High-Precision Bootstrapping of RNS-CKKS Homomorphic Encryption Using Optimal Minimax Polynomial Approxiamtion and Inverse Sine Function

Joon-Woo Lee[1], Eunsang Lee[1], Yongwoo Lee[1], Young-Sik Kim[2], and Jong-Seon No[1]

[1] Seoul National University, Republic of Korea
[2] Chosun University, Republic of Korea

Abstract. Approximate homomorphic encryption with the residue number system (RNS), called RNS-variant Cheon-Kim-Kim-Song (RNS-CKKS) scheme [13, 14], is a practical fully homomorphic encryption scheme that supports arithmetic operations for real or complex number data encrypted. Although the RNS-CKKS scheme is a fully homomorphic encryption scheme, most of the applications with RNS-CKKS scheme use it as only leveled homomorphic encryption scheme because of the lack of the practicality of the bootstrapping operation of the RNS-CKKS scheme. One of the crucial problems of the bootstrapping operation is its poor precision. While other basic homomorphic operations ensure sufficiently high precision for practical use, the bootstrapping operation only supports about 20-bit fixed-point precision at best, which is lower than the standard single-precision of the floating-point number system. Due to this limitation of the current bootstrapping technology, the RNS-CKKS scheme is difficult to be used for the reliable deep-depth homomorphic computations until now.
In this paper, we improve the message precision in the bootstrapping operation of the RNS-CKKS scheme. Since the homomorphic modular reduction process is one of the most important steps in determining the precision of the bootstrapping, we focus on the homomorphic modular reduction process. Firstly, we propose a fast algorithm of obtaining the optimal minimax approximate polynomial of modular reduction function and the scaled sine/cosine function over the union of the approximation regions, called the modified Remez algorithm. In fact, this algorithm derives the optimal minimax approximate polynomial of any continuous functions over any union of the finite number of intervals. Next, we propose the composite function method using inverse sine function to reduce the difference between the scaling factor used in the bootstrapping and the default scaling factor. With these methods, we reduce the approximation error in the bootstrapping of the RNS-CKKS scheme by 1/200$\sim$1/53 (5.8$\sim$7.6-bit precision improvement) for each parameter setting. While the bootstrapping without the composite function method has 16.5$\sim$20.1-bit precision at maximum, the bootstrapping with the composite function method has 22.4$\sim$27.8-bit precision, most of which are better precision than that of the single-precision number system.

Keywords: Approximate homomorphic encryption · Bootstrapping · Composite function approximation · Fully homomorphic encryption (FHE) · Inverse sine function · Minimax approximate polynomial · Modified Remez algorithm · RNS-variant Cheon-Kim-Kim-Song (RNS-CKKS) scheme

1 Introduction

Fully homomorphic encryption (FHE) is the encryption scheme enabling any logical operations [7, 16, 18, 21, 31] or arithmetic operations [13, 14] with encrypted data. The FHE scheme makes it possible to preserve security in the data processing. However, in the traditional encryption schemes, they are not encrypted to enable the processing of encrypted data, which causes clients to be dissuaded from receiving services and prevent companies from developing various related systems because of the lack of clients' privacy. FHE solves this problem clearly so that clients can receive many services by ensuring their privacy.

First, Gentry constructed the FHE scheme by coming up with the idea of bootstrapping [20]. After this idea was introduced, cryptographers constructed many FHE schemes using bootstrapping. Approximate homomorphic encryption, which is also called a Cheon-Kim-Kim-Song (CKKS) scheme [14], is one of the promising FHE schemes, which deals with any real and complex numbers. The CKKS scheme is particularly in the spotlight for much potential power in many applications such as machine learning [4–6, 8, 17, 25], in that data is usually represented by real numbers. Lots of researches for the optimization of the CKKS scheme have been done actively for practical use. Cheon et al. proposed the residue number system (RNS) variant CKKS scheme (RNS-CKKS) [13] so that the necessity of the arbitrary precision library can be removed and only use the word-size operations. The running time of the homomorphic operations in the RNS-CKKS scheme is 10 times faster than that of the original CKKS scheme with the single thread, and further, the RNS-CKKS scheme has an advantage in parallel computation, which leads to much better running time performance with the multi-core environment. Because of the fast homomorphic operations, most homomorphic encryption libraries, including `SEAL` [30] and `PALISADE` [2], are implemented using the RNS-CKKS scheme. Thus, we focus on the RNS-CKKS scheme in this paper.

Due to the fact that the CKKS scheme includes errors used to ensure the security as the approximate error in the message and the RNS-CKKS scheme has to use approximate rescaling procedure, the use of the RNS-CKKS scheme requires more sensitivity to the precision of the message than other homomorphic encryption schemes that support accurate decryption and homomorphic evaluation. This can be more sensitive for deep-depth homomorphic operations because errors are likely to be amplified by the operations and distort the data significantly. Fortunately, the basic homomorphic operations in the RNS-CKKS scheme can ensure sufficiently high precision for practical use, but its bootstrapping operation is not the case. Ironically, while the bootstrapping operation in other homomorphic encryption schemes reduces the effect of the errors on messages so that they do not distort messages, the bootstrapping operation in the CKKS scheme amplify the errors, which makes it the most major cause of data distortion among any other homomorphic operations in the RNS-CKKS scheme. Since advanced operations with large depth may require bootstrapping operation many times, the message precision problem in the bootstrapping operation is a crucial obstacle to applying the RNS-CKKS scheme to the advanced applications.

Although the RNS-CKKS scheme is currently one of the most potential solutions to implement privacy-preserving machine learning (PPML) system [4,5,17], the methods for the PPML studied so far have mainly been applied to simple models such as MNIST, which has so low depth that the bootstrapping is not required. Thus, the message precision problem in the bootstrapping operation in the RNS-CKKS scheme did not need to be considered in the PPML model until now. However, the advanced machine learning model currently presented requires a large depth, and thus we should introduce the bootstrapping operation and cannot avoid the message precision problem in the bootstrapping operation. Of course, the fact that the bootstrapping requires long running time and large depth than other homomorphic operations are also pointed out as a major limitation of the bootstrapping. While these points may be able to be improved by simple parameter adjustments and using hardware optimization, the message precision problem in the bootstrapping is difficult to solve with these simple methods.

Most of the works about PPML with FHE focused on the inference process rather than the training process because of the long running time. However, training neural networks with encrypted data is actually more important from a long-term perspective for solving the real security problem in machine learning, in that the companies cannot gather sufficiently many important but sensitive data, such as genetic or financial information, so that they cannot construct the deep learning model for them because of the privacy of the data owners. While the inference process does not need a high precision number system, the training process is affected sensitively by the precision of the number system. Chen et al. [10] showed that CNN network learning MNIST cannot converge when the model is trained using a 16-bit fixed-point number system. When the 32-bit fixed-point number system is used to train the CNN network with MNIST, the training performance was slightly lower than the case of using the single-precision floating-point number system, although all bits except one bit representing the sign are used to represent the data in 32-bit fixed-point number system, that is, it has 31-bit precision, which is much better precision than the single-precision floating point number system, which is 23-bit precision. Although many works proposed to use low-precision fixed-point numbers in the training procedure, they used additional special techniques, such as the stochastic rounding [22] or the dynamic fixed-point number system [23], which cannot be supported by the RNS-CKKS scheme until now.

While most of the deep learning systems use the single-precision floating-point numbers, the maximum precision achieved with the bootstrapping of the

CKKS scheme in the previous papers was about only 20 bits. Considering that the CKKS scheme only supports the fixed-point arithmetic, the 20-bit precision is not large enough to be applied wholly to the deep learning system. Thus, in order to apply the RNS-CKKS scheme to deep learning systems, it is necessary to achieve a precision sufficiently better than the 23-bit precision guaranteed by the single-precision floating-point number system, which requires a breakthrough for the bootstrapping in the RNS-CKKS scheme with regard to its precision.

One of the most important steps where the bootstrapping affects message precision is the homomorphic modular reduction process. The crucial point is that the difference between the modular reduction function and the sine/cosine function gives a significant precision loss. All previous works have used methods that approximate the modular reduction function as a part of the sine/cosine functions. This approximation has an inherent approximation error so that the limitation of the precision occurs. In addition, to ensure that these two functions are significantly close to each other, the approximate region has to be reduced a lot. They set the half-width of one interval in the approximation region as 2^{-10}, which is equal to the ratio of default scaling factor to the scaling factor used in the bootstrapping. The message has to be scaled by multiplying 2^{-10} to make the message into the approximation region, and it is scaled by multiplying 2^{10} at the end of the bootstrapping. Thus, the precision error in the computation for bootstrapping is amplified by 2^{10}, and the 10-bit precision loss occurs. If we try to reduce this precision loss by enlarging the approximation region, the approximation error by the sine/cosine function becomes large, and thus the overall precision becomes lower than before.

Another problem in the homomorphic modular reduction process is that the polynomial approximation method so far does not guarantee theoretically optimal minimax approximate polynomial. Since the modular reduction operation is a non-arithmetic operation, it should be approximated in a polynomial to be homomorphically computed with homomorphic arithmetic operations, that is, the homomorphic addition and the homomorphic multiplication. Although previous works have suggested methods to obtain polynomials that approximate the scaled sine/cosine function well from the minimax perspective, which is used to approximate the modular reduction function, these methods cannot obtain the optimal minimax approximate polynomial.

1.1 Previous Works

The CKKS scheme was firstly proposed in [14] without bootstrapping, which was a somewhat homomorphic encryption scheme supporting only the finite number of multiplications. Cheon et al. [12] firstly suggested bootstrapping operation with the homomorphic linear transformation enabling transformation between slots and coefficients, and approximation of homomorphic modulus reduction function as the sine function. In homomorphic modular reduction, they approximated the sine function by evaluating its Taylor approximation and applying the double-angle formula. Although the double-angle formula reduces the number of operations compared to the direct Taylor approximation, it requires large depths. Chen et al. [9] improved the running time of bootstrapping operation by making homomorphic linear transformation and homomorphic modular reduction efficient. They applied a modified fast Fourier transform (FFT) algorithm to evaluate homomorphic linear transformation and used Chebyshev interpolation and Paterson-Stockmeyer algorithm to approximate the sine function. Chebyshev interpolation is known as a good sub-optimal approximate polynomial in the minimax aspect, but this is not the optimal approximate polynomial. Han et al. [24] improved the homomorphic modular reduction in the bootstrapping operation. While Chen et al. approximated the sine function in one interval, Han et al. approximated the cosine function only in the separated approximation regions, which enables to reduce the degree of polynomials and use simpler double-angle formula than that of the sine function, but their approximate polynomial is also not optimal in the minimax aspect.

There is an example of using the exact minimax approximate polynomial of sign function for comparison operation in the CKKS scheme. That is, Cheon et al. [15] used an algorithm to obtain the minimax approximate polynomial only for sign function to accelerate their composition method for homomorphic comparison operations. However, they only used the well-known minimax approximation algorithm for sign function as a subroutine. There is no research to use the exact minimax approximate polynomial for bootstrapping of the CKKS scheme.

On the other hand, the RNS-CKKS scheme was proposed. Since big integers used to represent the ciphertexts in the CKKS scheme cannot be stored with the basic data type, the original CKKS scheme had to resort to the arbitrary precision data type libraries, such as the number theory library (NTL) library. To remove the reliance on the external libraries for performance improvement, Cheon et al. applied the RNS system in the CKKS scheme. Most practical homomorphic encryption libraries, such as `SEAL` and `PALISADE`, implement the RNS-CKKS scheme. The bootstrapping with the RNS-CKKS scheme is only dealt in [24] yet.

