. The message in the ciphertext is the signal, and as the final output has an additive error due to RLWE security, rounding, and approximation, the CKKS scheme itself can be considered as a noisy media.

The SNR is the most widely used measure of signal quality, which is defined as the ratio of the signal power to the noise power as

$$\text{SNR} = \frac{P_S}{P_N} = \frac{E[S^2]}{E[N^2]},$$

where S and N denote the signal (message) and noise (error), respectively. The power of a signal S is defined by $P_S = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} S(t)^2 dt$. As the signal and noise must be measured at the same or equivalent points in a system, the ratio of power is equivalent to the ratio of energy (or the second moment), $\frac{E[S^2]}{E[N^2]}$. As shown in the definition, the larger SNR, the better the signal quality.

The energy of noise should be minimized to maximize the SNR of the encrypted data because it is expensive to increase the energy of the message. An easy way to increase SNR is to increase the signal power, but in a real system, it is not easy due to regulation or physical constraints. This is the same for the CKKS scheme. The message can be multiplied by the larger scaling factor to increase the power of the message. However, if one uses a larger scaling factor, the level of the ciphertext decreases or larger parameters should be used to keep the encryption secure under the RLWE problem. In addition, usually in the RNS-CKKS scheme, the scaling factor is limited to 64 bits for efficient implementation. Hence, to increase SNR, it is important to reduce the power of noise in the CKKS scheme rather than to increase the power of the signal.

The CKKS scheme trades off the efficiency of computation and precision of the message, and improving the precision will make the CKKS scheme more reliable. Error estimation of the CKKS scheme so far has been focused on the high-probability upper bound of the error, and the upper bound was tracked by using the upper bound of the message [4, 19]. As the homomorphic operation continues, the error bound becomes quite loose, and its statistical significance may fade. In this dissertation, I propose methods to reduce the power (or energy) of error in encrypted data during homomorphic computation over ciphertext. I note that when the mean of error is zero, the energy of error is the same as its variance. Therefore, hereinafter, the energy and variance of errors are abused if its mean is zero.

2.5 Preliminary for Code-Based Cryptography

2.5.1 The McEliece Cryptosystem

Let $\mathbf{G}$ be the generator matrix of a q-ary (n, k) Goppa code with error correction capability t, where q is a prime power. Then, the key generation, encryption, and decryption of the McEliece cryptosystem are given as follows:

- $\mathsf{KeyGen}(\lambda)$

  - Given parameter λ, choose code length n, code dimension k, and error correction capability t.
  - Build the generator matrix $\mathbf{G}$ of an (n, k) Goppa code with error correction capability t.
  - Let $\mathbf{S}$ be a $k \times k$ random nonsingular matrix and $\mathbf{P}$ be an $n \times n$ permutation matrix.
  - Set the public key and the secret key as

$$\mathsf{pk} = (\mathbf{G}_{\text{pub}} = \mathbf{SGP}) \text{ and } \mathsf{sk} = (\mathbf{S}, \mathbf{G}, \mathbf{P}).$$

- $\mathsf{Enc}_{\mathsf{pk}}(\boldsymbol{m})$
  - Return $\boldsymbol{c}$ such that $\boldsymbol{c}^{\mathsf{T}} = \boldsymbol{m}^{\mathsf{T}}\mathbf{G}_{\text{pub}} + \boldsymbol{e}^{\mathsf{T}}$, where $\boldsymbol{e}$ is an error vector with Hamming weight t.
- $\mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{c})$
  - Compute $\boldsymbol{c}'^{\mathsf{T}} = \boldsymbol{c}^{\mathsf{T}}\mathbf{P}^{-1}$.
  - By decoding $\boldsymbol{c}'$, one then computes $\boldsymbol{m}'$.
  - Return $\hat{\boldsymbol{m}}^{\mathsf{T}} = \boldsymbol{m}'^{\mathsf{T}}\mathbf{S}^{-1}$.

2.5.2 CFS Signature Scheme

CFS signature scheme is an algorithm that applies the FDH methodology to the Niederreiter cryptosystem. The CFS signature scheme is based on Goppa codes, as McEliece cryptosystem. A summary of the CFS signature scheme is given in Algorithm 1.

As described in Algorithm 1, the signing process iterates until a decodable syndrome is obtained. The probability that a given random syndrome can be decoded is $\frac{\sum_{i=0}^{t}\binom{n}{i}}{2^{n-k}} \simeq \frac{1}{t!}$. Hence, the error correction capability $t = \frac{n-k}{\log n}$ should be sufficiently small to reduce the number of iterations. Thus, the high-rate Goppa codes should be used. Regarding the key size, the complexity of the decoding attack on the CFS signature scheme is known to be a small power of the key size, namely, $\approx \text{keysize}^{t/2}$. Hence, the key size should be fairly large to meet a certain security level. In summary, the CFS signature scheme is insecure and inefficient owing to the use of Goppa codes.

2.5.3 Reed–Muller Codes and Recursive Decoding

RM codes were introduced by Muller and Reed [35, 36] and its decoding algorithm, so-called recursive decoding, was proposed in [58]. There are various definitions of RM codes, but I adopt a recursive definition here as recursive decoding is defined by using this structure. An RM code $\mathrm{RM}_{(r,m)}$ is a linear binary $(n = 2^m, k = \sum_{i=0}^{r} \binom{m}{i})$ code, where r and m are integers. $\mathrm{RM}_{(r,m)}$ is defined as $\mathrm{RM}_{(r,m)} := \{(\boldsymbol{u}|\boldsymbol{u}+\boldsymbol{v})|\boldsymbol{u} \in \mathrm{RM}_{(r,m-1)}, \boldsymbol{v} \in \mathrm{RM}_{(r-1,m-1)}\}$, where $\mathrm{RM}_{(0,m)} := \{(0,\ldots,0),(1,\ldots,1)\}$ with code length 2^m and $\mathrm{RM}_{(m,m)} := \mathbb{F}_2^{2^m}$. This is the well-known Plotkin's construction, and its generator matrix is given by

$$\mathbf{G}_{(r,m)} = \begin{bmatrix} \mathbf{G}_{(r,m-1)} & \mathbf{G}_{(r,m-1)} \\ \mathbf{0} & \mathbf{G}_{(r-1,m-1)} \end{bmatrix},$$

where $\mathbf{G}_{(r,m)}$ is the generator matrix of $\mathrm{RM}_{(r,m)}$.

Recursive decoding is a soft-decision decoding algorithm that depends on the recursive structure of the RM codes; it is described in detail in Algorithm 2, where $\boldsymbol{y}' \cdot \boldsymbol{y}''$ denotes the component-wise multiplication of the vectors $\boldsymbol{y}'$ and $\boldsymbol{y}''$. In recursive decoding, a binary symbol $a \in \{0,1\}$ is mapped onto $(-1)^a$, and it is assumed that all codewords belong to $\{-1,1\}^n$.

First, $\boldsymbol{y}''$ (the second half of the received vector $\boldsymbol{y}$) is component-wisely multiplied by $\boldsymbol{y}'$ (the first half of the received vector). Then, a codeword from $\mathrm{RM}_{(r,m-1)}$ (i.e., $\boldsymbol{u}$) is removed from $\boldsymbol{y}''$ as it is both in $\boldsymbol{y}'$ and $\boldsymbol{y}''$, and then only $\boldsymbol{v}$ and the error vector remain. This is regarded as a codeword of $\mathrm{RM}_{(r-1,m-1)}$ added to an error vector and is referred to as $\hat{\boldsymbol{v}}$. Using $\hat{\boldsymbol{v}}$, we can remove the codeword of $\mathrm{RM}_{(r-1,m-1)}$ from the second half of the received vector. $\boldsymbol{y}'$ is then added to $\boldsymbol{y}'' \cdot \hat{\boldsymbol{v}}$, and the sum is divided

Algorithm 1 CFS signature scheme [38]

```
Key generation:
    $\mathbf{H}$ is the parity check matrix of an $(n, k)$ Goppa code
    The error correction capability $t$ is $\frac{n-k}{\log n}$
    $\mathbf{S}$ and $\mathbf{Q}$ are an $(n-k) \times (n-k)$ scrambler matrix and $n \times n$ permutation matrix,
    respectively
    Secret key: $\mathbf{H}, \mathbf{S},$ and $\mathbf{Q}$
    Public key: $\mathbf{H}' \leftarrow \mathbf{SHQ}$
Signing:
    $\boldsymbol{m}$ is a message to be signed
    $i \leftarrow 1$
    Do
        $i \leftarrow i + 1$
        Find syndrome $\boldsymbol{s} \leftarrow h(h(\boldsymbol{m})|i)$
        Compute $\boldsymbol{s}' \leftarrow \mathbf{S}^{-1}\boldsymbol{s}$
    Until a decodable syndrome $\boldsymbol{s}'$ is found
    Find an error vector satisfying $\mathbf{H}\boldsymbol{e}'^T \leftarrow \boldsymbol{s}'$
    * Compute $\boldsymbol{e}^T \leftarrow \mathbf{Q}^{-1}\boldsymbol{e}'^T$, and then the signature is $(\boldsymbol{m}, \boldsymbol{e}, i)$
Verification:
    Check $\mathrm{wt}(\boldsymbol{e}) \leqslant t$ and $\mathbf{H}'\boldsymbol{e}^T = h(h(\boldsymbol{m})|i)$
    If True, then return ACCEPT; else, return REJECT
```

by 2. This is regarded as a codeword of $\mathrm{RM}_{(r,m-1)}$ added to the error vector, and then decoding is performed. Recursively, the received vector is further divided into sub-vectors of length $n/4$, $n/8$, etc. Finally, we reach $\mathrm{RM}_{(m,m)}$ or $\mathrm{RM}_{(0,m)}$, then the division terminates and the minimum distance (MD) decoding of $\mathrm{RM}_{(m,m)}$ or $\mathrm{RM}_{(0,m)}$, which is trivial, is performed. The decoding for the entire code is performed by reconstructing these results into $(U, U+V)$ form.

Algorithm 2 Recursive decoding of RM code [58]

```
function RecursiveDecoding($\boldsymbol{y}, r, m$)
    if $r = 0$ then
        Perform MD decoding on RM$(0, m)$
    else if $r = m$ then
        Perform MD decoding on RM$(r, r)$
    else
        $(\boldsymbol{y}'|\boldsymbol{y}'') \leftarrow \boldsymbol{y}$
        $\boldsymbol{y}^{\boldsymbol{v}} = \boldsymbol{y}' \cdot \boldsymbol{y}''$
        $\hat{\boldsymbol{v}} \leftarrow$ RecursiveDecoding($\boldsymbol{y}^{\boldsymbol{v}}, r-1, m-1$)
        $\boldsymbol{y}^{\boldsymbol{u}} \leftarrow (\boldsymbol{y}' + \boldsymbol{y}'' \cdot \hat{\boldsymbol{v}})/2$
        $\hat{\boldsymbol{u}} \leftarrow$ RecursiveDecoding($\boldsymbol{y}^{\boldsymbol{u}}, r, m-1$)
        Output $(\hat{\boldsymbol{u}}|\hat{\boldsymbol{u}} \cdot \hat{\boldsymbol{v}})$
    end if
end function
```

2.5.4 IKKR Cryptosystems

Recently, Ivanov *et al.* proposed variants of the McEliece cryptosystem such as prototype, modified version of prototype, and the upgraded IKKR cryptosystems whose Hamming weights of error vectors are arbitrary value rather than t [52]. The prototype IKKR cryptosystem is composed of the following procedures.

- $\mathsf{KeyGen}(\lambda)$
  - Given parameter λ, choose code length n, code dimension k, and error correction capability t.
  - Build a $k \times n$ generator matrix $\mathbf{G}$ of an (n, k, t) linear code $\mathcal{C}$.
  - Construct an arbitrary nonsingular $n \times n$ matrix $\mathbf{M}$ and an $n \times n$ permutation matrix $\mathbf{P}$. Let $\mathbf{G}_0$ be a matrix whose rows are n codewords of code $\mathcal{C}$.
  - Set secret key and public key as $\mathsf{pk} := (\mathbf{G}', \mathbf{G}'_1) = (\mathbf{GM}, (\mathbf{G}_0 - \mathbf{P})\mathbf{M})$ and $\mathsf{sk} := (\mathbf{G}, \mathbf{M}, \mathbf{P})$, respectively.
- $\mathsf{Enc}_{\mathsf{pk}}(\boldsymbol{m})$
  - Generate an arbitrary vector $\boldsymbol{e}$ with Hamming weight at most t.
  - For given plaintext $\boldsymbol{m}$, return $\boldsymbol{c}$ such that

$$\boldsymbol{c}^{\mathsf{T}} = \boldsymbol{m}^{\mathsf{T}}\mathbf{G}' + \boldsymbol{e}^{\mathsf{T}}\mathbf{G}'_1.$$

- $\mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{c})$
  - $\boldsymbol{y}^{\mathsf{T}} = \boldsymbol{c}^{\mathsf{T}}\mathbf{M}^{-1}$.
  - Decode $\boldsymbol{y}$ by applying decoding algorithm for $\mathcal{C}$ and recover $\boldsymbol{e}^{\mathsf{T}}\mathbf{P}$. Then, find $\boldsymbol{e}^{\mathsf{T}} = \boldsymbol{e}^{\mathsf{T}}\mathbf{PP}^{-1}$.
  - Find and return $\hat{\boldsymbol{m}}$ from $\hat{\boldsymbol{m}}^{\mathsf{T}}\mathbf{G} = \boldsymbol{y} - \boldsymbol{e}^{\mathsf{T}}(\mathbf{G}_0 - \mathbf{P})$.

The upgraded IKKR cryptosystem is composed of the following procedures.

- $\mathsf{KeyGen}(\lambda)$
  - Given parameter λ, choose code length n, code dimension k, and error correction capability t.
  - Build a $k \times n$ generator matrix $\mathbf{G}$ of an (n, k, t) linear code $\mathcal{C}$.

- Construct $n \times n$ matrices $\mathbf{Q}$ and $\mathbf{T}$ such that $\mathbf{QT}$ has rank $n-k$ and has k zero columns at indices in $\mathcal{J}$, where $\mathcal{J}$ is an information set of $\mathcal{C}$.
- Generate $\mathbf{M}$ and $\mathbf{G}_0$ as in the prototype IKKR cryptosystem.
- Set secret key and public key as $\mathsf{pk} := (\mathbf{G}', \mathbf{G}'_2) = (\mathbf{GM}, \mathbf{Q}(\mathbf{G}_0 + \mathbf{T})\mathbf{M})$ and $\mathsf{sk} := (\mathbf{G}, \mathbf{M}, \mathbf{T}, \mathbf{Q}, \mathbf{G}_0, \mathcal{J})$, respectively.

• $\mathsf{Enc}_{\mathsf{pk}}(\boldsymbol{m})$

- Generate a q-ary random vector $\boldsymbol{e}$ with arbitrary Hamming weight.
- For given plaintext $\boldsymbol{m}$, return $\boldsymbol{c}$ such that $\boldsymbol{c}^\intercal = \boldsymbol{m}^\intercal \mathbf{G}' + \boldsymbol{e}^\intercal \mathbf{G}'_2$.

• $\mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{c})$

- $\boldsymbol{y}^\intercal = \boldsymbol{c}^\intercal \mathbf{M}^{-1}$.
- $\boldsymbol{e}_1^\intercal = \boldsymbol{y}^\intercal - \boldsymbol{y}_{\mathcal{J}}^\intercal \mathbf{G}_{\mathcal{J}}^{-1} \mathbf{G}$.
- Calculate $\boldsymbol{e}_2^\intercal = \boldsymbol{e}_1^\intercal \mathbf{T}^{-1} \mathbf{G}_0$.
- Find and return $\hat{\boldsymbol{m}}$ from $\hat{\boldsymbol{m}}^\intercal \mathbf{G} = \boldsymbol{y}^\intercal - \boldsymbol{e}_1^\intercal - \boldsymbol{e}_2^\intercal$.

It is noted that when $\boldsymbol{c}$ is a properly encrypted ciphertext by using $\mathsf{Enc}(\boldsymbol{m})$, $\boldsymbol{e}_1^\intercal = \boldsymbol{e}^\intercal \mathbf{QT}$ and $\boldsymbol{e}_2^\intercal = \boldsymbol{e}^\intercal \mathbf{QG}_0$; thus, $\hat{\boldsymbol{m}} = \boldsymbol{m}$.

The process to construct $\mathbf{Q}$ and $\mathbf{T}$ is given in Algorithm 3. As it will be detailed later, the success probability of the attack is 1 because of the construction of $\mathbf{Q}$ and $\mathbf{T}$.

In the modified version of prototype IKKR cryptosystem, $\mathbf{G}'_1 = \mathbf{WD}(\mathbf{G}_0 - \mathbf{P})\mathbf{M}$ is used instead of $\mathbf{G}'_2$, where $\mathbf{W}$ is a random $n \times n$ nonsingular matrix and $\mathbf{D}$ is a randomly chosen $n \times n$ diagonal matrix with t non-zero elements on the diagonal. I refer the readers to [52] for more details on the modified version of prototype IKKR cryptosystem.

Algorithm 3 Construction of $\mathbf{Q}$ and $\mathbf{T}$ [52]

```
Output: Matrices $\mathbf{Q}$ and $\mathbf{T}$ such that $\mathbf{QT}$ has rank $n-k$ and $k$ zero columns at indices in $\mathcal{J}$.
1: $\mathbf{T} \leftarrow$ an $n \times n$ random nonsingular matrix.
2: $\mathbf{L} \leftarrow$ an $n \times (n-k)$ arbitrary full-rank matrix.
3: $\mathbf{H}_{\mathcal{J}} \leftarrow$ a parity check matrix of code generated by $\mathbf{T}_{\mathcal{J}}$.
4: $\mathbf{Q} \leftarrow \mathbf{L}\mathbf{H}_{\mathcal{J}}$.
5: return $\mathbf{Q}$ and $\mathbf{T}$.
```

Chapter 3

Privacy-Preserving Machine Learning via FHE Without Bootstrapping

3.1 Introduction

In this chapter, I propose an efficient protocol that dismisses the most time-consuming operation of PPML using HE, bootstrapping, by using communication resources. Homomorphic encryption enables the computation of encrypted data, but one of its drawbacks is computational complexity. Usually, the complexity of basic operations of homomorphic encryption is $O(l^2)$, where l is the level of a ciphertext, which is the maximum depth of operation that can be performed using the ciphertext. However, for the accuracy of machine learning, depth is important, which inevitably increases the amount of computation. The bootstrapping initializes the level of the ciphertext; the complexity of the homomorphic operation can be maintained at any depth. However, bootstrapping is an expensive operation. In this dissertation, I propose a method of PPML without bootstrapping using communication resources. In other words, it replaces bootstrapping with a procedure that induces a small amount of communication between participants.

3.2 Information Theoretic Secrecy and HE for Privacy-Preserving Machine Learning

In this chapter, the use of information-theoretic secrecy to replace the bootstrapping in PPML is presented. The basic scenario of PPML via HE is a secure two-party computation protocol, which is composed of receiver and sender, the data owner, and computation participants, respectively. This dissertation aims at the inference scenario only. The receiver has data for inference, and the sender has a massive computation resource and the trained model.

The sender and receiver both have to keep their inputs secure while the final output of the computation is presented only to the receiver. In a secure two-party computation protocol, the receiver and the sender evaluate the neural network, $f(x, \theta)$ while keeping x and θ kept secure to the opposite, where x is a confidential data of receiver and θ is the trained weights of the sender.

However, when f is a deep circuit, bootstrapping is necessary, and it is an expensive operation. One may come up with the idea that sends back the intermediate value to the receiver and let the receiver decrypt and re-encrypt it and then send it back to the sender [59]. As a result, the sender could get the fresh ciphertext for the same message. There were several studies to avoid the bootstrapping using communication resources [55, 59]; however, a naive approach reveals the intermediate values to the receiver, and it leaks the information of the parameters of the neural network. Moreover, assigning some operation to the receiver using MPC techniques such as garbled circuit [55] might occur unnecessary communication resources, and the performance learn upon the computational ability of the receiver. A receiver can be a device with very limited computing power, such as an IoT device. Hence, computation and communication assigned to the receiver should be minimized to guarantee the quality of service.

The receiver's data is secured with homomorphic encryption, but how can one homomorphically encrypt and decrypt the intermediate values? The sender's conditions differ from the assumptions in a usual cryptosystem. The negative part is that the sender can only perform simple operations like addition and multiplication with low depth. However, on the positive side, the subject of encryption coincides with the subject of decryption. This means that no key setting is required, and thus he can use one-time keys. A one-time pad (OTP) is the encryption that the sender performs homomorphically to secure his data.

In the proposed protocol, when bootstrapping is required, the sender homomorphically encrypts the intermediate value by OTP with his own key and sends the ciphertext to the receiver; then, the receiver decrypts the given ciphertext, and he gets the encrypted intermediate value by OTP. Finally, the receiver re-encrypts the intermediate value with HE and sends it to the sender; the sender can decrypt the intermediate value by OTP, and then it is a fresh ciphertext for the intermediate value. All the operations the receiver should do is the encryption and decryption, which is the same as the succinct scenario with FHE. In other words, the proposed protocol does not require any additional hardware and software for additional work for MPC, such as symmetric key encryption and decryption, as in hybrid protocols.

In this dissertation, I mainly utilize the CKKS scheme as it is very useful for the machine learning application, but it has not been widely adopted because its bootstrapping is quite expensive and noisy. However, it is straightforward to adopt the proposed protocol to other HE scheme such as BGV [9], (B)FV [7, 8, 11]. Moreover, it is shown that even the average amount of computation and the number of communication can be reduced.

3.2.1 The Failure Probability of Ordinary CKKS Bootstrapping

The bootstrapping of the CKKS scheme has a certain probability of failure. In the CKKS bootstrapping proposed so far, the fact that the ciphertext is sparse and has small values is used [19]. Specifically, the Hamming weight of coefficients of s is $h = 64$

and it has ± 1 for its non-zero component. The result of decryption of a ciphertext, $\mathsf{ct} = (b, a)$ with the modulus q is given as

$$\begin{aligned} m &= \langle \mathsf{ct}, \mathsf{sk} \rangle \pmod q \\ &= \langle (b, a), (1, s) \rangle \pmod q \\ &= b + as \pmod q \,. \end{aligned}$$

As ct is a ciphertext, it is reasonable to assume that the coefficients of a and b are independent uniformly random elements in $\mathbb{Z}_q$ following the LWE assumption. Thus, each coefficient of $t = b + as$ follows the distribution of the sum of $h + 1$ random uniform distribution in $[-q/2, q/2)$. An heuristic approximation that $|t|_\infty < K$ is used in the bootstrapping so far [19]. Of course $|t|_\infty \geqslant K$ occurs very rarely, but is probable, and its probability is approximated as

$$\Pr(|t|_\infty \geqslant K \cdot q) \approx 2 \left(1 - \Phi\left(\frac{K - 0}{\sigma}\right)\right),$$

where $\sigma = (h + 1)/12$. I note that

$$\Pr(|t|_\infty \geqslant K \cdot q) \approx 2^{-24.06},$$

when $h = 64$ and $K = 12$. As the whole $2l$ coefficients must be less than K, the CKKS bootstrapping fails with the probability $2l \cdot 2\left(1 - \Phi\left(\frac{K-0}{\sigma}\right)\right)$, where l is number of slots. As a modulus reduction is made by the approximate polynomial, the input outside the domain approximate polynomial results in a huge value. However, enlarging the approximate domain will require a huge degree of the approximate polynomial.

The proposed protocol follows the scenario of privacy-preserving machine learning using FHE: the sender receives the encrypted input message from the receiver and perform all the operations homomorphically, and then returns the result to the receiver. Thus the activation function such as ReLU can be the polynomial approximation of a conventional activation function or an activation designed for HE as in [15, 16]. The difference is that this protocol replaces bootstrapping with network communication.

When the bootstrapping is required during the homomorphic computation, the sender encrypts the intermediate value via OTP, and transfer to the sender. Then, the sender decrypts the received ciphertext, re-encrypt it, and sends it to the sender. However, unlike the approaches using the garbled circuit, the receiver only performs the encryption and decryption. Thus, the ability of computation of the sender remains the same as the FHE scenario.

Table 3.1 illustrates how the proposed scheme replaces the bootstrapping with network communication. Let $\mathsf{Enc}_l(m)$ denote the ciphertext of the message m, where the level of the ciphertext is l. As I focus on the CKKS scheme, the level means the number of possible multiplications, likewise some other HE schemes. As shown in Table 3.1, when the ciphertext reaches level 0, the sender samples $r \leftarrow \mathcal{R}_q$ and add to $\mathsf{Enc}_0(m)$ which results in $\mathsf{Enc}_0(m + r)$ by OTP. Then $\mathsf{Enc}_0(m + r)$ is sent to the receiver. The receiver then gets $(m + r)$ by the decryption of $\mathsf{Enc}_0(m + r)$. $(m + r)$ does not leak information of m or r as it satisfies the perfect secrecy when m and r is in finite additive group of the integers. Then the receiver generates a fresh ciphertext that encrypts $m + r$ with the maximum level L, $\mathsf{Enc}_L(m + r)$. When $\mathsf{Enc}_L(m + r)$ is transferred to the sender, the sender can find $\mathsf{Enc}_L(m)$ using r.

I note here that the random key is generated as $r \leftarrow \mathcal{R}_q$, not $\mathbb{C}^{N/2}$. In the case of other word-wise encryption schemes, such as BGV and (B)FV schemes, the plaintext slots are finite values in $\mathbb{Z}_t$. Hence, the proposed scheme is straightforward and is performed without failure. In that case, sender chooses a random N-tuple $\boldsymbol{r} \rightarrow \mathbb{Z}_t^N$, then encode it, and add to the intermediate $\mathsf{Enc}_0(m + r)$. As the slot values are values in a finite field, the information-theoretic secrecy is naturally satisfied. However, in the CKKS scheme, the slot values are complex numbers. Hence, the information-theoretic secrecy should be considered in the coefficients, not slot values. Moreover, we should consider if m and $\mathsf{Dec}(\mathsf{Enc}_0(m) + r) - r$ are the same, as the decryption includes modulus reduction of coefficients by the ciphertext modulus q. There are two solutions for that: allowing failure or use restricted r.

Allowing Failure of Ciphertext Refreshing

The failure probability of refreshing the ciphertext in the proposed protocol is less than m/q, which can easily be lessened to $2^{-\lambda}$, where 2^{λ} is the desired security level. For example, when the best-known attack algorithm requires at least $\lambda = 128$ basic operations on average, it is called that the cryptosystem satisfies 128-bit security. In that case, we can ignore a probability of $2^{-\lambda}$ as it is very low as the probability that an attack on the cryptosystem succeeds.

Hence, following the security parameter, we can say that the protocol does not fail. Instead, the criterion can be relaxed to allow for some probability of failure, say p_0 and let $q \geqslant m \cdot p_0$.

Usage of Restricted r

The perfect secrecy is satisfied when

$$I(m;c) = 0.$$

The receiver has $c' = r + m$. In the case of $r \in [0, q)$, it offers perfect secrecy as it is a well-known OTP. However, there exists decryption failure.

The receiver and sender cannot figure out the failure until the termination of the protocol. However, it is the same in the ordinary bootstrapping of the CKKS scheme. In case of $r \in (0, q-p]$, it can be shown that $I(m;c') \leqslant \frac{1}{q-p} \cdot p \sim \frac{p}{q}$. As $q >> p$, it is a small value. Since knowing $\overline{m} = m + e$ is considered as decryption of m, where e is a small error, we could say that an information less than $H(m|\overline{m})$ is actually no information in CKKS setup (approximate arithmetic).

3.3 Comparison With Existing Methods

3.3.1 Comparison With the Hybrid Method

In the hybrid methods [55, 59], the linear layer is calculated homomorphically, and the non-linear layers, such as pooling and activation layers, are performed using MPC techniques. The garbled circuit is the most widely-used [60] 2PC technique, which can perform arbitrary binary circuit securely. As comparison is represented by a simple binary circuit, the garbled circuit is effective to evaluate activate functions and pooling layers such as ReLU and max pooling. However, there exist two drawbacks using the hybrid method: first, it reveals part of the network structure, and second, it requires much communication and computation resources of the receiver.

Not only the trained weights but also the structure of the network itself, such as how many layers it has, what activations and poolings it uses, etc., are the assets of the sender. Hence, the sender wants to keep such values secure, but such information is revealed by using the hybrid method. The receiver can guess the structure of the sender's model. This might be crucial in security because, once the structure of the model is known, it becomes much easier to attack or reproduce the model. Moreover, if the model is composed of simple comparisons, such as random forest, most of the operations are done by garbled circuits, and it reveals the structure.

A garbled circuit requires network communication between sender and receiver that is proportional to the depth of the circuit, and the basic building block of a garbled circuit is a symmetric-key cryptosystem such as AES. It requires a number of ciphertexts to perform a binary gate. Hence, it requires much communication resources to the receiver. Moreover, the receiver needs to do the encryption and decryption of AES, which requires additional hardware or computational ability. It is clear that homomorphic encryption and decryption of the hybrid method should be performed as frequently as the proposed method.

The advantage of the hybrid method over the proposed method is that the operation is accurate. The modern activation and pooling are composed of comparison operations, and it is not a polynomial operation. Hence, an approximate polynomial is used to perform the activation and pooling, and thus such layers cannot be performed exactly. This might affect the accuracy of the machine learning algorithm.

In summary, the proposed method is more suitable to secure the sender's information as well as it requires less communication and computation to the receiver. However, it requires more computation to the sender, and it might drop the precision of output, and thus a larger scaling factor should be used.

3.3.2 Comparison With FHE Method

In order to use the FHE method, bootstrapping should be performed. The bootstrapping of the CKKS scheme consumes certain ciphertext level, for example 10 [61] to 15 [19–21,25]. Hence, the parameter size should large enough. Table 3.2 shows the parameter size for the CKKS scheme with 128-bit security, where the non-sparse key is used. Recently, an attack algorithm for the CKKS scheme that uses the sparsity of the secret key is proposed [62], and thus, the security parameters for the sparse-key CKKS scheme should be adjusted. However, as the attack itself is not a critical threat of the CKKS scheme, and it is out of the scope of this paper, we use the parameters in Table 3.2 for the reference of the comparison of the proposed method and FHE method. It is noted that the max coefficient might be smaller than that of Table 3.2 to guarantee 128-bit security. A larger coefficient modulus $P \cdot q_L$ implies more rescaling budget, and thus more encrypted computation capabilities. However, an upper bound for the total bit-length of the coefficient modulus is determined by the polynomial degree N.

Assuming that one uses the $p = 2^{40}$ for enough precision, the bootstrapping roughly consumes the coefficient modulus from 500 to 750. Hence, it is required to use $N \geqslant 2^{16}$ for the existing methods [19–21, 25] and $N \geqslant 2^{15}$ for the proposed bootstrapping in this dissertation [61].

A larger parameter requirement of the bootstrapping is not desirable to the receiver.

One might argue that the use of a large parameter does not have a significant difference in single instruction multiple data (SIMD) manner as we have more slots than that of the smaller parameter. However, there are several reasons why the use of the larger parameter is not preferable. First of all, following the parameter size, the key sizes become larger, and it requires more number of key-switching keys. To perform bootstrapping, at least $log(N)$ rotation keys are required, and the size of each rotation key is $O(N^2)$. Hence, it requires $O(N^2 \cdot logN)$ memories and computations to the receiver for key generation. Second, on the sender's side, the operations become complex when the slot size is larger. When the operations between slot elements are required, the operations should be done by rotation; in other words, an arbitrary permutation is quite inefficient as it is composed of several rotations and plaintext multiplications. Hence, it is desirable to use fewer slots. Finally, the error in the CKKS scheme is affected by the slot size. Especially, the variance of bootstrapping error is proportional to the square of slot size, and other errors are proportional to the slot size. This means that to use a larger parameter, one should consider using a larger scaling factor or the use of sparse slots. The former results in less level, and the latter results in a throughput drop.

It is worth noting that the receiver is assumed to have very limited computation resources, such as IoT devices. It would be unreasonable to let such a small device perform key generation, encryption, and decryption of the CKKS scheme with $N = 2^{16}$.

3.4 Comparison for Evaluating Neural Network

Use of sparse slot is usual since the bootstrapping for full-slot ciphertext takes a long time, and the bootstrapping error is proportional to the square of slot size [19–21]. However, the use of a spare slot results in a throughput drop. In other words, there exists a trade-off between throughput and accuracy of results in the FHE approach.

In contrast, the proposed method does not occur any additional error to refresh the ciphertext, and thus it is always beneficial to utilize the full slots. Therefore, the proposed algorithm has a gain in throughput. It can be seen that the proposed method allows more data to be transmitted in one ciphertext, and thus the number of operations and communication per data is reduced compared to the FHE method.