The Remez algorithm [28], also called the Remez exchange algorithm, is an iterative algorithm that obtains the minimax approximate polynomial for a given continuous function on an interval $[a, b]$. There is its variant that obtains the minimax approximate polynomial for a given continuous function on the multiple sub-intervals of an interval [19, 26, 28], which is less well-known than the original one. There is a crucial difference between the two algorithms in determining the new set of reference points in each iteration, which is used to construct an approximate polynomial in the next iteration. While the new set of reference points can be chosen naturally in the original one for an interval, there are many candidate points for the new set of reference points in the variant for the multiple sub-intervals of an interval. The variant algorithm chooses the new set of reference points which alternates in the sign of error and includes the global extreme point. Although this selection method ensures the convergence to the minimax approximate polynomial, there are yet many candidate points for the new set of reference points satisfying these criteria. The Parks-McClellan filter design algorithm uses this variant of the Remez algorithm to design the optimal filter for a given condition, where its approximation domain is usually the union of two or three intervals.

1.2 Our Contribution

In this paper, we propose two methods to improve the message precision in the bootstrapping operation of the RNS-CKKS scheme. Firstly, we devise a fast algorithm of obtaining the optimal minimax approximate polynomial of modular reduction function and the scaled sine/cosine function over the union of the approximation regions, called the modified Remez algorithm. In fact, this algorithm derives the optimal minimax approximate polynomial of any continuous functions over any union of the finite number of intervals. Although the original variant of the Remez algorithm can obtain the optimal minimax approximate polynomial, the running time to obtain the polynomial is quite long and unstable, that is, the running time varies from time to time. To make the variant of the Remez algorithm practically, we modify the variant of the Remez algorithm, called the modified Remez algorithm. Since it can obtain the optimal minimax approximate polynomial in seconds, we can even adaptively obtain the polynomial when we abruptly change some parameters on processing the ciphertexts so that we have to update the approximate polynomial. All polynomial approximation method proposed in previous works for bootstrapping in the CKKS scheme can be replaced with the modified Remez algorithm, which ensures the best quality of the approximation.

Next, we propose the composite function method to reduce the difference between the scaling factor used in the bootstrapping and the default scaling factor. To reduce this difference, we have to enlarge the half-width of one interval in the approximate region, but this can make the difference between the modular reduction function and the sine/cosine function quite large. We solve this problem by composing the optimal approximate polynomial of the inverse sine function, since composing the inverse sine function to the sine/cosine function yields a part of the exact modular reduction function. Note that the inverse sine function we use has only one interval in the approximate region, and thus we can reach the small approximate error with low degree polynomials. We obtain the minimax approximate polynomials for the scaled cosine function and the inverse sine function with sufficiently small minimax error by the modified Remez algorithm, and we apply these polynomials in the homomorphic modular reduction process by homomorphically evaluating the approximate polynomial for the scaled cosine function, several double-angle formula, and the approximate polynomial for the inverse sine function. This enables us to minimize the inevitable precision loss by approximating the modular reduction function to the sine/cosine function.

In fact, these techniques can be applied in both the original variant and the RNS-variant of the CKKS scheme. Since the RNS-CKKS scheme is more practical but more vulnerable to the approximation errors than the original version, the additional analysis of the RNS-CKKS scheme should be dealt with for high-precision bootstrapping in the RNS-CKKS scheme. We deal with the relation between the scaling factor and the precision in the RNS-CKKS scheme. The rescaling procedure in the RNS-CKKS scheme actually rescales the ciphertext by the prime number used for one of the moduli in the RNS form instead of the exact scaling factor, and thus this makes the inherent upper bound of the precision of the RNS-CKKS scheme. This inherent precision bound is related to the scaling factor by the well-known prime number theorem, and it suggests that we have to use large scaling factors for ensuring precise homomorphic operations. This analysis also supports the necessity of the composite function method because it ensures the large default scaling factor by reducing the difference of the scaling factors, and thus the default precision for basic homomorphic operations is ensured to be large enough. With the proposed methods, we reduce the approximation error in the bootstrapping of the RNS-CKKS scheme by 1/200∼1/53 (5.8∼7.6-bit precision improvement) for each parameter setting. While the bootstrapping without the composite function method has 16.5∼20.1-bit precision at maximum, the bootstrapping with the composite function method has 22.4∼27.8-bit precision, most of which are better precision than that of the single-precision number system.

1.3 Outline

The outline of the paper is given as follows. Section 2 deals with some preliminaries for the RNS-CKKS scheme, approximation theory, and the Remez algorithm. In Section 3, we propose a modified Remez algorithm for obtaining the optimal minimax approximate polynomial. The theoretical and numerical relation between the message precision and several parameters in the RNS-CKKS scheme is dealt with in Section 4, and the upper bound of the message precision in the bootstrapping of the RNS-CKKS scheme is also included. In Section 5, we propose the composite function method which makes it possible to reduce the difference of the two scaling factors in default operations and in bootstrapping operations, and numerically shows the improvement of the message precision in the proposed bootstrapping operation in the RNS-CKKS scheme. Section 6 concludes the paper and discusses future works.

2 Preliminary

2.1 Notation

Let $\mathbb{Z}, \mathbb{Q}, \mathbb{R}$, and $\mathbb{C}$ be sets of integers, rational numbers, real numbers, and complex numbers, respectively. Let $C[D]$ be a set of continuous functions on the domain D. Let $[d]$ be a set of positive integers less than or equal to d, i.e., $\{1, 2, \cdots, d\}$. Let $\mathsf{round}(x)$ be the function that outputs the integer nearest to x, and we do not have to consider the case of tie in this paper. For a power of two, M, let $\Phi_M(X) = X^N + 1$ be an M-th cyclotomic polynomial, where $M = 2N$. Let $\mathcal{R} = \mathbb{Z}[X]/\langle \Phi_M(X) \rangle$ and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. Let $\mathbb{Q}[X]/\langle \Phi_M(X) \rangle$ be a M-th cyclotomic field. For positive real number α, $\mathcal{DG}(\alpha)$ is defined as the distribution in

$\mathbb{Z}^N$ whose entries are sampled independently from discrete Gaussian distribution of variance α^2. $\mathcal{HWT}(h)$ is a subset of $\{0, \pm 1\}^N$ with Hamming weight h. $\mathcal{ZO}(\rho)$ is the distribution in $\{0, \pm 1\}^N$ whose entries are sampled independently with probability $\rho/2$ for each of ± 1 and probability being zero, $1 - \rho$. The Chebyshev polynomials $T_n(x)$ are defined by $\cos n\theta = T_n(\cos\theta)$. The remainder of a divided by q is denoted as $[a]_q$. If $\mathcal{C} = \{q_0, q_1, \cdots, q_{\ell-1}\}$ is the set of positive integers coprime each other and $a \in \mathbb{Z}_Q$ where $Q = \prod_{i=0}^{\ell-1} q_i$, the RNS representation of a with regard to $\mathcal{C}$ is denoted by $[a]_{\mathcal{C}} = (a^{(0)}, a^{(1)}, \cdots, a^{(\ell-1)}) \in \mathbb{Z}_{q_0} \times \cdots \times \mathbb{Z}_{q_{\ell-1}}$. The base of logarithm in this paper is two.

2.2 CKKS Scheme and RNS-CKKS Scheme

It is known that the CKKS scheme supports several operations for encrypted data of real numbers or complex numbers. Since it deals with usually real numbers, the noise that ensures the security of the CKKS scheme can be embraced in the outside of the significant figures of the data, which is the crucial concept of the CKKS scheme.

The RNS-CKKS scheme [13] uses the RNS form to represent the ciphertexts and to perform the homomorphic operations efficiently. While the power-of-two modulus is used in the CKKS scheme, the product of large primes is used for ciphertext modulus in the RNS-CKKS scheme so that the RNS system can be applied. These large primes are chosen to be similar to the scaling factor, which is some power-of-two integer. There is a crucial difference in the rescaling operation between the CKKS scheme and the RNS-CKKS scheme. While the CKKS scheme can rescale the ciphertext by the exact scaling factor, the RNS-CKKS scheme has to rescale the ciphertext by one of the RNS modulus, which is not equal to the scaling factor. Thus, the RNS-CKKS scheme allows approximation in the rescaling procedure. Detailed procedures in the RNS-CKKS scheme are described as follows.

Several independent messages are encoded into one polynomial by the canonical embedding before encryption. The canonical embedding σ embeds $a \in \mathbb{Q}[X]/\langle \Phi_M(X) \rangle$ into an element of $\mathbb{C}^N$ whose elements are values of a evaluated at the distinct roots of $\Phi_M(X)$. It is a well-known fact that the roots of $\Phi_M(X)$ are exactly the power of odd integers of the M-th root of unity, and $\mathbb{Z}_M^* = \langle -1, 5 \rangle$. Let $\mathbb{H} = \{(z_j)_{j \in \mathbb{Z}_M^*} : z_j = \overline{z_{-j}}\}$, and π be a natural projection from $\mathbb{H}$ to $\mathbb{C}^{N/2}$. Then, it is easily known that the range of σ is exactly $\mathbb{H}$. When $N/2$ messages of complex number constitute an element in $\mathbb{C}^{N/2}$, each coordinate is called a slot. The encoding and decoding procedures are given as follows.

$\texttt{Ecd}(\boldsymbol{z}; \Delta)$: For a vector $\boldsymbol{z} \in \mathbb{C}^{N/2}$, return

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

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

$\mathtt{Dcd}(m;\Delta)$**:** For a polynomial $m(X) \in \mathcal{R}$, return a vector $\boldsymbol{z} \in \mathbb{C}^{N/2}$ whose entry of index j is $z_j = \left\lfloor \Delta^{-1} \cdot m(\zeta_M^{5^j}) \right\rceil$ for $j \in \{0, 1, \cdots, N/2-1\}$, where ζ_M is the M-th root of unity.

Before describing the RNS-CKKS scheme, several basic operations for RNS system is defined: Conv, ModUp, and ModDown. Let $\mathcal{B} = \{p_0, p_1, \cdots, p_{k-1}\}, \mathcal{C} = \{q_0, q_1, \cdots, q_{\ell-1}\}$, and $\mathcal{D} = \{p_0, p_1, \cdots, p_{k-1}, q_0, q_1, \cdots, q_{\ell-1}\}$, where p_i's and q_j's are all distinct primes.

$\mathtt{Conv}_{\mathcal{C}\to\mathcal{B}}$**:** It converts the RNS bases from $\mathcal{C}$ to $\mathcal{B}$ without the merge process of Chinese remainder theorem, which is defined as

$$\mathtt{Conv}_{\mathcal{C}\to\mathcal{B}}([a]_{\mathcal{C}}) = \left(\sum_{j=0}^{\ell-1} [a^{(j)} \cdot \hat{q}_j^{-1}]_{q_j} \cdot \hat{q}_j \mod p_i \right)_{0 \le i < k},$$

where $[a]_{\mathcal{C}} = (a^{(0)}, \cdots, a^{(\ell-1)}) \in \mathbb{Z}_{q_0} \times \cdots \times \mathbb{Z}_{q_{\ell-1}}$ and $\hat{q}_j = \prod_{j' \ne j} q_{j'} \in \mathbb{Z}$.

$\mathtt{ModUp}_{\mathcal{C}\to\mathcal{D}}$**:** It adds other moduli in $\mathcal{B}$ to the current RNS bases to expand the modulus space without changing the value as

$$\begin{aligned} \mathtt{ModUp}_{\mathcal{C}\to\mathcal{D}}(\cdot) : \prod_{j=0}^{\ell-1} R_{q_j} &\to \prod_{i=0}^{k-1} R_{p_i} \times \prod_{j=0}^{\ell-1} R_{q_j} \\ : [a]_{\mathcal{C}} &\to (\mathtt{Conv}_{\mathcal{C}\to\mathcal{B}}([a]_{\mathcal{C}}), [a]_{\mathcal{C}}). \end{aligned}$$

$\mathtt{ModDown}_{\mathcal{D}\to\mathcal{C}}$**:** It removes the moduli in $\mathcal{B}$ from the current RNS bases with dividing the value by $P = \prod_{i=0}^{k-1} p_i$ as

$$\begin{aligned} \mathtt{ModDown}_{\mathcal{D}\to\mathcal{C}}(\cdot) : \prod_{i=0}^{k-1} R_{p_i} \times \prod_{j=0}^{\ell-1} R_{q_j} &\to \prod_{j=0}^{\ell-1} R_{q_j} \\ : ([a]_{\mathcal{B}}, [b]_{\mathcal{C}}) &\to ([b]_{\mathcal{C}} - \mathtt{Conv}_{\mathcal{B}\to\mathcal{C}}([a]_{\mathcal{B}})) \cdot [P^{-1}]_{\mathcal{C}}. \end{aligned}$$

Then, each procedure of the RNS-CKKS scheme is given as follows.