The comparison of the proposed method and the FHE method in evaluating VGG-16 [1] is shown in Table 3.3. The activation function is ReLU in VGG, and max pooling is used. The depth of operation is computed in the table by assuming that the approximated ReLU and the max-pooling have depth 4 and 12, respectively [63]. One level is consumed to compute the convolution layer or fully-connected layer.

In the method of rescaling after encryption proposed in [23], the bottleneck of error is the rescaling error, not the encryption error. In case bootstrapping is done, the bottleneck of error is bootstrapping error, and it has been reduced by using the method proposed in [61], and it consumes 10 levels. For bootstrapping, enough levels are required, and thus, the $N = 2^{16}$ is used for the FHE method. The variance of rescaling error when $N = 2^{14}$ is $2^{16.43}$, and the variance of bootstrapping error when $N = 2^{16}$ and $l = 2^{14}$ is $2^{42.91}$. It is assumed that the Hamming weight of the secret key is 64. Therefore, for the same precision of encrypted message, the scaling factor 27 and 40 is used for the proposed method and FHE method, respectively, in order to maintain the precision of 16 bits below the decimal point.

For bootstrapping, N rotation keys are required, but by using only rotation keys for power-of-two indices, any rotation can be done within $\log(N)$ homomorphic rotations. Hence, it is assumed that the receiver generates only $\log(N)$ rotation keys in the table. Table 3.3 is for the RNS-CKKS scheme, and it is assumed that the key-switching in the SEAL library is used. Each integer element is assumed to be 64-bits.

Not to mention the reduction in computation on the sender side, it can be seen that the proposed method requires about ten times less communication because it uses smaller parameters and utilizes its entire slot. Rotation keys make up most of the calculations and communications. By omitting the bootstrapping, we can reduce the size of rotation and evaluation keys, and overall communication is reduced despite some communication being induced to refresh the ciphertext. The downside of the proposed method is that the sender and receiver have to communicate with each other until the operation is complete. However, since the reduction in computation and communication is very large, it can be endured enough. Most importantly, the proposed method can reduce the amount of communication by about 1/8 of the FHE method.

Table 3.1: Procedure of the proposed method for PPML without bootstrapping

Receiver (client)				Sender (server)		
Operation	**Secret**	**Public**		**Public**	**Secret**	**Operation**
encrypt the input message	m	$\boldsymbol{c} = \mathsf{Enc}_L(m)$	$\rightarrow$	$\boldsymbol{c}$		
						perform ML operations
$\dots$						
					$\boldsymbol{c}_{\text{inter}} = \mathsf{Enc}_0(m_{\text{inter}})$	at level 0
		$\boldsymbol{c}_{\text{otp}}$	$\leftarrow$	$\boldsymbol{c}_{\text{otp}} = \boldsymbol{c}_{\text{inter}} + r$	r	randomly generate and add r
decrypt $\boldsymbol{c}_{\text{otp}}$	$m_{\text{inter}} + r$					
encrypt and send back	$m_{\text{inter}} + r$	$\boldsymbol{c}'_{\text{otp}} = \mathsf{Enc}_L(m_{\text{inter}} + r)$	$\rightarrow$	$\boldsymbol{c}'_{\text{otp}}$		
					$\boldsymbol{c}_{\text{fresh}} = \boldsymbol{c}'_{\text{otp}} - r$	subtract r
					$\boldsymbol{c}_{\text{fresh}} = \mathsf{Enc}_L(m_{\text{inter}})$	continue ML operations
$\dots$						
		$\boldsymbol{c}_{\text{result}} = \mathsf{Enc}_l(m_{\text{result}})$	$\leftarrow$	$\boldsymbol{c}_{\text{result}} = \mathsf{Enc}_l(m_{\text{result}})$		send the final result
decrypt and get result	$m_{\text{result}} = \mathsf{Dec}(\boldsymbol{c}_{\text{result}})$					

Table 3.2: Parameter size for the CKKS scheme with 128-bit security, where non-sparse key is used

Polynomial degree (N)	Max coefficient modulus (Pq_L)
2^{13}	2^{218}
2^{14}	2^{438}
2^{15}	2^{881}
2^{16}	2^{1770}
2^{17}	2^{3540}

Table 3.3: Parameters and number of communications to evaluate VGG-16 [1] in the proposed method and the FHE method

	Proposed	FHE method
polynomial degree (N)	2^{14}	2^{16}
slot size (l)	2^{14}	2^{14}
depth	129	129
variance of error	$2^{16.43}$	$2^{42.91}$
scaling factor (Δ)	27	40
levels (after bootstrapping)	14	41 (29)
size of a ciphertext	2.63 MB	42 MB
size of evaluation keys	108.75 MB	1.76 GB
number of refreshing/bootstrapping	9	3
size of $\log(N)$ rotation keys	1.48 GB	28.16 GB
total amount of communication	3.86 GB	29.92 GB

Chapter 4

High-Precision Approximate Homomorphic Encryption and Its Bootstrapping by Error Variance Minimization and Convex Optimization

4.1 Introduction

In this chapter, I propose methods of improving the approximate HE using variance-minimizing and convex optimizations. The CKKS scheme is one of the highlighted FHE schemes as it is efficient to deal with encrypted complex(real) numbers, which are the usual data type for many applications such as machine learning [4]. In this dissertation, I propose a generally applicable method to achieve high-precision approximate FHE using the following two techniques. First, I apply the concept of SNR and propose a method of maximizing SNR of encrypted data by reordering homomorphic operations in the CKKS scheme. For that, the variance of error of encrypted data is minimized instead of the upper bound of error when we deal with the ciphertext. Second, from the same perspective of minimizing error variance, I propose a new method of finding the approximate polynomials for the CKKS scheme. The approximation method is especially applied to the bootstrapping of the CKKS scheme, where I achieve a smaller error variance in the bootstrapping compared to the prior arts. The performance improvement of the proposed methods for the CKKS scheme is verified by implementation over HE libraries, HEAAN, and SEAL. The implementation results show that by reordering homomorphic operations and using the proposed polynomial approximation, the message precision of the CKKS scheme is improved. Specifically, the proposed method uses only depth 8, although the bootstrapping error for the CKKS method is less than the error obtained using depth 11 of the previous method. I also suggest a loose lower bound for bootstrapping error in the CKKS scheme and show that the error by the proposed method is only 2.8 bits on average larger than the lower bound. Therefore, the quality of services of various applications using the proposed CKKS scheme, such as privacy-preserving machine learning, can be improved without compromising performance and security.

4.2 Optimization of Error Variance in the Encrypted Data

This section provides a new criterion of qualifying encrypted data of the CKKS scheme, that is, SNR. It is worth noting that the proposed method is popularly used in the communication theory. Measuring the quality of the signal by SNR is the main idea, which is natural and widely used in communication systems.

We can assume the following statements:

1. The mean of error is zero.

2. The message and error are statistically independent.

The first assumption is straightforward and clear. If the mean of error is not zero, one can simply subtract the mean value to reduce the error. In general, the second one is also true. When we deal with approximate polynomial, the approximation error is dependent on the message. However, the approximation error is usually small compared to the message, and the covariance is negligible. From these two assumptions, the variance of error introduced by multiplication can be obtained. Moreover, from the central limit theorem, the sum of independent random variables can be approximated to a Gaussian distribution. Now, since the power of noise and the variance of error are the same, I focus on error variance.

4.2.1 Tagged Information for Ciphertext

I propose new tagged information for the full ciphertext of the CKKS scheme to tightly manage the errors in encrypted data. The tagged information for ciphertext is introduced in [4], and it is used to estimate the magnitude of the error. The tagged information is composed of a level l, where $0 \leqslant l \leqslant L$, an upper bound $v \in \mathbb{R}$ of the message, and a high-probability upper bound $B \in \mathbb{R}$ of error. The upper bound is informative when there are few homomorphic operations. However, as the homomorphic operation continues, the upper bound becomes exponentially loose and thus useless.

I take a simple example of how the upper bound becomes loose. In [4], 6σ is used as the high-probability upper bound of the error that follows Gaussian distribution with variance σ^2. The probability that $Pr(|X| > 6\sigma)$ is $2^{-27.8}$. Previously, the error upper bound of a ciphertext, which was the sum of two ciphertexts with error upper bounds B_1 and B_2, was specified as $B_1 + B_2$ [4, 23]. Assume that there are 100 distinct fresh ciphertexts and let σ_o^2 be the error variance. Then the previous tagged information states that the upper bound of error is $6\sigma_o$. Hence, the upper bound of the error in the encrypted data, which is the sum of the hundred ciphertexts, is $600\sigma_o$. However, the ciphertexts are independent, and its error follows the Gaussian distribution with a variance of $100\sigma_o^2$. Therefore, the probability of the given upper bound, $600\sigma_o$, is $Pr(|e| > 600\sigma_o) \approx 2^{-2602.1}$, which is quite loose, where e is the summation of error. The previous upper bound is too loose to obtain useful information on the error. A real application such as deep learning requires a lot more computation than this, and thus the high-probability upper bound of error becomes looser than this example. In other words, managing the upper limit of errors is practically futile.

Instead of using these upper bounds, I propose to use the variance of messages and errors for tagged information. To manage the variance of error, the energy of message $E[x^2]$ is required. However, to tightly manage the error variance after multiplication, it is better to have the mean and variance of the message. In other words, three tuples are the tagged information, which are tuple of message mean, $\{E[\pi(\mathsf{Emb}(m))_i]\}_{i=0,\cdots N/2-1}$, tuple of message variance, $\{Var[\pi(\mathsf{Emb}(m))_i]\}_{i=0,\cdots N/2-1}$, and tuple, $\{Var[\pi(\mathsf{Emb}(e))_i]\}_{i=0,\cdots N/2-1}$, denoted by $\boldsymbol{\mu}_\mathsf{m} \in \mathbb{C}^{N/2}$ and $\boldsymbol{v}_\mathsf{m}, \boldsymbol{v}_\mathsf{e} \in \mathbb{R}^{N/2}$, respectively, where $\mathsf{Dec}_\mathsf{sk}(\boldsymbol{c}) = m + e$. If each slot value follows the identical distribution, the tagged information can be replaced as scalar values $\mu_\mathsf{m} \in \mathbb{C}$ and $v_\mathsf{m}, v_\mathsf{e} \in \mathbb{R}$. Hence, the full ciphertext is given as

$$(\boldsymbol{c}, l, \Delta, \boldsymbol{\mu}_\mathsf{m}, \boldsymbol{v}_\mathsf{m}, \boldsymbol{v}_\mathsf{e}),$$

where l and Δ are the level and scaling factor of ciphertext, respectively.

The distribution of messages and errors after homomorphic operations varies depending on the actual distribution of messages and the dependency between messages. However, it is difficult to know the exact correlation of the messages, and their distribution after several homomorphic operations is quite complicated. Therefore, while managing the mean and variance values, the message and error can be roughly treated as independent Gaussian distribution. It is shown through the implementation in Section 4.7 that errors in the encrypted data can strictly be managed in this way too.

4.2.2 Worst–Case Assumption

One might argue that an upper bound, and minimax approximation should be used as it is not appropriate to assume that someone other than the data owner knows the mean and variance of the message, $\boldsymbol{\mu}_\mathsf{m}$ and $\boldsymbol{v}_\mathsf{m}$. However, it is entirely reasonable to assume that someone other than the data owner knows minimal information about the distribution of the message, the mean and variance, for the following reasons. First, the previously used measure of error, the high-probability upper bound, is also related to the distribution. Second, in many applications such as deep learning, control of the distributions of intermediate node values is crucial. For example, the input is usually normalized or standardized, and many methods to normalize the intermediate values are widely used [64]- [65], which is crucial for the accuracy and speed up the training of neural networks. Finally, some information about the message distribution is known regardless of security, such as the message distribution after ModRaise in bootstrapping.

If one does not even want to provide the mean and variance of the message, the server can assume the worst-case that the coefficients of message $m(X)$ are distributed uniformly at random in $[-B, B]$. As the message is in $\mathcal{R}$, which is discrete, the uniform distribution maximizes the entropy, and then it is obviously the worst case. Similarly, it can also be assumed that the slot values $\boldsymbol{z} \in \mathbb{Z}^{N/2}$ follow centered Gaussian distribution with the variance that the coefficients of its encoded value are in $[-B, B]$ with high-probability, which maximizes the differential entropy. In the experimental results in Section 4.7, even though the worst-case scenario is used, it is shown that the error value in the proposed method is smaller than that of the previous methods.

4.2.3 Error in Homomorphic Operations of the CKKS Scheme

The error analysis in each operation, such as encoding, encryption, addition, multiplication, and key switching, is shown in this subsection. It should be noted that the proposed error variance minimization method can be applied for all the variants of the CKKS scheme, such as the original CKKS scheme [4], RNS variants [20, 56], and the reduced-error variants [23]. The only difference for the above variants is the variance of errors.

The following lemmas are basically based on the lemmas in [4, 19, 23]. The difference is that the lemmas so far have focused on high-probability upper bounds of errors, but in the proposed method, I focus on the variance of errors in encrypted data because the upper bounds become loose after successive homomorphic operations. Error variances in encrypted data for the original CKKS scheme and the RNS-CKKS scheme are given in the following lemmas. In the RNS-CKKS scheme, the rescaling factors q_i's are different for each level, and thus the errors in rescaling are different from that of the original CKKS scheme.

Lemma 4.1 (Encoding and encryption)**.** *Given a secret key* sk *with Hamming weight* h*, we have the following variance of encryption noise in the CKKS scheme:*

$$\frac{1}{2}\sigma^2 N^2 + \sigma^2(h+1)N.$$

Proof. Since $a(\zeta_M^j)$ is the inner product of coefficient vector of a polynomial $a(X)$ and the fixed vector $(1, \zeta_M^j, \ldots, \zeta_M^{j(N-1)})$ with $|\zeta_M^j| = 1$, the random variable $a(\zeta_M^j)$ has variance $\sigma^2 N$, where σ^2 is the variance of each coefficient of a. Therefore, $a(\zeta_M^j)$ has the variances $q^2N/12, \sigma^2 N, \rho N$, and h, when $a(X)$ is sampled from $U(\mathcal{R}_q), \mathcal{DG}(\sigma^2), \mathcal{ZO}(\rho)$, and $\mathcal{HWT}(h)$, respectively.

v and e_0, e_1 are chosen from $\mathcal{ZO}(0.5)$ and $\mathcal{DG}(\sigma^2)$, respectively. We have a ciphertext $\boldsymbol{c} \leftarrow v \cdot \mathsf{pk} + (m + e_0, e_1)$ with error given by

$$\langle \boldsymbol{c}, \mathsf{sk} \rangle - m \pmod{q_L} = v \cdot e + e_0 + e_1 \cdot s.$$

As v, e, e_0, e_1, and s are independent, its variance is given as

$$N/2 \cdot \sigma^2 N + \sigma^2 N + \sigma^2 N \cdot h = \frac{1}{2}\sigma^2 N^2 + \sigma^2(h+1)N.$$

$$\square$$

□

Lemma 4.2 (Rescaling)**.** *Let* $(\boldsymbol{c}, l, \Delta, \boldsymbol{\mu}_{\mathsf{m}}, \boldsymbol{v}_{\mathsf{m}}, \boldsymbol{v}_{\mathsf{e}})$ *be an encryption of the encoded message* $m(X) \in \mathcal{R}$ *of* $\boldsymbol{z} \in \mathbb{C}^{N/2}$*, where* l *is its level. Then*

$$\left(\boldsymbol{c}_{\mathsf{RS}}, l', p^{(l'-l)} \cdot \Delta, p^{(l'-l)} \cdot \boldsymbol{\mu}_{\mathsf{m}}, p^{2(l'-l)} \cdot \boldsymbol{v}_{\mathsf{m}}, p^{2(l'-l)} \cdot \boldsymbol{v}_{\mathsf{e}} + \boldsymbol{v}_{\mathsf{scale}}\right)$$

is a valid encryption of the rescaled message $p^{(l'-l)} \cdot m(X)$ *for* $\boldsymbol{c}_{\mathsf{RS}} \leftarrow \mathsf{RS}_{l \to l'}(\boldsymbol{c})$ *and* $\boldsymbol{v}_{\mathsf{scale}} = \frac{1}{12}(h+1)N$.

Proof. The rescaled ciphertext $\boldsymbol{c}_{\mathsf{RS}} \leftarrow \left\lfloor \frac{q_{l'}}{q_l} \boldsymbol{c} \right\rceil$ satisfies $\langle \boldsymbol{c}_{\mathsf{RS}}, \mathsf{sk} \rangle = \frac{q_{l'}}{q_l}(m+e) + e_{\mathsf{scale}}$, where $e_{\mathsf{scale}} = \langle \boldsymbol{\tau}, \mathsf{sk} \rangle$ and $\boldsymbol{\tau} = (\tau_0, \tau_1)$ is the rounding error vector. We can assume that the coefficients of polynomials τ_0 and τ_1 are distributed uniformly at random on $\frac{q_{l'}}{q_l}\mathbb{Z}_{q_l/q_{l'}}$, and thus the variance of $\tau_0 + \tau_1 \cdot s$ is $N/12 + hN/12$. Therefore, the variance of e_{scale} is given as $\frac{1}{12}(h+1)N$. □ □

Lemma 4.3 (Addition and multiplication)**.** *Let* $(\boldsymbol{c}_i, l, \Delta_i, \boldsymbol{\mu}_{m,i}, \boldsymbol{v}_{m,i}, \boldsymbol{v}_{e,i})$ *be two independent encryptions of the encoded messages* $m_i(X)$ *of values* $\boldsymbol{z}_i \in \mathbb{C}^{N/2}$ *for* $i = 1, 2$*, and let*

$$\boldsymbol{c}_{add} \leftarrow Add(\boldsymbol{c}_1, \boldsymbol{c}_2) \text{ and } \boldsymbol{c}_{mult} \leftarrow Mult(\boldsymbol{c}_1, \boldsymbol{c}_2).$$

Then,

$$\left(\boldsymbol{c}_{add}, l, \Delta_1, \boldsymbol{\mu}_{m,1} + \boldsymbol{\mu}_{m,2}, \boldsymbol{v}_{m,1} + \boldsymbol{v}_{m,2}, \boldsymbol{v}_{e,1} + \boldsymbol{v}_{e,1}\right)$$

and

$$\begin{aligned}(\boldsymbol{c}_{mult}, l, \Delta_1\Delta_2, \boldsymbol{\mu}_{m,1} \times \boldsymbol{\mu}_{m,2}, \boldsymbol{v}_{m,1} \times \boldsymbol{v}_{m,2} \\ , (\boldsymbol{v}_{m,1} + |\boldsymbol{\mu}_{m,1}^2|) \times \boldsymbol{v}_{e,2} + (\boldsymbol{v}_{m,2} + |\boldsymbol{\mu}_{m,2}^2|) \times \boldsymbol{v}_{e,1} + \boldsymbol{v}_{e,1} \times \boldsymbol{v}_{e,2} + \boldsymbol{v}_{mult})\end{aligned}$$

are valid encryptions of $m_1(X) + m_2(X)$ *and* $m_1(X) \cdot m_2(X)$*, respectively, where* $\boldsymbol{v}_{mult} = \left(\frac{q_l}{P}\right)^2 \boldsymbol{v}_{ks} + \boldsymbol{v}_{scale}$, $\boldsymbol{v}_{ks} = \frac{1}{12}N^2\sigma^2$*, and* $|\boldsymbol{\mu}^2|$ *refers* $\boldsymbol{\mu} \times \overline{\boldsymbol{\mu}}$*. For addition,* $\Delta_1 = \Delta_2$ *should be satisfied.*

Proof. Addition is trivial. The ciphertext of $m_1(X) \cdot m_2(X)$

$$\boldsymbol{c}_{\mathsf{mult}} \leftarrow (d_0, d_1) + \left\lfloor P^{-1} \cdot d_2 \cdot \mathsf{evk} \right\rceil \pmod{q_l}$$

contains additional error $e'' = P^{-1} \cdot d_2 e'$ and the error by scaling. As $d_2 = a_1 a_2$ and from RLWE assumption, we can assume that d_2 is distributed uniformly at random on $\mathcal{R}_{q_l}$. Thus, the variance of Pe'' is derived as $q_l^2 N/12 \cdot \sigma^2 N = \frac{1}{12}q_l^2\sigma^2 N^2$. The total error is given as

$$m_1 e_2 + m_2 e_1 + e_1 e_2 + e'' + e_{\mathsf{scale}}$$

and as the means of e_1 and e_2 are zero and m_1 and m_2 are independent, the variance of $m_1e_2 + m_2e_1 + e_1e_2$ is given as

$$(\boldsymbol{v}_{\mathsf{m},1} + |\boldsymbol{\mu}^2_{\mathsf{m},1}|) \times \boldsymbol{v}_{\mathsf{e},2} + (\boldsymbol{v}_{\mathsf{m},2} + |\boldsymbol{\mu}^2_{\mathsf{m},2}|) \times \boldsymbol{v}_{\mathsf{e},1} + \boldsymbol{v}_{\mathsf{e},1} \times \boldsymbol{v}_{\mathsf{e},2}.$$

□ □

Lemma 4.4 (Key-switching). *Let $(\boldsymbol{c}, l, \Delta, \boldsymbol{\mu}_{\mathsf{m}}, \boldsymbol{v}_{\mathsf{m}}, \boldsymbol{v}_{\mathsf{e}})$ be a ciphertext with respect to a secret key $\mathsf{sk}' = (1, s')$ and let $\mathsf{swk} \leftarrow \mathsf{KSGen}_{\mathsf{sk}}(s')$, where $\mathsf{sk} = (1, s)$. Then*

$$\left(\boldsymbol{c}', l, \Delta, \boldsymbol{\mu}_{\mathsf{m}}, \boldsymbol{v}_{\mathsf{m}}, \boldsymbol{v}_{\mathsf{e}} + \left(\frac{q_l}{P}\right)^2 \boldsymbol{v}_{\mathsf{ks}} + \boldsymbol{v}_{\mathsf{scale}}\right)$$

is a valid ciphertext with respect to a secret key sk for the same message, where $\boldsymbol{c}' \leftarrow \mathsf{KS}_{\mathsf{swk}}(\boldsymbol{c})$.

Proof. The proof is similar to that of Lemma 4.3 and thus, it is omitted. □ □

Lemma 4.5 (Addition and multiplication by constant). *Let $(\boldsymbol{c}, l, \Delta, \boldsymbol{\mu}_{\mathsf{m}}, \boldsymbol{v}_{\mathsf{m}}, \boldsymbol{v}_{\mathsf{e}})$ be an encryption of the encoded message $m(X)$ of $\boldsymbol{z} \in \mathbb{C}^{N/2}$. For a constant tuple $\boldsymbol{a} \in \mathbb{C}^{N/2}$, let*

$$\boldsymbol{c}_{\mathsf{cadd}} \leftarrow \mathsf{cAdd}(\boldsymbol{c}_1, \boldsymbol{a}; \Delta) \text{ and } \boldsymbol{c}_{\mathsf{cmult}} \leftarrow \mathsf{cMult}(\boldsymbol{c}_1, \boldsymbol{a}; \Delta'),$$

where $\boldsymbol{c}_{\mathsf{cadd}}$ and $\boldsymbol{c}_{\mathsf{cmult}}$ correspond to the constant multiplication and addition with constant $\boldsymbol{a}$, scaled by Δ', respectively. Then,

$$(\boldsymbol{c}_{\mathsf{cadd}}, l, \Delta, \boldsymbol{\mu}_{\mathsf{m}} + \Delta\boldsymbol{a}, \boldsymbol{v}_{\mathsf{m}}, \boldsymbol{v}_{\mathsf{e}})$$

and

$$\left(\boldsymbol{c}_{\mathsf{cmult}}, l, \Delta\Delta'\boldsymbol{a} \times \boldsymbol{\mu}_{\mathsf{m}}, \Delta'^2\boldsymbol{a}^2 \times \boldsymbol{v}_{\mathsf{m}}, \Delta'^2\boldsymbol{a}^2 \times \boldsymbol{v}_{\mathsf{e}}\right)$$

are valid encryptions of $\Delta\boldsymbol{a} + \boldsymbol{z}$ and $\Delta'\boldsymbol{a} \times \boldsymbol{z}$, respectively, where $\boldsymbol{a}^2 = \boldsymbol{a} \times \boldsymbol{a}$.

Proof. The rounding during encoding, introduces a rounding error. However, we could assume that the scaling factor is large enough so that there are no errors. Then, it is self-evident. □ □

The following two lemmas are slightly modified from the original lemmas in [20]. It is noted that in the RNS rescaling, the error introduced by approximate scaling factor is eliminated by managing the exact scaling factor after rescaling, Δ/p_l.

Lemma 4.6 (RNS rescaling)**.** *For the RNS-CKKS scheme, let* $(\boldsymbol{c}, l, \Delta, \boldsymbol{\mu}_{\mathsf{m}}, \boldsymbol{v}_{\mathsf{m}}, \boldsymbol{v}_{\mathsf{e}})$ *be an encryption of the encoded message* $m(X) \in \mathcal{R}$ *of* $\boldsymbol{z} \in \mathbb{C}^{N/2}$*. Then*

$$\left(\boldsymbol{c}_{\mathsf{RS}}, l-1, p_l^{-1} \cdot \Delta, p_l^{-1} \cdot \boldsymbol{\mu}_{\mathsf{m}}, p_l^{-2} \cdot \boldsymbol{v}_{\mathsf{m}}, p_l^{-2} \cdot \boldsymbol{v}_{\mathsf{e}} + \boldsymbol{v}_{\mathsf{scale}}\right)$$

is a valid encryption of the rescaled message $p_l^{-1} \cdot m(X)$ *for* $\boldsymbol{c}_{\mathsf{RS}} \leftarrow \mathsf{RS}(\boldsymbol{c})$ *and* $\boldsymbol{v}_{\mathsf{scale}} = \frac{1}{12}(h+1)N$*. Thus, the scaling factor of* $\boldsymbol{c}_{\mathsf{RS}}$ *is* $p_l^{-1} \cdot \Delta$*.*

Proof. The proof is the same as that of the original CKKS scheme except for the scaling factor. In RNS-CKKS, the scaling factor is slightly different depending on the operations done to the ciphertext, and thus when adding different ciphertexts, an error occurs according to the ratio of p_l and p in the process of forcibly treating the scaling factor as p. The methods to remove such error was proposed in [23, 62]. $\square$ $\square$

Lemma 4.7 (RNS key-switching)**.** *For the RNS-CKKS scheme, let*

$$(\boldsymbol{c}, l, \Delta, \boldsymbol{\mu}_{\mathsf{m}}, \boldsymbol{v}_{\mathsf{m}}, \boldsymbol{v}_{\mathsf{e}})$$

be a ciphertext with respect to a secret key $\mathsf{sk}' = (1, s')$ *and let* $\mathsf{swk} \leftarrow \mathsf{KSGen}_{\mathsf{sk}}(s')$*, where* $\mathsf{sk} = (1, s)$*. Then* $\left(\boldsymbol{c}', l, \Delta, \boldsymbol{\mu}_{\mathsf{m}}, \boldsymbol{v}_{\mathsf{m}}, \boldsymbol{v}_{\mathsf{e}} + \frac{1}{P^2}\boldsymbol{v}_{\mathsf{ksrns}} + \boldsymbol{v}_{\mathsf{scale}}\right)$ *is a valid ciphertext with respect to a secret key* sk *for the same message, where* $\boldsymbol{c}' \leftarrow \mathsf{KS}_{\mathsf{swk}}(\boldsymbol{c})$ *and* $\boldsymbol{v}_{\mathsf{ksrns}} = \frac{\mathsf{dnum} \cdot p^{\alpha}}{12}\sigma N$*.*

Proof. The key switching noise comes from the rounding terms $\boldsymbol{\tau}$ as in Lemma 4.2 and from the error terms e'_k in swk_0. The variance of error from $\boldsymbol{\tau}$ is v_{scale}. The other error is given as

$$\frac{\langle \mathcal{WD}_l(c_1), \{e'_k\}_{0 \leqslant k < \mathsf{dnum}-1} \rangle}{P}. \tag{4.1}$$

It can be assumed that the i-th component of $\mathcal{WD}_l(c_1)$ follows uniform distribution in $\tilde{q}_i$. Then, its variance is $\frac{\tilde{q}_i}{12}$ and the variance of each coefficient of e'_k is σ^2. Thus, the variance of error in (4.1) is derived as

$$P^{-2} \cdot \sum_{0 \leqslant i < \mathsf{dnum}' - 1} \frac{\tilde{q}_i}{12} \sigma N \approx P^{-2} \cdot \frac{\mathsf{dnum}' \cdot p^{\alpha}}{12} \sigma N.$$

□ □

When the sparse packing method [19] is applied, N in the above lemmas can be replaced by $2n$ when there are n slots.

4.2.4 Reordering Homomorphic Operations

As the proposed method enables tight management of errors in encrypted data, homomorphic operations can be effectively reordered to reduce errors. The main advantage of reordering homomorphic operations is that the errors in the encrypted data are reduced without compromising security and performance. Using Lemmas 4.1 to 4.7, one can reorder homomorphic operations to minimize the error variance. In this subsection, I show some operations patterns that can be reordered to reduce the error in encrypted data. Considering that the error increases cumulatively in the CKKS scheme, the small differences in the following examples greatly affect the error variance as the depth of operations advances. It is worth noting that the most beneficial application of the CKKS scheme, deep learning has a deep of operations, too.

In addition to the below examples, there are many methods to reorder homomorphic operations corresponding to the inputs and the operations themselves. Thus the reordering of homomorphic operations can be done in an on-the-fly manner. In the field of optimizing compilers, there have been many sequences of research on instruction reordering [66], and I leave the adoption of compiler techniques as future work.

Polynomial Basis With Smaller Magnitude of Error

In this subsection, I propose a way to reduce the error in the encrypted polynomial basis by reordering homomorphic operations. The error in each encrypted polynomial basis depends on the order of homomorphic operations obtaining it. Polynomials are frequently evaluated not only for bootstrapping [19]- [25] but also in various applications using HE [13, 14]- [18] as all the homomorphic operations are polynomial operations, except for the rotation and conjugation.

In general, it is beneficial to find a polynomial basis first and then evaluate the polynomial. Doing so consumes less level of ciphertext, and obtaining a polynomial basis is necessary for efficient evaluation algorithms such as the baby-step giant-step algorithm or the Paterson-Stockmeyer algorithm. Hence, it is essential to reduce error in the encrypted polynomial basis.

As the Chebyshev polynomial basis will be used in the later sections for polynomial approximation in bootstrapping, I use here the Chebyshev polynomial basis as an example. However, I note that the method in this subsection can also be applied to other polynomial bases. $T_n(x)$ is usually obtained by the following recursive equation

$$T_n(x) = 2T_{2^k}(x) \cdot T_{n-2^k} - T_{2^{k+1}-n},$$

where k is the greatest integer satisfying $2^k < n$. This is beneficial in terms of depth and simplicity; $T_{2^k}(x)$ is the maximum degree term that can be obtained within the depth k and by using just $T_{2^k}(x)$ and $T_i(x)$'s for $0 \leqslant i \leqslant 2^k$, we can obtain all $T_i(X)$'s for $2^k < i \leqslant 2^{k+1}$.

Let $\boldsymbol{c}_i$ be the ciphertext of message $T_i(x)$ with scaling factor Δ for $i = 0, \dots, n$. Considering that $\boldsymbol{c}_i$ contains error e_i, the error in $\boldsymbol{c}_n$ obtained by $T_n(x) = 2T_k(x) \cdot T_{n-2^k}(x) - T_{2^{k+1}-n}(x)$ is $(2T_{2^k}(x)e_{n-2^k} + 2T_{n-2^k}(x)e_{2^k})\Delta + 2e_{2^k}e_{n-2^k} - e_{2^{k+1}-n}$ by Lemma 4.3, if we ignore the key-switching error for brevity. When the ciphertext is rescaled by Δ, the error becomes

$$2T_{2^k}(x)e_{n-2^k} + 2T_{n-2^k}(x)e_{2^k} + e_{\textsf{scale,rs}} + (2e_{2^k}e_{n-2^k} - e_{2^{k+1}-n})/\Delta,$$

where $e_{\textsf{scale,rs}}$ is another scaling error as described in Lemma 4.2. It is noted that Δ is large enough so that $(2e_{2^k}e_{n-2^k} - e_{2^{k+1}-n})/\Delta$ can be ignored. Roughly, as $T_{n-2^k}(x)$

Table 4.1: Mean, variance, and second moment of $T_i(x)$ when x follows the Gaussian distribution with zero mean and variance σ^2 for $\sigma = 1/6$ and $1/24$, so that the input is highly probable to be in $[-1, 1]$.

i	$\sigma = 1/6$			$\sigma = 1/24$		
	$E[T_i(x)]$	$Var[T_i(x)]$	$E[T_i(x)^2]$	$E[T_i(x)]$	$Var[T_i(x)]$	$E[T_i(x)^2]$
0	1.000	0.000	1.000	1.000	0.000	1.000
1	0.000	0.028	0.028	0.000	0.002	0.002
2	-0.944	0.006	0.898	-0.997	0.000	0.993
3	0.000	0.200	0.200	0.000	0.015	0.0154
4	0.796	0.070	0.704	0.986	0.0004	0.973
5	0.000	0.376	0.376	0.000	0.042	0.042
6	-0.601	0.208	0.569	-0.970	0.002	0.941
7	0.000	0.465	0.465	0.000	0.078	0.078

and e_{2^k} are independent and $E[e_{2^k}]$ is zero, the variance of error $T_{n-2^k}(x)e_{2^k}$ is given as

$$\begin{aligned} Var[T_{n-2^k}(x)e_{2^k}] &= Var[T_{n-2^k}(x)]Var[e_{2^k}] + E[T_{n-2^k}(x)]^2 Var[e_{2^k}] \\ &= E[T_{n-2^k}(x)^2]Var[e_{2^k}]. \end{aligned} \tag{4.2}$$

Since T_i's are not independent variables, calculating the exact error distribution in encrypted T_i's is not straightforward, but roughly according to (4.2) and Table 4.1, we can see that the error multiplied by the even term tends to remain and the error multiplied by the odd term tends to decrease. In Table 4.1, it is shown that $E[T_i(x)^2]$ is close to one when i is an even number. Meanwhile, $E[T_i(x)]$ is zero and $Var[T_i(x)]$ is a small value when i is an odd number. That is, since even degree terms have a large mean of squares, the error is large when multiplication of even terms is used when calculating the Chebyshev basis.

Therefore, when n is even, T_n should be calculated by $T_n(x) = 2T_{2^k-1}(x) \cdot T_{n+1-2^k} - T_{2^{k+1}-n-2}$ rather than $T_n(x) = 2T_{2^k}(x) \cdot T_{n-2^k} - T_{2^{k+1}-n}$. The power-of-two degree terms T_{2^k} can only be found by the product of $T_{2^{k-1}}$, to consume the least level. Thus, the power-of-two degree terms have a large error, and it is also shown in implementation in Subsection 4.7.1. Now, it is clear that it is better to avoid using the power-of-two degree terms as possible as we can when evaluating polynomials.

As a simple example, it is assumed in Table 4.1 that the input messages follow the Gaussian distribution with zero means. It is common to normalize or standardize the input value in deep learning [64, 65, 67, 68], which is the most attractive application of HE. Thus, I am interested in inputs that follow Gaussian distribution. Besides, x should be in $[-1, 1]$ to use Chebyshev polynomials, and thus the input is concentrated in the center (zero). Table 4.1 shows that the smaller the order of the messages and more centered, the larger the even terms and the smaller the odd terms.

There are several reasons why this property is essential. In practice, errors in lower degree terms are essential in homomorphic polynomial evaluation because the evaluation algorithms such as the baby-step giant-step algorithm mostly utilize the lower degree terms. Most importantly, as the higher degree terms are obtained from lower degree terms, and the error is accumulated, minimizing error in lower degree terms is crucial. Finally, in the case of bootstrapping, the input distribution is much more concentrated in the center compared to the case of examples in Table 4.1 and thus, the proposed reordering method is quite efficient in bootstrapping.

In Subsection 4.7.1, the implementation shows that the error in encrypted polynomial basis can be reduced by reordering homomorphic operations. For example, the error variance for the proposed method becomes smaller by $1/1973$ compared to the case without minimization for T_{74}. I can also conclude that the modified baby-step giant-step algorithm in [25] is advantageous in terms of error because the baby-step giant-step algorithm proposed in [20] utilizes power-of-two degree terms.

Lazy Rescaling

I generalize the technique to treat rescaling as a part of multiplication, which is proposed in [62] to do rescaling as lazy as possible. In the method suggested by Kim et al. [23], the scaling factor of the ciphertext is a value around Δ^2, and scaling is performed right before a multiplication, not after a multiplication. It was shown that error could be reduced, and especially, the encryption error can be removed by this method [23]. I generalize this so that the ciphertext can have any scaling factor such as $\approx \Delta^3$ or Δ^4, and the rescaling is done as lazy as possible.

As shown in Lemma 4.2, error in encrypted data is divided by the scaling factor, and rounding error is added when the ciphertext is rescaled. The critical point is that the rescaling has a distributive property, and thus, rescaling can be reordered to reduce errors. In other words, since the rounding errors occur through rescaling in general, it should be done as lazy as possible. For example, as suggested in [23], ciphertexts can be rescaled right before a multiplication. This method prevents further amplification of rescaling errors by addition as well as reduces the number of required rescalings. Moreover, this method reduces the number of rescaling, and thus the more additions out of the total operation, the better the effect.

To reduce the error, rescaling can also be reordered with constant multiplications as well as additions. For example, when one calculating encryption of $a\boldsymbol{x}^8$ by using $\boldsymbol{c}$, which is the encryption of $\boldsymbol{x}$ with scaling factor $\approx \Delta^2$, previously, the calculation was as follows [23]

$$\begin{aligned}
\boldsymbol{c}_2 &\leftarrow \mathsf{Mult}(\mathsf{RS}(\boldsymbol{c}), \mathsf{RS}(\boldsymbol{c})) \\
\boldsymbol{c}_4 &\leftarrow \mathsf{Mult}(\mathsf{RS}(\boldsymbol{c}_2), \mathsf{RS}(\boldsymbol{c}_2)) \\
\boldsymbol{c}_8 &\leftarrow \mathsf{Mult}(\mathsf{RS}(\boldsymbol{c}_4), \mathsf{RS}(\boldsymbol{c}_4)) \\
\boldsymbol{c}_{output} &\leftarrow \mathsf{cMult}(\mathsf{RS}(\boldsymbol{c}_8), a; \Delta),
\end{aligned} \tag{4.3}$$

and it consumes four levels. However, we can reorder the operation as

$$\begin{aligned} \boldsymbol{c}_a &\leftarrow \mathsf{cMult}(\mathsf{RS}\left(\boldsymbol{c}\right), a^{1/8}; \Delta) \\ \boldsymbol{c}_{2a} &\leftarrow \mathsf{Mult}(\mathsf{RS}(\boldsymbol{c}_a), \mathsf{RS}(\boldsymbol{c}_a)) \\ \boldsymbol{c}_{4a} &\leftarrow \mathsf{Mult}(\mathsf{RS}(\boldsymbol{c}_{2a}), \mathsf{RS}(\boldsymbol{c}_{2a})) \\ \boldsymbol{c}_{8a} &\leftarrow \mathsf{Mult}(\mathsf{RS}(\boldsymbol{c}_{4a}), \mathsf{RS}(\boldsymbol{c}_{4a})). \end{aligned} \tag{4.4}$$

When $a\boldsymbol{x}^8$ is obtained using (4.3), the error introduced by rescaling is amplified by a unlike (4.4). However, when a is an integer, it is not necessary to consume level; in other words, a is not scaled by Δ, and thus (4.3) may be advantageous in terms of depth. In that case, unless a is a prime number, depth and error can be reduced simultaneously by multiplying the factors of a in advance.

Equation (4.4) can be improved further by replacing the first line in (4.4) as

$$\boldsymbol{c}'_a \leftarrow \mathsf{RS}\left(\mathsf{cMult}(\boldsymbol{c}, a^{1/8}; \Delta)\right).$$

Let us compare the error in $\boldsymbol{c}_a$ and $\boldsymbol{c}'_a$. Let $\boldsymbol{c}$ be a ciphertext whose the full ciphertext is expressed as

$$\left(\boldsymbol{c}, l, \Delta^2, \boldsymbol{\mu}_{\mathsf{m}}, \boldsymbol{v}_{\mathsf{m}}, \boldsymbol{v}_{\mathsf{e}}\right).$$

Then, from Lemma 4.2, the full ciphertext of $\mathsf{RS}(\boldsymbol{c})$ is expressed as

$$\left(\mathsf{RS}(\boldsymbol{c}), l-1, \Delta, \Delta^{-1}\boldsymbol{\mu}_{\mathsf{m}}, \Delta^{-2}\boldsymbol{v}_{\mathsf{m}}, \Delta^{-2}\boldsymbol{v}_{\mathsf{e}} + \boldsymbol{v}_{\mathsf{scale}}\right)$$

and thus, the full ciphertext of $\boldsymbol{c}_a$ is obtained as

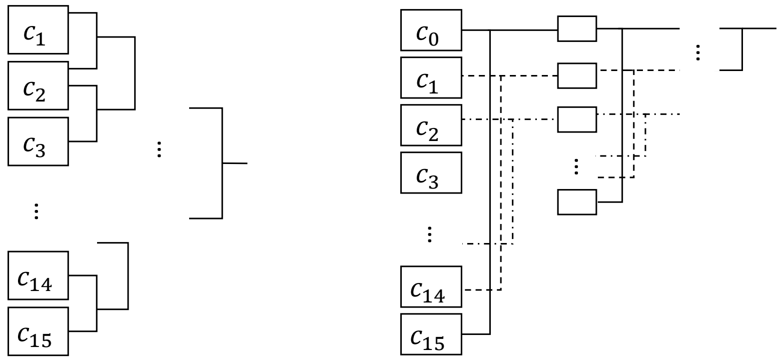
$$\left(\boldsymbol{c}_a, l-1, \Delta^2, a^{1/8}\boldsymbol{\mu}_{\mathsf{m}}, a^{1/4}\boldsymbol{v}_{\mathsf{m}}, a^{1/4}\boldsymbol{v}_{\mathsf{e}} + a^{1/4}\Delta^2\boldsymbol{v}_{\mathsf{scale}}\right).$$

However, the full ciphertext of $\mathsf{cMult}(\boldsymbol{c}, a^{1/8}; \Delta)$ is obtained as

$$\left(\mathsf{cMult}(\boldsymbol{c}, a^{1/8}; \Delta), l-1, \Delta^3, a^{1/8}\Delta\boldsymbol{\mu}_{\mathsf{m}}, a^{1/4}\Delta^2\boldsymbol{v}_{\mathsf{m}}, a^{1/4}\Delta^2\boldsymbol{v}_{\mathsf{e}}\right)$$

and thus, the full ciphertext of $\boldsymbol{c}'_a$ is given derived as

$$\left(\boldsymbol{c}'_a, l-1, \Delta^2, a^{1/8}\boldsymbol{\mu}_{\mathsf{m}}, a^{1/4}\boldsymbol{v}_{\mathsf{m}}, a^{1/4}\boldsymbol{v}_{\mathsf{e}} + \boldsymbol{v}_{\mathsf{scale}}\right).$$



(a) Multiply in order of magnitude of messages with error variance $2^{44.4}$ (b) Multiply the larger and smaller ones first with error variance $2^{21.5}$

Figure 4.1: Two different methods of obtaining $\prod_{i=0}^{15} \boldsymbol{c}_i$.

I note that $\boldsymbol{v}_{\mathsf{scale}}$ is negligible but $a^{1/4}\Delta^2\boldsymbol{v}_{\mathsf{scale}}$ is not. In (4.4), the rescaling error introduced by $\mathsf{RS}(\boldsymbol{c})$ is amplified by a, but we can even rule out this by lazy rescaling.

This technique is quite powerful when evaluating high-degree polynomials, such as approximate modulus reduction in bootstrapping. If rescaling is done before multiplying coefficients, the rounding error is amplified by the coefficients and added by the number of terms, but if rescaling is done as late as possible, it is added only once.

Successive Multiplication of Ciphertexts With Distinct Magnitudes of Messages

When multiplying many ciphertexts, it is not difficult to see that the error can be reduced by pairing the large and small values and multiplying the largest and smallest values first. Let us give an example of how to reduce errors while the calculation time is maintained. There are 16 ciphertexts with level l as

$$(\boldsymbol{c}_i, l, \Delta, 0, 2^{52+i}, 2^{30}),$$

for $i = 0, \ldots, 15$, where $\Delta = 2^{30}$ and $N = 2^{14}$. I compare two ways to obtain the multiplication $\prod_{i=0}^{15} \boldsymbol{c}_i$ in Fig. 4.1. The results and computation time are the same. However, the variances of errors are $2^{44.4}$ for the operation in Fig. 4.1(a) and $2^{21.5}$ for the other.

In summary, I propose three methods of reordering the homomorphic operations to minimize the errors as follows:

1. Mean and variance of the message should be considered when I find the polynomial basis.
2. Resizing should be done as lazy as possible.
3. The error can be reduced by pairing the large and small values and multiplying the largest and smallest values first when successively multiplying ciphertexts.

Aside from the given examples in this dissertation, there must exist tons of optimization methods. It is expected that techniques in optimizing compilers can be adopted to reduce error in approximate homomorphic encryption without compromising performance.

4.3 Near-Optimal Polynomial for Modulus Reduction

By scaling the modulus reduction function by $\frac{1}{q}$, I define $[t]_q$ as $t - k$ for $t \in I_k$, where $I_k = [k - \epsilon, k + \epsilon]$ and k is an integer $|k| < K$. Here, ϵ denotes the rate of the maximum coefficient of the message polynomial and the ciphertext modulus, that is, $\frac{|m|}{q} < \epsilon$. The domain of $[t]_q$ is given by $\bigcup_{k=-K+1}^{K-1} I_k$. In other words, $q \cdot \left[\frac{t}{q}\right]_q \approx m$ for $t = q \cdot I + m$.

4.3.1 Approximate Polynomial Using L2-Norm optimization

Here, I propose how to find an approximate polynomial $p_o(t)$ of $[t]_q$ without using an intermediate approximation, such as a sine or cosine function. The proposed method uses the well-known least-squares method or L2-norm optimization. The objective is to find a set of coefficients $\boldsymbol{c} = (c_0, c_1, \ldots, c_n)$ to minimize $\|[t]_q - p(t)\|_\infty$, where a polynomial of degree n is defined by $p(t) = \sum_{i=0}^{n} c_i \cdot t^i$. Such a polynomial is referred to as the minimax polynomial. It is worth noting that $p(t)$ is equivalent to the inner product of $\boldsymbol{c}$ and $\mathbf{T} = (1, t^1, \ldots, t^n)$.

Here, t_i's are sampled uniformly at intervals of $\delta \ll \epsilon$ in each I_k, namely, $k - \epsilon, k - \epsilon + \delta, \ldots, k + \epsilon - \delta, k + \epsilon$. There are $\frac{2\epsilon}{\delta} + 1$ samples in I_k, and thus we have $N_{tot} = (2K - 1)(\frac{2\epsilon}{\delta} + 1)$ samples. With N_{tot} samples of t_i, one can build a vector of the powers of t_i, that is, $\mathbf{T}_i = (1, t_i, t_i^2, \ldots, t_i^n)$ for $1 \leqslant i \leqslant N_{tot}$.

The object function to be minimized is given as

$$\begin{aligned}\max_i |[t_i]_q - p(t_i)| &= \| ([t_0]_q - p(t_0), \cdots, [t_{N_{tot}}]_q - p(t_{N_{tot}})) \|_\infty \\ &= \|\boldsymbol{y} - \mathbf{T} \cdot \boldsymbol{c}\|_\infty,\end{aligned}$$

where $\mathbf{T}$ is an $N_{tot} \times (n + 1)$ matrix such that $\mathbf{T}[i, j] = t_i^j$ and $\boldsymbol{y}$ is a vector such that $\boldsymbol{y}[i] = [t_i]_q$. Instead of the L-infinity norm, I replace the above objective function by a loss function using the L2-norm. Then, the optimal solution for L2-norm minimization can be efficiently computed. The L2-norm minimization was used in [14] to find an approximate polynomial of logistic function. In this dissertation, I apply discrete L2-norm minimization to find the approximate polynomial of modulus reduction in an efficient way. Let $L_{\boldsymbol{c}}$ denote the L2-norm with the coefficient $\boldsymbol{c}$. Then, we can find $\boldsymbol{c}$ that minimizes the following

$$\begin{aligned}L_{\boldsymbol{c}} &= \|\boldsymbol{y} - \mathbf{T} \cdot \boldsymbol{c}\|_2 \\ &= (\boldsymbol{y} - \mathbf{T} \cdot \boldsymbol{c})^T (\boldsymbol{y} - \mathbf{T} \cdot \boldsymbol{c}).\end{aligned}$$

Unfortunately, the entries of $\mathbf{T}$ become considerably big or small values close to zero, as the degree of the polynomial, n, is high.

Thus, I utilize the Chebyshev polynomials as the basis of the polynomial instead of the power basis. In other words, I redefine the $N_{tot} \times (n + 1)$ matrix $\mathbf{T}$ with entries $\mathbf{T}[i, j] = T_j\left(\frac{t_i}{K}\right)$. As $t_i \in \bigcup_{k=-K+1}^{K-1} I_k$, we have $|\frac{t_i}{K}| < 1$. Hence, the entries of $\mathbf{T}$ are well-distributed in $[-1, 1]$ rather than considerably big values or small values around 0.

Then, the optimal coefficient vector $\boldsymbol{c}^*$ is given as

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} L_{\boldsymbol{c}}.$$

As the loss is a convex function, the optimum solution $\boldsymbol{c}^*$ lies at the gradient zero. The gradient of the loss function $L_{\boldsymbol{c}}$ is given by

$$\nabla L_{\boldsymbol{c}} = -2\boldsymbol{y}^T\mathbf{T} + 2\boldsymbol{c}^T\mathbf{T}^T\mathbf{T}.$$

Setting the gradient to zero produces the optimum coefficient, as follows:

$$\begin{aligned} \nabla L_{\boldsymbol{c}^*} &= 0 \\ \implies \boldsymbol{c}^* &= \left(\mathbf{T}^T\mathbf{T}\right)^{-1}\mathbf{T}^T\boldsymbol{y}. \end{aligned}$$

To sum up, the modulus reduction function can be approximated by

$$[t]_q \approx p_o(t) = \sum_{i=0}^{n} \boldsymbol{c}^*[i] \cdot T_i\left(\frac{t}{K}\right),$$

where $t \in \bigcup_{k=-K+1}^{K-1} I_k$.

Maximum Error of Samples and the Approximation Error

Theorem 4.8. *The approximation error is bounded by the multiplication of the maximum error of the sampled points and* $\mathcal{O}(1 + \frac{n}{N_{tot}})$.

Proof. For $t \in I_k$, let us define the approximation error as the absolute value of following

$$E(t) = (t - k) - p_o(t).$$

Note that $E(t)$ is a polynomial for the domain $t \in I_k$. Denote $E(t) = \sum_j \hat{c}_j x^j$. I have optimized $|E(t_i)|$ for discrete points t_i's.

Consider $|E(t)|$ for t in small intervals of $[t_i, t_i + \delta)$. Then, we have $|E(t)| \leqslant |E(t_i)| + |E(t) - E(t_i)|$ and $|E(t) - E(t_i)|$ is bounded as follows

$$\begin{aligned}
|E(t) - E(t_i)| &= |\sum_j \hat{c}_j \left((t_i + \Delta t)^j - t_i^j \right)| \\
&\approx |\sum_j \hat{c}_j t_i^j \left(j \frac{\Delta t}{t_i} \right)| \\
&\leqslant \left| n \frac{\delta}{t_i} \right| \cdot |\sum_j \hat{c}_j t_i^j| \\
&= \mathcal{O}(n \frac{1}{N_{tot}}) |E(t_i)|,
\end{aligned}$$

where $\Delta t = t - t_i$ for $t \in [t_i, t_i + \delta)$. As $\Delta t < \delta << t_i$, the linear approximation $(1 + \frac{\Delta t}{t_i})^j \approx (1 + j\frac{\Delta t}{t_i})$ is applied. Moreover, we have $\frac{\Delta t}{t_i} \leqslant \frac{\delta}{\epsilon} = \mathcal{O}(\frac{1}{N_{tot}})$, where $t_i > \epsilon$. Otherwise, at least we can always make $\frac{\delta}{t_i} < 1$.

Hence, it is concluded that

$$\max_{t \in \bigcup_{k=-K+1}^{K-1} I_k} |[t]_q - p_o(t)| = \max_i \left([t_i]_q - p_o(t_i) \right) \cdot \mathcal{O}(1 + \frac{n}{N_{tot}}).$$

□

In summary, with fine sampling, the maximum error of the sampled points is close to the global maximum of approximation error. Moreover, as the domain of the objective function is in the real numbers with errors in the CKKS scheme, it is reasonable to handle the sampled values.

L2-norm Instead of L-infinity Norm

Clearly, we can bound the L-infinity norm by the L2-norm:

$$\frac{1}{\sqrt{N_{tot}}} \|\boldsymbol{x}\|_2 \leqslant \|\boldsymbol{x}\|_\infty \leqslant \|\boldsymbol{x}\|_2.$$

Thus, minimizing the L2-norm reduces the L-infinity norm. As it is not a tight bound, we have room for optimization using a higher norm. However, the solution of the L2-norm is clear and can be computed effortlessly. It is difficult to apply minimax polynomial approximation algorithms to the modulus reduction function because it is not a continuous function, and the domain is not a closed interval. However, through the L2-norm optimization problem, it is possible to find a near-optimal solution of the minimax polynomial in a considerably efficient manner without iteration. The next section shows that it is possible to find polynomials with fewer errors than with the currently best-known methods.

Time Complexity for Finding $\boldsymbol{c}^*$

Considering $N_{tot} > n$, the matrix inversion $\left(\mathbf{T}^T\mathbf{T}\right)^{-1}$ is the dominant computation. Hence, the time complexity is $\mathcal{O}(N_{tot}^{2.37})$ when the Coppersmith–Winograd algorithm is used. This is acceptable because $\boldsymbol{c}^*$ is pre-computed and stored as coefficients for the baby-step giant-step algorithm to be explained later or also, the Paterson-Stockmeyer algorithm in [21].

4.3.2 Efficient Homomorphic Evaluation of the Approximate Polynomial

The difference between the proposed and conventional methods in [20] are the coefficients of the approximate polynomial, which is more optimized with the same polynomial basis, the Chebyshev polynomial. Hence, the baby-step giant-step algorithm [20] and modified Paterson-Stockmeyer algorithm [21] can be applied for an efficient homomorphic evaluation of the proposed polynomial. Using Algorithm 4, we can evaluate $p_o(t)$ homomorphically with at most $2^l + 2^{m-l} + m - l - 3$ nonscalar multiplication while consuming m depth, where 2^m is greater than the degree n.

I revisit Algorithm 4, and the number of operations per step is given in Table 4.2. When the Chebyshev polynomials are evaluated, $T_{2n} = 2T_n^2 - T_0$ and $T_{2n+1} = 2T_nT_{n+1} - T_1$ are used and the multiplication of 2 can be replaced by an addition. Hence, one nonscalar multiplication and two additions are required.

In the baby-step, polynomials of degree $2^l - 1$ are evaluated and there are at most $2^m/2^l$ such polynomials. However, when $2^m > n + 1$, there are polynomials with all-

Table 4.2: Number of operations for each step of the baby-step giant-step algorithm in Algorithm 4

	Nonscalar multiplication	Scalar multiplication	Addition
$T_2, \ldots, T_{2^l}$	$2^l - 1$	0	$2 \cdot (2^l - 1)$
$T_{2^{l+1}}, \ldots, T_{2^{m-1}}$	$m - l - 1$	0	$2 \cdot (m - l - 1)$
Baby-step	0	$(n+1) - \left\lceil \frac{(n+1)}{2^l} \right\rceil$	$(n+1) - \lceil \frac{n+1}{2^l} \rceil$
Giant-step	$\lceil \frac{n+1}{2^l} \rceil - 1$	0	$\lceil \frac{n+1}{2^l} \rceil - 1$
Total	$2^l + \lceil \frac{n+1}{2^l} \rceil + m - l - 3$	$(n+1) - \lceil \frac{n+1}{2^l} \rceil$	$n + 2(2^l + m - l - 2)$

Algorithm 4 Baby-step giant-step algorithm [20]

Instance: A ciphertext for t, a polynomial of degree n, $p(t) = \sum_{i=0}^{n} c_i T_i(t)$.

Output: A ciphertext encrypting $p(t)$.

```
1: Let $m$ be the smallest integer satisfying $2^m > n$ and $l \approx m/2$.
2: Evaluate $T_2(t), T_3(t), \ldots, T_{2^l}(t)$ inductively.
3: Evaluate $T_{2^{l+1}}(t), T_{2^{l+2}}(t), \ldots, T_{2^{m-1}}(t)$ inductively.
4: Find polynomials of degree $\leqslant 2^{m-1}$ which satisfy $p = r + qT_{2^{m-1}}$ in forms of
   linear combinations of the Chebyshev basis.
5: Evaluate $q(t)$ and $r(t)$ recursively.
6: Evaluate $p(t)$ using $T_{2^{m-1}}(t), q(t)$, and $r(t)$.
```

zero coefficients. By ignoring them, there are $\lceil (n+1)/2^l \rceil$ polynomials with degree at most $2^l - 1$ in the baby-step. In other words, as 2^m and $n+1$ differ, there are $2^{m-l} - \lceil (n+1)/2^l \rceil$ zero polynomials, that is, $0 \cdot T_0(t) + 0 \cdot T_1(t) + \cdots + 0 \cdot T_{2^l-1}(t)$, in Algorithm 4. Hence, we could ignore these zero polynomials and in the recursive structure, exactly $2^{m-l} - \lceil (n+1)/2^l \rceil$ nonscalar multiplications are ignored in the giant-step. Hence, taking $2^{m'} > n \geqslant 2^{m'-1}$, we have

$$N(n) = N(n - 2^{m'-1}) + N(2^{m'-1} - 1) + 1,$$

which yields

$$N(n) = \left\lceil (n+1)/2^l \right\rceil - 1,$$

where $N(k)$, $k \geqslant 2^l$, is the number of nonscalar multiplications in the giant-step and $N(k) = 0$ for $k < 2^l$. Thus, the number of nonscalar multiplications is given as

$$\left\lceil (n+1)/2^l \right\rceil - 1 + 2^l - 1 + m - l - 1.$$

As shown in Table 4.2, the number of scalar multiplications is $(n+1) - \lceil (n+1)/2^l \rceil$ and the number of addition is $n + 2(2^l + m - l - 2)$. Note that the depth and number of nonscalar multiplications can be minimized when m is the smallest integer satisfying $2^m > n$ and $l \approx m/2$.

4.4 Optimal Approximate Polynomial and Bootstrapping of the CKKS Scheme

Usually, HE schemes support addition and multiplication, and thus only polynomials can be evaluated. However, non-polynomial functions such as ReLU, min/max function, multiplicative inverse, and modulus reduction are frequently required in their applications [69]. Hence, approximate polynomials are used to replace those non-polynomial functions in real-world applications [16]. This subsection proposes a new method to find the optimal approximate polynomial for the CKKS scheme using the generalized least mean square method.

4.4.1 Polynomial Basis Error and Polynomial Evaluation in the CKKS Scheme

Rounding of ciphertexts introduces an additional error during rescaling and key switching in the CKKS scheme. Also, these errors and encryption errors are amplified through homomorphic operations. Generalized polynomial basis of degree n is denoted by $\{\phi_0(t), \phi_1(t), \ldots, \phi_n(t)\}$. For instance, monomial basis $\{1, x, x^2, \ldots, x^n\}$ and Chebyshev polynomial basis $\{T_0(x), T_1(x), \ldots, T_n(x)\}$ are polynomial bases. We can assume that each polynomial basis has independent errors due to rounding and encryption errors, namely, the basis errors.

When a polynomial $f(x) = \sum c_i\phi_i(x)$ is evaluated homomorphically, it is expected that the result is $f(x) + e$ for a given input x and small error e. However, in the CKKS scheme, there exists an error in encrypted data and thus, each basis of polynomial, $\phi_i(x)$ contains independent basis error $e_{\mathsf{b},i}$. Hence, the output is given as

$$\begin{aligned}\sum c_i(\phi_i(x) + e_{\mathsf{b},i}) &= \sum c_i\phi_i(x) + \sum c_i e_{\mathsf{b},i}\\ &= f(x) + \sum c_i e_{\mathsf{b},i}.\end{aligned}$$

As $e_{\mathsf{b},i}$ is a small value, the error $\sum c_i e_{\mathsf{b},i}$ is small in general. However, when $|c_i|$

is much larger than $\|f(x)\|_\infty$ such as a high-degree polynomial for bootstrapping, $\sum c_i e_{\mathsf{b},i}$ might overwhelm $f(x)$.

High-degree approximate polynomials have large coefficients in general. The outputs of the approximate polynomial for modulus reduction, which is essential for the bootstrapping of the CKKS scheme, are in $[-\epsilon, \epsilon]$, where $\epsilon = \frac{|m|}{q}$. There have been series of studies in approximate polynomials in the CKKS scheme [4]– [25], but the error amplified by coefficients were not considered in the previous studies. The magnitude of coefficients of approximate polynomial should be controlled when we find the approximate polynomial of any non-polynomial functions, as well as the modulus reduction, which deteriorates message precision by the successive homomorphic operations.

4.4.2 Variance-Minimizing Polynomial Approximation

In the encrypted data of the CKKS scheme, errors include errors added for security, rounding errors, approximation errors, and errors added during homomorphic operations. Therefore, from the central limit theorem, the basis errors can be considered Gaussian random variables with zero means. The approximate polynomial can be optimized by minimizing the variance of the approximation error, rather than using minimax approximate polynomial [25].

As shown in Subsection 4.4.1, basis error is amplified by coefficients of the approximate polynomial. Thus, the magnitude of its coefficients should not be large values and using the generalized least squares method, the optimal coefficients vector $\boldsymbol{c}^*$ of the approximate polynomial is obtained as

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} \left(Var[e_{\mathsf{aprx}}] + \sum w_i \boldsymbol{c}_i^2 \right) \tag{4.5}$$

for weight constants w_i, where w_i's are determined by basis error and e_{aprx} is the approximation error. I call the proposed approximate polynomial obtained by (4.5) as the *error variance-minimizing approximate polynomial*, and there exists an analytic solution. It is noted that when w_i's are all zero, the approximate polynomial minimizes the variance of approximation error only.

Especially, error variance-minimizing approximate polynomial is more efficient and suitable for the bootstrapping of the CKKS scheme as it reduces the bootstrapping error compared to the minimax approximate polynomial. Considering SlotToCoeff, the error in the j-th slot is given as $e(\zeta_M^j) = \sum_{i=0}^{N-1} e_i \cdot \zeta_M^{ji}$, which is an addition of independent and identically distributed random variables, $e_i \cdot \zeta_M^{ji}$. Hence, the minimax approximate polynomial does not minimize the variance of errors, which are the actual errors in the encrypted data in bootstrapping. Instead, minimizing the error variance of each coefficient minimizes errors in slot values after bootstrapping. This implies that minimizing the variance of the approximate polynomial error is optimal to reduce the error during the bootstrapping of the CKKS scheme compared to the minimax approximation. The error variance-minimizing approximate polynomial is described in detail by taking bootstrapping as a specific example in the next subsection.

4.4.3 Optimal Approximate Polynomial for Bootstrapping and Magnitude of Its Coefficients

The key part of the bootstrapping of the CKKS scheme is the homomorphic evaluation of the modulus reduction. In [19], the modulus reduction is approximated by the sine function, and the approximate polynomial for the sine function is homomorphically evaluated using a Taylor series expansion and the double-angle formula. Moreover, with optimized nodes for the Chebyshev interpolation, the polynomial approximation is significantly improved [20]. The least-square method to find a better approximate polynomial without trigonometric functions is proposed in [24] and the method to find minimax approximate polynomial is also proposed in [25].

In [25], the modulus reduction function, $t \pmod q$ is considered as

$$\frac{q}{2\pi} \arcsin \circ \sin\left(\frac{2\pi t}{q}\right)$$

and the approximate polynomials for $\arcsin(\cdot)$ and $\sin(\cdot)$ are evaluated sequentially. This dissertation focuses on the direct-approximation of modulus reduction rather than trigonometric function approximation to minimize the bootstrapping error and depth. However, as the proposed error variance-minimizing method can be applied to any function, the composite method and double-angle formula for faster evaluation in [20, 25] can also be applied to the proposed method.

By scaling the modulus reduction function by $\frac{1}{q}$, I define $f_{\mathsf{mod}}(t) = t - i$ if $t \in I - i$, that is, $f_{\mathsf{mod}} : \bigcup_{i=-K+1}^{K-1} I_i \to [-\epsilon, \epsilon]$, where $I_i = [i - \epsilon, i + \epsilon]$ and i is an integer such that $|i| < K$. Here, ϵ denotes the ratio of the maximum coefficient of the message polynomial and the ciphertext modulus, that is, $|m_i|/q \leqslant \epsilon$, where m_i denotes a coefficient of message polynomial $m(X)$.