$\mathtt{Setup}(q, L; 1^\lambda)$**:** Given a scaling factor Δ, the number of levels L, and a security parameter λ, we choose several parameters as follows.

- A power-of-two degree N of the polynomial modulus of the ring is chosen so that the number of level L can be supported with the security parameter λ.
- A secret key distribution χ_{key}, an encryption key distribution χ_{enc}, and an error distribution χ_{err} over R are chosen considering the security parameter λ.
- A basis with prime numbers $\mathcal{B} = \{p_0, p_1, \cdots, p_{k-1}\}$ and $\mathcal{C} = \{q_0, q_1, \cdots, q_L\}$ is chosen so that $p_i \equiv 1 \mod 2N$, $q_j \equiv 1 \mod 2N$ for $0 \le i \le k-1, 0 \le j \le L$, and $|q_i - \Delta|$ is as small as possible. All prime numbers are distinct and $\mathcal{D} = \mathcal{B} \cup \mathcal{C}$. Let $\mathcal{C}_\ell = \{q_0, q_1, \cdots, q_\ell\}$ and $\mathcal{D}_\ell = \mathcal{B} \cup \mathcal{C}_\ell$ for $0 \le \ell \le L$.

Let $P = \prod_{i=0}^{k-1} p_i, Q = \prod_{j=0}^{L} q_j, \hat{p}_i = \prod_{0 \le i' \le k-1, i' \ne i} p_{i'}$ for $0 \le i \le k-1$, and $\hat{q}_{\ell,j} = \prod_{0 \le j' \le \ell, j' \ne j} q_{j'}$ for $0 \le j \le \ell \le L$. Then, we compute the following numbers.

- $[\hat{p}_i]_{q_j}$ and $[\hat{p}_i^{-1}]_{p_i}$ for $0 \le i \le k-1, 0 \le j \le L$
- $[P^{-1}]_{q_j}$ for $0 \le j \le L$
- $[\hat{q}_{\ell,j}]_{p_i}$ and $[\hat{q}_{\ell,j}^{-1}]_{q_j}$ for $0 \le i \le k-1, 0 \le j \le \ell \le L$

$\texttt{KSGen}(s_1, s_2)$**:** This procedure generates the switching key for switching the secret key s_1 to s_2 without changing the message in a ciphertext. Given $s_1, s_2 \in R$, sample $(a'^{(0)}, \cdots, a'^{(k+L)}) \leftarrow U\left(\prod_{i=0}^{k-1} R_{p_i} \times \prod_{j=0}^{L} R_{q_j}\right)$ and an error $e' \leftarrow \chi_{\mathsf{err}}$, and generate the switching key swk as

$$\left(\mathsf{swk}^{(0)} = (b'^{(0)}, a'^{(0)}), \cdots, \mathsf{swk}^{(k+L)} = (b'^{(k+L)}, a'^{(k+L)})\right) \in \prod_{i=0}^{k-1} R_{p_i}^2 \times \prod_{j=0}^{L} R_{q_j}^2,$$

where $b'^{(i)} \leftarrow -a'^{(i)} \cdot s_2 + e' \mod p_i$ for $0 \le i \le k-1$ and $b'^{(k+j)} \leftarrow -a'^{(k+j)} \cdot s_2 + [P]_{q_j} \cdot s_1 + e' \mod q_j$ for $0 \le j \le L$.

$\texttt{KeyGen}$**:** This procedure generates the secret key, the evaluation key, and the public key. Sample $s \leftarrow \chi_{\mathsf{key}}$ and set $\mathsf{sk} \leftarrow (1, s)$ as the secret key. The evaluation key is set by $\mathsf{evk} \leftarrow \texttt{KSGen}(s^2, s)$. Also, sample $(a^{(0)}, a^{(1)}, \cdots, a^{(L)}) \leftarrow U\left(\prod_{j=0}^{L} R_{q_j}\right)$ and $e \leftarrow \chi_{\mathsf{err}}$ and the public key is generated as

$$\mathsf{pk} \leftarrow \left(\mathsf{pk}^{(j)} = (b^j, a^{(j)}) \in R_{q_j}^2\right)_{0 \le j \le L},$$

where $b^{(j)} \leftarrow -a^{(j)} \cdot s + e \mod q_j$ for $0 \le j \le L$.

$\texttt{Enc}_{\mathsf{pk}}(m)$**:** For a message slot $\mathbf{z} \in \mathbb{C}^{N/2}$, generate the message polynomial by $m = \texttt{Ecd}(\mathbf{z}; \Delta)$. Then, sample $v \leftarrow \chi_{\mathsf{enc}}$ and $e_0, e_1 \leftarrow \chi_{\mathsf{err}}$ and generate the ciphertext $\mathsf{ct} = (\mathsf{ct}^{(j)})_{0 \le j \le L} \in \prod_{j=0}^{L} R_{q_j}^2$, where $\mathsf{ct}^{(j)} \leftarrow v \cdot \mathsf{pk}^{(j)} + (m + e_0, e_1) \mod q_j$ for $0 \le j \le L$.

$\texttt{Dec}_{\mathsf{sk}}(\mathsf{ct})$**:** For a ciphertext $\mathsf{ct} = (\mathsf{ct}^{(j)})_{0 \le j \le \ell} \in \prod_{j=0}^{\ell} R_{q_j}^2$, compute $\tilde{m} = \langle \mathsf{ct}^{(0)}, \mathsf{sk} \rangle \mod q_0$ and output $\mathbf{z} = \texttt{Dcd}(\tilde{m}; \Delta)$.

$\texttt{Add}(\mathsf{ct}_1, \mathsf{ct}_2)$**:** For two ciphertexts $\mathsf{ct}_r = \left(\mathsf{ct}_r^{(j)}\right)_{0 \le j \le \ell}$ for $r = 1, 2$, output the ciphertext $\mathsf{ct}_{\mathsf{add}} = \left(\mathsf{ct}_{\mathsf{add}}^{(j)}\right)_{0 \le j \le \ell}$, where $\mathsf{ct}_{\mathsf{add}}^{(j)} \leftarrow \mathsf{ct}_1^{(j)} + \mathsf{ct}_2^{(j)} \mod q_j$ for $0 \le j \le \ell$.

$\texttt{Mult}_{\mathsf{evk}}(\mathsf{ct}_1, \mathsf{ct}_2)$**:** For two ciphertexts $\mathsf{ct}_r = \left(\mathsf{ct}_r^{(j)} = (c_{r0}^{(j)}, c_{r1}^{(j)})\right)_{0 \le j \le \ell}$, compute the followings and output the ciphertext $\mathsf{ct}_{\mathsf{mult}} \in \prod_{j=0}^{\ell} R_{q_j}^2$.

- $d_0^{(j)} = c_{00}^{(j)} c_{10}^{(j)} \mod q_j, d_1^{(j)} = c_{00}^{(j)} c_{11}^{(j)} + c_{01}^{(j)} c_{10}^{(j)} \mod q_j$, and $d_2^{(j)} = c_{01}^{(j)} c_{11}^{(j)} \mod q_j$ for $0 \le j \le \ell$.
- $\texttt{ModUp}_{\mathcal{C}_\ell \to \mathcal{D}_\ell}(d_2^{(0)}, d_2^{(1)}, \cdots, d_2^{(\ell)}) = (\tilde{d}_2^{(0)}, \tilde{d}_2^{(1)}, \cdots, \tilde{d}_2^{(k-1)}, d_2^{(0)}, d_2^{(1)}, \cdots, d_2^{(\ell)})$.
- $\tilde{\mathsf{ct}} = (\tilde{\mathsf{ct}}^{(k+\ell)} = (\tilde{c}_0^{(j)}, \tilde{c}_1^{(j)}))_{0 \le j \le k+\ell}$, where $\tilde{\mathsf{ct}^{(i)}} = \tilde{d}_2^{(i)} \cdot \mathsf{evk}^{(i)} \mod p_i$ and $\tilde{\mathsf{ct}^{(k+j)}} = d_2^{(j)} \cdot \mathsf{evk}^{(k+j)} \mod q_j$ for $0 \le i \le k-1$ and $0 \le j \le \ell$.

- $\left(\hat{c}_r^{(0)}, \hat{c}_r^{(1)}, \cdots, \hat{c}_r^{(\ell)}\right) = \texttt{ModDown}_{\mathcal{D}_\ell \to \mathcal{C}_\ell}\left(\tilde{c}_r^{(0)}, \tilde{c}_r^{(1)}, \cdots, \tilde{c}_r^{(k+\ell)}\right)$ for $r = 0, 1$.
- $\mathsf{ct}_{\mathsf{mult}} = (\mathsf{ct}_{\mathsf{mult}}^{(j)})_{0 \le j \le \ell}$, where $\mathsf{ct}_{\mathsf{mult}}^{(j)} = \left(\hat{c}_0^{(j)} + d_0^{(j)}, \hat{c}_1^{(j)} + d_1^{(j)}\right) \mod q_j$ for $0 \le j \le \ell$.

$\texttt{RS}(\mathsf{ct})$**:** For a ciphertext $\mathsf{ct} = \left(\mathsf{ct}^{(j)} = (c_0^{(j)}, c_1(j))\right)_{0 \le j \le \ell}$, output the ciphertext $\mathsf{ct}' = \left(\mathsf{ct}'^{(j)} = (c_0'^{(j)}, c_1'^{(j)})\right)_{0 \le j \le \ell - 1}$, where $c_r'^{(j)} = q_\ell^{-1} \cdot \left(c_r^{(j)} - c_r^{(\ell)}\right) \mod q_j$ for $r = 0, 1$ and $0 \le j \le \ell - 1$.

There are additional homomorphic operations, rotation, and complex conjugation, which are used for homomorphic linear transformation in the bootstrapping of the RNS-CKKS scheme. Since these operations are not used in this paper, we omit these operations in this section.

2.3 Bootstrapping for CKKS Scheme

The framework of the bootstrapping of the CKKS scheme was introduced in [14], which is the same as the case of the RNS-CKKS scheme. The purpose of bootstrapping is to refresh the ciphertext of level 0, whose multiplication cannot be performed anymore, to the fresh ciphertext of level L having the same messages. The bootstrapping is composed of the following four steps:

i) Modulus raising
ii) Homomorphic linear transformation; CoeffToSlot
iii) Homomorphic modular reduction
iv) Homomorphic linear transformation; SlotToCoeff

Modulus Raising: The starting point of bootstrapping is modulus raising, where we simply consider the ciphertext of level 0 as an element of $\mathcal{R}_Q^2$, instead of $\mathcal{R}_{q_0}^2$. Since the ciphertext of level 0 is supposed to be $\langle \mathsf{ct}, \mathsf{sk} \rangle \approx m \mod q_0$, we have $\langle \mathsf{ct}, \mathsf{sk} \rangle \approx m + q_0 I \mod Q$ for some $I \in \mathcal{R}$ when we try to decrypt it. We are assured that the absolute values of coefficients of I are rather small, for example, usually smaller than 12, because coefficients of sk consist of small numbers [12]. The crucial part of the bootstrapping of the CKKS scheme is to make ct' such that $\langle \mathsf{ct}', \mathsf{sk} \rangle \approx m \mod q_L$. This is divided into two parts: homomorphic linear transform and homomorphic evaluation of modular reduction function.