Let T be the random variable of input t of $f_{\mathsf{mod}}(t)$. Then, $T = R + I$, where R is the random variable of r, the rational part of t and I is the random variable of i, for $t \in I_i$. It should be noted that $\Pr_T(t) = \Pr_I(i) \cdot \Pr_R(r)$ is satisfied for $t = r + i$ as i and r are independent and $\bigcup_i I_i = [-\epsilon, \epsilon] \times \{0, \pm 1, \ldots, \pm(K-1)\}$, where $\Pr_T, \Pr_I$, and $\Pr_R$ are probability mass functions or probability density functions of T, I, and R, respectively.

The approximation error in t is given as

$$\begin{aligned} e_{\mathsf{aprx}}(t) &= p(t) - f_{\mathsf{mod}}(t) \\ &= p(t) - (t - i), \end{aligned}$$

where $p(t)$ is the approximate polynomial of $f_{\mathsf{mod}}(t)$. Then the variance of e_{aprx} is given as

$$\begin{aligned} Var[e_{\mathsf{aprx}}] &= E[e_{\mathsf{aprx}}^2] - E[e_{\mathsf{aprx}}]^2 \\ &= E[e_{\mathsf{aprx}}^2] \\ &= \int_t e_{\mathsf{aprx}}(t)^2 \cdot \Pr_T(t)\, dt, \end{aligned}$$

where the mean of e_{aprx} is zero by Lemma 4.9. This gives us the following equation

$$\begin{aligned}Var[E_{\mathsf{aprx}}] &= \sum_i \int_m e_{\mathsf{aprx}}(t)^2 \cdot \Pr_R(r) \cdot \Pr_I(i)\, dt \\ &= \sum_i \Pr_I(i) \int_{t=i-\epsilon}^{i+\epsilon} e_{\mathsf{aprx}}(t)^2 \cdot \Pr_R(t-i)\, dt.\end{aligned}$$

It is noted that the integral can be directly calculated or replaced by the sum of discretized values as in [24].

Lemma 4.9. *Let $p(t)$ be an approximate polynomial that minimizes $Var[f(t) - p(t)]$ for a function f. Then, $E[f(t) - p(t)] = 0$ is satisfied.*

Proof. Assume that $E[f(t) - p(t)] = \mu \neq 0$. Then, we can see that it is always satisfied that

$$\begin{aligned}Var[f(t) - (p(t) + \mu)] &= E[(f(t) - p(t))^2] - \mu^2 \\ &< Var[f(t) - p(t)],\end{aligned}$$

which is a contradiction.

□ □

Let $\{\phi_0(t), \phi_1(t), \ldots, \phi_n(t)\}$ be a generalized polynomial basis. Then, I represent the approximate polynomial by $p(t) = \sum_{k=0}^{n} c_k \phi_k(t)$, where c_k's are coefficients. In this dissertation, polynomial approximation aims to find the coefficients that minimize $Var[e_{\mathsf{aprx}}]$. However, it should be noted that the approximation of the modulus reduction, $f_{\mathsf{mod}}(t), t \in \bigcup_{i=-K+1}^{K-1} I_i$, is required to be very accurate, especially for the bootstrapping and thus, a high dimensional approximate polynomial should be used. The problem is that high-degree approximate polynomials usually have large coefficients. Generally, it is not a problem, but in the case of the CKKS scheme, large coefficients amplify the errors on the polynomial basis. Therefore, a high-degree approximate polynomial with small coefficients is required. Hence, we find $\boldsymbol{c}^*$ such that

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} \left(Var[e_{\mathsf{aprx}}] + \sum_{i=0}^{n} w_i c_i^2 \right), \tag{4.6}$$

and the solution satisfies

$$\nabla_{\boldsymbol{c}}\left(Var[e_{\mathsf{aprx}}]+\sum_{i=0}^{n} w_i c_i^2\right)=0,$$

where $\boldsymbol{c}=(c_0,c_1,\dots,c_n)$ and $\boldsymbol{w}=(w_0,w_1,\dots,w_n)$ are coefficient and weight constant vectors, respectively.

It is noted that the variance of error in each $\phi_i(t)$ may be different. For example, when the ciphertext of x^4 is obtained by multiplying the ciphertext of x^2, while the ciphertext of x^2 contains a rounding error $e_{\mathsf{rnd},2}$. Then, the ciphertext of x^4 has error $2e_{\mathsf{rnd},2}x^2+e_{\mathsf{rnd},2}^2+e_{\mathsf{rnd},4}$. In general, we can say that a high-degree term of polynomial basis causes a larger error. Hence, a precise adjustment of the magnitude of polynomial coefficients can also be made using multiple weight constants, w_i's.

Theorem 4.10. *There exists a polynomial-time algorithm that finds $\boldsymbol{c}=(c_0,\dots,c_n)$ satisfying*

$$\arg\min_{\boldsymbol{c}}\left(Var[e_{\mathit{aprx}}]+\sum_{i=0}^{n} w_i \boldsymbol{c}_i^2\right).$$

Proof. From Lemma 4.9, we can assume that $E[e_{\mathsf{aprx}}]=0$. Then, we have the following equation

$$\begin{aligned}Var[e_{\mathsf{aprx}}]&=E[e_{\mathsf{aprx}}^2]-E[e_{\mathsf{aprx}}]^2\\&=E[e_{\mathsf{aprx}}^2]\\&=E[f_{\mathsf{mod}}(t)^2]2E[f_{\mathsf{mod}}(t)\cdot p(t)]+E[p(t)^2].\end{aligned}$$

By substituting $p(t)=\sum_{k=0}^{n} c_k\phi_k(t)$, we have

$$\frac{\partial}{\partial c_j}Var[e_{\mathsf{aprx}}]=-2E[f_{\mathsf{mod}}(t)\phi_j(t)]+2\sum_{k=0}^{n} c_k\cdot E[\phi_k(t)\phi_j(t)].$$

The solution of the following system of linear equations, $\boldsymbol{c}^*$ satisfies

$$\boldsymbol{c}^*=\arg\min_{\boldsymbol{c}}\left(Var[e_{\mathsf{aprx}}]+\sum_{i=0}^{n} w_i c_i^2\right):$$

$$\mathbf{T}\cdot\boldsymbol{c}=\boldsymbol{y}, \tag{4.7}$$

where

$$\mathbf{T} = \begin{bmatrix} E[\phi_0\phi_0] + w_0 & E[\phi_0\phi_1] & \dots & E[\phi_0\phi_n] \\ E[\phi_1\phi_0] & E[\phi_1\phi_1] + w_1 & \dots & \vdots \\ \vdots & & \ddots & \vdots \\ E[\phi_n\phi_0] & E[\phi_n\phi_1] & \dots & E[\phi_n\phi_n] + w_n \end{bmatrix}$$

and

$$\boldsymbol{y} = \begin{bmatrix} E[f_{\mathsf{mod}}(t)\phi_0(t)] \\ E[f_{\mathsf{mod}}(t)\phi_1(t)] \\ \vdots \\ E[f_{\mathsf{mod}}(t)\phi_n(t)] \end{bmatrix}.$$

As $E[\phi_i\phi_j]$ and $E[f_{\mathsf{mod}}(t)\phi_i(t)]$ are integral of polynomials, these are easy to calculate. □ □

Theorem 4.10 states that the approximate polynomial for $p(t)$ is found efficiently. In other words, the computation time of solving this system of linear equations is the same as that of finding an interpolation polynomial for given points, which is faster than the modified Remez algorithm [25].

4.4.4 Reducing Complexity and Error Using Odd Function

When the approximate polynomial is an odd function, one can save time to find and homomorphically evaluate the approximate polynomial. Using the fact that $f_{\mathsf{mod}}(t)$ is an odd function and the minimax approximate polynomial of an odd function is also an odd function, the approximate polynomial for $f_{\mathsf{mod}}(t)$ with only odd degree terms was derived [24, 25]. Moreover, the number of operations to evaluate the approximate polynomial can also be reduced by omitting even-order terms. Besides, the amplified basis error is also reduced as there are only half of the terms to be added when the approximate polynomial is an odd function. Theorem 4.11, it is shown that when the target function of polynomial approximation such as $f_{\mathsf{mod}}(t)$ is odd and the probability density function is even, the error variance-minimizing approximate polynomial is also an odd function from the following theorem. In the following section, odd approximate polynomials are obtained and implemented based on the following theorem.

Theorem 4.11. *When* $\Pr_T(t)$ *is an even function and* $f(t)$ *is an odd function, the error variance-minimizing approximate polynomial for* $f(t)$ *is an odd function.*

Proof. Existence and uniqueness: The error variance-minimizing approximate polynomial minimizes $Var[e_{\mathsf{apprx}}] + \sum w_i c_i^2$, which is a quadratic polynomial for the coefficients $\boldsymbol{c}$. Hence, there exists the one and only solution.

Oddness: Let P_m denote the subspace of the polynomial function of degree at most m and $f_m(t)$ denote the unique element of P_m that is closest to $f(t)$ in the variance of difference. Then, $Var[-f(-t)-p(t)]+\sum w_i c_i^2$ is minimized when $p(t) = -f_m(-t)$, because

$$\begin{aligned} Var\left[-f(-t) - p(t)\right] &= \int_t (-f(-t) - p(t))^2 \cdot Pr(t)dt \\ &= \int_{-u} -(f(u) + p(-u))^2 \cdot Pr(-u)du \\ &= \int_u (f(u) - (-p(-u)))^2 \cdot Pr(u)du \end{aligned}$$

is satisfied and the squares of coefficients of $f_m(t)$ and $-f_m(-t)$ are the same. As the error variance-minimizing approximate polynomial is unique, I conclude that $f_m(t) = -f_m(-t)$.

□ □

4.4.5 Generalization of Weight Constants and Numerical Method

Earlier it is noted that the weight constant vector $\boldsymbol{w}$ provides the trade-off between the magnitude of coefficients and variance of polynomial approximation error. In this subsection, I generalize the amplified basis error term $\sum_{i=0}^{n} w_i \boldsymbol{c}_i^2$ and find the optimal approximate polynomial for baby-step giant-step algorithm. The numerical method to select the weight-constant vector $\boldsymbol{w}$ is also proposed.

Algorithm 5 Generalized odd baby-step giant-step algorithm [25]

Instance: A ciphertext for t, a polynomial of degree n, $p(t) = \sum_{i=0}^{n} c_i T_i(t)$.

Output: A ciphertext encrypting $p(t)$.

```
1: Let $l$ be the smallest integer satisfying $2^l k > n$ for an even number $k$.
2: Evaluate $T_2(t), T_3(t), \ldots, T_k(t)$ inductively, but even degree polynomials other than $T_k(t)$ are not necessary to be obtained unless it is used to obtain other polynomials.
3: Evaluate $T_{2k}(t), T_{2^2 k}(t), \ldots, T_{2^{l-1}k}(t)$ inductively.
4: Find polynomials $r(t), q(t)$ of degree $\leqslant 2^{l-1}k$, which satisfy $p(t) = r(t) + q(t)T_{2^{l-1}k}(t)$ in forms of linear combinations of the Chebyshev polynomial basis.
5: Evaluate $q(t)$ and $r(t)$ recursively, by using the quotient and remainder polynomials when those are divided by $T_{2^{l-2}k}(t)$.
6: Evaluate $p(t)$ using $T_{2^{l-1}k}(t), q(t)$, and $r(t)$.
```

The basis error can be found using the method proposed in Subsection 4.2.4 or numerically. Let $v_{\mathsf{b},i}$ be the variance of basis error in a slot of ciphertext which is an encryption of $T_i(x)$. Then, the basis errors are multiplied by c_i and the amplified error is given as $\sum_{i=0}^{n} c_i^2 v_{\mathsf{b},i}$. Considering the approximation error, it should be noted that a large scaling factor $\Delta_{\mathsf{bs}} = O(q)$ is multiplied to the result of EVALMOD [19, 24]. For brevity of description, I let $\Delta_{\mathsf{bs}} = q$; it is proper, because there are coefficients of $m(X)$, m_i's in the slots after COEFFTOSLOT, and by letting its scaling factor q, the slot values become m_i/q with scaling value q. Hence, the approximation error is given as $q^2 \cdot Var[e_{\mathsf{aprx}}]$. Therefore, it is optimal when w_i has the value $v_{\mathsf{b},i}/q^2$.

However, in practice, fast evaluation algorithms such as the baby-step giant-step and the Paterson-Stockmeyer algorithms are used to evaluate the polynomial efficiently. Thus, the coefficients are changed and the errors are not simply added.

The generalized baby-step giant-step algorithm for an odd degree polynomial is given in Algorithm 5. The basic blocks of the baby-step giant-step algorithm are polynomials of degree less than k, so-called baby-step polynomials, $p_i(t) = \sum_{j\in\{1,3,\dots,k-1\}} d_{i,j}T_j(t)$ for $i = 0, 1, \dots, 2^l - 1$. Then, it can easily be seen that $p(t)$ consists of $p_i(t)$'s and $T_k(t), \dots, T_{2^{l-1}k}(t)$. For example, when $l = 2$, we have

$$p(t) = (p_3(t)T_k(t) + p_2(t))T_{2k}(t) + p_1(t)T_k(t) + p_0(t). \tag{4.8}$$

Hence, when $p(t)$ is evaluated, the coefficients of $p_i(t)$'s amplify the basis error, and thus, minimizing the basis error of basis elements with a degree less than k is crucial.

Let E_p be a function of $\boldsymbol{d}$, which is the variance of basis error amplified by coefficients $\boldsymbol{d} = (d_{0,1}, d_{0,3}, \dots, d_{2^l-1,k-1})$. A heuristic assumption that T_i's are independent and the encryptions of $T_k(t), \dots, T_{2^{l-1}k}(t)$ have small error simplifies E_p. Let $\hat{T}_i$ be the product of all T_{2^jk}'s multiplied by p_i, for example, $\hat{T}_0 = 1$ and $\hat{T}_3 = T_kT_{2k}$ in (4.8). Considering the error multiplied by $d_{i,j}$, $e_j \cdot \hat{T}_i$ is the dominant term as T_i has zero mean and very small variance. Thus, it can be said that $E_p = \sum_i \sum_j d_{i,j}^2 E[\hat{T}_i^2] v_{\mathsf{b},j}$, which is a quadratic function of $\boldsymbol{d}$. In other words, we have $E_p = \boldsymbol{d}^{\mathsf{T}}\mathbf{H}\boldsymbol{d}$, where $\mathbf{H}$ is a diagonal matrix that

$$\mathbf{H}_{ki+j,ki+j} = E[\hat{T}_i^2] v_{\mathsf{b},j}.$$

Thus, (4.6) is generalized as

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} \left(Var[q \cdot e_{\mathsf{aprx}}] + E_p \right).$$

Since $\boldsymbol{c}$ and $\boldsymbol{d}$ have linearity, $\nabla_{\boldsymbol{c}} E_p$ can easily be calculated. Specifically, we have

$$\begin{aligned} \boldsymbol{c} &= \mathbf{L}\boldsymbol{d} \\ &= \begin{bmatrix} \mathbf{A}_{2^{l-1}k} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{A}_{2^{l-2}k} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{2^{l-2}k} \end{bmatrix} \cdots \begin{bmatrix} \mathbf{A}_k & & \\ & \ddots & \\ & & \mathbf{A}_k \end{bmatrix} \cdot \boldsymbol{d}, \end{aligned} \tag{4.9}$$

where

$$\mathbf{A}_k = \begin{bmatrix} \mathbf{I}_{k/2} & \frac{1}{2}\mathbf{J}_{k/2} \\ \mathbf{0} & \frac{1}{2}\mathbf{I}_{k/2} \end{bmatrix},$$

$\mathbf{I}_{k/2}$ is the $k/2 \times k/2$ identity matrix, and $\mathbf{J}_{k/2}$ is the $k/2 \times k/2$ exchange matrix. Equation (4.9) is derived from $T_m(x)T_n(x) = \frac{1}{2}(T_{m+n}(x) + T_{|m-n|}(x))$. Then, we have $\nabla_{\boldsymbol{c}} E_p = 2\mathbf{L}^{-1^{\mathsf{T}}}\mathbf{H}\mathbf{L}^{-1}\boldsymbol{c}$. Hence, the optimal coefficient $\boldsymbol{c}^*$ satisfies

$$\left(\mathbf{T} + \mathbf{L}^{-1^{\mathsf{T}}}\mathbf{H}\mathbf{L}^{-1}\right)\boldsymbol{c}^* = y.$$

Instead of finding E_p, a simple numerical method can also be used. Actually, the value of $\hat{T}_i$ is close to one, and moreover, the numerical method shows good error performance in the implementation in Subsection 4.7.2. I let $w_i = w$ for all i and find the value using the numerical method for brevity. When w increases, the coefficients $\boldsymbol{c}$ decreases, and $Var[e_{\text{aprx}}]$ increases, and thus its sum is a convex function of w. Thus, the optimal polynomial is found by using numerical methods by finding the optimal w. $\boldsymbol{c}$ is uniquely determined by w, and using $\boldsymbol{c}$, the coefficients for the baby-step giant-step algorithm or the Paterson-Stockmeyer algorithm can be calculated. The magnitude of the basis errors that are amplified by coefficients is similar to the rounding error whose variance is $\frac{N(h+1)}{12}$. After multiplying $\|\mathbf{L}^{-1}\boldsymbol{c}\|_2$ with the variance, it is added to $q^2 \cdot Var[e_{\text{aprx}}]$. In other words, we adjust w to minimize

$$Var[e_{\text{aprx}}] + \frac{w}{q^2} \cdot \|\mathbf{L}^{-1}\boldsymbol{c}\|_2^2,$$

where w is close to $\frac{N(h+1)}{12}$.

The proposed method is efficient when an accurate approximation is required. In [62], a bootstrapping for the CKKS scheme with a non-sparse key was proposed; in other words, the secret key has Hamming weight $h \approx N/2$. In that case, K is a considerable value, so that a high-degree approximate polynomial is required. Therefore, if the method proposed in this dissertation is applied to the non-sparse key case, its impact on bootstrapping error reduction will be significant.

4.5 Comparison and Implementation

I conduct an experiment to compare the proposed method with previous work in [20], which, to our knowledge, is the best current method. Maximum errors between $[t]_q$ and the approximate polynomials are numerically computed and compared. Note that we can analytically obtain the maximum error once the polynomial is known and that the approximate error is an absolute value of a polynomial. However, the numerically computed maximum error is sufficient as it is approximately equal to the real value, and we are dealing with approximate arithmetic here. For example, we can see that the numerically computed maximum error for the polynomial is almost the same as the error bound presented in [20].

In Fig. 4.2, I plot the maximum error in log scale, $\log_2(|[t]_p - p_o(t)|)$, while fixing n and varying ϵ or fixing $\epsilon = 2^{-7}, 2^{-10}$ and varying n. It is noteworthy that the proposed method gives an approximation (error below 2^{-21}) for a large ϵ $(= 2^{-7})$ with the depth of 7, whereas the previous method cannot achieve this even when using polynomials of a higher degree. This is because the sine function is not a suitable approximation for the modulus reduction when ϵ is large. As the proposed method does not depend on the sine function, even large-sized messages that could not be handled by the previous method can be handled by low-degree polynomials in the proposed method.

A staircase shape is shown in Fig. 4.2(b), in other words, the maximum approximation errors are similar when the degrees are $2n - 1$ and $2n$. This is because the target of the approximation, the modulus reduction function $[t]_q$, is an odd function. The following proposition shows that the minimax polynomial for an odd function is an odd function.

Proposition 4.1. *If $f(t)$ is an odd function, the best approximation among the polynomials of degree n is also odd.*

Proof. Let P_m denote the subspace of the polynomial function of a degree of at most

m and $f_m(t)$ denote the unique element of P_m that is closest to $f(t)$ in the supreme norm. I define $p(t) \in P_m$ by $p(t) = \frac{1}{2}(f_m(t) - f_m(-t))$. Then, for all u in the domain of $f(t)$, we have

$$\begin{aligned}
|f(u) - p(u)| &= \left| f(u) - \frac{1}{2}\left(f_m(u) - f_m(-u)\right) \right| \\
&\leqslant \frac{1}{2}|f(u) - f_m(u)| + \frac{1}{2}|f(u) + f_m(-u))| \\
&= \frac{1}{2}|f(u) - f_m(u)| + \frac{1}{2}|f(-u) - f_m(-u))| \\
&\leqslant \sup_t |f(t) - f_m(t)|.
\end{aligned}$$

If $p(t) \neq f_m(t)$, it contradicts that $f_m(t)$ is the closest to $f(t)$. Hence, $f_m(t) = p(t) = \frac{1}{2}(f_m(t) - f_m(-t))$ and this is an odd function. $\square$

From the polynomial coefficients of the proposed method, it can be observed that the coefficient of an even-order term has a significantly small value close to zero in $p_o(t)$. This is evidence for the fact that the proposed method finds a polynomial near the minimax polynomial because the modulus reduction function is an odd function. It can be seen that the even-order terms are rather a handicap for finding an approximate polynomial. Therefore, approximating using only odd-order Chebyshev polynomials yields a more accurate approximate polynomial.

It is one of the advantages of the proposed method that the nature of the odd function can be utilized. In contrast, the previous method [20] cannot make use of odd function because their cosine function in the constrained domain is not an odd function nor even function. Using the fact that the odd functions are symmetric with respect to the origin, we can solve the L2-norm minimization only with samples whose value is greater than zero. Thus, the number of rows and columns of the matrix $\mathbf{T}$ is reduced by half each. As a result, the time complexity of matrix inversion is reduced to about 1/8. Also, some operations on even-order terms may be ignored during evaluation.

In Table 4.3, I compare previous results from [20, 21] and the results of the proposed method for $\epsilon = 2^{-10}$. The criterion is the maximum value of the approximation

Table 4.3: Comparison of approximate polynomial performance of various methods ($K = 12$ and $\epsilon = 2^{-10}$)

Methods	Degree	Max err ($\log_2$)	Nonscalar multiplication	Scalar multiplication	Addition	Depth
Proposed polynomial (L2-norm min.)	73	-27.18	17 (PS alg.*)	68	109	7
	75	-27.78	17 (PS alg.)	68	109	7
	119	-35.91	20 (PS alg.)	113	160	7
	127	-40.10	24 (BSGS**)	120	161	7
[20] (Modified Chebyshev)	74	-26.42	17 (PS alg.)	68	109	7
	119	-27.28	20 (PS alg.)	113	160	7
	127	-27.28	24 (BSGS)	120	161	7
[21] (Chebyshev interpolation)	119	-	20 (PS alg.)	113	160	7

*PS alg.: Paterson-Stockmeyer algorithm.

**BSGS: Baby-step giant-step algorithm.

error. As shown in Table 4.3, I reduce the approximation error from $2^{-26.42}$ to $2^{-27.18}$, while also reducing the degree from 74 to 73. Note that due to the method of selecting nodes, the method of [20] is restricted in the degree of an approximate polynomial. It is evident that the difference is greater when a more precise approximation is needed; moreover, in some cases, the number of nonscalar multiplications, scalar multiplications, and additions are reduced by reducing the degree of an approximate polynomial. Moreover, notice that the maximum error of the proposed method is always smaller than the previous state-of-the-art results even with the same degree polynomial.

It can be seen that the proposed method provides a trade-off between approximation error and the degree of the approximate polynomial. When a polynomial of degree 127 is used, the proposed method provides an approximation error below 2^{-40}. However, when the previous method is used, the error cannot be reduced below $2^{-27.28}$ as it is bounded by the error between sine function and $[t]_q$ as in Table 4.3 and Fig. 4.2(b). Table 4.3 and Fig. 4.2(b) show that that increasing the degree of the polynomial does not lower the approximation error to some extent when using the previous methods.

A comparison of the minimum degrees necessary to achieve the desired error bounds is given in Table 4.4. For $\epsilon = 2^{-6}$, it is shown that the proposed method achieves an approximation error of less than 2^{-20} with only a depth of 7. When a polynomial $p_{cos}(t)$ approximates a sine or cosine function as in [19–21], the approximate error is bounded by the sine function. In other words, it is bounded by

$$\begin{aligned}\max_t |[t]_q - p_{cos}(t)| &\geqslant \max_{m\in[-\epsilon q,\epsilon q]} \left| m - \frac{1}{2\pi}\sin\left(2\pi\frac{m}{q}\right)\right| \\ &\approx \frac{1}{2\pi}\cdot\frac{1}{3!}\left(\frac{2\pi|m|}{q}\right)^3, \end{aligned} \tag{4.10}$$

which is small when $\frac{|m|}{q}$ is small. However, as $\frac{|m|}{q}$ increases, the bound increases in the third order. For $\epsilon = 2^{-10}, 2^{-9}, 2^{-8}$, and 2^{-7}, the bounds are given as $2^{-27}, 2^{-24}, 2^{-21}$, and 2^{-18}. Table 4.4 shows that the approximation error of a polynomial found by the method in [20] is above those bounds. Therefore, for applications that require a more accurate approximation than this range, the proposed method should be used.

Table 4.4: Comparison of minimum degree of approximate polynomials to achieve desired error bound

	$\|[t]_q - p(t)\| < 2^{-25}$		$\|[t]_q - p(t)\| < 2^{-21}$	
	Proposed	Method in [20]	Proposed	Method in [20]
ϵ	Deg	Deg	Deg	Deg
2^{-11}	69	70	63	63
2^{-10}	73	74	65	65
2^{-9}	75	converge to 2^{-24}	71	72
2^{-8}	119	converge to 2^{-21}	73	76
2^{-7}	127	converge to 2^{-18}	121	converge to 2^{-18}
2^{-6}	137	converge to 2^{-15}	127	converge to2^{-15}

The proposed method is implemented in `SageMath 9.0`. It requires 1.01 s in average on `Intel Core i7-6700k` (4.0 GHz) to find the optimal coefficients with 32 samples for each I_k, the degree $n = 73$, and $\epsilon = 2^{-10}$. Note that most of the results in Table 4.3, 4.4, and Fig. 4.2 are driven by 32 samples for each I_k. This implies that massive samples are not required for good approximations. Instead, with only ~ 300 samples (depends on the degree of polynomial), the proposed method surpasses the best-known method [20].

4.6 Reduction of Level Loss in Bootstrapping

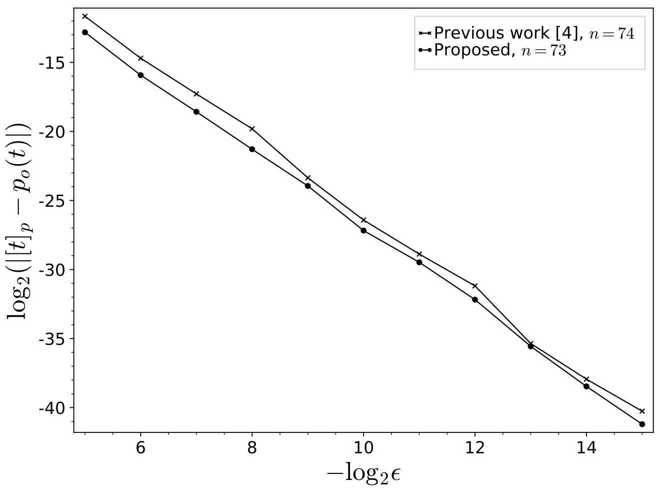
By using the proposed method, better parameters that reduce the loss of level during the bootstrapping can be selected. As discussed in the previous section, the proposed method finds a more accurate approximate polynomial for relatively large ϵ than the previous best method. This section explains how such property leads to better parameters.

I will make use of the following lemmas from [4, 19] for noise estimation.

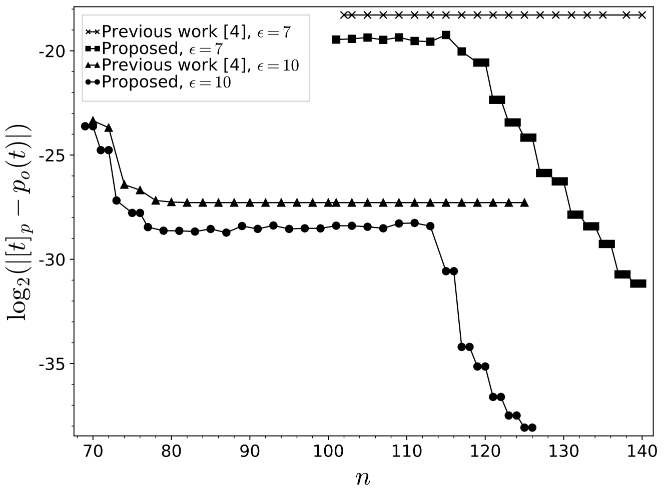
Lemma 4.12 ([4], Lemma 2). *Let $\boldsymbol{c}' \leftarrow \texttt{RS}_{l\to l'}(\boldsymbol{c})$ for a ciphertext $\boldsymbol{c} \in \mathcal{R}_{q_l}^2$. Then $\langle \boldsymbol{c}', sk\rangle = \frac{q_{l'}}{q_l}\langle \boldsymbol{c}, sk\rangle + e \pmod{q_{l'}}$ for some $e \in \mathcal{R}$ satisfying $\|e\|_\infty^{\mathsf{can}} \leqslant B_{\mathsf{rs}}$ for $B_{\mathsf{rs}} = \sqrt{N/3}\cdot(3 + 8\sqrt{h})$.*

Lemma 4.13 ([19], Lemma 4). *Let $\boldsymbol{c} \in \mathcal{R}_q^2$ be a ciphertext with respect to a secret key $sk' = (1, s')$ and let $swk \leftarrow \texttt{KSGen}_{sk}(s')$. Then $\boldsymbol{c}' \leftarrow \texttt{KS}_{swk}(\boldsymbol{c})$ satisfies $\langle \boldsymbol{c}', sk\rangle = \langle \boldsymbol{c}, sk'\rangle + e_{\mathsf{ks}} \pmod{q}$ for some $e_{\mathsf{ks}} \in \mathcal{R}$ with $\|e_{\mathsf{ks}}\|_\infty^{\mathsf{can}} \leqslant P^{-1}\cdot q \cdot B_{\mathsf{ks}} + B_{\mathsf{rs}}$ for $B_{\mathsf{ks}} = 8\sigma N/\sqrt{3}$.*

A sufficiently large scaling factor $\Delta_{bs} = \mathcal{O}(q)$ is multiplied during the CoeffToSlot step in order to keep the precision of values in slots. Note that Δ_{bs} differs from the scaling factor of the message Δ. From Lemma 4.13, the total error in the CoeffToSlot step is $\mathcal{O}(B_{\mathsf{rs}})$ when a sufficiently large P is chosen [4].



(a) Approximation error for various message width ϵ



(b) Approximation error for various degree n

Figure 4.2: Maximum value of the error $\log_2(|[t]_p - p_o(t)|)$ for the proposed method and previous method ($K = 12$).

In the EVALMOD step, each component of the corresponding plaintext slot contains $t_j + e_j$ for some small error e_j such that $|e_j| \leqslant \mathcal{O}(B_{\mathtt{rs}})$. An approximate polynomial $p_o(t_j)$ is evaluated with scaling factor Δ_{bs}, and thus the approximate error is given as

$$\begin{aligned}
&\Delta_{bs}\left|\left[\frac{t_j}{q}\right]_q - p_o\left(\frac{t_j+e_j}{q}\right)\right| \\
&\quad\leqslant \Delta_{bs}\left|\left[\frac{t_j}{q}\right]_q - \left[\frac{t_j+e_j}{q}\right]_q\right| + \Delta_{bs}\left|\left[\frac{t_j+e_j}{q}\right]_q - p_o\left(\frac{t_j+e_j}{q}\right)\right| \\
&\quad\leqslant \Delta_{bs}\cdot\frac{|e_j|}{q} + \Delta_{bs}\max_t\left|[t]_q - p_o(t)\right|.
\end{aligned}$$

In order to bound the error in the EVALMOD step to $\mathcal{O}(B_{\mathtt{rs}})$, it should be guaranteed that

$$\max_t |[t]_q - p_o(t)| < \frac{|e_j|}{q}. \tag{4.11}$$

When the error in the EVALMOD step is bounded to $\mathcal{O}(B_{\mathtt{rs}})$, we have the error bound after the SLOTTOCOEFF step as $\mathcal{O}(\sqrt{N}\cdot B_{\mathtt{rs}})$ [19].

Note that from Lemma 4.12, the error in bootstrapping is independent from the scaling factor of message Δ and bounded to $\mathcal{O}(N\sqrt{h})$. Thus, the plaintext precision is proportional to $\log\Delta$, where Δ determines $|m|$. Combining (4.10) and (4.11), q is restricted to be greater than $\mathcal{O}(m^{3/2})$ in all the methods proposed so far [19–21]. Considering that a scaling factor $\Delta_{bs} = \mathcal{O}(q)$ is used in the bootstrapping, the level consumption is given as $\mathcal{O}(m^{3/2})$. Thus, the previous methods do not scale well for applications that require accurate computations.