Homomorphic Linear Transformation: The ciphertext ct after modulus raising can be considered as the ciphertext encrypting $m + q_0 I$, and thus we now have to perform modular reduction to coefficients of message polynomial homomorphically. However, the operations we have are all for slots, not coefficients of the message polynomial. Thus, to perform some meaningful operations on coefficients, we have to convert ct into a ciphertext that encrypts coefficients of $m + q_0 I$ as its slots, and after evaluation of homomorphic modular reduction function, we have to reversely convert this ciphertext into the other ciphertext

ct' that encrypts the slots of the previous ciphertext as the coefficients of its message. These two operations are called CoeffToSlot and SlotToCoeff operations. These operations are regarded as homomorphic evaluation of encoding and decoding of messages, which are a linear transformation by some variants of Vandermonte matrix for roots of $\Phi_M(x)$. This can be performed by general homomorphic matrix multiplication [12], or FFT-like operation [9].

Homomorphic Modular Reduction Function: After CoeffToSlot is performed, we now have to perform modular reduction homomorphically on each slot in modulus q_0. This procedure is called EvalMod. This modular reduction function is not an arithmetic function, and even not a continuous function. Fortunately, by restricting the range of the messages such that m/q_0 is small enough, the approximation region can be given only near multiples of q_0. This allows us to approximate the modular reduction function more effectively. Since the operations that the CKKS supports are arithmetic operations, most of the research [9, 12, 24] dealing with CKKS bootstrapping approximate the modular reduction function with some polynomials, which is sub-optimal approximate polynomials.

The scaling factor is increased when the bootstrapping is performed because m/q_0 needs to be very small in the homomorphic modular reduction function. In this paper, the default scaling factor means the scaling factor used in the intended applications, and the bootstrapping scaling factor means the scaling factor used in the bootstrapping. The bit-length difference between these two scaling factors is usually 10.

2.4 Approximation Theory

Approximation theory is needed to prove the convergence of the minimax polynomial obtained by the proposed modified Remez algorithm. Assume that functions are defined on a union of the finite number of closed and bounded intervals in the real line. From the following well-known theorem [29] in real analysis, we are convinced that this domain of functions is a compact set.

Theorem 2.1 ([29] Bolzano-Weierstrass Theorem). *A subset of $\mathbb{R}^n$ is a compact set if and only if it is closed and bounded.*

A union of the finite number of closed and bounded intervals in the real line is trivially closed and bounded, and thus this domain is a compact set by Bolzano-Weierstrass theorem. This theorem will be used in the convergence proof of the modified Remez algorithm in Section 3.

The next theorem [29] states that any continuous function on compact set in the real line can be approximated with an arbitrarily small error by polynomial approximation. In fact, the theorem includes the case of continuous functions on more general domains, but we only use the special case on compact sets in the real line in this paper.

Theorem 2.2 (**[29] Stone-Weierstrass Theorem**). *Assume that f is a continuous function on the compact subset D of the real line. For every $\epsilon > 0$, there is a polynomial p such that $\|f - p\|_\infty < \epsilon$.*

There are many theorems for the minimax approximate polynomials of a function defined on a compact set in approximation theory. Before the introduction of these theorems, we refer to a definition of Haar condition of a set of functions that deals with the generalized version of power bases used in polynomial approximation and its equivalent statement. It is a well-known fact that the power basis $\{1, x, x^2, \cdots, x^d\}$ satisfies the Haar condition. Thus, if an argument deals with the polynomials with regard to a set of basis functions satisfying Haar condition, it naturally includes the case of polynomials.

Definition 2.3 (**[11] Haar's Condition**). *A set of functions $\{g_1, g_2, \cdots, g_n\}$ satisfies the Haar condition if each g_i is continuous and if each determinant*

$$D[x_1, \cdots, x_n] = \begin{vmatrix} g_1(x_1) & \cdots & g_n(x_1) \\ \vdots & \ddots & \vdots \\ g_1(x_n) & \cdots & g_n(x_n) \end{vmatrix}$$

for any n distinct points $x_1, \cdots, x_n$ is not zero.

Lemma 2.4 (**[11]**). *A set of functions $\{g_1, \cdots, g_n\}$ satisfies the Haar condition if and only if zero polynomial is the only polynomial $\sum_i c_i g_i$ that has more than $n - 1$ roots.*

Firstly, we are convinced that there is the unique minimax approximate polynomial in the union of the finite number of closed and bounded intervals as in the following two theorems.

Theorem 2.5 (**[11] Existence of Best Approximations**). *Let $\mathcal{F}$ be a normed linear space, and f is any fixed element in $\mathcal{F}$. If $\mathcal{S}$ is a linear subspace of $\mathcal{F}$ with finite dimension, $\mathcal{S}$ contains at least one element of minimum distance from f.*

Theorem 2.6 (**[11] Haar's Unicity Theorem**). *Let f be any continuous function on a compact set K. Then the minimax polynomial $\sum_i c_i g_i$ of f is unique if and only if $\{g_1, g_2, \cdots, g_n\}$ satisfies the Haar condition.*

In Theorem 2.5 for the existence of the best approximation, consider a set $\mathcal{F}$ of continuous functions on a union D of the finite number of closed and bounded intervals. We can easily know that $\mathcal{F}$ is a linear space with a max-norm $\|f\|_\infty = \max_{x \in D} |f(x)|$. The set $\mathcal{P}_d$ of polynomials with regard to the finite number of basis functions on D is a finite-dimensional linear subspace. Then, from Theorem 2.5, there is at least one minimax approximate polynomial for any $f \in \mathcal{F}$.

We now introduce the core property of the minimax approximate polynomial for a function on D.

Theorem 2.7 ([11] **Chebyshev Alternation Theorem). ***Let $\{g_1, \cdots, g_n\}$ be a set of functions in $C[a, b]$ satisfying the Haar condition, and let D be a closed subset of $[a, b]$. A polynomial $p = \sum_i c_i g_i$ is the minimax approximation on D to any given $f \in C[D]$ if and only if there are $n+1$ distinct elements $x_0 < \cdots < x_n$ in D such that for the error function $r = f - p$ restricted on D,*

$$r(x_i) = -r(x_{i-1}) = \pm\|r\|_\infty.$$

This condition is also called equioscillation condition. This means that if we find a polynomial satisfying the equioscillation condition, then this is the unique minimax approximate polynomial, needless to compare with the maximum approximation error of any polynomials.

The following three theorems are used to prove the convergence of the modified Remez algorithm in Section 3.

Theorem 2.8 ([11] **de La Vallee Poussin Theorem). ***Let $\{g_1, \cdots, g_n\}$ is a set of continuous functions on $[a, b]$ satisfying the Haar condition. Let f be a continuous on $[a, b]$, and p be a polynomial such that $p - f$ has alternately positive and negative values at $n+1$ consecutive points x_i in $[a, b]$. Let p^* be a minimax approximate polynomial for f, and $e(f)$ be the minimax approximation error of p^*. Then, we have*

$$e(f) \geq \min_i |p(x_i) - f(x_i)|.$$

Lemma 2.9 ([11]). *Let $\{g_1, \cdots, g_n\}$ be a set of continuous functions satisfying the Haar condition. Assume that $x_1 < \cdots < x_n$ and $y_1 < \cdots < y_n$. Then the determinants $D[x_1, \cdots, x_n]$ and $D[y_1, \cdots, y_n]$, defined by Definition 2.3, have the same sign.*

Theorem 2.10 ([11] **Strong Unicity Theorem). ***Let $\{g_1, \cdots, g_n\}$ be a set of functions satisfying the Haar condition, and let p^* be the minimax polynomial of a given continuous function u. Then, there is a constant $\gamma > 0$ determined by f such that for any polynomial p, we have*

$$\|p - f\|_\infty \geq \|p^* - f\|_\infty + \gamma\|p - p^*\|_\infty.$$

2.5 Algorithms for Minimax Approximation

Remez Algorithm Remez algorithm [11, 27, 28] is an iterative algorithm that always returns the minimax approximate polynomial for any continuous function on an interval of $[a, b]$. This algorithm strongly uses Chebyshev alternation theorem [11] in that its purpose is finding the polynomial satisfying equioscillation condition. In fact, the Remez algorithm can be applied to obtain the minimax approximate polynomial, whose basis function $\{g_1, \cdots, g_n\}$ satisfies the Haar condition. The following explanation includes the generalization of the Remez algorithm, and if we want to obtain the minimax approximate polynomial of degree d, we choose the basis function $\{g_1, \cdots, g_n\}$ by the power basis $\{1, x, \cdots, x^d\}$, where $n = d + 1$.

Remez algorithm firstly initializes the set of reference points $\{x_1, \cdots, x_{n+1}\}$, which will be converged to the extreme points of the minimax approximate polynomial. Then, it obtains the minimax approximate polynomial in regard to only the set of reference points. Since the set of reference points is the set of finite points in $[a, b]$, it is a closed subset of $[a, b]$, and thus Chebyshev alternation theorem holds on the set of reference points. Let $f(x)$ be a continuous function on $[a, b]$. The minimax approximate polynomial on the set of reference points is exactly the polynomial $p(x)$ with the basis $\{g_1, \cdots, g_n\}$ satisfying

$$p(x_i) - f(x_i) = (-1)^i E \quad i = 1, \cdots, d + 2$$

for some real number E. This forms a system of linear equations having $n + 1$ equations and $n + 1$ variables of n coefficients of $p(x)$ and E, which is ensured to be not singular by Haar's condition, and thus we can obtain the polynomial $p(x)$. Then, we can find n zeros of $p(x) - f(x)$, z_i between x_i and x_{i+1}, $i = 1, 2, \cdots, n$, and we can find $n + 1$ extreme points $y_1, \cdots, y_{n+1}$ of $p(x) - f(x)$ in each $[z_{i-1}, z_i]$, where $z_0 = a$ and $z_{n+1} = b$. That is, we choose the minimum point of $p(x) - f(x)$ in $[z_{i-1}, z_i]$ if $p(x_i) - f(x_i) < 0$, and we choose the maximum point of $p(x) - f(x)$ in $[z_{i-1}, z_i]$ if $p(x_i) - f(x_i) > 0$. Thus, we find a new set of extreme points $y_1, \cdots, y_{n+1}$. If this satisfies equioscillation condition, the Remez algorithm returns $p(x)$ as the minimax approximate polynomial from the Chebyshev alternation theorem. Otherwise, it replaces the set of reference points with these extreme points $y_1, \cdots, y_{n+1}$ and processes above steps again. This is the Remez algorithm in Algorithm 1. The Remez algorithm is proved to be always converged to the minimax approximate polynomial by the following theorem.

Theorem 2.11 ([11] **Convergence of Remez Algorithm**). *Let $\{g_1, \cdots, g_n\}$ be a set of functions satisfying the Haar condition, p_k be a polynomial generated in the k-th iteration of Remez algorithm, and p^* be the minimax polynomial of a given f. Then, p_k converges uniformly to p^* by the following inequality,*

$$\|p_k - p^*\|_\infty \le A\theta^k,$$

where A is a non-negative constant, and $0 < \theta < 1$.

A Variant of the Remez Algorithm Since the Remez algorithm works only when the approximation region is one interval, we need another variant of the Remez algorithm which works when the approximation region is the union of several intervals. The above Remez algorithm can be extended to the multiple sub-intervals of an interval [19,26,28]. The variant of the Remez algorithm is the same as Algorithm 1, except Steps 3 and 4. For each iteration, firstly, we find all of the local extreme points of the error function $p - f$ whose absolute error

```
Algorithm 1: Remez
    Input : An input domain $[a, b]$, a continuous function $f$ on $[a, b]$, an
            approximation parameter $\delta$, and a basis $\{g_1, \cdots, g_n\}$.
    Output: The minimax approximate polynomial $p$ for $f$
 1  Select $x_1, x_2, \cdots, x_{d+2} \in [a, b]$ in strictly increasing order.
 2  Find the polynomial $p(x)$ in terms of $\{g_1, \cdots, g_n\}$ with
      $p(x_i) - f(x_i) = (-1)^i E$ for some $E$.
 3  Divide the interval into $n + 1$ sections $[z_{i-1}, z_i]$, $i = 1, \cdots, n + 1$, from zeros
      $z_1, \cdots, z_n$ of $p(x) - f(x)$, where $x_i < z_i < x_{i+1}$, and boundary points
      $z_0 = a, z_{n+1} = b$.
 4  Find the maximum (resp. minimum) points for each section when $p(x_i) - f(x_i)$
      has positive (resp. nagative) value. Denote these extreme points $y_1, \cdots, y_{n+1}$.
 5  $\epsilon_{\mathsf{max}} \leftarrow \max_i |p(y_i) - f(y_i)|$
 6  $\epsilon_{\mathsf{min}} \leftarrow \min_i |p(y_i) - f(y_i)|$
 7  if $(\epsilon_{\mathsf{max}} - \epsilon_{\mathsf{min}})/\epsilon_{\mathsf{min}} < \delta$ then
 8      return $p(x)$
 9  else
10      Replace $x_i$'s with $y_i$'s and go to line 2.
11  end
```

values are larger than the absolute error values at the current reference points. Then, we choose $n + 1$ new extreme points among these points satisfying the following two criteria:

i) The error values alternate in sign.
ii) A new set of extreme points includes the global extreme point.

These two criteria are known to ensure the convergence to the minimax polynomial, even though we could not find the exact proof of its convergence. However, it is noted that there are many choices of sets of extreme points satisfying these criteria. In the next section, we modify the variant of the Remez algorithm, where one of the two criteria is changed.

3 Efficient Algorithm for Optimal Minimax Approximate Polynomial

In this section, we propose an algorithm for obtaining the optimal minimax approximate polynomial, called the modified Remez algorithm. With this proposed algorithm, we can obtain the optimal minimax approximate polynomial for continuous function on the union of finitely many closed intervals in order to apply the Remez algorithm to the bootstrapping of the CKKS scheme. The function we are going to approximate is the normalized modular reduction function defined in only near finitely many integers given as

$$\mathsf{normod}(x) = x - \mathsf{round}(x), \quad x \in \bigcup_{i=-(K-1)}^{K-1} [i-\epsilon, i+\epsilon],$$

where K determines the number of intervals in the domain. normod function corresponds to the modular reduction function scaled for both its domain and range.

In addition, Han et al. [24] uses the cosine function to approximate $\mathsf{normod}(x)$ to use double-angle formula for efficient homomorphic evaluation. If we use double-angle formula ℓ times, we have to approximate the following cosine function

$$\cos\left(\frac{2\pi}{2^{\ell}}\left(x - \frac{1}{4}\right)\right), \quad x \in \bigcup_{i=-(K-1)}^{K-1} [i-\epsilon, i+\epsilon].$$

To design an approximation algorithm that deals with the above two functions, we assume the general continuous function defined on an union of finitely many closed intervals, which is given as

$$D = \bigcup_{i=1}^{t} [a_i, b_i] \subset [a, b] \subset \mathbb{R}$$

where $a_i < b_i < a_{i+1} < b_{i+1}$ for all $i = 1, \cdots, t-1$.

When we modify the variant of the Remez algorithm to approximate a given continuous function on D with a polynomial having a degree less than or equal to d, called the modified Remez algorithm, we have to consider two crucial points. One is to establish an efficient criterion for choosing new $d+2$ reference points among several extreme points, and the other is to make some steps in the modified Remez algorithm efficient. We deal with these two issues for the modified Remez algorithm in Sections 3.1 and 3.3, respectively.

3.1 Modified Remez Algorithm with Criteria for Choosing Extreme Points

Assume that we apply the variant of the Remez algorithm on D and use $\{g_1, \cdots, g_n\}$ satisfying Haar condition on $[a, b]$ as the basis of polynomials. After obtaining the minimax approximate polynomial in regard to the set of reference points for each iteration, we have to choose a new set of reference points for the next iteration. However, there are many boundary points in D and all these boundary points have to be considered as extreme points of the error function. For this reason, there are many cases of selecting $n+1$ points among these extreme points. For bootstrapping in the CKKS scheme, there are many intervals to be considered, and thus there are lots of candidate extreme points. Since the criterion of the variant of the Remez algorithm cannot determine the unique new set of reference points for each iteration, it is necessary to make how to choose $n+1$ points for each iteration to reduce the number of iterations as small as possible. Otherwise, it requires a large number of iteration for convergence to the minimax approximate polynomial. In addition, it is very important to ensure that this criterion always leads to convergence to the minimax approximate polynomial. If the criterion is not designed properly, the modified Remez algorithm may not converge into a single polynomial in some cases.

In order to set the criterion for selecting $n+1$ reference points, we need to define a simple function for extreme points, $\mu_{p,f} : D \to \{-1, 0, 1\}$ as follows,

$$\mu_{p,f}(x) = \begin{cases} 1 & p(x) - f(x) \text{ is a local maximum value at } x \text{ on } D \\ -1 & p(x) - f(x) \text{ is a local minimum value at } x \text{ on } D \\ 0 & \text{otherwise,} \end{cases}$$

where $p(x)$ is a polynomial obtained in that iteration and $f(x)$ is a continuous function on D to be approximated. We abuse the notation $\mu_{p,f}$ as μ.

If we gather all extreme points of $p(x) - f(x)$ into a set B, we can assume that B is a finite set, that is, $B = \{x_1, x_2, \cdots, x_m\}$. If there is an interval in B, we can choose a point in that interval. Assume that B is ordered in increasing order, $x_1 < x_2 < \cdots < x_m$, and then the values of μ at these points are always 1 or -1. Let $\mathcal{S}$ be a set of functions defined as

$$\mathcal{S} = \{\sigma : [n+1] \to [m] \mid \sigma(i) < \sigma(i+1) \text{ for all } i = 1, \cdots, n\}.$$

Clearly, $\mathcal{S}$ has only the identity function if $n+1 = m$.

Then, we set three criteria for selecting $n+1$ extreme points as follows:

i) *Local extreme value condition.* If E is the absolute value of error at points in the set of reference points, then we have

$$\min_i \mu(x_{\sigma(i)})(p(x_{\sigma(i)}) - f(x_{\sigma(i)})) \geq E.$$

ii) *Alternating condition.* $\mu(x_{\sigma(i)}) \cdot \mu(x_{\sigma(i+1)}) = -1$ for $i = 1, \cdots, n$.

iii) *Maximum absolute sum condition.* Among σ's satisfying the above two conditions, choose σ maximizing the following value

$$\sum_{i=1}^{n+1} |p(x_{\sigma(i)}) - f(x_{\sigma(i)})|.$$

It is noted that the local extreme value condition in i) means that the extreme points are discarded if the local maximum value of $p(x) - f(x)$ is negative or the local minimum of $p(x) - f(x)$ is positive.

Note that the first two conditions are also included in the variant of the Remez algorithm, and the third condition, the maximum absolute sum condition, is the replacement of the condition that the new set of reference points includes the global extreme point. The numerical analysis will show that the third condition makes the proposed modified Remez algorithm to converge to the optimal minimax approximate polynomial fast. Although there are some cases in which the global maximum point is not included in the new set of reference points chosen by the maximum absolute sum condition, we prove that the maximum absolute sum condition is enough for the modified Remez algorithm to converge to the minimax approximate polynomial in Subsection 3.2.

Now we propose the modified Remez algorithm for the continuous function on the union of finitely many closed intervals as in Algorithm 2. The local extreme value condition is reflected in Step 3, and the alternating condition and the maximum absolute sum condition are reflected in Step 4.

Algorithm 2: Modified Remez Algorithm

```
Input  : An input domain $D = \bigcup_{i=1}^{t}[a_i, b_i] \subset \mathbb{R}$, a continuous function $f$ on
         $D$, an approximation parameter $\delta$, and a basis $\{g_1, \cdots, g_n\}$
Output: The minimax approximate polynomial $p$ for $f$
1  Select $x_1, x_2, \cdots, x_{n+1} \in D$ in strictly increasing order.
2  Find the polynomial $p(x)$ with $p(x_i) - f(x_i) = (-1)^i E$ for some $E$.
3  Gather all extreme and boundary points such that $\mu_{p,f}(x)(p(x) - f(x)) \geq |E|$
    into a set $B$.
4  Find $n + 1$ extreme points $y_1 < y_2 < \cdots < y_{n+1}$ with alternating condition
    and maximum absolute sum condition in $B$.
5  $\epsilon_{\mathsf{max}} \leftarrow \max_i |p(y_i) - f(y_i)|$
6  $\epsilon_{\mathsf{min}} \leftarrow \min_i |p(y_i) - f(y_i)|$
7  if $(\epsilon_{\mathsf{max}} - \epsilon_{\mathsf{min}})/\epsilon_{\mathsf{min}} < \delta$ then
8      return $p(x)$
9  else
10     Replace $x_i$'s with $y_i$'s and go to line 2.
11 end
```

3.2 Correctness of Modified Remez Algorithm

We now have to prove that the proposed algorithm always converges to the minimax approximate polynomial for given piecewise continuous function on D. This proof is similar to the convergence proof of the original Remez algorithm on one closed interval [11, 27], but there are a few more general arguments than the original proof. This convergence proof includes the proof for both the variant of the Remez algorithm and the modified Remez algorithm.

We have to check that $\mathcal{S}$ always contains at least one element σ_0 that satisfies the local extreme value condition and the alternating condition, and has $\sigma_0(i_0)$ for some i_0 such that $|p(x_{\sigma_0(i_0)}) - f(x_{\sigma_0(i_0)})| = \|p - f\|_\infty$. This existence is in fact the basic assumption of the variant of Remez algorithm, but we prove this existence for mathematical clarification.

Theorem 3.1. *Let B and $\mathcal{S}$ be the sets defined above. Then, there is at least one element in $\mathcal{S}$ which satisfies the local extreme value condition and the alternating condition and has $\sigma_0(i_0)$ for some i_0 such that $|p(x_{\sigma_0(i_0)}) - f(x_{\sigma_0(i_0)})| = \|p - f\|_\infty$.*

Proof. Let a_i and b_i be the boundary points in D defined above and let $t_1, t_2, \cdots, t_{n+1} \in D$ be the reference points used to construct $p_k(x)$ at the k-th iteration. Without loss of generality, $t_i < t_{i+1}$ for all $i = 1, \cdots, n$, and the following equation for some proper positive value E is satisfied as

$$p(t_i) - f(t_i) = (-1)^{i-1}E.$$

Let u_{2i-1} be the largest point among all a_j and t_{2j}'s which are less than or equal to t_{2i-1}, and let v_{2i-1} be the smallest point among all b_j and t_{2j}'s which are larger than or equal to t_{2i-1}. Then, firstly, we prove that there exists at least one local maximum point c_{2i-1} of $p_k(x)-f(x)$ in $[u_{2i-1}, v_{2i-1}]$, and $c_{2i-1} < t_{2i} < c_{2i+1}$ for all possible i. From the extreme value theorem for continuous function on interval [29], there exists at least one maximum point of $p_k(x) - f(x)$ in $[u_{2i-1}, v_{2i-1}]$, since $p_k(x)-f(x)$ is continuous on D. We denote this value at the maximum point as c_{2i-1}. Since t_{2i-1} is in $[u_{2i-1}, v_{2i-1}]$, $p_k(c_{2i-1}) - f(c_{2i-1}) \geq E > -E = p_k(t_{2j}) - f(t_{2j})$ for all possible j, and thus c_{2i-1} cannot be equal to any t_{2i}'s. Since elements appeared more than once in intervals $[u_{2i-1}, v_{2i-1}]$, $i = 1, 2, \cdots, \lfloor \frac{n+2}{2} \rfloor$, are only t_{2i}'s and $v_{2i-1} \leq t_{2i} \leq u_{2i+1}$ for all possible i, we now prove that $c_{2i-1} < t_{2i}$ and $t_{2i} < c_{2i+1}$.