However, by using the proposed method, the upper bound from (4.10) does not exist. Hence, the level loss in bootstrapping is roughly proportional to $\mathcal{O}(m)$ rather than $\mathcal{O}(m^{3/2})$. This is one of the advantages of the proposed method, and it overcomes the limitations of the existing methods. The more precise calculations are required, the greater the gain we have.

Various factors, such as the number of slots, affect plaintext precision. Hence, the plaintext precision is obtained using the numerical methods, and it can be used to

Table 4.5: Probability mass function of I.

i	$\Pr_I(i)$	i	$\Pr_I(i)$
0	0.3343019	± 6	0.00342346
± 1	0.13919181	± 7	0.00091685
± 2	0.09646158	± 8	0.00020066
± 3	0.05556329	± 9	0.00003567
± 4	0.02655144	± 10	0.00000511
± 5	0.01049854	± 11	0.00000059

determine the parameters as in [19,21]. Using the proposed method, relatively small q can be used, and thus in some cases, it may leave more levels after bootstrapping.

4.7 Implementation of the Proposed Method and Performance Comparison

The proposed method of minimizing error variance is implemented on HEAAN and SEAL, which can be widely applied to many different applications. I compare the experimental error of the Chebyshev polynomial of the first kind in the case of applying the reordering method proposed in the dissertation and not. Finding error variance-minimizing approximate polynomial is implemented by SageMath. Recently, I implemented the bootstrapping algorithm for SEAL, which will be released soon. The bootstrapping using the proposed approximate polynomial is implemented by modifying HEAAN and SEAL. In this section, several implementation results and comparisons for the bootstrapping algorithms of the state-of-the-art methods are also presented.

4.7.1 Error Variance Minimization

In this subsection, I show how to find the Chebyshev polynomial basis with smaller errors in HEAAN and SEAL. The input for bootstrapping is $t = r + i$, and for the worst-case assumption, I assume that r follows the uniform distribution $[-\epsilon, \epsilon]$, where $\epsilon = 2^{-10}$. The probability mass function of I is given in Table 4.5.

As the domain of Chebyshev polynomial is $[-1, 1]$, $\{T_i(t/K)\}_{0 \leqslant i \leqslant n}$ is used as the polynomial basis, and it can be seen that the distribution of the input t/K is concentrated at zero. As shown in Subsection 4.2.4, multiplication between two even degree terms should be avoided when we calculate the even degree terms. Fig. 4.3 shows the variance of error in $T_i(t)$ for even i's. It can be seen that error in encrypted data can be greatly reduced by only changing the order of calculating $T_i(t)$. In particular, for $T_{74}(t)$, the variance of error for the proposed method becomes smaller by $1/1973$ compared to that without minimization.

4.7.2 Weight Constant and Minimum Error Variance

In Subsection 4.4.5, I discussed analytic solution and numerical method for optimal approximate polynomial. In this subsection, the above methods are implemented and verified, together with the theoretical values of approximation error and the amplified basis error. Besides, I confirm that although the numerical method finds a polynomial that is very close to the value obtained through Subsection (4.4.5), it has a slightly larger error than this.

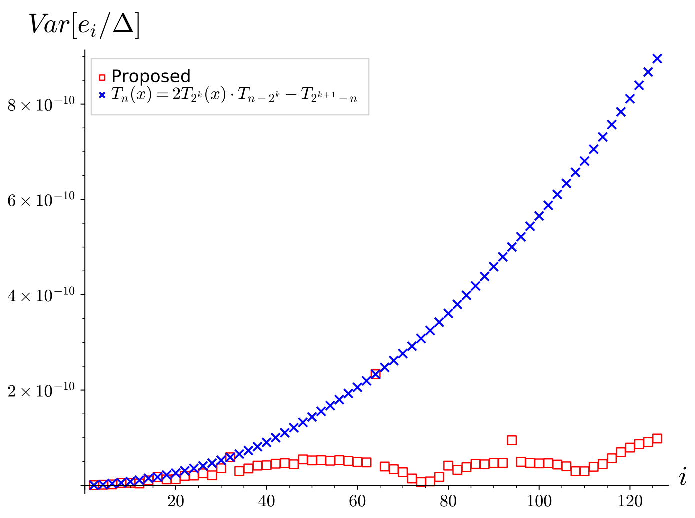
The approximate polynomial minimizing $Var[e_{\mathsf{aprx}}] + w \cdot \|\boldsymbol{c}\|_2^2$ can be found for a given weight constant w. For the scaling factor $\Delta_{\mathsf{bs}} = q$ of bootstrapping, the variance of approximation error in the slot after EVALMOD is given as

$$q^2 \cdot Var[e_{\mathsf{aprx}}].$$

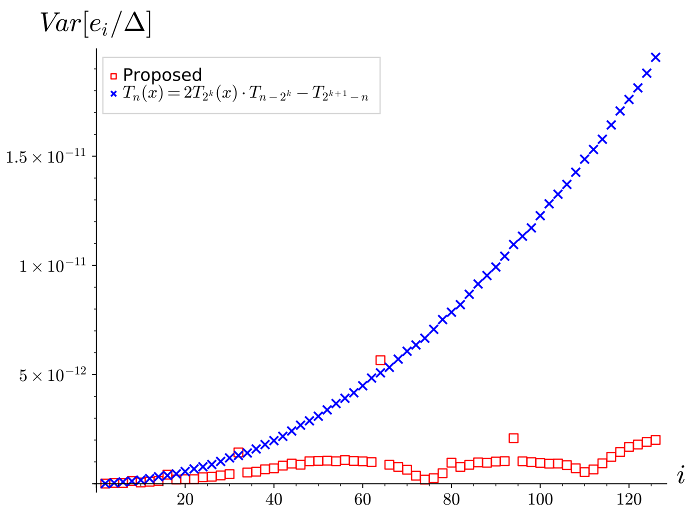
The variance of amplified basis errors by coefficients are given as

$$E_p = (\mathbf{L}^{-1}\boldsymbol{c})^{\mathsf{T}}\mathbf{H}(\mathbf{L}^{-1}\boldsymbol{c}).$$

Finally, in the SLOTTOCOEFF step, the plaintext vectors are multiplied with the first half of the encoding matrix $\mathbf{U}$ and their diagonal vectors have the magnitude of one.



(a) Result in HEAAN, where $\epsilon = 2^{-10}$ and $q = 2^{40}$



(b) Result in SEAL, where $\epsilon = 2^{-5}$ and $q = 2^{45}$

Figure 4.3: Variance of error in $T_i(t)$ for even i using HEAAN and SEAL with various parameters.

Hence, the variance of the approximation error is multiplied by the number of slots n. In summary, the total errors by bootstrapping are given as

$$n \cdot \left(q^2 \cdot Var[e_{\mathsf{aprx}}] + E_p\right).$$

The experimental results, the theoretical variance of the approximate error, and the basis error are shown in Fig. 4.4. The default parameters in HEAAN library are used for the experiment: $N = 2^{16}, h = 64, \sigma = 3.2$ and the number of slots is $n = 2^3$. The experimental result is averaged over 256 experiments, where the scaling factors are $\Delta = 40, 45$, the number of slots l is 8 and $\epsilon = 2^{-10}$ in this experiment. Therefore, $q = 50, 55$ for $\Delta = 40, 45$, respectively.

The blue lines with triangular legend show the error by polynomial approximation as

$$n \cdot q^2 \cdot Var[e_{\mathsf{aprx}}].$$

The green lines with x mark legend show the amplified basis errors as

$$n \cdot E_p$$

and the red lines with square legend are for the mean square of errors obtained by experiments. The gray dot line is the variance of bootstrapping error achieved by using the error variance-minimizing approximate polynomial of the same degree. For the worst-case assumption, we assume that m is distributed uniformly at random. However, we use m that is not uniformly distributed. Therefore, the total error can further be reduced when the distribution of m is known.

In Fig. 4.4, the sum of blue lines with triangular legend and green lines with x mark legend meets the red lines with the square legend. Thus, it shows that the theoretical derivation and experimental results are agreed upon. It can also be seen that it is possible to obtain an approximate polynomial with a small error even by using the numerical method, but the error is slightly larger than that of an accurately calculated approximate polynomial. It is noted that the variance of the rescaling error is $\frac{(h+2)2n}{12}$

and the optimal w is close to $\frac{(h+2)2n}{12q^2} \approx 2^{6.4}$ because each element of Chebyshev polynomials has error mainly introduced by rescaling.

4.7.3 Comparison of the Proposed Method With the Previous Methods

Minimized Error Variance by the Proposed Method and Error Variance of Minimax Polynomial

The best-known approximation method for the CKKS bootstrapping so far is the modified Remez algorithm [25]. The modified Remez algorithm is an iterative method that finds the minimax approximate polynomial for piece-wise continuous functions such as $f_{\mathsf{mod}}(t)$. Using the modified Remez algorithm, the minimax approximate polynomial for $f_{\mathsf{mod}}(t)$ can be found. The minimax approach is reasonable when the input distribution is unknown. However, in the CKKS bootstrapping, the input distribution is partially known; the probability mass function of I follows a distribution similar to the Irwin–Hall distribution. I use the worst-case assumption that r is uniformly distributed when I derive the variance-minimizing approximate polynomial. However, in the experiment of finding the variance of approximate error for the given approximate polynomials for both methods, I let r follow the Gaussian distribution, not the uniform distribution, because the message polynomial is assumed to be the resultant value of compound operations and summations. In other words, by assuming the distribution of the message polynomial differently for finding the variance-minimizing approximate polynomial and actually calculating the variance, the experiment is conducted in an unfavorable environment to the proposed method. It is noteworthy that a lower error variance than minimax polynomial is achieved when using the proposed method, despite the worst-case assumption. It is shown that the distribution of I has a dominant effect on the error.

It is shown in Fig. 4.5 that the variance of approximation error is smaller when the error variance-minimizing polynomial is used, as expected. This means that the proposed method reduces the approximation error during bootstrapping. As the magnitude of the coefficients of the approximate polynomial cannot be controlled in the modified Remez algorithm, the approximate polynomials for both methods are compared without controlling the magnitude of coefficients in Fig. 4.5. It is noted here that the variance of approximation errors shown in Fig. 4.5 is not practical in the CKKS scheme due to the basis error and the enormous coefficients of the approximate polynomials. However, it is possible to reduce the magnitude of the coefficients of the approximate polynomial in the proposed method with slightly increased error variance. In contrast, the previous methods cannot control the magnitude of its coefficients, and thus the use of the double-angle formula is essential, which results in a large error variance and more depth.

Experimental Result of Bootstrapping Error

The experimental result of bootstrapping errors with practical approximate polynomials using various methods are compared in Tables 4.6 and 4.7 for HEAAN and SEAL, respectively. In those tables, the proposed variance-minimizing polynomial directly approximates $f_{\mathsf{mod}}(t)$ and the lazy rescaling proposed in this dissertation is applied to the proposed method. The polynomials of previous methods approximate the scaled sine function, and the double-angle formula is used. For a very low error achieved by a method in [25], the composite function method is applied, and the composition of $\frac{1}{2\pi}\arcsin(t)$ consumes at least two more depth. The scale-invariant evaluation [23,62] is applied to the previous methods implemented in SEAL.

In Table 4.6, the additional error introduced by the bootstrapping implemented in HEAAN is presented, including SUBSUM, COEFFTOSLOT, EVALMOD, and SLOTTOCOEFF. In the table, it is shown that the proposed method achieves the average precision of 29.87 bits with only modulus consumption of 400, while the previous method in [25] achieves 29.18 bits with modulus consumption of 550, where modulus consumption is defined as $\log_2(q)$ times depth. Therefore, the proposed method restores about 4 more levels with less error through bootstrapping, compared to the

Table 4.6: Comparison of the minimum variance of bootstrapping error of the proposed variance-minimizing polynomial and prior arts. It is noted that the depth for modulus reduction is displayed only; depth for CoeffToSlot and SlotToCoeff is not counted. The variance of the error is obtained by 256 samples of experiments implemented in HEAAN. I try several trials for each method and display the polynomial with the least error in this table.

Algorithm	$\log_2 p$	$\log_2 \epsilon$	depth	Modulus consumption	Deg. of polynomial			Variance of bootstrapping error	Average precision (bits)
					deg. cos	# of double angle formula	deg. arcsin		
Proposed	40	-8	8	384		239		$2^{21.51}$	29.57
		-10	7	350		111		$2^{41.06}$	19.79
			8	400		239		$2^{20.91}$	**29.87**
		-12	7	364		111		$2^{33.11}$	23.77
			8	416		223		$2^{20.55}$	30.05
Taylor [19]	40	-10	11	594	7	8	-	$2^{44.38}$	18.13
Han et al. [20]	40	-8	8	384	61	2	-	$2^{45.94}$	17.36
		-10	8	400	63	2	-	$2^{37.27}$	21.69
		-16	7	392	63	1	-	$2^{28.56}$	26.05
Method in [25]	40	**-10**	8	400	63	2	1	$2^{40.24}$	20.20
			10	500	63	2	3	$2^{22.55}$	29.05
			11	550	63	2	5	$2^{22.29}$	**29.18**

previous methods. The default parameters of HEAAN, $N = 2^{16}, h = 64$, and $\sigma = 3.2$ are used and the number of slots is $n = 2^3$.

In Table 4.7, the additional error introduced by the bootstrapping implemented in SEAL is presented. It is shown in this table that the proposed method achieves the average precision of $XX.XX$ with modulus consumption of 440 while the previous method achieves $XX.XX$ with modulus consumption of 550. By modifying SEAL, the same parameters as HEAAN are used.

In conclusion, we can see that the proposed method achieves the least error with the least modulus consumption from the implementation results. As a result of the experiment, it is shown that only depth eight is required to achieve the error that could have been achieved using the previous method with depth 11 [25]. Alternatively, when

Table 4.7: Comparison of the minimum variance of bootstrapping error of the proposed variance-minimizing polynomial and prior arts. It is noted that the depth for modulus reduction is displayed only; depth for CoeffToSlot and SlotToCoeff is not counted. The variance of the error is obtained by 256 samples of experiments implemented in SEAL. I try several trials for each method and display the polynomial with the least error in this table.

Algorithm	$\log_2 p$	$\log_2 \epsilon$	depth	Modulus consumption	Deg. of polynomial			$\log_2 q^2 \cdot Var[e_{\mathsf{aprx}}]$	$\log_2 E[\lvert e_{\mathsf{aprx}} \rvert]/p$
					deg. cos	# of double angle formula	deg. arcsin		
Proposed	45	-10	7	385	111			48.80	-20.92
		-10	**8**	**440**	239			17.19	**-36.73**
		-8	8	424	239			16.87	-36.89
		-6	8	408	239			21.58	-34.53
Han et al. [20]	45	-8	8	424	55	2	-	55.25	-17.70
		-10	8	440	55	2	-	56.94	-16.86
		-12	8	456	55	2	-	57.09	-16.78
Method in [25]	45	-10	8	440	55	2	1	56.75	-16.88
			10	**550**	55	2	3	56.89	**-16.95**
		-3	12	576	55	2	13	56.62	-17.02
	57	-3	12	720	55	2	13	56.39	-29.13

using the same depth of 8, the average precision of the proposed method is much higher than that of the previous methods [19, 25].

The use of the double-angle formula and the composite function method is for the magnitude of the coefficients and the fast evaluation. It is worth noting that all the previous techniques, such as the double-angle formula and composition method for efficient evaluation, can also be applied to the proposed method; these techniques are not for just minimax approximate polynomials. It is evident that the use of the proposed error variance-minimizing polynomial for sine function and inverse sine function will also reduce the error compared to the minimax approach in [25]. One might argue that the bootstrapping algorithm of the proposed method may fail because the approximation error is too large where the probability is very low. In fact, it is not; by experiments, I check that the maximum approximation error is also small for the error variance-minimizing polynomial.

Fundamental Error of Baby-Step Giant-Step Algorithm

This subsection discusses a very loose lower bound of bootstrapping error, which is the constant term of bootstrapping error, and shows that the proposed method is very close to the lower bound. As the lazy rescaling method is applied, the rescaling is performed after a baby-step polynomial is obtained. In other words, we have ciphertext $\boldsymbol{c}_j$'s with scaling factor close to Δ^2, which are encryptions of $T_j(t)$'s. Then, the rescaling is not performed to $\boldsymbol{c}_j$ and coefficients are multiplied as

$$\boldsymbol{c}_j' \leftarrow \mathsf{cMult}(\boldsymbol{c}_j, d_{i,j}; \Delta).$$

Then, $\boldsymbol{c}_j'$'s are added up and rescaled by one level. Let $\boldsymbol{c}_{p_i}$ denote the summation of $\boldsymbol{c}_j'$ and then it is an encryption of $p_i(t)$, where $\boldsymbol{c}_{p_i}$ includes amplified basis error and additional basis error.

Then the giant-step such as

$$\mathsf{Mult}(\mathsf{RS}(\boldsymbol{c}_{p_i}), \mathsf{RS}(\boldsymbol{c}_k)) + \boldsymbol{c}_{p_{i+1}}$$

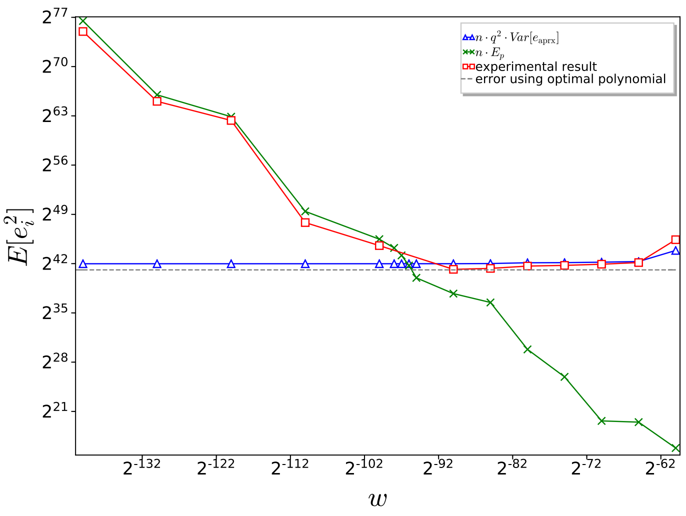
is performed. Of course, $\mathsf{RS}(\boldsymbol{c}_{p_i})$, $\mathsf{RS}(\boldsymbol{c}_k)$, and $\boldsymbol{c}_{p_{i+1}}$ have independent rounding errors, whose variances are $\frac{(h+1)\cdot 2n}{12}$. Although $E[p_i(t)^2]$ is usually greater than one, but for a very loose bound, I let $E[p_i(t)^2] = E[T_k(t)^2] \leqslant 1$, and then the rounding errors are maintained and added. It is noted that the number of rescaling cannot further be reduced by the commutative property since the level and a scaling factor of $\boldsymbol{c}_{p_i}$ and $\boldsymbol{c}_k$ are the same; these errors are independent of the coefficients $\boldsymbol{d}$, in other words, it is the constant term of modulus reduction error. There are $2^l - 1$ such operations in the giant-step.

The error in ModRaias is further amplified by SlotToCoeff. During SlotToCoeff, key switching makes the $2n$ shifted copy of the ciphertext (introduces rounding error), and the slot values are multiplied by ζ_i^j's, whose magnitudes are one, and added up.

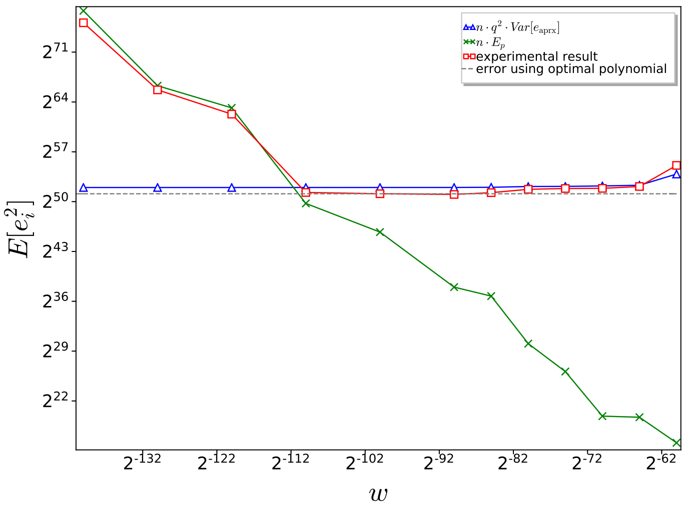
There are $3 \times (2^l - 1)$ independent rounding errors that occur during the baby-step giant-step algorithm, and one more rounding error occurs during key switching. There are n copies of such ciphertexts, and they are all added up. Roughly, the variance of error introduced by such rounding is given as

$$\left(\frac{(h+1)\cdot 2n}{12} \times (3 \times (2^l - 1) + 1)\right) \times 2n \approx 2^{14.9},$$

where $n = 2^3$ and $l = 3$ in the experiment. I note that this is a very loose lower bound of error, but the proposed method achieves an error of only 2.8 bits greater than this lower bound on average.



(a) Theoretical variance of errors and experimental result when scaling factor is 2^{40}



(b) Theoretical variance of errors and experimental result when scaling factor is 2^{45}

Figure 4.4: Theoretical energy of approximation error, amplified basis error, and energy of experimental results, implemented in HEAAN. A polynomial of degree 81 is used. The gray dot line is the variance of bootstrapping error that is achieved by using the polynomial with coefficients that $\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} \left(Var[q \cdot e_{\mathsf{aprx}}] + E_p \right)$, which is the lower bound of bootstrapping error.

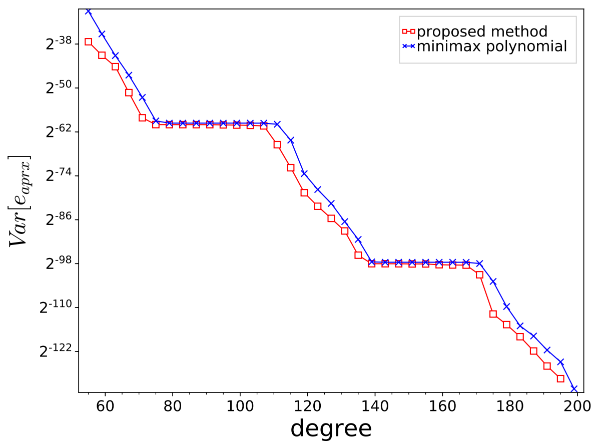


Figure 4.5: Comparison of the minimum achievable variance of approximation error of the proposed method and that of the minimax polynomial, when $\epsilon = 2^{-10}$ and no controlling of coefficients of both approximate polynomials.

Chapter 5

Efficient Code-Based Signature Scheme and Cryptanalysis of Code-Based Cryptosystems

5.1 Introduction

In this chapter, I propose an efficient code-based signature scheme and cryptanalysis of code-based cryptosystems. Especially, the pqsigRM, a first-round candidate of PQC standardization by NIST and its modification, are included. By using the proposed modified RM codes and their decoding, one can find a small-Hamming-weight error vector for any given received vector. Hence, it reduces the required iteration in code-based signature schemes, such as the signature scheme proposed by Courtois, Finiasz, and Sendrier (CFS). The proposed signature scheme has a small parameter size. In addition, I propose here that one of the IKKR cryptosystems is equivalent to the McEliece cryptosystem and cryptanalysis for the other two. The implementation results show that the proposed attack algorithm is efficient so that it performs faster than the legitimate decryption.

5.2 Modified Reed–Muller Codes and Proposed Signature Scheme

In this section, I propose new codes, their decoder, and a signature scheme that uses these codes and decoders. The proposed code essentially has a $(U, U+V)$-code as its subcode, and recursively, U and V are also $(U, U+V)$-codes. This recursive structure allows the decoding of any given vector in $\mathbb{F}_2^n$. Then, we can find an error vector with a small Hamming weight for any given syndrome corresponding to the received vector. Starting from $(U, U+V)$-codes, we replace certain rows and append random rows on the generator matrix of $(U, U+V)$-codes. Thus, these codes are no longer $(U, U+V)$-codes. However, they have a $(U, U+V)$-subcode and can use the decoder for $(U, U+V)$-codes.

5.2.1 Partial Permutation of Generator Matrix and Modified Reed–Muller Codes

New codes named modified RM codes are defined in this section. I first present the core of the proposed codes, which is a $(U, U+V)$-code. Subsequently, I describe which rows are replaced or appended to the generator matrix. The rationale for these operations is provided in Section 5.4.

For a code $\mathcal{C}$, I define its hull by the intersection of the code and its dual, in other words, $hull(\mathcal{C}) = \mathcal{C} \cap \mathcal{C}^{\perp}$. The proposed $(U, U+V)$-code is designed to have a high-dimensional hull, where $dim(U^{\perp} \cap V)$, dimenstion of $U^{\perp} \cap V$, is large. In general, for a $(U, U+V)$-code $\mathcal{C}$, a codeword $(\boldsymbol{u}|\boldsymbol{u}+\boldsymbol{v}) \in hull(\mathcal{C})$ satisfies $\boldsymbol{v} = \boldsymbol{u}^{\perp}$ and $\boldsymbol{u}+\boldsymbol{v} = \boldsymbol{v}^{\perp}$, where $\boldsymbol{u} \in U$ and $\boldsymbol{v} \in V$. Hence, when $U^{\perp} \cap V = \{\mathbf{0}\}$, $hull(\mathcal{C})$ has only $(\boldsymbol{u}|\boldsymbol{u})$ codewords, and this may reveal the secret key. To avoid this, the proposed code is designed so that $dim(U^{\perp} \cap V)$ is large.

For convenience, I focus on the generator matrix. First, I construct the generator matrix $\mathbf{G}_{(r,m)}$ of an RM code and then permute its submatrices. An example is shown

$\mathbf{G}_{(r,m-2)}^{\sigma_p^1}$	$\mathbf{G}_{(r,m-2)}^{\sigma_p^1}$	$\mathbf{G}_{(r,m-2)}^{\sigma_p^1}$	$\mathbf{G}_{(r,m-2)}^{\sigma_p^1}$
$\mathbf{0}$	$\mathbf{G}_{(r-1,m-2)}$	$\mathbf{0}$	$\mathbf{G}_{(r-1,m-2)}$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{G}_{(r-1,m-2)}$	$\mathbf{G}_{(r-1,m-2)}$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{G}_{(r-2,m-2)}^{\sigma_p^2}$

Figure 5.1: Generator matrix of partially permuted RM code with parameter (r, m).

in Figure 5.1, where σ_p^1 and σ_p^2 denote two independent partial permutations that randomly permute only p out of $n/4$ columns. As will be explained in Section 5.5.2, p is related to the decoding performance. To generate σ_p^1 and σ_p^2, p column indices are randomly selected from the index set $\{0, 1, \ldots, n/4 - 1\}$, and the selected indices are randomly permuted, whereas the others are not. Then, σ_p^1 is used to permute the submatrices corresponding to $\mathbf{G}_{(r,m-2)}$'s in the first $dim(\mathrm{RM}_{(r,m-2)})$ rows, and σ_p^2 is used to permute the submatrix corresponding to $\mathbf{G}_{(r-2,m-2)}$ in the last $dim(\mathrm{RM}_{(r-2,m-2)})$ rows, as shown in Figure 5.1. The codes generated by the generator matrix in Figure 5.1 are called partially permuted RM codes. It should be noted that, unlike in the case of code-based cryptographic algorithms, we permute submatrices of the generator matrix rather than the entire matrix here. It is noted that the entire matrix should also be permuted to design a signature scheme. This will be discussed on the key generation in Section 5.2.3.

$dim(U^\perp \cap V)$ is large for the following reasons. Let $\mathbf{G}_U$ and $\mathbf{G}_V$ denote the generator matrices of U and V, respectively:

$$\mathbf{G}_U = \begin{bmatrix} \mathbf{G}^{\sigma_p^1}_{(r,m-2)} & \mathbf{G}^{\sigma_p^1}_{(r,m-2)} \\ \mathbf{0} & \mathbf{G}_{(r-1,m-2)} \end{bmatrix},$$

$$\mathbf{G}_V = \begin{bmatrix} \mathbf{G}_{(r-1,m-2)} & \mathbf{G}_{(r-1,m-2)} \\ \mathbf{0} & \mathbf{G}^{\sigma_p^2}_{(r-2,m-2)} \end{bmatrix}.$$

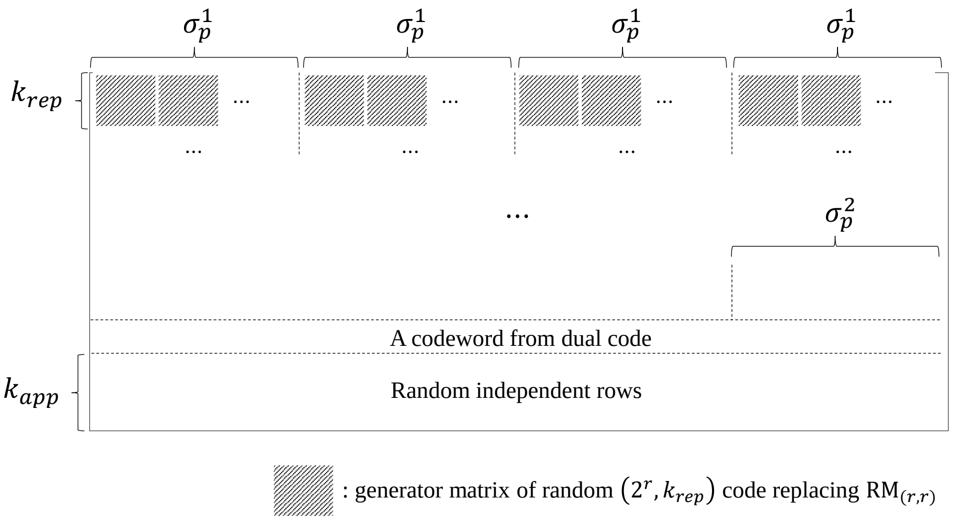
Then, the generator matrix of the dual code of U is

$$\mathbf{G}_U^{\perp} = \begin{bmatrix} \mathbf{G}^{\perp\sigma_p^1}_{(r,m-2)} & \mathbf{0} \\ \mathbf{G}^{\perp}_{(r-1,m-2)} & \mathbf{G}^{\perp}_{(r-1,m-2)} \end{bmatrix}.$$

Thus, $U^{\perp} \cap V$ has a subcode that is the intersection of the codewords generated by $\begin{bmatrix} \mathbf{G}_{(r-1,m-2)} & \mathbf{G}_{(r-1,m-2)} \end{bmatrix}$ and the codewords generated by $\begin{bmatrix} \mathbf{G}^{\perp}_{(r-1,m-2)} & \mathbf{G}^{\perp}_{(r-1,m-2)} \end{bmatrix}$. Its dimension is $\min(dim(\mathrm{RM}_{(r-1,m-2)}, dim(\mathrm{RM}_{(m-r-2,m-2)}))$, as the dual of $\mathrm{RM}_{(r,m)}$ is equal to $\mathrm{RM}_{(m-r-1,m)}$ and $\mathrm{RM}_{(r',m)} \subseteq \mathrm{RM}_{(r,m)}$, where $r' \leqslant r$.

With the partially permuted RM codes, the received vector and the syndrome have the same parity, causing the signature leak. Thus, the generator matrix in Figure 5.1 should be further modified. That is, some rows are replaced with repetitions of random codewords and random rows are appended to the generator matrix. Considering $\mathbf{G}_U$, it is also an $(U, U+V)$-code, which can similarly be divided into (permuted) $(U, U+V)$-codes. By repeating this process 2^{m-r} times, the rows of the partially permuted RM code consist of the 2^{m-r} repeated generator matrices of $\mathrm{RM}_{(r,r)}$, which are $2^r \times 2^r$ identity matrices. Then, $\mathrm{RM}_{(r,r)}$ is replaced by a repeated random $(2^r, k_{rep})$ code such that its dual code has at least one non-zero codeword with odd Hamming weight.

We now append random independent rows to the generator matrix. One row to be appended is a random codeword of the dual code. This should be independent of the existing rows; i.e., it should not belong to the hull of the code. Furthermore, it should be verified that the hull has codewords with Hamming weight that is not a multiple of four as a result of appending this row. The others are k_{app} random independent



σ_p^1
σ_p^1
σ_p^1
σ_p^1
k_{rep}
σ_p^2
A codeword from dual code
k_{app}
Random independent rows
: generator matrix of random $(2^r, k_{rep})$ code replacing $\mathrm{RM}_{(r,r)}$

Figure 5.2: Generator matrix of modified RM code.

vectors, including at least one vector of odd Hamming weight. These k_{app} vectors are independent of the partially permuted RM codes and independent of each other.

After all these modifications, the resulting code is called a modified RM code. An example of its generator matrix is given in Figure 5.2.

5.2.2 Decoding of Modified Reed–Muller Codes

Unlike the Niederreiter cryptosystem and THE CFS signature scheme, it is required to find an error vector whose Hamming weight is greater than the error correction capability. Hence, there may exist several solutions $\boldsymbol{e}$ satisfying $\mathbf{H}\boldsymbol{e}^T = \boldsymbol{s}^T$ and $\mathrm{wt}(\boldsymbol{e}) \leqslant w$ for a given syndrome $\boldsymbol{s}$. Such decoding can be achieved by the modified Prange decoder using the $(U, U+V)$ structure, as in the signature schemes in [32,50]. However, in this section, a new decoder is proposed that uses the recursive structure of the subcode of modified RM codes, and it achieves better performance than the modified Prange decoder. In other words, it finds error vectors whose Hamming weights are less than the result in [50]. This results in the smaller parameters, considering attacks as in [70].

In addition to the decoding performance, a major difference between the proposed decoder and the modified Prange decoder is their input. The input of the modified Prange decoder used in [32] and [50] is a syndrome vector. In contrast, the input of the proposed decoder is an n dimensional vector $\boldsymbol{r}$ satisfying $\mathbf{H}\boldsymbol{r}^T = \boldsymbol{s}$, which is called received vector in coding theory, and the decoder outputs codewords close to the received vector. An error vector with a small Hamming weight is obtained by subtracting the output from the received vector. Even if two different received vectors in the same coset are given, the proposed decoder can return different outputs. Besides, as the input of the decoder is the received random vector, decoding can be performed even if random rows are appended to the generator matrix.

As stated in the previous section, random rows (one from the dual code and the others being k_{app} independent random vectors) are appended to the generator matrix of the partially permuted RM codes. Let $\mathcal{C}_{app}$ be the code spanned by the added $k_{app}+1$ rows. The number of codewords increases by $2^{k_{app}+1}$ times when rows are appended by adding codewords of $\mathcal{C}_{app}$ to each $(U, U+V)$-codeword. Choosing a codeword of $\mathcal{C}_{app}$ (including $\mathbf{0}$), subtracting it from the received vector $\boldsymbol{r}$, decoding it, and adding the subtracted codewords back is the decoding process when rows are appended. Thus, the code is decodable even if arbitrary random codes are appended to its generator matrix.

Hence, it suffices to explain the decoding algorithm for the $(U, U+V)$-subcode of a modified RM code. This decoding basically follows the recursive decoding of RM codes [58]. The difference is the partial permutation and the replacement of $\mathrm{RM}_{(r,r)}$. Considering the decoding proposed in [58], we have $\boldsymbol{c} = (\boldsymbol{u}|\boldsymbol{u}+\boldsymbol{v})$ for all $\boldsymbol{c} \in \mathrm{RM}_{(r,m)}$, where $\boldsymbol{u} \in \mathrm{RM}_{(r,m-1)}$ and $\boldsymbol{v} \in \mathrm{RM}_{(r-1,m-1)}$. $\mathrm{RM}_{(r,m-1)}$ and $\mathrm{RM}_{(r-1,m-1)}$ are also $(U, U+V)$-codes, except for $r = 0$ or $r = m$. Here, if the code corresponding to $\boldsymbol{u}$ or $\boldsymbol{v}$ is replaced with a code other than the RM code and the decoding of the replaced code can be performed appropriately, the entire code $\boldsymbol{c}$ can also be decoded [71].

When the subcode of the RM code is replaced with its permutation, the entire code can also be decoded by slightly modifying the recursive decoding. Moreover, no decoding failure occurs because the recursion eventually reaches $\mathrm{RM}_{(0,m')}$, $\mathrm{RM}_{(r',r')}$, or the $(2^r, k_{rep})$ code to replace $\mathrm{RM}_{(r,r)}$ and there exists polynomial-time MD decoder for these codes. Even the $(2^r, k_{rep})$ random code is MD decodable in constant time because it is a small code. To handle partial permutations, when the code is decodable, it uses the fact that the permutation is always decodable if the permutation is known. Depermutation and decoding followed by permutation is the decoding process for permuted codes.

In general, the output distribution of decoding is crucial for security. Thus, I also propose a randomized decoding method, the output of which is almost uniformly distributed. With the algorithm described above, a random decoder can easily be designed. Algorithm 6 summarizes the randomized decoding. It is easy to find a received vector (regardless of its Hamming weight) for any given syndrome; a coset element corresponding to the syndrome is randomly selected. This is given to the decoder as input. Finally, the decoder finds a different error vector with a small Hamming weight for different inputs.

5.2.3 Proposed Signature Scheme

Herein, the proposed modified pqsigRM signature scheme using the codes in the previous section is presented. Its decoding algorithm is presented in Section 5.2.2.

Key Generation

Let $\mathbf{G}$ be the generator matrix of a modified (n, k) RM code, and $\mathbf{H}$ be the parity check matrix. Let $\mathbf{S}$ be an $(n-k) \times (n-k)$ random non-singular matrix and $\mathbf{Q}$ be an $n \times n$ random permutation matrix. Then, the public key is $\mathbf{H}' = \mathbf{SHQ}$, and the secret keys are $\mathbf{H}$, $\mathbf{S}$, and $\mathbf{Q}$.

Signing

To sign a given message $\boldsymbol{m}$, we randomly select a coin $\boldsymbol{i}$ from $\{0,1\}^{\lambda_0}$. A binary vector $\boldsymbol{s} = h(h(\boldsymbol{m}|\mathbf{H}')|\boldsymbol{i})$ is calculated, where $h : \{0,1\}^* \rightarrow \{0,1\}^{n-k}$ is a cryptographic hash function. Our goal is to find the error vector $\boldsymbol{e}$ satisfying $\mathbf{H}'\boldsymbol{e}^T = \mathbf{SHQ}\boldsymbol{e}^T = \boldsymbol{s}$. Let $\boldsymbol{s}' = \mathbf{S}^{-1}\boldsymbol{m}$.

Performing the decoding as in Algorithm 6, we find an error vector $\boldsymbol{e}'$ satisfying $\mathbf{H}\boldsymbol{e}'^T = \boldsymbol{s}'$. If $\mathrm{wt}(\boldsymbol{e}') \leqslant w$, we compute $\boldsymbol{e}^T = \mathbf{Q}^{-1}\boldsymbol{e}'^T$, and the signature is then given as $(\boldsymbol{m}, \boldsymbol{e}, \boldsymbol{i})$.

Verification

If $\mathrm{wt}(\boldsymbol{e}) \leqslant w$ and $\mathbf{H}'\boldsymbol{e}^T = h(h(\boldsymbol{m}|\mathbf{H}')|\boldsymbol{i})$, we return ACCEPT; otherwise, we return REJECT.

The key generation, signing, and verification processes are summarized in Algorithm 7. For simplicity, let $\mathbf{H}$ represent all the secrets such as partial permutations σ_p^1 and σ_p^2, appended rows, and replaced codes. It should be noted that in the signing process, we choose a random coset element and perform ModDec($\cdot$). As ModDec($\cdot$) returns different outputs for different inputs even in the same coset, we can achieve randomized decoding. The output distribution of this randomized decoding output is analyzed in Section 5.4. We add a salt λ_0 to obtain a tight security proof.

5.3 Security Analysis of Modified pqsigRM

In this section, the security of the proposed modified pqsigRM will be analyzed. I will consider the best-known algorithms for solving DOOM. Thereafter, I will discuss the resistance of the proposed signature scheme against key substitution attacks. Finally, it will be proved that the modified pqsigRM is EUF-CMA secure.

As the public key of the proposed signature scheme is a modification of an RM code, one may consider key recovery attacks on RM codes, such as Minder–Shokrollahi [72] and Chizhov–Borodin [73] attacks, as well as square code attacks [74]. However, owing to the partial permutation as well as the appending and replacement of codewords in the generator matrix, these attacks cannot be adopted here. Table 5.1 shows the comparison between the proposed modified pqsigRM and the original pqsigRM.

Table 5.1: Comparison of the proposed modified pqsigRM and the original pqsigRM

	Modified pqsigRM	Original pqsigRM [37]
Key generation method	partial permutation, row appending and replacement	column puncturing and insertion
Randomized decoding	yes	no
Attack	none	finding puncturing with hull

5.3.1 Decoding One Out of Many

Information set decoding is a brute-force attack method that finds an error vector $\boldsymbol{e}$ such that $\mathbf{H}\boldsymbol{e}^T = \boldsymbol{s}$ and $\mathrm{wt}(\boldsymbol{e}) \leqslant w$, where Stern improved the attack complexity in [75]. It has been extensively studied, and Dumer's algorithm [76] as well as more involved variants in [77, 78] have been proposed.

In the variants of the CFS signature scheme, there are several hash queries. Therefore, to launch a forgery attack, it suffices to find an error vector with a small Hamming weight for any of the syndromes. Hence, the decoding problem DOOM given below is adequate for a tight security proof. The usual FDH proof for existential forgery using syndrome decoding would require a work factor $\geqslant q_{\mathcal{H}} \cdot 2^{\lambda}$, where $q_{\mathcal{H}} \leqslant 2^{\lambda}$ is the number of hash queries. However, with DOOM, the work factor is required to be $\geqslant 2^{\lambda}$. Although the work factor of DOOM is greater than that of syndrome decoding, it provides tighter bounds for security.

Problem 5.1. (DOOM)

Instance: *A parity check matrix* $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ *of an* (n, k) *linear code, syndromes* $\boldsymbol{s}_1, \boldsymbol{s}_2, \cdots, \boldsymbol{s}_q \in \mathbb{F}_2^{n-k}$*, and an integer* w*.*

Output: $(\boldsymbol{e}, i) \in \mathbb{F}_2^n \times [1, q]$ *such that* $\mathrm{wt}(\boldsymbol{e}) \leqslant w$ *and* $\mathbf{H}\boldsymbol{e}^T = \boldsymbol{s}_i^T$.

We consider the case in which the adversary has q instances and $M = \max\left(1, \binom{n}{w}/2^{n-k}\right)$ solutions for each instance. Of course, in our case, w is not small, and thus M is $\binom{n}{w}/2^{n-k}$. In [70], the work factor of solving DOOM is given as

$$\mathrm{WF}_q^M = \min_{p,l}\left(\frac{C_q(p,l)}{\mathcal{P}_{qM}(p,l)}\right),$$

where

$$C_q(p,l) = \max\left(\sqrt{q\binom{k+l}{p}}, \frac{q\binom{k+l}{p}}{2^l}\right), q \leqslant \binom{k+l}{p}$$

is the complexity of solving the DOOM problem using Dumer's algorithm, and

$$\mathcal{P}_{qM}(p,l) = 1 - \left(1 - \frac{\binom{n-k-l}{w-p}\binom{k+l}{p}}{\binom{n}{w}}\right)^{qM}$$

is the success probability. This work factor is the reference for choosing the parameters of the signature scheme. Although advanced algorithms for information set decoding can be adapted to DOOM to reduce complexity, this has not yet been conducted. The proposed signature scheme is designed to use codes with a high-dimensional hull. Hence, the attacker can exploit this. However, to our knowledge, there is no algorithm for information set decoding or DOOM that considers this.

5.3.2 Security Against Key Substitution Attacks

In a key substitution attack, the adversary attempts to find a valid key that is different from the correct key and can be used for signature verification. If the adversary knows the secret key and the public key corresponding to a message–signature pair, we have a weak-key substitution attack, whereas if the adversary knows only the public key, we have a strong-key substitution attack. Both polynomial-time weak- and strong-key substitution attacks on the CFS signature scheme were proposed in [79]. A modification of the CFS scheme that resists such attacks was also proposed in [79]. In this modification, the syndrome $\boldsymbol{s}$ is generated by hashing the message, counter, and public key, rather than hashing only the message and counter. It has been demonstrated that this modified CFS signature scheme is secure against key substitution attacks [40]. In the modified pqsigRM, the syndrome is given as $\boldsymbol{s} = h(h(\boldsymbol{m}|\mathbf{H}')|\boldsymbol{i})$, and thus it is also secure against key substitution attacks.

5.3.3 EUF–CMA Security

Here, I prove the EUF-CMA security of the modified pqsigRM. The methods presented below are adapted from the EUF-CMA security proof of SURF and Wave [32, 50]. It should be noted that although a key attack for SURF is presented in [50], its proof technique is valid and generally applicable. The proof is essentially the same except for the code used for the key and the decoding algorithm for signing.

Basic Techniques for EUF-CMA Security Proof

EUF-CMA is a widely used attack model against signature schemes. In the security reduction task, EUF-CMA is viewed as a game played between an adversary and a challenger. The public key PK, hash oracle $\mathcal{H}$, and signing oracle Σ are given to a $(t, q_{\mathcal{H}}, q_{\Sigma}, \epsilon)$-adversary $\mathcal{A}$, where $\mathcal{A}$ can query at most $q_{\mathcal{H}}$ hash values and q_{Σ} signatures for inputs of its own choice. Within a maximum computation time t, $\mathcal{A}$ attempts to find a valid message–signature pair $(\boldsymbol{m}^*, \boldsymbol{\sigma}^*)$. $\mathcal{A}$ wins the game if

Verifying$(\boldsymbol{m}^*, \boldsymbol{\sigma}^*, PK) = 1$ and σ^* has not been provided by Σ; otherwise, the challenger wins the game. The winning probability of the $(t, q_{\mathcal{H}}, q_{\Sigma}, \epsilon)$-adversary is at least ϵ.

Definition 5.1. (EUF-CMA Security)
Let $\mathcal{S}$ be a signature scheme. I define the EUF-CMA *success probability against $\mathcal{S}$ as*

$$Succ_{\mathcal{S}}^{\mathrm{EUF-CMA}}(t, q_{\mathcal{H}}, q_{\Sigma}) := \max(\epsilon | \exists (t, q_{\mathcal{H}}, q_{\Sigma}, \epsilon)\textit{-adversary}).$$

The signature scheme $\mathcal{S}$ is called $(t, q_{\mathcal{H}}, q_{\Sigma})$-secure in EUF-CMA *if the above success probability is a negligible function of the security parameter λ.*

I use the statistical and computational distance as basic metrics.

Definition 5.2. (Statistical distance)
The statistical distance between two discrete probability distributions $\mathcal{D}^0$ and $\mathcal{D}^1$ over the same space $\mathcal{E}$ is defined as

$$\rho(\mathcal{D}^0, \mathcal{D}^1) := \frac{1}{2} \sum_{x \in \mathcal{E}} |\mathcal{D}^0(x) - \mathcal{D}^1(x)|.$$

Proposition 5.1. [50] *Let $(\mathcal{D}_1^0, \ldots, \mathcal{D}_n^0)$ and $(\mathcal{D}_1^1, \ldots, \mathcal{D}_n^1)$ be two n-tuples of discrete probability distributions over the same space. For all $n \geqslant 0$, we have*

$$\rho(\mathcal{D}_1^0 \otimes \cdots \otimes \mathcal{D}_n^0, \mathcal{D}_1^1 \otimes \cdots \otimes \mathcal{D}_n^1) \leqslant \sum_{i=1}^{n} \rho(\mathcal{D}_i^0, \mathcal{D}_i^1).$$

Definition 5.3. (Computational distance and indistinguishability)
The computational distance between two distributions $\mathcal{D}^0$ and $\mathcal{D}^1$ in time t is

$$\rho_c(\mathcal{D}^0, \mathcal{D}^1) := \frac{1}{2} \max_{|\mathcal{A}| \leqslant t} \left(Adv^{\mathcal{D}^0, \mathcal{D}^1}(\mathcal{A}) \right),$$

where $|\mathcal{A}|$ denotes the running time of $\mathcal{A}$, and $Adv^{\mathcal{D}^0, \mathcal{D}^1}$ is the advantage of distinguisher $\mathcal{A}$, which returns $b \in \{0, 1\}$ against $\mathcal{D}^0$ and $\mathcal{D}^1$:

$$Adv^{\mathcal{D}^0, \mathcal{D}^1} := \mathbb{P}_{\xi \sim \mathcal{D}^0}(\mathcal{A}(\xi) \textit{ outputs } 1) - \mathbb{P}_{\xi \sim \mathcal{D}^1}(\mathcal{A}(\xi) \textit{ outputs } 1).$$

The EUF-CMA security of the modified pqsigRM is reduced to the *modified RM code distinguishing problem* and *DOOM with high-dimensional hull*, which are defined as follows.

Problem 5.2. (Modified RM code distinguishing problem)

Instance: *A code $\mathcal{C}$ with high-dimensional hull.*

Output: *A bit $b \in \{0, 1\}$, where $b = 1$ if $\mathcal{C}$ is a permutation of the modified RM code; otherwise, $b = 0$.*

Problem 5.3. (DOOM with high-dimensional hull)

Instance: *A parity check matrix $\mathbf{H}' \in \mathbb{F}_2^{(n-k)\times n}$ of an (n, k) code with high-dimensional hull, syndromes $\boldsymbol{s}_1, \boldsymbol{s}_2, \cdots, \boldsymbol{s}_q \in \mathbb{F}_n^{(n-k)}$, and an integer w.*

Output: $(\boldsymbol{e}, i) \in \mathbb{F}_2^n \times [1, q]$ *such that* $\mathrm{wt}(\boldsymbol{e}) \leqslant w$ *and* $\mathbf{H}\boldsymbol{e}^T = \boldsymbol{s}_i^T$.

Definition 5.4. (One-wayness of DOOM with high-dimensional hull)
I define the success of an algorithm $\mathcal{A}$ against DOOM with high-dimensional hull and parameters n, k, q, w as

$$Succ^{n,k,q,w}(\mathcal{A}) = \mathbb{P}(\mathcal{A}(\mathbf{H}, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_q) \quad \textit{is a solution of Problem 5.3}),$$

where $\mathbf{H}$ is chosen uniformly from the parity check matrix of (n, k) codes with a high-dimensional hull, $\boldsymbol{s}_i$ is chosen uniformly in $\mathbb{F}_2^{n-k}$, and the probability is taken over these choices and the internal coin of algorithm $\mathcal{A}$. The computational success of breaking DOOM with a high-dimensional hull in time t is defined by

$$Succ_{DOOMHull}^{n,k,q,w}(t) = \max_{|\mathcal{A}|\leqslant t} \left(Succ^{n,k,q,w}(\mathcal{A}) \right).$$

We assume here that the probability is negligible (as a function of λ) for the parameters given in Table 5.2.

I will discuss these problems in greater detail in Section 5.4. It is worth noting that there are sufficiently many codes with high-dimensional hull for the parameters given in Tables 5.2 and 5.4 [80].

EUF-CMA Security Proof

Let $\mathcal{S}_{pqsigRM}$ denote the proposed modified pqsigRM. The following definitions as well as the theorem and its proof are adopted from those in [32, 50].

Definition 5.5. (Challenger procedures in the EUF-CMA game)
The challenger procedures in the EUF-CMA game corresponding to $\mathcal{S}_{pqsigRM}$ are defined as follows:

proc Init(λ)	proc Hash($\boldsymbol{m}, \boldsymbol{i}$)
$(PK, SK) \leftarrow \texttt{Gen}(1^\lambda)$ $\mathbf{H}' \leftarrow PK$ $(\mathbf{H}, \mathbf{S}, \mathbf{Q}) \leftarrow SK$ return $\mathbf{H}'$	return $h(\boldsymbol{m}, \boldsymbol{i})$
proc Sign($\boldsymbol{m}$)	proc Finalize($\boldsymbol{m}, \boldsymbol{e}, \boldsymbol{i}$)
$\boldsymbol{i} \hookleftarrow \{0,1\}^{\lambda_0}$ $\boldsymbol{s} \leftarrow \texttt{Hash}(\boldsymbol{m}, \boldsymbol{i})$ $\boldsymbol{e} \leftarrow \textsc{Decode}(\mathbf{S}^{-1}\boldsymbol{s}^T; \mathbf{H})$ return $(\boldsymbol{e}\mathbf{Q}, \boldsymbol{i})$	$\boldsymbol{s} \leftarrow \texttt{Hash}(\boldsymbol{m}, \boldsymbol{i})$ return $\mathbf{H}'\boldsymbol{e}^T = \mathbf{S}^T \wedge wt(\boldsymbol{e}) = w$

It is noted that the procedures in Definition 5.5 simplify Algorithm 7. We can now modify the security reduction in [32, 50] and prove the EUF-CMA security of the modified pqsigRM as follows.

Theorem 5.1. (Security reduction)
Let $Succ^{\mathrm{EUF-CMA}}_{\mathcal{S}_{pqsigRM}}(t, q_{\mathcal{H}}, q_{\Sigma})$ be the success probability of the EUF-CMA game corresponding to $\mathcal{S}_{pqsigRM}$ for time t when the number of queries to the hash oracle (resp.

signing oracle) is $q_{\mathcal{H}}$ (resp. q_{Σ}). Then, in the random oracle model, we have for all t

$$\begin{aligned} Succ^{\mathrm{EUF-CMA}}_{\mathcal{S}_{pqsigRM}}(t, q_{\mathcal{H}}, q_{\Sigma}) \leqslant \\ &2Succ^{n,k,q,w}_{DOOMHull}(t_c) + q_{\mathcal{H}}\mathbb{E}_{\mathbf{H}'}\left(\rho(\mathcal{D}^{\mathbf{H}'}_{w}, \mathcal{U}_s)\right) \\ &\quad + q_{\Sigma}\rho(\mathcal{D}_w, \mathcal{U}_w) + \rho_c(\mathcal{D}_{pub}, \mathcal{D}_{rand})(t_c) + \frac{1}{2^{\lambda}}, \end{aligned}$$

where $t_c = t + O(q_{\mathcal{H}} \cdot n^2)$, $\mathcal{D}^{\mathbf{H}'}_{w}$ is the distribution of the syndromes $\mathbf{H}'\boldsymbol{e}^T$ when $\boldsymbol{e}$ is drawn uniformly from the binary vectors of weight w, $\mathcal{U}_s$ is the uniform distribution over $\mathbb{F}_2^{n-k}$, $\mathcal{D}_w$ is the distribution of the decoding result of Algorithm 6, $\mathcal{U}_w$ is the uniform distribution over the binary vectors of weight w, $\mathcal{D}_{rand}$ is the uniform distribution over the random codes with high-dimensional hull, and $\mathcal{D}_{pub}$ is the uniform distribution over the public keys of modified pqsigRM.

Proof. Let $\mathcal{A}$ be a $(t, q_{\mathcal{H}}, q_{\Sigma}, \epsilon)$-adversary against $\mathcal{S}_{pqsigRM}$, and let $(\mathbf{H}_0, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_{q_{\mathcal{H}}})$ be a random instance of DOOM with high-dimensional hull for the parameters $n, k, q_{\mathcal{H}}$, and w. I stress that $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{q_{\mathcal{H}}}$ are random independent vectors of $\mathbb{F}_2^{n-k}$. Let $\mathbb{P}(S_i)$ denote the probability that $\mathcal{A}$ wins Game i.

Game 0 is the EUF-CMA game for $\mathcal{S}_{pqsigRM}$.

Game 1 is the same as Game 0 except for the following failure event F: There is a collision in a signature query. From the difference lemma in [81], we have

$$\mathbb{P}(S_1) \leqslant \mathbb{P}(S_0) + \mathbb{P}(F). \tag{5.1}$$

The following lemma is from [32].

Lemma 5.2. *For $\lambda_0 = \lambda + 2\log_2(q_{\mathcal{H}})$, we have $\mathbb{P}(F) \leqslant \frac{1}{\lambda}$.*

Game 2 is obtained from Game 1 by changing `Hash` and `Sign` as follows, where S_w denotes the set of vectors with Hamming weight w in $\mathbb{F}_2^n$: Index j is initialized to 0 in the `Init` procedure. I introduce the list $L_{\boldsymbol{m}}$, which contains $q_{\mathcal{H}}$ random elements of $\mathbb{F}_2^{\lambda_0}$ for each message $\boldsymbol{m}$. The list is sufficiently large so that all queries are satisfied.

```
proc Hash($\boldsymbol{m}, \boldsymbol{i}$)
if $\boldsymbol{i} \in L_{\boldsymbol{m}}$
  $\boldsymbol{e}_{\boldsymbol{m},\boldsymbol{i}} \hookleftarrow S_w$
  return $\mathbf{H}'\boldsymbol{e}^T_{\boldsymbol{m},\boldsymbol{i}}$
else
  $j \leftarrow j + 1$
  return $\boldsymbol{s}_j$
```

```
proc Sign($\boldsymbol{m}$)
$\boldsymbol{i} \leftarrow L_{\boldsymbol{m}}$.next()
$\boldsymbol{s} \leftarrow$ Hash($\boldsymbol{m}, \boldsymbol{i}$)
$\boldsymbol{e} \leftarrow$ DECODE($\mathbf{S}^{-1}\boldsymbol{s}^T; \mathbf{H}$)
return ($\boldsymbol{e}\boldsymbol{Q}, \boldsymbol{i}$)
```

The Hash procedure returns $\mathbf{H}'\boldsymbol{e}^T_{\boldsymbol{m},\boldsymbol{r}}$ if and only if $\boldsymbol{i} \in L_{\boldsymbol{m}}$; otherwise, it returns $\boldsymbol{s}_j$. The Sign process is unchanged unless $\boldsymbol{i} \in L_{\boldsymbol{m}}$.

The statistical distance between the syndromes generated by matrix $\mathbf{H}'$ and the uniform distribution over $\mathbb{F}_2^{n-k}$ is $\rho(\mathcal{D}_w^{\mathbf{H}'}, \mathcal{U}_s)$. This is the difference between Hash in Game 1 and Game 2 when $\boldsymbol{i} \in L_{\boldsymbol{m}}$. There are at most $q_{\mathcal{H}}$ such instances. Thus, by Proposition 5.1, it follows that

$$\mathbb{P}(S_2) \leqslant \mathbb{P}(S_1) + q_{\mathcal{H}}\mathbb{E}_{\mathbf{H}'}\left(\rho(\mathcal{D}_w^{\mathbf{H}'}, \mathcal{U}_s)\right). \tag{5.2}$$

Game 3 is obtained from Game 2 by replacing DECODE with $\boldsymbol{e}_{\boldsymbol{m},\boldsymbol{i}}$ in Sign procedure as follows: $\boldsymbol{e}$ is drawn according to the proposed decoding algorithm DECODE

Game 3

```
proc Sign($\boldsymbol{m}$)
$\boldsymbol{i} \leftarrow L_{\boldsymbol{m}}$.next()
$\boldsymbol{s} \leftarrow$ Hash($\boldsymbol{m}, \boldsymbol{i}$)
$\boldsymbol{e} \leftarrow \boldsymbol{e}_{\boldsymbol{m},\boldsymbol{i}}$
return ($\boldsymbol{e}, \boldsymbol{i}$)
```

Game 5

```
proc Finalize($\boldsymbol{m}, \boldsymbol{e}, \boldsymbol{i}$)
$\boldsymbol{s} \leftarrow$ Hash($\boldsymbol{m}, \boldsymbol{i}$)
$b \leftarrow \mathbf{H}'\boldsymbol{e}^T = \mathbf{S}^T \wedge wt(\boldsymbol{e}) = w$
return $b \wedge (\boldsymbol{i} \notin L_{\boldsymbol{m}})$
```

in Game 2, whereas it is now drawn according to the uniform distribution $\mathcal{U}_w$. By Proposition 5.1, we have

$$\mathbb{P}(S_3) \leqslant \mathbb{P}(S_2) + q_{\Sigma}\rho(\mathcal{D}_w, \mathcal{U}_w). \tag{5.3}$$

Game 4 is the game in which $\mathbf{H}'$ is replaced with $\mathbf{H}_0$. This implies that the adversary is forced to construct a solution for DOOM with the high-dimensional hull. Here, if a difference between Game 3 and Game 4 is detected, then this yields a distinguisher between $\mathcal{D}_{pub}$ and $\mathcal{D}_{rand}$. According to [50], the cost to call `Hash` does not exceed $O(n^2)$, and thus the running time of the challenger is $t_c = t + O(q_{\mathcal{H}} \cdot n^2)$. Therefore, we have

$$\mathbb{P}(S_4) \leqslant \mathbb{P}(S_3) + \rho_c(\mathcal{D}_{pub}, \mathcal{D}_{rand})(t_c). \tag{5.4}$$

Game 5 is modified in `Finalize`. The success of Game 5 implies $\boldsymbol{i} \notin L_{\boldsymbol{m}}$ and the success of Game 4. A valid forgery $\boldsymbol{m}^*$ has never been queried by `Sign`, and the adversary has never accessed $L_{\boldsymbol{m}^*}$. As there are q_Σ signing queries, we have

$$\mathbb{P}(S_5) = (1 - 2^{\lambda_0})^{q_\Sigma} \mathbb{P}(S_4).$$

Moreover, $(1 - 2^{\lambda_0})^{q_\Sigma} \geqslant \frac{1}{2}$ because we assumed $\lambda_0 = \lambda + 2\log_2(q_\Sigma)$. Thus, this can be simplified to

$$\mathbb{P}(S_5) \geqslant \frac{1}{2} \mathbb{P}(S_4). \tag{5.5}$$

$\mathbb{P}(S_5)$ is the probability that $\mathcal{A}$ returns a solution for DOOM with high-dimensional hull, which yields

$$\mathbb{P}(S_4) \leqslant 2Succ^{n,k,q,w}_{DOOMHull}(t_c). \tag{5.6}$$

Combining (5.1)–(5.6) concludes the proof. □

5.4 Indistinguishability of the Public Code and Signature

It is challenging to prove the hardness of distinguishing a public code of a code-based cryptographic algorithm from a random code. As it is difficult to prove the hardness of distinguishing the public code from a random code, several cryptographic algorithms are designed by assuming it. In this section, I will consider possible attack algorithms and consider the difficulty of distinguishing the public code and signatures. Moreover, the difficulty of distinguishing signatures from random errors is also analyzed.

5.4.1 Modifications of Public Code

For successful decoding of any received vector, a $(U, U + V)$-code should be used in the modified RM codes. To resist the attack on $(U, U+V)$-codes proposed in [50], I design a code with a high-dimensional hull. Generally, the expected dimension of the hull of a random code is $O(1)$, which is smaller than d with probability $\geqslant 1 - O(d)$ [80]. This is a difference between random and public codes. However, there is currently no algorithm for solving the syndrome decoding problem by taking advantage of the hull. We consider that a high-dimensional hull is not a significant drawback unless the hull has a certain structure that may reveal the secret. Moreover, in [80], it is demonstrated that there are a large number of codes with the high-dimensional hull. Hence, we can expect the one-wayness of DOOM with the high-dimensional hull as in Definition 5.4.

Cryptanalysis using hulls is widely used in code-based cryptography. However, this is valid if the hull has a specific structure that allows information leakage about the secret key. Therefore, using only the fact that the dimension of the hull is large, it is difficult to distinguish whether the code is public or random code with the high-dimensional hull.

The EUF-CMA security proof requires the indistinguishability between public and random codes, i.e., $\rho_c(\mathcal{D}_{pub}, \mathcal{D}_{rand})(t_c)$ should be negligible. I will discuss the design methodology and how these modifications can ensure indistinguishability.

Considering the key recovery attack in [50], a $(U, U+V)$-code used in code-based crypto-algorithms should have a high-dimensional hull for security. Even though the public code of the proposed signature scheme is not a $(U, U + V)$-code, it should contain a $(U, U + V)$ subcode for efficient decoding.

The attack on SURF in [50] uses the fact that for any $(U, U + V)$-code, the hull of the public code is highly probable to have a $(\boldsymbol{u}|\boldsymbol{u})$ structure when $U^{\perp} \cap V = \{\mathbf{0}\}$, $dim(U) \geqslant dim(V)$. This $(\boldsymbol{u}|\boldsymbol{u})$ reveals information about the secret permutation Q and enables the attacker to locate the U and $U + V$ codes. To avoid this, we should maintain the high dimension of $U^{\perp} \cap V$, implying that the public code should have a high-dimensional hull. Hence, I define DOOM with high-dimensional hull and assume that the public code of pqsigRM is indistinguishable from a random code with a hull of the same dimension as that of the public code, rather than any random linear code.

Moreover, k_{app} random rows are appended to the generator matrix, and 2^r rows of the generator matrix, that is the repeated $\mathrm{RM}_{(r,r)}$, are replaced by k_{rep} random rows; furthermore, a codeword from the dual code is appended to the generator matrix. These modifications are equivalent to increasing the dimension of the code itself, the hull, and the dual of the code, respectively, by appending random codewords. Moreover, by adding random codewords, the code is no longer a $(U, U+V)$-code, and thus distinguishing attacks are more difficult to perform.

I now explain the rationale for the aforementioned modifications, which are applied in addition to partial permutation.

Appending k_{app} Random Rows to the Generator Matrix

The Hamming weights of a random code are distributed. However, the partially permuted RM code has only codewords with even Hamming weight. This is because the Hamming weights of codewords of $\mathrm{RM}_{(r,m)}$ are even numbers, and partial permutations do not affect parity.

By appending a random row with odd Hamming weight to the generator matrix, the Hamming weights of the public code become distributed binomially. The problem is that if only one row with an odd Hamming weight is appended, it can easily be extracted. This can be resolved by appending more than one codeword. Hence, I append k_{app} random rows such that at least one has an odd Hamming weight. By the nature of the decoding process, it is still possible to decode the resulting code.