Let u_{2i} be the largest point among all a_j and c_{2j-1}'s which are less than or equal to t_{2i}, and let v_{2i} be the smallest point among all b_j and c_{2j-1}'s which are larger than or equal to t_{2i}. Then, we prove that there exists at least one local minimum point c_{2i} of $p_k(x) - f(x)$ in $[u_{2i}, v_{2i}]$, and c_i's are sorted in strictly increasing order. Again, from the extreme value theorem for continuous function on interval, there exists at least one minimum point of $p_k(x) - f(x)$ in $[u_{2i}, v_{2i}]$. We denote this value at the minimum point as c_{2i}. Since t_{2i} is in $[u_{2i}, v_{2i}]$, $p_k(c_{2i}) - f(t_{2j}) \leq -E < E \leq p_k(c_{2j-1}) - f(c_{2j-1})$ for all possible j, and thus c_{2i} cannot be equal to any c_{2j-1}. Since elements appeared more than once in intervals $[u_{2i}, v_{2i}]$, $i = 1, 2, \cdots, \lfloor \frac{n+2}{2} \rfloor$, are only c_{2i-1}'s and $v_{2i} \leq c_{2i+1} \leq u_{2i+2}$ for all possible i, we now prove that c_i's are sorted in strictly increasing order.

Since c_i's are all local extreme points, $c_i \in B$ for all i. Then, we can set $\sigma' \in \mathcal{S}$ such that $x_{\sigma'(i)} = c_i$. Since c_{2i-1}'s are local maximum points and c_{2i}'s are local minimum points, σ' satisfies alternating condition. Since $\mu(c_i)(p(c_i)-f(c_i)) \geq E$, σ' also satisfies the local extreme value condition. If one of c_i has the maximum absolute value of $p - f$, we are done.

Assume that all of c_i's do not have the maximum absolute value of $p_k - f$. Let x_m be the global extreme point of $p_k - f$. If there is some k such that $c_k < x_m < c_{k+1}$, either c_k or c_{k+1} has the same value of μ at x_m. Then, we replace it with x_m and define this function as σ_0. Since σ_0 satisfies all of conditions in Theorem 3.1, we are done.

If $x_m < c_1$ or $x_m > c_{n+1}$, we separate it into two cases again. If $\mu(x_m) = \mu(c_1)$ (resp. $\mu(x_m) = \mu(c_{n+1})$), we just replace c_1 (resp. c_{n+1}) with x_m and define this function as σ_0, and σ_0 satisfies all these conditions. If $\mu(x_m) \neq \mu(c_1)$ (resp. $\mu(x_m) \neq \mu(c_{n+1})$), we replace c_{n+1} (resp. c_1) with x_m, and relabel the points to define the new function σ_0. This also satisfies all three conditions. Thus, we prove it.

□

Remark. This theorem also ensures that $m \geq n+1$. If $m < n+1$, $\mathcal{S}$ has to be empty. This theorem ensures that there is at least one element in $\mathcal{S}$, we can be convinced that $m \geq n+1$.

Before proving the convergence, we have to generalize the de La Vallee Poussin theorem, which was used to prove the convergence of the original Remez algorithm on an interval. Since original de La Vallee Poussin theorem [11] only deals with a single interval, we generalize it in the following theorem that deals with the closed subset of an interval, whose proof is almost the same as that of the original theorem.

Lemma 3.2 (**Generalized de La Vallee Poussin Theorem**). *Let $\{g_1, \cdots, g_n\}$ is a set of continuous functions on $[a,b]$ satisfying the Haar condition, and let D be a closed subset of $[a,b]$. Let f be a continuous on D, and p be a polynomial such that $p - f$ has alternately positive and negative values at $n+1$ consecutive points x_i in D. Let p^* be a minimax approximate polynomial for f on D, and $e(f)$ be the minimax approximation error of p^*. Then, we have*

$$e_D(f) \geq \min_i |p(x_i) - f(x_i)|.$$

Proof. Assume that the above statement is false. Then, there is a polynomial p_0 such that $p_0 - f$ has alternately positive and negative values at $n+1$ consecutive points in D, and

$$\|p^* - f\|_\infty < |p_0(x_i) - f(x_i)| \tag{1}$$

for all i. Then, $p_0 - p^* = (p_0 - f) - (p^* - f)$ has alternately positive and negative values at the consecutive x_i, which leads to the fact that there is n roots in $[a,b]$. From Lemma 2.4, $p_0 - p^*$ has to be zero, which is contradiction. □

We now prove the convergence of Algorithm 2.

Theorem 3.3. *Let $\{g_1, \cdots, g_n\}$ be a set of functions satisfying the Haar condition on $[a,b]$, D be the multiple sub-intervals of $[a,b]$, and f be a continuous function on D. Let p_k be an approximate polynomial generated in the k-th iteration of the modified Remez algorithm, and p^* be the optimal minimax approximate polynomial of f. Then, as k increases, p_k converges uniformly to p^* as in the following inequality*

$$\|p_k - p^*\|_\infty \leq A\theta^k,$$

where A is a non-negative constant and $0 < \theta < 1$.

Proof. Let $\{x_1^{(0)}, \cdots, x_{n+1}^{(0)}\}$ be the initial set of reference points and $\{x_1^{(k)}, \cdots, x_{n+1}^{(k)}\}$ be the new set of reference points chosen at the end of iteration k. Let $r_k = p_k - f$ be the error function of p_k and $r^* = p^* - f$ be the error function of p^*. Since p_k is generated such that the absolute values of the error function r_k at the reference points $x_i^{(k-1)}$, $i = 1, 2, \cdots, n+1$ are the same. For $k \geq 1$, we define

$$\alpha_k = \min_i |r_k(x_i^{(k-1)})| = \max_i |r_k(x_i^{(k-1)})|,$$

$$\beta_k = \|r_k\|_\infty,$$

$$\gamma_k = \min_i |r_k(x_i^{(k)})|. \tag{2}$$

Define $\beta^* = \|r^*\|_\infty$. We have $\beta^* \geq \gamma_k$ from Lemma 3.2, $\beta_k \geq \beta^*$ by definition of p^*, and $\gamma_k \geq \alpha_k$ by the local extreme value condition for new set of reference points. Then, we have

$$\alpha_k \leq \gamma_k \leq \beta^* \leq \beta_k.$$

Let $c^{(k)} = [c_1^{(k)}, \cdots, c_n^{(k)}]^T$ be the coefficient vector of p_k. Then, $c^{(k)}$ is the solution vector of the following system of linear equations

$$(-1)^{i+1}h^{(k)} + \sum_{j=1}^{n} c_j^{(k)} g_j(x_i^{(k-1)}) = f(x_i^{(k-1)}), \quad i = 1, \cdots, n+1 \tag{3}$$

for the $n+1$ unknowns $h^{(k)}$ and $c_j^{(k)}$'s, and $|h^{(k)}| = \alpha_k$. From Theorem 2.6, we assure that the system of linear equations in (3) is nonsingular, which can be rewritten as in the matrix equation for $k+1$, instead of k,

$$\begin{bmatrix} 1 & g_1(x_1^{(k)}) & \cdots & g_n(x_1^{(k)}) \\ -1 & g_1(x_2^{(k)}) & \cdots & g_n(x_2^{(k)}) \\ \vdots & \vdots & \ddots & \vdots \\ (-1)^n & g_1(x_{n+1}^{(k)}) & \cdots & g_n(x_{n+1}^{(k)}) \end{bmatrix} \begin{bmatrix} h^{(k+1)} \\ c_1^{(k+1)} \\ \vdots \\ c_n^{(k+1)} \end{bmatrix} = \begin{bmatrix} f(x_1^{(k)}) \\ f(x_2^{(k)}) \\ \vdots \\ f(x_{n+1}^{(k)}) \end{bmatrix}. \tag{4}$$

From Cramer's rule, we can find $h^{(k+1)}$ as

$$h^{(k+1)} = \begin{vmatrix} f(x_1^{(k)}) & g_1(x_1^{(k)}) & \cdots & g_n(x_1^{(k)}) \\ f(x_2^{(k)}) & g_1(x_2^{(k)}) & \cdots & g_n(x_2^{(k)}) \\ \vdots & \vdots & \ddots & \vdots \\ f(x_{n+1}^{(k)}) & g_1(x_{n+1}^{(k)}) & \cdots & g_n(x_{n+1}^{(k)}) \end{vmatrix} \Bigg/ \begin{vmatrix} 1 & g_1(x_1^{(k)}) & \cdots & g_n(x_1^{(k)}) \\ -1 & g_1(x_2^{(k)}) & \cdots & g_n(x_2^{(k)}) \\ \vdots & \vdots & \ddots & \vdots \\ (-1)^n & g_1(x_{n+1}^{(k)}) & \cdots & g_n(x_{n+1}^{(k)}) \end{vmatrix}. \tag{5}$$

Let $M_i^{(k)}$ be the minor of the matrix in (4) removing the first column and the i-th row. Then, (5) can be written as

$$h^{(k+1)} = \frac{\sum_{i=1}^{n+1} f(x_i^{(k)}) M_i^{(k)} (-1)^{i+1}}{\sum_{j=1}^{n+1} M_j^{(k)}}. \tag{6}$$

If f is replaced by any polynomial $p = \sum_{j=1}^{n} c'_j g_j$ in (4), the minimax approximation on $\{x_1^{(k)}, \cdots, x_{n+1}^{(k)}\}$ is p itself. This leads to

$$\frac{\sum_{i=1}^{n+1} p_k(x_i^{(k)}) M_i^{(k)} (-1)^{i+1}}{\sum_{j=1}^{n+1} M_j^{(k)}} = 0. \tag{7}$$

By substracting (6) from (7), and $r_k = p_k - f$, we have

$$-h^{(k+1)} = \frac{\sum_{i=1}^{n+1} r_k(x_i^{(k)}) M_i^{(k)} (-1)^{i+1}}{\sum_{j=1}^{n+1} M_j^{(k)}}.$$

By the fact that $r_k(x_i^{(k)})$'s alternate in sign by the alternating condition for new set of reference points, and all minors M_i have the same sign by Lemma 2.9, we have

$$\left| \sum_{i=1}^{n+1} r_k(x_i^{(k)})(-1)^i M_i^{(k)} \right| = \sum_{i=1}^{n+1} |r_k(x_i^{(k)})||M_i^{(k)}|$$

$$\alpha_{k+1} = |h^{(k+1)}| = \frac{\sum_{i=1}^{n+1} |M_i^{(k)}||r_k(x_i^{(k)})|}{\sum_{j=1}^{n+1} |M_j^{(k)}|}. \tag{8}$$

Now let

$$\theta_i^{(k)} = \frac{|M_i^{(k)}|}{\sum_{j=1}^{n+1} |M_j^{(k)}|}.$$

Since α_{k+1} is weighted average of $|r_k(x_i^{(k)})|$ by $\theta_i^{(k)}$'s as weights, we have $\alpha_{k+1} \geq \gamma_k$ from (2). Note that $\alpha_k \leq \gamma_k \leq \alpha_{k+1}$, and thus α_k is a non-decreasing sequence. This fact is used in the last part of the proof.

Note that there are some $n+1$ alternating points, where the approximate error values alternates, including the global extreme point by Theorem 3.1, and the approximate error values at $x_1^{(k)}, \cdots, x_{n+1}^{(k)}$ have the maximum absolute sum by the maximum absolute sum condition for new set of reference points. That is, $\sum_{i=1}^{n+1} |r_k(x_i^{(k)})|$ is larger than or equal to sum of any $n+1$ absolute error values including β_k and thus we have

$$\sum_{i=1}^{n+1} |r_k(x_i^{(k)})| \geq \beta_k. \tag{9}$$

First, we will prove that $\theta_i^{(k)}$ is larger than a constant $1 - \theta > 0$ throughout the iterations. It is known that $M_i \neq 0$ for all i from the Haar condition. We firstly prove an inequality

$$x_{i+1}^{(k)} - x_i^{(k)} \geq \epsilon > 0, \quad i = 0, \cdots, n, \tag{10}$$

where ϵ does not depend on k. Assume that (10) is not true. Let $x^{(k)} = (x_1^{(k)}, \cdots, x_{n+1}^{(k)})$ be a sequence defined on D^{n+1}. Then $x^{(k)}$ has its subsequence such that $\min_i |x_{i+1}^{(k)} - x_i^{(k)}|$ converges to zero. Since D^{n+1} is a closed and bounded subset of $\mathbb{R}^{n+1}$, it is a compact set, and thus this subsequence also has its subsequence converging to a point $(x_1^*, \cdots, x_{n+1}^*)$. Since $\min_i |x_{i+1}^* - x_i^*| = 0$, there is some i such that $x_i^* = x_{i+1}^*$. Let p be the minimax polynomial of f on $(x_1^*, \cdots, x_{n+1}^*)$. Since there is actually less than or equal to n points, p is the approximate polynomial generated by the Lagrange interpolation, and thus

$$p(x_i^*) = f(x_i^*), \quad i = 1, \cdots, n+1.$$

It is known that $\alpha_1 > 0$ by the fact that α_k is weighted average of absolute approximation errors at the previous set of reference points. Then, there exists a number $\delta > 0$ such that whenever $|y_1 - y_2| < \epsilon$ and $y_1, y_2 \in D$, we have

$$|(p-f)(y_1) - (p-f)(y_2)| < \alpha_1$$

because $p - f$ is a continuous function on the compact set, and thus it is also uniformly continuous. Since there is a subsequence of $(x_1^{(k)}, \cdots, x_{n+1}^{(k)})$ converging to $x^{(k)} = (x_1^*, \cdots, x_{n+1}^*)$, there is some k_0 such that

$$|x_i^{(k_0)} - x_i^*| < \delta, \quad i = 1, \cdots, n+1.$$

Then,

$$|(p-f)(x_i^{(k_0)}) - (p-f)(x_i^*)| = |p(x_i^{(k_0)}) - f(x_i^{(k_0)})| < \alpha_1$$

because $(p-f)(x_i^*) = 0$. In fact, p is not the minimax approximate polynomial in regard to the k-th set of reference points $\{x_1^{(k_0)}, \cdots, x_{n+1}^{(k_0)}\}$. Since α_{k+1} is the error value of the minimax approximate polynomial on $\{x_1^{(k_0)}, \cdots, x_{n+1}^{(k_0)}\}$, we have

$$\alpha_{k+1} \leq \max_i |p(x_i^{(k_0)}) - f(x_i^{(k_0)})| < \alpha_1.$$

This contradicts the fact that α_k is non-decreasing sequence, and thus we have (10).

Now, we will prove that $\theta_i^{(k)}$ is larger than a constant $1 - \theta$. Consider the subset D' of D^{n+1} such that for $(y_1, \cdots, y_{n+1}) \in D'$, $y_{i+1} - y_i \geq \epsilon$. We easily see that D' is a closed and bounded subset in $\mathbb{R}^{n+1}$, and thus D' is a compact set. Then, $|M_i|$, which is the same function as $|M_i^{(k)}|$ except that the inputs are y_i's instead of $x_i^{(k)}$'s, is a continuous function on D^{n+1}, so is on D', and thus there is an element at which $|M_i|$ has the mininum value on D' from the extreme value theorem. From the Haar condition in Definition 2.3, $|M_i|$ cannot be zero because y_i's are distinct and the minimum value of $|M_i|$ is not zero. Since we consider the finite number of functions $|M_i|$'s, the lower bound of all $|M_i|$'s is bigger than zero. In addition, since $\sum_{j=1}^{n+1} |M_j|$ is also a continuous function on D', there is an upper bound of $\sum_{j=1}^{n+1}|M_j|$ on D' from the extreme value theorem. This leads to the fact that θ_i's are lower-bounded beyond zero.

From $\gamma_{k+1} \geq \alpha_{k+1}$, (8), and (9), we have

$$\begin{aligned}\gamma_{k+1}-\gamma_k &\geq \alpha_{k+1}-\gamma_k\\ &= \sum_{i=1}^{n+1}\theta_i^{(k)}(|r_k(x_i^{(k)})|-\gamma_k) && (11)\\ &\geq (1-\theta)(\beta_k-\gamma_k) && (12)\\ &\geq (1-\theta)(\beta^*-\gamma_k). && (13)\end{aligned}$$

From (13), we have

$$\begin{aligned}\beta^*-\gamma_{k+1} &= (\beta^*-\gamma_k)-(\gamma_{k+1}-\gamma_k)\\ &\leq (\beta^*-\gamma_k)-(1-\theta)(\beta^*-\gamma_k)\\ &= \theta(\beta^*-\gamma_k).\end{aligned}$$

Then, we obtain the following inequality for some nonnegative B as

$$\beta^*-\gamma_k \leq B\theta^k. \tag{14}$$

From (12) and (14), we have

$$\begin{aligned}\beta_k-\beta^* &\leq \beta_k-\gamma_k\\ &\leq \frac{1}{1-\theta}(\gamma_{k+1}-\gamma_k)\\ &\leq \frac{1}{1-\theta}(\beta^*-\gamma_k)\\ &\leq \frac{1}{1-\theta}B\theta^k\\ &\leq C\theta^k. && (15)\end{aligned}$$

From Theorem 2.10, there is a constant $\gamma > 0$ such that

$$\|p^*-f\|_\infty+\gamma\|p_k-p^*\|_\infty \leq \|p_k-f\|_\infty.$$

Since $\beta_k = \|p_k-f\|_\infty$, $\beta^* = \|p^*-f\|_\infty$, and (15), we complete the proof by the following inequality

$$\begin{aligned}\|p_k-p^*\|_\infty &= \frac{\beta_k-\beta^*}{\gamma}\\ &\leq A\theta^k\end{aligned}$$

for nonnegative constant A.

□

Note that (11) and Theorem 3.3 can be satisfied if we include the global extreme point to the new set of reference points in the variant of Remez algorithm, instead of the maximum absolute sum condition. Thus, this proof naturally includes the convergence proof of the original variant of the Remez algorithm. From (11), we can easily know that the maximum absolute sum condition is better for the choice of the new set of reference points than the simple inclusion of the global extreme point, in that the rate of the convergence is determined by the value of the absolute sum of errors values, which is confirmed as in Table 1 for the power basis $\{1, x, x^2, \cdots, x^d\}$.

3.3 Efficient Implementation of Modified Remez Algorithm

In this section, we have to consider the issues in each step of Algorithm 2 and suggest how to implement Steps 1, 2, 3, and 4 of Algorithm 2 as follows.

Initialization: The modified Remez algorithm has proven to converge on the optimal minimax approximate polynomial regardless of how initialization is performed in the previous section. However, depending on the initialization method, there can be a large difference in the number of iterations required. Therefore, the closer the polynomial produced by initializing the initial reference points to the optimal minimax approximation polynomial, the less number of iterations is required. We use the node setting method of Han et al. to effectively set the initial reference points in the modified Remez algorithm. Since Han et al's node setting method was for polynomial interpolation, it chooses the $d+1$ number of nodes when we need the approximate polynomial of degree d. Instead, if we need to obtain the optimal minimax approximate polynomial of degree d, we choose the $d+2$ number of nodes with Han et al's method as if we need the approximate polynomial of degree $d+1$, and uses them for the initial reference points.

Finding Approximate Polynomial: A naive approach is finding coefficients of the approximate polynomial with power basis at the current reference points for the continuous function $f(x)$, i.e., we can obtain c_j's in the following equation

$$\sum_{j=0}^{d} c_j x_i^j - f(x_i) = (-1)^i E,$$

where E is also an unknown variable in this system of linear equations. However, this method suffers from the precision problem for the coefficients. It is known that as the degree of the basis of approximate polynomial increases, the coefficients usually decreases, and we have to set higher precision for the coefficients of the higher degree basis. Han et al. [24] use the Chebyshev basis for this coefficient precision problem since the coefficients of a polynomial with the Chebyshev basis usually have the almost same order. Thus, we also use the Chebyshev basis instead of the power basis.

Obtaining Extreme Points: Since we are dealing with a very small minimax approximation error, we have to obtain the extreme points as precisely as possible. Otherwise, we cannot reach the extreme point for the minimax approximate polynomial precisely, and then the minimax approximation error obtained with this algorithm becomes large. Basically, in order to obtain the extreme points, we can scan $p(x) - f(x)$ with a small scan step and obtain the extreme points where the increase and decrease are exchanged. A small scan step increases the accuracy of the extreme point but causes a long scan time accordingly. To be more specific, it takes approximately 2^ℓ proportional time to find the extreme points with the accuracy of ℓ-bit. Therefore, it is necessary to devise a method to obtain high accuracy extreme points more quickly.

In order to obtain the exact point of the extreme value, we use a method of finding the points where the increase and decrease are exchanged and then finding the exact extreme point using a kind of binary search. Let $r(x) = p(x) - f(x)$ and sc be the scan step. If we can find x_0 where $\mu(x_0)r(x_0) \geq |E|$, and $(r(x_0) - r(x_0 - \mathsf{sc}))(r(x_0 + \mathsf{sc}) - r(x_0)) \leq 0$, we obtain the i-th extreme points using the following process successively ℓ times,

$$x_{i,k} = \operatorname*{arg\,max}_{x \in \{x_{i,k-1} - \mathsf{sc}/2^k, x_{i,k-1}, x_{i,k-1} + \mathsf{sc}/2^k\}} |r(x)|, k = 1, 2, \cdots, \ell.$$

Then, we obtain the extreme point with $O(\log(\mathsf{sc}) + \ell)$-bit precision. Since sc needs not to be a too small value, we can find the extreme point with arbitrary precision with linear time to precision ℓ. In summary, we propose that the ℓ-bit precision of the extreme points can be obtained by the linear time of ℓ instead of 2^ℓ.

This procedure for each interval in the approximation region can be performed independently with each other, and thus it can be performed effectively with several threads. Since this step is the slowest step among any other steps in the modified Remez algorithm, the parallel processing for this procedure is desirable to make the whole algorithm much fast.

Obtaining New Reference Points: When we find the new reference points satisfying the local extreme value condition, the alternating condition, and maximum absolute sum condition, there is a naive approach: among local extreme points which satisfy the local extreme value condition, find all $n+1$ points satisfying the alternating condition and choose the $n+1$ points which have the maximum absolute sum value. If we have m local extreme points, we have to investigate $\binom{m}{n+1}$ points, and this value is too large, and thus it makes this algorithm impractical. Thus, we have to find a more efficient method than this naive approach.

We propose a very efficient and provable algorithm for finding the new reference points. The proposed algorithm always gives the $n+1$ points satisfying the three criteria. It can be considered as an elimination method, in that we eliminate some elements for each iteration in the proposed algorithm until we obtain $n+1$ points. It is clear that as long as $m > n+1$, we can find at least one element which may not be included in the new reference points. This proposed algorithm is given in Algorithm 3. Algorithm 3 takes $O(m \log m)$ running time, which is a quasi-linear time algorithm.

To understand the last part of Algorithm 3, the example can be given that if the extreme point x_2 is removed, $T = \{|r(x_1)|+|r(x_2)|, |r(x_2)|+|r(x_3)|, |r(x_3)|+|r(x_4)|, \cdots\}$ is changed to $T = \{|r(x_1)| + |r(x_3)|, |r(x_3)| + |r(x_4)|, \cdots\}$. It is assumed that whenever we remove an element in the ordered set B in Algorithm 3, the remaining points remain sorted and indices are relabeled in increasing order. When we compare the values to remove some extreme points, there are the cases that the compared values are equal or the smallest element is more than one. In such cases, we randomly remove one of these elements. The correctness of Algorithm 3 is proven in the following theorem.