Appending a Random Codeword of the Dual Code to the Generator Matrix

The Hamming weights of the codewords in the hull of the partially permuted RM code are only multiples of four. However, the Hamming weight of the codewords in the hull of a random code may be an arbitrary even number, not only a multiple of four. As in the previous modification, a random codeword is appended to the hull. Thereby, I force the codewords of the hull of the public code to have arbitrary even Hamming weights. As a randomly appended row to the generator matrix is unlikely to be appended to its hull, appending a codeword to the hull is more complicated. The following is the process for appending a random codeword to the hull.

Let $hull(\mathcal{C})$ be the hull of a code $\mathcal{C}$. I define $\mathcal{C}'$ and $\mathcal{C}''$ by $\mathcal{C} = hull(\mathcal{C}) + \mathcal{C}'$ and $\mathcal{C}^{\perp} = hull(\mathcal{C}) + \mathcal{C}''$, where $hull(\mathcal{C})$, $\mathcal{C}'$, and $\mathcal{C}''$ are linearly independent. We can then generate a code with a hull with dimension $dim(hull(\mathcal{C})) + 1$ by the following procedure:

i) Find a codeword $\boldsymbol{c}_{dual} \in \mathcal{C}''$ such that $\boldsymbol{c}_{dual} \cdot \boldsymbol{c}_{dual} = 0$. This is easy because a codeword with even Hamming weight satisfies it.

ii) Let $\mathcal{C}_{inc} = \mathcal{C} + \{\boldsymbol{c}_{dual}\} = (hull(\mathcal{C}) + \{\boldsymbol{c}_{dual}\}) + \mathcal{C}'$.

iii) As $\boldsymbol{c}_{dual} \cdot (hull(\mathcal{C}) + \{\boldsymbol{c}_{dual}\}) = \{0\}$ and $\boldsymbol{c}_{dual} \cdot \mathcal{C}' = \{0\}$, we have $\boldsymbol{c}_{dual} \in \mathcal{C}_{inc}^{\perp}$, where for a vector x and a set of vectors A, $x \cdot A$ is the set of all inner products of x and elements of A.

iv) It can be seen that $\mathcal{C}_{inc} \cap \mathcal{C}_{inc}^{\perp} = (hull(\mathcal{C}) + \{\boldsymbol{c}_{dual}\})$. Hence, $\mathcal{C}_{inc}$ is a code that has a hull of which dimension is $dim(hull(\mathcal{C})) + 1$.

If the Hamming weights of the codewords of the hull are only multiples of 4, then another c_{dual} is selected, and the above process is repeated.

Replacing Repeated $\mathrm{RM}_{(r,r)}$ With Random $(2^r, k_{rep})$ Codes

It is noted that by replacing repeated $\mathrm{RM}_{(r,r)}$ by random $(2^r, k_{rep})$ codes, the dimension of the code is reduced by $2^r - k_{rep}$; this is equivalent to appending $2^r - k_{rep}$ rows to the parity check matrix. The codewords of the dual code of the partially permuted RM code have only codewords of even Hamming weight owing to a subcode of the partially permuted RM code. This can be resolved by replacing this subcode with another random code such that its MD decoder exists. The partially permuted RM code includes $(\mathrm{RM}_{(r,r)}|\dots|\mathrm{RM}_{(r,r)})$, and the dual code of this has only codewords of even Hamming weight by the proposition below. It is easy to verify that the dual code of the partially permuted RM code is a subset of the dual code of $(\mathrm{RM}_{(r,r)}|\dots|\mathrm{RM}_{(r,r)})$. That is, $(\mathrm{RM}_{(r,r)}|\dots|\mathrm{RM}_{(r,r)})$ causes the dual code of the partially permuted RM code to have only codewords of even Hamming weight.

Proposition 5.2. *Let $\mathcal{C}$ be a code such that its dual code has only codewords of even Hamming weight. Then, the dual of the concatenated code, $\{(\boldsymbol{c}|\boldsymbol{c})|\boldsymbol{c} \in \mathcal{C}\}$, has only codewords of even Hamming weight.*

Proof. Let $\boldsymbol{h} \in (\mathcal{C}|\mathcal{C})^{\perp}$, where $\mathcal{C}$ is an (n, k) code and $\mathcal{C}|\mathcal{C}$ is a concatenated code given as $\{(\boldsymbol{c}|\boldsymbol{c})|\mathbf{c} \in \mathcal{C}\}$. I define vectors $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ of length n so that $\boldsymbol{h} = (\boldsymbol{h}_1|\boldsymbol{h}_2)$. Clearly, if $\boldsymbol{h}_1 \in \mathcal{C}^{\perp}$, then $\boldsymbol{h}_2 \in \mathcal{C}^{\perp}$. If $\boldsymbol{h}_1 \notin \mathcal{C}^{\perp}$, we have $\boldsymbol{h}_1 \cdot \boldsymbol{c} + \boldsymbol{h}_2 \cdot \boldsymbol{c} = 0$, i.e., $\boldsymbol{h}_1 \cdot \boldsymbol{c} = \boldsymbol{h}_2 \cdot \boldsymbol{c}$. This implies that $\boldsymbol{h}_1 = \boldsymbol{h}_2$. Hence, $\mathrm{wt}(\boldsymbol{h})$ is even. □

By replacing the repeated $\mathrm{RM}_{(r,r)}$ with a random code such that its dual code has codewords of odd Hamming weight, we can force the dual of the public code to have codewords with odd Hamming weight.

Clearly, the dual code of $\mathrm{RM}_{(r,r)}$ is $\{\mathbf{0}\}$. I replace $\mathrm{RM}_{(r,r)}$ with a random $(2^r, k_{rep})$ code. It is noted that the dual code of this $(2^r, k_{rep})$ code must have codewords with odd Hamming weight. The generator matrix is modified in this manner, rather than by appending rows to the parity check matrix, to ensure that the entire code is decodable.

5.4.2 Public Code Indistinguishability

In the EUF-CMA security proof, $\rho_c(\mathcal{D}_{pub}, \mathcal{D}_{rand})$ is required to be negligible; that is, the modified RM code distinguishing problem should be hard. As it is challenging to find the computational distance between public and random codes, in this section, we study the randomness of the public code and consider possible attacks.

Public Code is Not a $(U, U+V)$-Code

After random rows have been appended to the generator matrix of a $(U, U+V)$-code, the resulting code is unlikely to be a $(U, U+V)$-code. Considering the following proposition, it can be seen that with probability $O(2^{k_U - n/2})$, a $(U, U+V)$-code remains a $(U, U+V)$-code after a row has been appended to its generator matrix.

Proposition 5.3. *Let $\mathcal{C}$ be a $(U, U+V)$-code. Then, for all codewords $(\boldsymbol{c}'|\boldsymbol{c}'') \in \mathcal{C}$, $(\mathbf{0}|\boldsymbol{c}' - \boldsymbol{c}'') \in \mathcal{C}$.*

It is expected that attacking the modified RM code is difficult because the appended codewords change the algebraic structure of the code (i.e., the $(U, U+V)$ structure), there is considerable randomness, and there is currently no recovery algorithm.

Distinguishing Using Hull

When a random row is appended to the generator matrix, it is unlikely to be included in the hull. The appended row should be a codeword of the dual code to achieve this, and its square should be zero. Hence, we append a codeword from the dual code to the generator matrix.

The appended row can be omitted when the attacker collects several independent codewords with Hamming weight four from the hull. However, for any random code with a high-dimensional hull, the same process can be applied, and finally, there only remain codewords of which the Hamming weight is a multiple of 4. Hence, this is not a valid distinguishing attack.

The hull of a random $(U, U+V)$-code is $\{\mathbf{0}\}$ when $k_U < k_V$ and is highly probable to have codewords of $(\boldsymbol{u}|\boldsymbol{u})$ form when $k_U \geqslant k_V$. However, the hull of an RM code is also an RM code, and in our case, the partial permutation randomizes its hull and retains its large dimension. As shown in Section 5.5, the hull is neither a subcode of the RM code nor a $(U, U+V)$-code. Moreover, most of the hull depends on the secret partial permutations σ_p^1 and σ_p^2.

5.4.3 Signature Leaks

In the EUF-CMA security proof, it is required that $\rho(\mathcal{D}_w, \mathcal{U}_w)$ is a negligible function of the security parameter λ. If this is true, then the signature does not leak information. In several signature schemes, such as Durandal, SURF, and Wave, this is achieved and proved. In SURF and Wave, the rejection sampling method is applied to render $\mathcal{D}_w$ indistinguishable.

To apply rejection sampling, the distribution of the decoding output should be known. In SURF and Wave, a simple and efficient decoding algorithm is used, and thus it is easy to find the distribution of the decoding output. However, in our case, the decoding output exhibits a high degree of randomness, and the structure of the decoder is complex. Therefore, it is difficult to analyze the distribution of the decoding output. Instead, I conduct a proof-of-concept implementation of the modified pqsigRM using SageMath. Then, I perform statistical randomness tests under NIST SP 800-22 [82] on the decoding output, and I compare the results with random errors in $\mathbb{F}_2^n$ with Hamming weight w. No significant difference is observed. However, it should be noted that the success of a statistical randomness test does not imply indistinguishability. Thus, the indistinguishability of the signature should be rigorously studied as future work.

5.5 Parameter Selection

5.5.1 Parameter Sets

The constraint here is that n is a power of two. We can numerically find the feasible ranges of w once n and k are determined. If the security level λ is achieved in this range, the value is accepted; otherwise, we increase n. Considering DOOM, a smaller value of w implies higher security. If w is so small that a large number of decoding iterations are required, we could reduce the partial permutation parameter p. p is at most $n/4$, and the characteristics of the codes are retained by lowering p to a certain degree. The method for obtaining the minimum values is described in the following

Table 5.2: Parameters for each security level

λ (security)	80	128	256
(r, m)	(5,11)	(6,12)	(6,13)
n	2048	4096	8192
k	1025	2511	4097
w	325	495	1370
k_{rep}	30	62	62
k_{app}	2	2	2
p (recommended)	$\geqslant$130	$\geqslant$386	$\geqslant$562
Signature length (bits)	2048	4096	8192
Public key size (MB)	0.249	0.773	3.99
$\log_2$ WF	80	128	256

subsection. The discussed state-of-the-art algorithm for DOOM is used as a basis for the parameters proposed in Table 5.2. I set $k_{app} = 2$ (the minimum value) and $k_{rep} = 2^r - 2$ (the maximum value).

Regarding the key size, the public key is a parity check matrix given in the systematic form and requires $(n-k)n$ bits. The secret key does not include a scrambler matrix $\mathbf{S}$ because it can be obtained from $\mathbf{H}'$, $\mathbf{Q}$, and $\mathbf{H}$. Moreover $\mathbf{H}$ can be represented by σ_p^1, σ_p^2, replacing code, and appending rows.

The comparison of parameter sets is given in Table 5.3. The key size of the proposed modified pqsigRM is small compared to other algorithms. It is noted that it is for reference only, and the actual parameter size is given variously along with trade-off with signing complexity, etc. The security level in parallel-CFS is based on the generalized birthday algorithm [84], and the distinguisher for high-rate Goppa code [29] is not considered. For detailed information, see [83] and [32].

Table 5.3: Comparison of parameter sets of several code-based signature schemes for given security.

λ		Proposed	Wave [32]	Parallel-CFS [83]
80	pk size	0.249	1.214	20.0
	sgn. len.	2048	8234	294
128	pk size	0.773	3.108	2.7×10^5
	sgn. len.	4096	13174	474
256	pk size	3.99	12.432	9.4×10^{15}
	sgn. len.	8192	26347	1242

5.5.2 Statistical Analysis for Determining Number of Partial Permutations

If w is excessively small, there is a low probability of finding an error vector with Hamming weight less than equal to w. I present two solutions. One is iterating until an appropriate error vector is obtained, and the other is improving the decoder. The number p of columns permuted in the partial permutation varies from 0 to $n/4$. From the numerical analysis, it is demonstrated that small values of p result in low Hamming weight of the decoding output. However, it should be noted that when $p = 0$, the $(U, U + V)$ part of the modified RM codes becomes identical to the RM code except that $\mathrm{RM}_{(r,r)}$ is replaced. Hence, I propose the lower bound of p that does not affect the randomness of the hull.

Regarding the modified RM code, its hull overlaps with (but is not a subset of) the original RM code. If the hull is a subset of the original RM code, and its dimension is large, the codeword of minimum Hamming weight of the original RM code may be included in the hull. Then, attacks such as the Minder–Shokrollahi attack may be applied using codewords with minimum Hamming weight. Therefore, to prevent attacks, the hull of the public code should not be a subset of the original RM code, and $hull(\mathcal{C}_{pub}) \setminus$ (RM$_{(r,m)}$ permuted by Q) should occupy a large portion of the hull, where $\mathcal{C}_{pub}$ denotes the public code, and $\setminus$ denotes the relative complement.

As the permutation Q is not important for determining the parameter p, we ignore it in this subsection, and the term permutation refers to the partial permutations σ_p^1 and σ_p^2. When $p = n/4$, which implies that σ_p^1 and σ_p^2 are full permutations, the average dimension of the hull and the dimension of $hull(\mathcal{C}_{pub}) \setminus \mathrm{RM}_{(r,m)}$ are given in Table 5.4. The values may slightly change according to the permutation.

If p is small, the Hamming weight of the errors decreases. Hence, the signing time can be reduced by using partial permutation with p rather than full permutation. The aim is to find a smaller value for p maintaining the dimension of $hull(\mathcal{C}_{pub}) \setminus \mathrm{RM}_{(r,m)}$ as large as that by the full permutation. It can be seen that the average of the dimension of $hull(\mathcal{C}_{pub}) \setminus \mathrm{RM}_{(r,m)}$ tends to increase as p increases, and it is saturated when p is above a certain value, as in Figure 5.3. Specifically, the dimension of $hull(\mathcal{C}_{pub}) \setminus \mathrm{RM}_{(r,m)}$ is saturated when p is approximately equal to the average dimension of $hull(\mathcal{C}_{pub}) \setminus \mathrm{RM}_{(r,m)}$ with full permutation. Hence, I determine p as 130, 386, and 562 in Table 5.2.

Table 5.4: Average dimension of $hull(\mathcal{C}_{pub})$ and $hull(\mathcal{C}_{pub}) \setminus \mathrm{RM}_{(r,m)}$ with $p = n/4$

(r,m)	(5,11)	(6,12)	(6,13)
n	2048	4096	8192
k	1025	2511	4097
$dim(hull(\mathcal{C}_{pub}))$	766	1236	2974
$dim(hull(\mathcal{C}_{pub}) \setminus \mathrm{RM}_{(r,m)})$	130	386	562

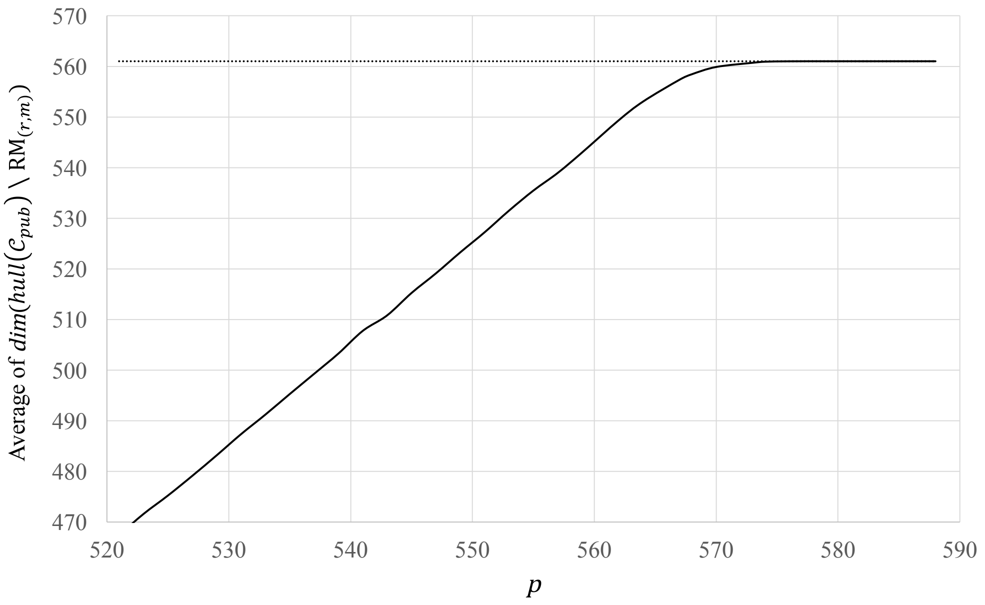


Figure 5.3: Dimension of $hull(\mathcal{C}_{pub}) \setminus \mathrm{RM}_{(6,12)}$ for 128-bit security parameters.

5.6 Equivalence of the Prototype IKKR and the McEliece Cryptosystems

The IKKR cryptosystems are not designed for indistinguishability or non-malleability; they do not even satisfy the indistinguishability under chosen-plaintext attack (IND-CPA). Thus, for a fair comparison, the equivalence of the prototype IKKR and the McEliece cryptosystems for one-wayness (OW) is shown in this section. All matrices regarding keys for the prototype IKKR and the McEliece cryptosystems are already defined in Sections 2.5.1 and 2.5.4.

A fair comparison of the prototype IKKR and the McEliece cryptosystems dictates that both of them are based on the same error-correcting (n, k) linear code $\mathcal{C}$ with error correction capability t. Let $\mathbf{G}$ and $\mathbf{H}$ denote the generator matrix and the parity check matrix of $\mathcal{C}$, respectively.

It is proved that the prototype IKKR cryptosystem is not more secure than the ordinary McEliece cryptosystem in [52]. In other words, we have following lemma.

Lemma 5.3. *(Prototype IKKR $\leqslant$ McEliece [52]) If there exists an efficient adversary that decrypts the ciphertext of the McEliece cryptosystem, then there exists an efficient adversary that decrypts the ciphertext of the prototype IKKR cryptosystem.*

It is proved in the following lemma that the McEliece cryptosystem is also reduced to the prototype IKKR cryptosystem.

Lemma 5.4. *(Prototype IKKR $\geqslant$ McEliece) If there exists an efficient adversary that decrypts the ciphertext of the prototype IKKR cryptosystem, then there exists an efficient adversary that decrypts the ciphertext of the McEliece cryptosystem.*

Proof. I define $Adv_{\mathrm{IKKR}}(\mathbf{G}', \mathbf{G}'_1, \boldsymbol{c})$ an adversary for the prototype IKKR cryptosystem that returns $\boldsymbol{m}$ if there exists such $\boldsymbol{m}$ satisfying $\boldsymbol{c}^{\intercal} = \boldsymbol{m}^{\intercal}\mathbf{G}' + \boldsymbol{e}^{\intercal}\mathbf{G}'_1$, and returns $\perp$, otherwise. The adversary $Adv_{\mathrm{McEliece}}(\mathbf{G}_{\mathrm{pub}}, \boldsymbol{c})$ can find $\boldsymbol{e}$ and $\boldsymbol{m}$ satisfying $\boldsymbol{c}^{\intercal} = \boldsymbol{m}^{\intercal}\mathbf{G}_{\mathrm{pub}} + \boldsymbol{e}^{\intercal}$ using Adv_{IKKR} from the following procedure, where

$\mathbf{G}_{\text{pub}} = \mathbf{G}\mathbf{P}^{\intercal}$ for a permutation matrix $\mathbf{P}$.

1. Key translation from McEliece to prototype IKKR
   (a) The adversary chooses a random nonsingular matrix $\mathbf{M}'$. Let $\mathbf{G}''$ be a matrix whose rows are n distinct codewords from $\mathbf{G}_{\text{pub}}$.
   (b) Then, the adversary can make a public key for the prototype IKKR cryptosystem by
   $$(\mathbf{G}', \mathbf{G}'_1) = (\mathbf{G}_{\text{pub}}\mathbf{M}', (\mathbf{G}'' - \mathbf{I})\mathbf{M}').$$
   (c) This is because by letting $\mathbf{M} = \mathbf{P}^{\intercal}\mathbf{M}'$ (even if $\mathbf{P}$ is unknown), $\mathbf{G}' = \mathbf{G}\mathbf{M}$ and $\mathbf{G}'_1 = (\mathbf{G}_0 - \mathbf{P})\mathbf{M}$ are satisfied, where $\mathbf{G}_0$ is a matrix whose rows are n codewords from $\mathbf{G}$ (even if $\mathbf{G}_0$ is unknown).
2. Ciphertext translation from McEliece to prototype IKKR
   (a) For a ciphertext $\boldsymbol{c}^{\intercal} = \boldsymbol{m}^{\intercal}\mathbf{G}_{\text{pub}} + \boldsymbol{e}^{\intercal}$, the following equation is satisfied as
   $$\begin{aligned} \boldsymbol{c}^{\intercal}\mathbf{M}' &= \boldsymbol{m}^{\intercal}\mathbf{G}_{\text{pub}}\mathbf{M}' + \boldsymbol{e}^{\intercal}\mathbf{M}' \\ &= \boldsymbol{m}^{\intercal}\mathbf{G}' + \boldsymbol{e}^{\intercal}\mathbf{P}\mathbf{M} \\ &= \boldsymbol{m}'^{\intercal}\mathbf{G}' - \boldsymbol{e}^{\intercal}\mathbf{G}'_1 \end{aligned}$$
   for some $\boldsymbol{m}'$.
   (b) Thus, $Adv_{\text{IKKR}}(\mathbf{G}', \mathbf{G}'_1, \mathbf{M}'^{\intercal}\boldsymbol{c})$ finds $\boldsymbol{m}'$ and $Adv_{\text{McEliece}}(\mathbf{G}_{\text{pub}}, \boldsymbol{c})$ can find $\boldsymbol{e}$ and $\boldsymbol{m}$ using $\boldsymbol{m}'$.

□

If an adversary is provided a decryption oracle, both the cryptosystems are OW-insecure, and thus the decryption oracle is not considered. Although it was not involved in the above, it is easy to see that the above two reductions are similarly applied to the key-recovery attack. The following theorem appears directly from the above two lemmas.

Theorem 5.5. *The prototype IKKR and the McEliece cryptosystems are equivalent.*

5.7 Cryptanalysis of the IKKR Cryptosystems

5.7.1 Linearity of Two Variants of IKKR Cryptosystems

In the proposed attack algorithm, I mainly utilize the linearity of encryption of IKKR cryptosystems and the fact that there exist a small number of possible solutions for the system of linear equations. I remark that the encryption of the upgraded IKKR cryptosystem is given as

$$\boldsymbol{c}^{\mathsf{T}} = \boldsymbol{m}^{\mathsf{T}}\mathbf{G}' + \boldsymbol{e}^{\mathsf{T}}\mathbf{G}_2'. \tag{5.7}$$

As the rank of $\mathbf{G}_2'$ is $(n-k)$, there exists $\boldsymbol{e}'$ satisfying

$$\boldsymbol{e}'^{\mathsf{T}}\mathbf{G}_2'' = \boldsymbol{e}^{\mathsf{T}}\mathbf{G}_2', \tag{5.8}$$

where $\mathbf{G}_2''$ is a matrix formed by $(n-k)$ linearly independent rows of $\mathbf{G}_2'$. Thus, (5.7) can be rewritten as

$$\boldsymbol{c}^{\mathsf{T}} = \boldsymbol{m}^{\mathsf{T}}\mathbf{G}' + \boldsymbol{e}'^{\mathsf{T}}\mathbf{G}_2''.$$

Considering an augmented matrix, the encryption is given as the following linear transformation

$$\boldsymbol{c} = \left[\mathbf{G}'^{\mathsf{T}} | \mathbf{G}_2''^{\mathsf{T}}\right] \cdot \begin{bmatrix} \boldsymbol{m} \\ \boldsymbol{e}' \end{bmatrix}. \tag{5.9}$$

Similarly, the linearity can be found as in (5.9) for the modified version of prototype IKKR cryptosystem as

$$\boldsymbol{c} = \left[\mathbf{G}'^{\mathsf{T}} | \mathbf{G}_1''^{\mathsf{T}}\right] \cdot \begin{bmatrix} \boldsymbol{m} \\ \boldsymbol{e}' \end{bmatrix}, \tag{5.10}$$

where $\mathbf{G}_1''$ consists of t linearly independent rows of $\mathbf{G}_1'$ as the rank of $\mathbf{G}_1'$ is t.

(5.9) and (5.10) depict the linearity of encryption. The following section describes a polynomial-time attack algorithm for the two variants of IKKR cryptosystems using this linearity.

5.7.2 The Attack Algorithm

When $\left[\mathbf{G}'^{\mathsf{T}}|\mathbf{G}_2''^{\mathsf{T}}\right]$ is a nonsingular matrix, the plaintext is found from only the public key and ciphertext as

$$\begin{bmatrix} \boldsymbol{m} \\ \boldsymbol{e}' \end{bmatrix} = \left[\mathbf{G}'^{\mathsf{T}}|\mathbf{G}_2''^{\mathsf{T}}\right]^{-1} \boldsymbol{c}.$$

In fact, as in the following theorem, $\left[\mathbf{G}'^{\mathsf{T}}|\mathbf{G}_2''^{\mathsf{T}}\right]$ is always a nonsingular matrix when $\mathbf{Q}$ and $\mathbf{T}$ are constructed according to Algorithm 3.

Theorem 5.6. *$\left[\mathbf{G}'^{\mathsf{T}}|\mathbf{G}_2''^{\mathsf{T}}\right]$ is a nonsingular matrix when $\mathbf{Q}$ and $\mathbf{T}$ are constructed according to Algorithm 3.*

Proof. The row spaces of $\mathbf{G}_2''$ and $\mathbf{G}_2'$ are equivalent. Thus, it is sufficient to show that the rank of $\left[\mathbf{G}'^{\mathsf{T}}|\mathbf{G}_2'^{\mathsf{T}}\right]$ is n. Using the fact that $\left[\mathbf{G}'^{\mathsf{T}}|\mathbf{G}_2'^{\mathsf{T}}\right] = \mathbf{M}^{\mathsf{T}}\left[\mathbf{G}^{\mathsf{T}}|(\mathbf{Q}(\mathbf{G_0}+\mathbf{T}))^{\mathsf{T}}\right]$ and $\mathbf{M}$ is nonsingular, we omit $\mathbf{M}$ when discussing the rank of $\left[\mathbf{G}'^{\mathsf{T}}|\mathbf{G}_2'^{\mathsf{T}}\right]$.

Without loss of generality, we assume that $\mathcal{J} = \{1, 2, \ldots, k\}$. Let $\mathbf{G}'_{\text{sys}} = \left[\mathbf{I}_k|\mathbf{G}'_p\right]$ be a systematic form of $\mathbf{G}'$. Let $\mathbf{T} = [\mathbf{T}_{\mathcal{J}}|\mathbf{T}_{\mathcal{J}^c}]$ and then $\mathbf{QT} = [\mathbf{0}|\mathbf{LH}_{\mathcal{J}}\mathbf{T}_{\mathcal{J}^c}]$. It is noted that $\mathbf{LH}_{\mathcal{J}}\mathbf{T}_{\mathcal{J}^c}$ is an $n \times (n-k)$ full rank matrix. Given $\mathbf{G}_2' = \mathbf{Q}(\mathbf{G}_0 + \mathbf{T})$, we have

$$\begin{bmatrix} \mathbf{G}'_{\text{sys}} \\ \mathbf{G}'_2 \end{bmatrix} = \begin{bmatrix} \mathbf{G}'_{\text{sys}} \\ \mathbf{QG}_0 + \mathbf{QT} \end{bmatrix}. \tag{5.11}$$

As the rows of $\mathbf{QG}_0$ are codewords of $\mathcal{C}$, $\mathbf{QG}_0$ can be eliminated by some linear combinations of rows in $\mathbf{G}'_{\text{sys}}$. Thus, the rank of the matrix (5.11) is the same as

$$\begin{bmatrix} \mathbf{G}'_{\text{sys}} \\ \mathbf{QT} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_k & \mathbf{G}'_p \\ \mathbf{0} & \mathbf{LH}_{\mathcal{J}}\mathbf{T}_{\mathcal{J}^c} \end{bmatrix}.$$

Applying Gaussian elimination on the $(k+1)$-th to $(n+k)$-th rows of the above matrix, we have

$$\begin{bmatrix} \mathbf{I}_k & \mathbf{G}'_p \\ \mathbf{0} & \mathbf{I}_{n-k} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

Hence, the rank of

$$\begin{bmatrix} \mathbf{G}'_{\text{sys}} \\ \mathbf{G}'_2 \end{bmatrix}$$

is n and thus $\left[\mathbf{G}'^{\intercal} | \mathbf{G}''^{\intercal}_2\right]$ is a nonsingular matrix. $\square$

Therefore, the proposed attack algorithm always succeeds.

In the modified version of prototype IKKR cryptosystem, $\mathbf{G}'_1 = \mathbf{WD}(\mathbf{G}_0 - \mathbf{P})\mathbf{M}$ is used instead of $\mathbf{G}'_2$ and the rank of $\mathbf{G}'_1$ is $t < (n - k)$. Similarly, $\left[\mathbf{G}'^{\intercal} | \mathbf{G}''^{\intercal}_1\right]$ is a full-rank $n \times (k + t)$ matrix from the following theorem.

Theorem 5.7. *The rank of* $\left[\mathbf{G}'^{\intercal} | \mathbf{G}'^{\intercal}_1\right]$ *is* $(k + t)$*, where* $\mathbf{G}'_1 = \mathbf{WD}(\mathbf{G}_0 - \mathbf{P})\mathbf{M}$*,* $\mathbf{W}$ *is an* $n \times n$ *nonsingular matrix, and* $\mathbf{D}$ *is an* $n \times n$ *diagonal matrix with* t *non-zero elements on the diagonal.*

Proof. As in the proof of Theorem 5.6, we omit $\mathbf{M}$. The rank of matrix

$$\begin{bmatrix} \mathbf{G}' \\ \mathbf{G}'_1 \end{bmatrix} = \begin{bmatrix} \mathbf{G}_0 \\ \mathbf{WDG}_0 - \mathbf{WDP} \end{bmatrix}$$

is equal to

$$\begin{bmatrix} \mathbf{G}_0 \\ \mathbf{WDP} \end{bmatrix}$$

as the rows of $\mathbf{WDG}_0$ are codewords generated by $\mathbf{G}_0$. The Hamming weight of the rows of $\mathbf{WDP}$ is at most t while the minimum distance of the code generated by $\mathbf{G}_0$ is larger than $2t$. Hence, every row of $\mathbf{WDP}$ is independent of the rows of $\mathbf{G}_0$ while the rank of $\mathbf{WDP}$ is t. This implies that the rank of $\left[\mathbf{G}'^{\intercal} | \mathbf{G}'^{\intercal}_1\right]$ is $(k + t)$. $\square$

Therefore, (5.10) also has a unique solution.

5.7.3 Implementation

I have done a proof-of-concept implementation of the upgraded IKKR cryptosystem and the attack algorithm in Section 5.7.2 using `SageMath9.0`[1]. It is noted that the

[1]Source code: `https://github.com/Yongwoo-Lee-ccl/crypt_ikkr`

Table 5.5: Average execution time for each step with 100 trials when $q = 2, n = 1024$, and $k = 524$.

KeyGen	Enc	Dec	Proposed attack
5.97 s	161 ms	513 ms	188 ms

decryption does not utilize the decoding of the code $\mathcal{C}$ and thus I could consider $\mathcal{C}$ as a random code rather than a well-designed code with efficient decoding such as the Goppa codes. Table 5.5 shows the execution time on average for 100 trials running in `Intel Core i7-6700k` (4.0 GHz). It can be seen that the proposed attack algorithm finds the plaintext corresponding to a given ciphertext within 0.2s in a stock PC that is faster than the elapsed time for legitimate decryption. During the 100 trials of cryptanalysis with different keys, the attack algorithm always succeeds.

Algorithm 6 Decoding for modified RM code

```
1:  function DECODE($\boldsymbol{s}; \mathbf{H}$)
2:      $\boldsymbol{r} \leftarrow$ PRANGE($\mathbf{H}, \boldsymbol{s}$)
3:      while True do
4:          $\boldsymbol{r} \leftarrow \boldsymbol{r}+$ random codeword
5:          $\boldsymbol{c} \leftarrow$ MODDEC($\boldsymbol{r}, r, m$)
6:          if $\mathrm{wt}(\boldsymbol{r}+\boldsymbol{c}) \leqslant w$ then
7:              Output $\boldsymbol{r}+\boldsymbol{c}$
8:          end if
9:      end while
10: end function
11: function MODDEC($\boldsymbol{y}, r, m$)
12:     $\boldsymbol{y} \leftarrow \boldsymbol{y}^{\sigma^{-1}}$
13:     if $r = 0$ then
14:         Output MD decoding on RM$(0, m)$
15:     else if $r = m$ then
16:         Output MD decoding on RM$(r, r)$ or replaced $(2^r, k_{rep})$ code
17:     else
18:         $(\boldsymbol{y}'|\boldsymbol{y}'') \leftarrow \boldsymbol{y}$
19:         $\boldsymbol{y}^{\boldsymbol{v}} = \boldsymbol{y}' \cdot \boldsymbol{y}''$
20:         $\hat{\boldsymbol{v}} \leftarrow$ MODDEC($\boldsymbol{y}^{\boldsymbol{v}}, r-1, m-1$)
21:         $\boldsymbol{y}^{\boldsymbol{u}} \leftarrow (\boldsymbol{y}' + \boldsymbol{y}'' \cdot \hat{\boldsymbol{v}})/2$
22:         $\hat{\boldsymbol{u}} \leftarrow$ MODDEC($\boldsymbol{y}^{\boldsymbol{u}}, r, m-1$)
23:         $\boldsymbol{y} \leftarrow (\hat{\boldsymbol{u}}|\hat{\boldsymbol{u}} \cdot \hat{\boldsymbol{v}})$
24:     end if
25:     Output $\boldsymbol{y}^{\sigma}$
26: end function
```

*σ is σ_p^1 or σ_p^2 for permuted block and identity, otherwise.

Algorithm 7 Modified pqsigRM signature scheme

```
Key Generation:
    Using $\sigma_p^1$ and $\sigma_p^2$, generate a partially permuted generator matrix $\mathbf{G}$
    Generate $\mathbf{H}$ from $\mathbf{G}$
    Generate $\mathbf{S}$ and $\mathbf{Q}$
    Compute $\mathbf{H}' \leftarrow \mathbf{SHQ}$
    Secret key: $\mathbf{H}, \mathbf{S}, \mathbf{Q}$
    Public key: $\mathbf{H}'$
Signing:
    $\boldsymbol{m}$ is a message to be signed
    $\boldsymbol{i} \hookleftarrow \{0,1\}^{\lambda_0}$
    Find syndrome $\boldsymbol{s} \leftarrow h(h(\boldsymbol{m}|\mathbf{H}')|\boldsymbol{i})$
    $\boldsymbol{s}'^T \leftarrow \mathbf{S}^{-1}\boldsymbol{s}^T$
    Perform decoding $\boldsymbol{e}' \leftarrow \textsc{Decode}(\boldsymbol{s}; \mathbf{H})$
    * Compute $\boldsymbol{e}^T \leftarrow \mathbf{Q}^{-1}\boldsymbol{e}'^T$, and then the signature is $(\boldsymbol{m}, \boldsymbol{e}, \boldsymbol{i})$
Verification:
    Check $\mathrm{wt}(\boldsymbol{e}) \leqslant w \wedge \mathbf{H}'\boldsymbol{e}^T = h(h(\boldsymbol{m}|\mathbf{H}')|\boldsymbol{i})$
    If True, then return ACCEPT; else, return REJECT
```

Chapter 6

Conclusion

In this dissertation, three main contributions are given as; i) a protocol of privacy-preserving machine learning using network resources, ii) the development of approximate homomorphic encryption that achieves less error and high-precision bootstrapping algorithm without compromising performance and security, iii) the cryptanalysis and the modification of code-based cryptosystems: cryptanalysis on IKKR cryptosystem and modification of the pqsigRM, a digital signature scheme proposed to the PQC standardization of NIST.

6.1 Privacy-Preserving Machine Learning Without Bootstrapping

I introduced a method of privacy-preserving machine learning using the CKKS scheme without bootstrapping. In the proposed method, bootstrapping is replaced by network communication. The ciphertext of intermediate value is sent to the receiver, and the receiver decrypts and re-encrypts the ciphertext and sends the message back. The information-theoretic secrecy is adopted to secure the intermediate values during this process.

This protocol is effective in the CKKS scheme. A deep neural network is impractical in the CKKS scheme because of the expensive computation as well as error. Hence, a larger scaling factor should be used for high precision. In this protocol, unlike bootstrapping, the refreshing of ciphertext does not introduce additional error, and thus the more level can be used. Moreover, full slots can be utilized, and thus the throughput is much larger than the method with bootstrapping. In conclusion, following the application might require the receiver less computation and less communication.

6.2 Variance-Minimization in the CKKS Scheme

I introduced two novel methods to improve the precision of the CKKS scheme. First, SNR, a widely-used measure of performance when we deal with erroneous media such as communication systems, was adopted for the error variance minimization of the CKKS scheme. To maximize the SNR of encrypted data, I proposed a method to minimize the variance of errors. To do this, I replaced the high-probability upper bound that has been in the tagged information so far with the variance of errors. As a result, I could tightly manage the error, and the homomorphic operations were effectively reordered to minimize the error variance. Second, I proposed a method to find the optimal approximate polynomial for the CKKS scheme in the same aspect of minimizing the error variance. Especially, the newly proposed operation reordering and approximate polynomial were applied to the bootstrapping of the CKKS scheme, and thus, the error performance of the bootstrapping of the CKKS scheme was improved. To our best knowledge, this is the first bootstrapping algorithm that contemplates various parameters, slot size, the error characteristics of the CKKS scheme, and the polynomial evaluation algorithm. From its implementation on HEAAN and SEAL, it was shown that the proposed bootstrapping algorithm achieves less variance of error in encrypted data while consuming less level, compared to the previous works.

From the proposed method, now there are two criteria to reorder homomorphic operations when I use the CKKS scheme: error variance reduction and computation time reduction. In addition to the proposed three examples in this dissertation, there are various methodologies to reorder homomorphic operations to minimize the error, and it will affect the error performance of the CKKS scheme significantly as the operation becomes deep. Since a lot of studies to adjust the order of operations for a general-purpose have been done in the field of compilers, applying the results of these researches will lead to significant improvement in many applications using the CKKS scheme. I leave application-specific reordering of homomorphic operations with compiler techniques as further work.

6.3 L2-Norm Minimization for the Bootstrapping of the CKKS Scheme

I determined the near-optimal approximate polynomial of a modulus reduction function for bootstrapping of the CKKS scheme. I cast the problem of finding approximate polynomials for a modulus reduction into an L2-norm minimization problem for which the solution can be directly found without intermediates, such as a sine function. As the approximation error in the proposed method is not subject to the sine function, it approximates the modulus reduction better than the best-known method [20]. Using the Chebyshev polynomials as a basis, I achieved a lower approximation error even with a lower degree compared with the best-known method. Moreover, the proposed polynomial can utilize the baby-step giant-step algorithm [20] and Paterson-Stockmeyer algorithm [21]. I re-investigated the number of nonscalar multiplications, scalar multiplications, and additions needed for the baby-step giant-step algorithm and showed that the proposed polynomial reduces the required number of operations for the homomorphic approximate modulus reduction.

By casting the problem into a simple L2-norm optimization problem, I free the approximation problem from the sine function. The proposed method can offer a bootstrapping with fewer errors, particularly when a large scaling factor is selected. Thus, one can say that the choice of parameters has been expanded. Most importantly, the proposed method is essential for applications that require accurate approximation because the approximation error cannot be lowered when previous methods are used. In contrast, as the proposed method does not have such a lower bound, a better parameter can be selected. Consequently, bootstrapping consumes fewer levels when the proposed method is used.

I proposed loose upper and lower bounds, which were far from the numerical result. The challenge of a tighter bound or a better method for finding the minimax polynomial can be addressed in future work. In [20], the number of operations is reduced by using the double angle formula of the cosine function, but it is challenging to apply to the proposed method. A double angle formula-like approach for the proposed method also requires further study.

6.4 Modified pqsigRM: RM Code-Based Signature Scheme

I introduced a new signature scheme, called modified pqsigRM, based on modified RM codes with partial permutation as well as row appending and replacement in the generator matrix. For any given syndrome, an error vector with a small Hamming weight can be obtained. Moreover, the decoding method achieves indistinguishability to some degree because it is collision-resistant. The proposed signature scheme resists all known attacks against cryptosystems based on the original RM codes. The partially permuted RM code improves the signature success condition in previous signature schemes such as the CFS signature scheme and can improve signing time and key size.

I further modified the RM code using row appending/replacement. The resulting code is expected to be indistinguishable from random codes with the same hull dimension; moreover, the decoding of the partially permuted RM code is maintained. Assuming indistinguishability and the hardness of DOOM with a high-dimensional hull, I proved the EUF-CMA security of the proposed signature scheme. The challenge of rigorously verifying these two assumptions will be addressed in the future.

6.5 Cryptanalysis of the IKKR Cryptosystem

It was shown that the prototype IKKR cryptosystem is equal to the McEliece cryptosystem. The linearity of encryption allows the proposed cryptanalysis on the other two IKKR cryptosystems. The proposed attack algorithm runs in polynomial-time; further, although it depends on the implementation, it turns out that the attack algorithm is even faster than the decryption in [52]. It is worth noting that the error vectors in the McEliece-type cryptosystem should not have a certain structure; instead, it is desirable to use random error vectors. When designing or using variants of the McEliece cryptosystem, careful attention should be paid to the attacks using the structure error structure in addition to the key distinguishing attacks.

As the proposed attack algorithm finds the plaintext corresponding to the given ciphertext in polynomial-time only with the public key, even if conversion techniques such as the one in [85] is used, the modified version of prototype and the upgraded IKKR cryptosystems are vulnerable.

Bibliography

[1] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[2] C. Gentry, S. Halevi, and N. P. Smart, “Homomorphic evaluation of the AES circuit,” in *Proceedings of Annual International Cryptology Conference (Crypto)*, Santa Barbara, CA, USA, Aug. 2012, pp. 850–867.

[3] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” *SIAM J. of Comput.*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997.

[4] J. H. Cheon, A. Kim, M. Kim, and Y. Song, “Homomorphic encryption for arithmetic of approximate numbers,” in *Proceedings of International Conference on the Theory and Application of Cryptology and Information Security (Asiacrypt)*, Dec. 2017, pp. 409–437.

[5] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, “TFHE: Fast fully homomorphic encryption over the torus,” *Journal of Cryptology*, vol. 33, no. 1, pp. 34–91, Apr. 2020.

[6] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachene, “Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE,” in *Proceedings of International Conference on the Theory and Application of Cryptology and Information Security (Asiacrypt)*, Dec. 2017, pp. 377–408.

[7] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-LWE and security for key dependent messages," in *Proceedings of Annual International Cryptology Conference (Crypto)*, Santa Barbara, CA, USA, Aug. 2011, pp. 505–524.

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

[9] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) Fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, pp. 1–36, 2014.

[10] M. Albrecht, M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, S. Halevi, J. Hoffstein, K. Laine, K. Lauter, S. Lokam, D. Micciancio, D. Moody, T. Morrison, A. Sahai, and V. Vaikuntanathan, "Homomorphic Encryption Security Standard," HomomorphicEncryption.org, Toronto, Canada, Tech. Rep., Nov. 2018.

[11] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *IACR Cryptol. ePrint Arch.*, 2012.

[12] L. Ducas and D. Micciancio, "FHEW: bootstrapping homomorphic encryption in less than a second," in *Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt)*, Sofia, Bulgaria, Apr. 2015, pp. 617–640.

[13] J. H. Cheon, D. Kim, Y. Kim, and Y. Song, "Ensemble method for privacy-preserving logistic regression based on homomorphic encryption," *IEEE Access*, vol. 6, pp. 46 938–46 948, Aug. 2018.

[14] M. Kim, Y. Song, S. Wang, Y. Xia, and X. Jiang, "Secure logistic regression based on homomorphic encryption: Design and evaluation," *JMIR medical informatics*, vol. 6, no. 2, p. e19, Apr.-Jun. 2018.

[15] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, “Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy,” in *Proceedings of International Conference on Machine Learning*, New York City, NY, USA, Jun. 2016, pp. 201–210.

[16] E. Chou, J. Beal, D. Levy, S. Yeung, A. Haque, and L. Fei-Fei, “Faster cryptonets: Leveraging sparsity for real-world encrypted inference,” *arXiv preprint arXiv:1811.09953*, Nov. 2018.

[17] H. Chabanne, A. de Wargny, J. Milgram, C. Morel, and E. Prouff, “Privacy-preserving classification on deep neural network,” *IACR Cryptol. ePrint Arch.*, Mar. 2017.

[18] E. Hesamifard, H. Takabi, and M. Ghasemi, “Cryptodl: Deep neural networks over encrypted data,” *arXiv preprint arXiv:1711.05189*, 2017.

[19] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, “Bootstrapping for approximate homomorphic encryption,” in *Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt)*, Tel Aviv, Israel, Apr. 2018, pp. 360–384.

[20] K. Han and D. Ki, “Better bootstrapping for approximate homomorphic encryption,” in *Proceedings of Cryptographers’ Track at the RSA Conference*, San Francisco, CA, USA, Feb. 2020, pp. 364–390.

[21] H. Chen, I. Chillotti, and Y. Song, “Improved bootstrapping for approximate homomorphic encryption,” in *Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt)*, May 2019, pp. 34–54.

[22] K. Han, M. Han, and J. H. Cheon, “Improved homomorphic discrete Fourier transforms and FHE bootstrapping,” *IEEE Access*, vol. 7, pp. 57 361–57 370, Apr. 2019.

[23] A. Kim, A. Papadimitriou, and Y. Polyakov, "Approximate homomorphic encryption with reduced approximation error," *IACR Cryptol. ePrint Arch*, Oct. 2020.

[24] Y. Lee, J.-W. Lee, Y.-S. Kim, and J.-S. No, "Near-optimal polynomial for modulus reduction using l2-norm for approximate homomorphic encryption," *IEEE Access*, vol. 8, pp. 144 321–144 330, Aug. 2020.

[25] J.-W. Lee, E. Lee, Y. Lee, Y.-S. Kim, and J.-S. No, "High-precision bootstrapping of RNS-CKKS homomorphic encryption using optimal minimax polynomial approxiamtion and inverse sine function," *IACR Cryptol. ePrint Arch*, Oct. 2020.

[26] E. Berlekamp, R. McEliece, and H. Van Tilborg, "On the inherent intractability of certain coding problems," *IEEE Transactions on Information Theory*, vol. 24, no. 3, pp. 384–386, 1978.

[27] R. J. McEliece, "A public-key cryptosystem based on algebraic," *DSN Progress Report*, vol. 4244, pp. 114–116, 1978.

[28] A. Otmani, J.-P. Tillich, and L. Dallot, "Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes," *Special Issues of Mathematics in Computer Science*, vol. 3, no. 2, pp. 129–140, 2010.

[29] J.-C. Faugere, V. Gauthier-Umana, A. Otmani, L. Perret, and J.-P. Tillich, "A distinguisher for high-rate McEliece cryptosystems," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6830–6844, 2013.

[30] Y. Lee, Y.-S. Kim, and J.-S. No, "Ciphertext-only attack on linear feedback shift register-based Esmaeili-Gulliver cryptosystem," *IEEE Communications Letters*, vol. 21, no. 5, pp. 971–974, May 2017.

[31] Round 2 submissions. https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions

[32] T. Debris-Alazard, N. Sendrier, and J.-P. Tillich, "Wave: A new family of trapdoor one-way preimage sampleable functions based on codes," in *Proceedings of International Conference on the Theory and Application of Cryptology and Information Security (Asiacrypt)*, Dec. 2019, pp. 21–51.

[33] N. Aragon, O. Blazy, P. Gaborit, A. Hauteville, and G. Zémor, "Durandal: A rank metric based signature scheme," in *Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt)*, May 2019, pp. 728–758.

[34] Y. Lee, W. Lee, Y. S. Kim, and J.-S. No, "Modified pqsigRM: RM code-based signature scheme," *IEEE Access*, vol. 8, pp. 177 506–177 518, Sep. 2020.

[35] I. S. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *Transactions of the IRE Professional Group on Information Theory*, vol. 4, no. 4, pp. 38–49, Sep. 1954.

[36] D. E. Muller, "Application of boolean algebra to switching circuit design and to error detection," *Transactions of the IRE professional group on electronic computers*, vol. EC-3, no. 3, pp. 6–12, 1954.

[37] W. Lee, Y.-S. Kim, Y. Lee, and J.-S. No, "Post-quantum signature scheme based on modified Reed-Muller code (pqsigRM)," in *First Round Submission to the NIST Postquantum Cryptography Call*, Fort Lauderdale, FL, USA, Nov. 2017.

[38] N. T. Courtois, M. Finiasz, and N. Sendrier, "How to achieve a McEliece-based digital signature scheme," in *Proceedings of International Conference on the Theory and Application of Cryptology and Information Security (Asiacrypt)*, Gold Coast, Australia, Dec. 2001, pp. 157–174.

[39] L. Dallot, "Towards a concrete security proof of Courtois, Finiasz and Sendrier signature scheme," in *Proceedings of Western European Workshop on Research in Cryptology*, Bochum, Germany, 2007, pp. 65–77.

[40] K. Morozov, P. S. Roy, R. Steinwandt, and R. Xu, "On the security of the Courtois-Finiasz-Sendrier signature," *Open Mathematics*, vol. 16, no. 1, pp. 161–167, Mar. 2018.

[41] M. Baldi, M. Bianchi, F. Chiaraluce, J. Rosenthal, and D. Schipani, "Using LDGM codes and sparse syndromes to achieve digital signatures," in *Proceedings of International Workshop on Post-Quantum Cryptography (PQCrypto)*, Limoges, France, 2013, pp. 1–15.

[42] D. Gligoroski, S. Samardjiska, H. Jacobsen, and S. Bezzateev, "McEliece in the world of Escher," *IACR Cryptol. ePrint Arch.*, 2014.

[43] A. Phesso and J.-P. Tillich, "An efficient attack on a code-based signature scheme," in *Proceedings of International Conference on Post-Quantum Cryptography (PQCrypto)*, Fukuoka, Japan, Feb. 2016, pp. 86–103.

[44] D. Moody and R. Perlner, "Vulnerabilities of "McEliece in the World of Escher"," in *Proceedings of International Conference on Post-Quantum Cryptography (PQCrypto)*, Fukuoka, Japan, Feb. 2016, pp. 104–117.

[45] G. Kabatianskii, E. Krouk, and B. Smeets, "A digital signature scheme based on random error-correcting codes," in *Proceedings of IMA International Conference on Cryptography and Coding*, Cirencester, United Kingdom, Dec. 1997, pp. 161–167.

[46] G. Kabatiansky, E. Krouk, and S. Semenov, *Error correcting coding and security for data networks: Analysis of the Superchannel Concept*. Wiley, Mar. 2005.

[47] P. S. Barreto, R. Misoczki, and M. A. Simplicio Jr, "One-time signature scheme from syndrome decoding over generic error-correcting codes," *Journal of Systems and Software*, vol. 84, no. 2, pp. 198–204, 2011.

[48] P.-L. Cayrel, A. Otmani, and D. Vergnaud, “On Kabatianskii-Krouk-Smeets signatures,” in *Proceedings of International Workshop on the Arithmetic of Finite Fields*, Madrid, Spain, June, 2007, pp. 237–251.

[49] A. Otmani and J.-P. Tillich, “An efficient attack on all concrete KKS proposals,” in *Proceedings of International Workshop on Post-Quantum Cryptography (PQCrypto)*, Taipei, Taiwan, Jul. 2011, pp. 98–116.

[50] T. Debris-Alazard, N. Sendrier, and J.-P. Tillich, “SURF: A new code-based signature scheme,” *arXiv preprint arXiv:1706.08065*, 2017.

[51] C. Gentry, C. Peikert, and V. Vaikuntanathan, “Trapdoors for hard lattices and new cryptographic constructions,” in *Proceedings of the fortieth annual ACM symposium on Theory of computing*, Victoria, British Columbia, May 2008, pp. 197–206.

[52] F. Ivanov, G. Kabatiansky, E. Krouk, and N. Rumenko, “A new code-based cryptosystem,” in *Proceedings of Code-Based Cryptography Workshop (affiliated with Eurocrypt 2020)*, May 2020, pp. 41–49.

[53] A. C. Yao, “Protocols for secure computations,” in *Proceedings of Annual Symposium on Foundations of Computer Science*, Jul. 1982, pp. 160–164.

[54] O. Goldreich, S. Micali, and A. Wigderson, “How to play any mental game, or a completeness theorem for protocols with honest majority,” in *Proceedings of ACM Symposium on Theory of Computing*, Jan. 1987, pp. 218–229.

[55] F. Boemer, Y. Lao, R. Cammarota, and C. Wierzynski, “nGraph-HE: A graph compiler for deep learning on homomorphically encrypted data,” in *Proceedings of the 16th ACM International Conference on Computing Frontiers*, Alghero, Italy, Apr. 2019, pp. 3–13.

[56] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, “A full RNS variant of approximate homomorphic encryption,” in *Proceedings of International Conference on Selected Areas in Cryptography*, Calgary, Canada, Aug. 2018, pp. 347–368.

[57] J. C. Mason and D. C. Handscomb, *Chebyshev Polynomials.* FL, USA: CRC Press, 2002, ch. Chebyshev interpolation, pp. 154–172.

[58] I. Dumer, “Recursive decoding and its performance for low-rate Reed-Muller codes,” *IEEE Transactions on Information Theory*, vol. 50, no. 5, pp. 811–823, 2004.

[59] F. Boemer, A. Costache, R. Cammarota, and C. Wierzynski, “nGraph-HE2: A high-throughput framework for neural network inference on encrypted data,” in *Proceedings of the ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, Nov. 2019, pp. 45–56.

[60] A. C.-C. Yao, “How to generate and exchange secrets,” in *Proceedings of 27th Annual Symposium on Foundations of Computer Science*, 1986, pp. 162–167.

[61] Y. Lee, J.-W. Lee, Y.-S. Kim, H. Kang, and J.-S. No, “High-precision approximate homomorphic encryption by error variance minimization,” *IACR Cryptol. ePrint Arch*, 2020.

[62] J.-P. Bossuat, C. Mouchet, J. Troncoso-Pastoriza, and J.-P. Hubaux, “Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys,” *IACR Cryptol. ePrint Arch*, Oct. 2020.

[63] E. Lee, J.-W. Lee, J.-S. No, and Y.-S. Kim, “Minimax approximation of sign function by composite polynomial for homomorphic comparison,” *IACR Cryptol. ePrint Arch*, 2020.

[64] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[65] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” *arXiv preprint arXiv:1607.06450*, 2016.

[66] C. Lattner and V. Adve, “LLVM: a compilation framework for lifelong program analysis & transformation,” in *Proceedings of International Symposium on Code Generation and Optimization*, Palo Alto, CA, USA, Mar. 2004, pp. 75–86.

[67] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proceedings of International Conference on Machine Learning*, 2015, pp. 448–456.

[68] T. Salimans and D. P. Kingma, “Weight normalization: A simple reparameterization to accelerate training of deep neural networks,” *arXiv preprint arXiv:1602.07868*, 2016.

[69] J. H. Cheon, D. Kim, D. Kim, H. H. Lee, and K. Lee, “Numerical method for comparison on homomorphically encrypted numbers,” in *Proceedings of International Conference on the Theory and Application of Cryptology and Information Security (Asiacrypt)*, Dec. 2019, pp. 415–445.

[70] N. Sendrier, “Decoding one out of many,” in *Proceedings of International Workshop on Post-Quantum Cryptography (PQCrypto)*, Taipei, Taiwan, Jul. 2011, pp. 51–67.

[71] F. Hemmati, “Closest coset decoding of $U|U+V$ codes,” *IEEE Journal on Selected Areas in Communications*, vol. 7, no. 6, pp. 982–988, Jan. 1989.

[72] L. Minder and A. Shokrollahi, “Cryptanalysis of the Sidelnikov cryptosystem,” in *Proceedings of Annual International Conference on the Theory and Applications*

of Cryptographic Techniques (Eurocrypt), Barcelona, Spain, May. 2007, pp. 347–360.

[73] I. V. Chizhov and M. A. Borodin, "The failure of McEliece PKC based on Reed-Muller codes," *IACR Cryptol. ePrint Arch.*, 2013.

[74] A. Otmani and H. T. Kalachi, "Square code attack on a modified Sidelnikov cryptosystem," in *Proceedings of International Conference on Codes, Cryptology, and Information Security*, Rabat, Morocco, May 2015, pp. 173–183.

[75] J. Stern, "A method for finding codewords of small weight," in *Proceedings of International Colloquium on Coding Theory and Applications*, 1988, pp. 106–113.

[76] I. Dumer, "On minimum distance decoding of linear codes," in *Proceedings of 5th Joint Soviet-Swedish Int. Workshop Inform. Theory*, 1991, pp. 50–52.

[77] A. Becker, A. Joux, A. May, and A. Meurer, "Decoding random binary linear codes in $2^{n/20}$: How 1+1= 0 improves information set decoding," in *Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt)*, Cambridge, United Kingdom, Apr. 2012, pp. 520–536.

[78] A. May and I. Ozerov, "On computing nearest neighbors with applications to decoding of binary linear codes," in *Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt)*, Sofia, Bulgaria, Apr. 2015, pp. 203–228.

[79] B. Dou, C.-H. Chen, and H. Zhang, "Key substitution attacks on the CFSsignature," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 95, no. 1, pp. 414–416, 2012.

[80] N. Sendrier, “On the dimension of the hull,” *SIAM Journal on Discrete Mathematics*, vol. 10, no. 2, pp. 282–293, 1997.

[81] V. Shoup, “Sequences of games: A tool for taming complexity in security proofs,” *IACR Cryptol. ePrint Arch.*, 2004.

[82] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker, “A statistical test suite for random and pseudorandom number generators for cryptographic applications,” National Institute of Standards and Technology, Tech. Rep., 2001.

[83] M. Finiasz, “Parallel-CFS,” in *Proceedings of International Workshop on Selected Areas in Cryptography*, Waterloo, Ontario, Canada, Aug. 2010, pp. 159–170.

[84] D. Wagner, “A generalized birthday problem,” in *Proceedings of Annual International Cryptology Conference (Crypto)*, Santa Barbara, CA, USA, Aug. 2002, pp. 288–304.

[85] K. Kobara and H. Imai, “Semantically secure McEliece public-key cryptosystems-conversions for McEliece PKC,” in *Proceedings of International Workshop on Public Key Cryptography*, Cheju Island, Korea, Feb. 2001, pp. 19–35.

초 록

본 논문은 크게 다음의 세 가지의 기여를 포함한다. i) 네트워크를 활용해서 정보 보호 딥러닝을 개선하는 프로토콜 ii) 근사 동형 암호에서 보안성과 성능의 손해 없이 에러를 낮추고 높은 정확도로 부트스트래핑 하는 방법 iii) IKKR 암호 시스템과 pqsigRM 등 부호 기반 암호를 공격하는 방법과 효율적인 부호 기반 전자 서명 시스템.

근래의 기계학습과 블록체인 기술의 발전으로 인해서 기밀 데이터에 대한 연산을 어떻게 외주할 수 있느냐에 대한 새로운 보안 문제가 대두되고 있다. 또한, 양자 컴퓨터에 관한 연구가 성공을 거듭하면서, 이를 이용한 공격에 저항하는 포스트 양자 암호의 필요성 또한 커지고 있다. 다자간 컴퓨팅은 데이터를 공개하지 않고 데이터에 대한 연산을 수행할 수 있도록 하는 암호학적 프로토콜의 총칭이다. 다자간 컴퓨팅은 동형 암호와 포스트 양자 암호에 기반하고 있으므로, 효율적인 동형 암호와 포스트 양자 암호에 관한 연구가 활발하게 수행되고 있다.

동형 암호는 암호화된 데이터에 대한 연산이 가능한 특수한 암호화 알고리즘이다. 일반적으로 동형 암호의 암호문에 대해서 수행 가능한 연산의 깊이가 정해져 있으며, 이를 암호문의 레벨이라고 칭한다. 레벨을 모두 소비한 암호문의 레벨을 다시 복원하는 과정을 부트스트래핑 (bootstrapping)이라고 칭한다. 일반적으로 부트스트래핑은 매우 오래 걸리는 연산이며 시간 및 공간 복잡도가 크다. 그러나, 딥러닝과 같이 깊이가 큰 연산을 수행하는 경우 부트스트래핑이 필수적이다. 본 논문에서는 정보 보호 기계학습을 위한 새로운 프로토콜을 제안한다. 이 프로토콜에서는 입력 메시지와 더불어 신경망의 중간값들 또한 안전하게 보호된다. 그러나 여전히 사용

자의 통신 및 연산 복잡도는 낮게 유지된다.

Cheon, Kim, Kim 그리고 Song (CKKS)가 제안한 암호 시스템 (Asiacrypt' 17)은 기계학습 등에서 가장 널리 쓰이는 데이터인 실수를 효율적으로 다룰 수 있으므로 가장 촉망받는 완전 동형 암호 시스템이다. 그러나, 오류의 증폭과 전파가 CKKS 암호 시스템의 가장 큰 단점이다. 이 논문에서는 아래의 기술을 활용하여 CKKS 암호 시스템의 오류를 줄이는 방법을 제안하며, 이는 근사 동형 암호에 일반화하여 적용할 수 있다. 첫째, 신호 대비 잡음 비 (signal-to-noise ratio, SNR)의 개념을 도입하여, SNR를 최대화하도록 연산의 순서를 재조정한다. 그러기 위해서는, 오류의 최대치 대신 분산이 최소화되어야 하며, 이를 관리해야 한다. 둘째, 오류의 분산을 최소화한다는 같은 관점에서 새로운 다항식 근사 방법을 제안한다. 이 근사 방법은 특히, CKKS 암호 시스템의 부트스트래핑에 적용되었으며, 종래 기술보다 더 낮은 오류를 달성한다. 위의 방법에 더하여, 근사 다항식을 구하는 문제를 L2-norm 최소화 문제로 치환하는 방법을 제안한다. 이를 통해서 사인 함수의 도입 없이 근사 다항식을 구하는 방법을 제안한다. 제안된 방법을 사용하면, $q = O(m^{3/2})$라는 제약을 $q = O(m)$으로 줄일 수 있으며, 부트스트래핑에 필요한 레벨 소모를 줄일 수 있다. 성능 향상은 HEAAN과 SEAL 등의 동형 암호 라이브러리를 활용한 구현을 통해 증명했으며, 구현을 통해서 연산 재정렬과 새로운 부트스트래핑이 CKKS 암호 시스템의 성능을 향상함을 확인했다. 따라서, 보안성과 성능의 타협 없이 근사 동형 암호를 사용하는 서비스의 질을 향상할 수 있다.

양자 컴퓨터를 활용하여 전통적인 공개키 암호를 공격하는 효율적인 알고리즘이 공개되면서, 포스트 양자 암호에 대한 필요성이 증대했다. 부호 기반 암호는 포스트 양자 암호로써 널리 연구되었다. 작은 키 크기를 갖는 새로운 부호 기반 전자 서명 시스템과 부호 기반 암호를 공격하는 방법이 논문에 제안되어 있다. pqsigRM이라 명명한 전자 서명 시스템이 그것이다. 이 전자 서명 시스템은 수정된 Reed-Muller (RM) 부호를 활용하며, 서명의 복잡도와 키 크기를 종래 기술보다 많이 줄인다. pqsigRM은 hull의 차원이 큰 $(U, U + V)$ 부호와 이의 복호화를 이용하여, 서명에서 큰 이득이 있다. 이 복호화 알고리즘은 주어진 모든 코셋 (coset)의 원소에 대하여 작은 헤밍 무게를 갖는 원소를 반환한다. 또한, 수정된 RM 부호를 이용하여, 알려진

모든 공격에 저항한다. 128비트 안정성에 대해서 서명의 크기는 4096 비트이고, 공개 키의 크기는 1MB보다 작다. 최근, Ivanov, Kabatiansky, Krouk, 그리고 Rumenko (IKKR)가 McEliece 암호 시스템의 세 가지 변형을 발표했다 (CBCrypto 2020, Eurocrypt 2020와 함께 진행). 본 논문에서는 IKKR 암호 시스템중 하나가 McEliece 암호 시스템과 동치임을 증명한다. 또한 나머지 IKKR 암호 시스템에 대한 다항 시간 공격을 제안한다. 제안하는 공격은 IKKR 암호 시스템의 선형성을 활용한다. 또한, 이 논문은 제안한 공격의 구현을 포함하며, 제안된 공격은 0.2초 이내에 메시지를 복원하고, 이는 정상적인 복호화보다 빠른 속도이다.

주요어: 격자 기반 암호, 공개 키 암호, 근사 동형 암호, 부트스트래핑, 부호 기반 암호, 암호 공격, 암호학, 완전 동영 암호, 오류 정정 부호, 전자 서명, 정보 보호, 정보 보호 머신 러닝, McEliece 암호 시스템, 포스트 양자 암호, CKKS 암호시스템, Reed-Muller (RM) 부호
학번: 2017-34837

ACKNOWLEGEMENT

This dissertation is a winner of the Distinguished Dissertation Award.