Theorem 3.4. *Algorithm 3 always returns the $n+1$ points satisfying alternating condition and maximum absolute sum condition.*

Proof. Let $B_{\mathsf{init}} = \{t'_1, t'_2, \cdots, t'_m\}$ be the initial elements in B , and let t'_ℓ be an element removed in the first while statement. We first show that each element removed in the first while statement in Algorithm 3 is not included in the new reference; that is, if the subset A of B_{init} having $n+1$ elements satisfies alternating condition and contain this removed element, there is another subset A' of B having $n+1$ elements satisfying alternating condition, not containing the removed element, and having absolute sum larger than or equal to A. Since it is removed in the first while statement, there is an element $t'_{\ell'}$ such that $|r(t'_{\ell'})| \geq |r(t'_\ell)|$, $\mu(t'_{\ell'}) = \mu(t'_\ell)$, and $|\ell' - \ell| = 1$. Clearly, A cannot contain $t'_{\ell'}$, because A satisfies the alternating condition. Let A' be the same set as A except that t'_ℓ is replaced with $t'_{\ell'}$. Then A' also satisfies alternating condition, does not contain t'_ℓ, and has absolute sum larger than or equal to that of A.

We now observe that at the end of the first while statement, B itself satisfies the alternating condition. Then we now have to prove that elements removed in the second while statement in Algorithm 3 are not included in the new reference points. In other words, if the subset A of B_{init} having $n+1$ elements satisfies the alternating condition and contains these removed elements, there is another subset A' of B having $n+1$ elements satisfying alternating condition, not containing these removed elements, and having absolute sum larger than or equal to A. Let t'_ℓ be an element removed in the second while statement be.

Then, there are three cases: at the time of removal of t'_ℓ, the remaining set B can have $n+2$, $n+3$, or larger than $n+3$ elements as in Algorithm 3. We consider each case separately. By the induction argument, we can assume that the remaining set B in each iteration has $n+1$ points that satisfy the alternating condition and have the maximum absolute sum among all possible $n+1$ points in B_{init}. In other words, we can assume that if we have $n+1$ points in B_{init} that satisfy the alternating condition and contain at least one of the removed elements before that time, there are $n+1$ points in the remaining set B at that time such that they satisfy the alternating condition and have absolute sum larger than or equal to the previous ones. This inductive assumption makes us consider only the remaining set B at that iteration instead of all B_{init} in the proof.

Algorithm 3: New Reference

Input : An increasing ordered set of extreme points $B = \{t_1, t_2, \cdots, t_m\}$ with $m \geq n+1$, and number of basis n.

Output: $n+1$ points in B satisfying alternating condition and maximum absolute sum condition.

```
1  $i \leftarrow 1$
2  while $t_i$ is not the last element of $B$ do
3      if $\mu(t_i)\mu(t_{i+1}) = 1$ then
4          Remove from $B$ one of two points $t_i, t_{i+1}$ having less value among
             $\{|r(t_i)|, |r(t_{i+1})|\}$.
5      else
6          $i \leftarrow i+1$
7      end
8  end
9  if $|B| > n+2$ then
10     Calculate all $|r(t_i)| + |r(t_{i+1})|$ for $i = 1, \cdots, |B|-1$ and sort and store this
         values into the array $T$.
11 while $|B| > n+1$ do
12     if $|B| = n+2$ then
13         Remove from $B$ one of two points $t_1, t_{|B|}$ having less value among
             $\{|r(t_1)|, |r(t_{|B|})|\}$.
14     else if $|B| = n+3$ then
15         Insert $|r(t_1)| + |r(t_{|B|})|$ into $T$ and sort $T$. Remove from $B$ the two
             element having the smallest value in $T$.
16     else
17         if $t_1$ or $t_{|B|}$ is included in the smallest element in $T$ then
18             Remove from $B$ only $t_1$ or $t_{|B|}$.
19         else
20             Remove from $B$ the two elements having the smallest element in $T$.
21         end
22         Remove from $T$ all elements related to the removed extreme points,
             and insert into $T$ the sum of absolute error values of the two newly
             adjacent extreme points.
23     end
24 end
```

i) Case of $n+2$: If the remaining set B has $n+2$ elements at the time of removal of t'_ℓ, elements in B at that time are labeled as $t_1, t_2, \cdots, t_{n+2}$, and t'_ℓ is labeled as t_1 or t_{n+2}. Then, $|r(t)|$ at one of the two points t_1 and t_{n+2}, which is not t'_ℓ has the value of $|r(x)|$ larger than or equal to the value at t'_ℓ. We denote this element $t'_{\ell'}$. If we have a subset A of $n+1$ points in the remaining set B that satisfy alternating condition and contain t'_ℓ, $t'_{\ell'}$ must not be in these $n+1$ points due to alternating condition. Let A' be the same set as A except that t'_ℓ is replaced with $t'_{\ell'}$. Then, A' also satisfies the alternating condition, does not contain t'_ℓ, and has absolute sum larger than or equal to that of A.

ii) Case of $n+3$: If the remaining set B has $n+3$ elements at the time of removal of t'_ℓ, the elements in B at that time are labeled as $t_1, t_2, \cdots, t_{n+3}$, and we have to remove two elements. Then, there must be a different element t'_p which is also removed at the time of removal of t'_ℓ. $\{t'_\ell, t'_p\}$ can be $\{t_i, t_{i+1}\}$ for some i or $\{t_1, t_{n+3}\}$ as in Algorithm 3. Since all of the subsets of B having $n+1$ elements that satisfy the alternating condition are the cases of $B\backslash\{t_i, t_{i+1}\}$ for some i or $B\backslash\{t_1, t_{n+3}\}$, one subset of B with the alternating condition that has the maximum absolute sum has to be $B\backslash t'_\ell, t'_p$. Therefore, we can obtain the resulting subset by removing these two elements.

iii) Case of larger than $n+3$: If the remaining set B has elements larger than $n+3$ elements at the time of removal of t'_ℓ, the elements in B at that time are labeled as $t_1, t_2, \cdots, t_j$, where $j > n+3$. Then, there are two cases: One is that t'_ℓ is labeled as t_1 or t_j, and the other is not the first case.

iii)-1 If t'_ℓ is labeled as t_1 or t_j, let t'_p be the adjacent element in B. If the subset A in B that satisfies the alternating condition and contains t'_ℓ also contains t'_p, there is at least one pair of adjacent elements $t'_{\ell'}$ and $t'_{p'}$ in B that is not contained in A, since A satisfies the alternating condition and more than three elements are removed from B. Note that $|r(t'_{\ell'})| + |r(t'_{p'})| \geq |r(t'_\ell)| + |r(t'_p)|$. Let A' be the same set as A except that t'_ℓ and t'_p are replaced with $t'_{\ell'}$ and $t'_{p'}$. A' also satisfies alternating condition, does not contain t'_ℓ, and has absolute sum larger than or equal to that of A.

If A does not contain t'_p, the adjacent element of t'_p which is not t'_ℓ cannot be contained in A, since A satisfies the alternating condition. Let $t'_{\ell'}$ be the adjacent element of t'_p. Note that $|r(t'_{\ell'})| + |r(t'_p)| \geq |r(t'_\ell)| + |r(t'_p)|$. Let A' be the same set with A except that t'_ℓ is replaced with $t'_{\ell'}$. A' also satisfies the alternating condition, does not contain t'_ℓ, and has absolute sum larger than or equal to that of A.

iii)-2 If t'_ℓ is not labeled as t_1 or t_j, the adjacent element t'_p of t'_ℓ in B which $|r(t'_\ell)| + |r(t'_p)|$ is the smallest value in T cannot be t_1 or t_j. If this is the case, t'_ℓ cannot be removed but t'_p is removed in that iteration. If the alternated subset A in B that contains t'_ℓ also contain t'_p, there is at least one pair of adjacent elements $t'_{\ell'}$ and $t'_{p'}$ in B that is not contained in A. Note also that $|r(t'_{\ell'})| + |r(t'_{p'})| \geq |r(t'_\ell)| + |r(t'_p)|$. Let A' be the same set as A except that t'_ℓ and t'_p are replaced with $t'_{\ell'}$

and $t'_{p'}$. A' also satisfies the alternating condition, does not contain t'_ℓ, and has absolute sum larger than or equal to that of A.
If A does not contain t'_p, there is the adjacent element of t'_p which is not t'_ℓ, since t'_p is not t_1 or t_j. Let $t'_{\ell'}$ be adjacent element of t'_p. Then, $t'_{\ell'}$ cannot be contained in A, since A satisfies the alternating condition. Note that $|r(t'_{\ell'})| + |r(t'_p)| \geq |r(t'_\ell)| + |r(t'_p)|$. Let A' be the same set as A except that t'_ℓ is replaced with $t'_{\ell'}$. A' also satisfies alternating condition, does not contain t'_ℓ, and has absolute sum larger than or equal to that of A.

Thus, we prove the theorem. □

3.4 Numerical Analysis with Modified Remez Algorithm

In this subsection, we show the result of the numerical analysis of the modified Remez algorithm for its efficiency and the optimal minimax approximation error.

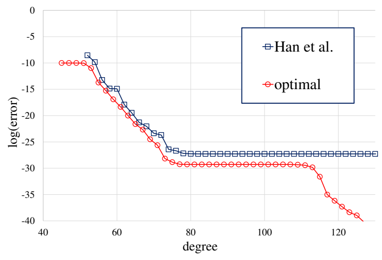
Maximum Sum Condition: Table 1 shows the number of iterations required to converge to the optimal minimax approximate polynomial in the variant of the Remez algorithm and the modified Remez algorithm. The initial set of reference points is selected uniformly in each interval as soon as possible since we want to observe their performances in the worst case. While the selection for new reference points is not unique for each iteration in the variant of the Remez algorithm, the modified Remez algorithm selects the new reference points uniquely for each iteration. Thus, when we analyze the variant of the Remez algorithm, we select the new reference points randomly for each iteration among the possible sets of reference points that satisfy the local extreme value condition and the alternating condition and have the global extreme point. We set the approximation parameter δ in Algorithm 2 as 2^{-40} and repeat this simulation with 100 times. It shows that the modified Remez algorithm is much better to reduce the iteration number of the Remez algorithm.

Note that the number of iterations depends on the initial set of reference points. In fact, the uniformly distributed reference points are not desirable as an initial set of reference points because these reference points are far from the converged reference points. In fact, the modified Remez algorithm with the initialization method explained in the previous subsection only needs 4$\sim$14 iterations. The overall running time of the modified Remez algorithm with the method in the previous subsection is 1$\sim$3 seconds by PC with AMD Ryzen Threadripper 1950X 16-core CPU @ 3.40GHz.

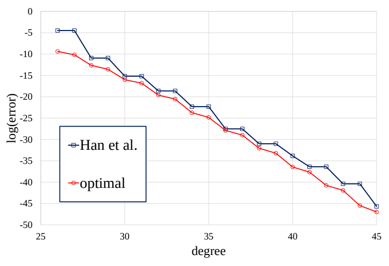
Minimax Error: We obtain the optimal minimax approximate polynomials for the modular reduction function and the scaled cosine function with the scaling number two. Fig. 1(a) shows the minimax approximation error of the approximate polynomial of the modular reduction function derived by the modified

Table 1: Comparison of iteration numbers between the modified Remez algorithm and the variant of the Remez algorithm for $\delta = 2^{-40}$

degree of approx. poly.	modified Remez algorithm	variant of Remez algorithm			
		average	standard deviation	max	min
79	**28**	60.0	9.38	82	41
99	**8**	17.1	3.34	28	11
119	**26**	53.4	8.10	79	37
139	**39**	60.3	4.71	79	48
159	**39**	72.1	9.71	98	42
179	**48**	72.3	9.72	105	53
199	**56**	80.4	7.28	94	60



(a) Modular reduction function



(b) Cosine function with scaling number two

Fig. 1: Comparison of minimax approximatio error between the previous approximation method and the modified Remez algorithm.

Remez algorithm and the minimax approximation error of the previous homomorphic modular reduction method with scaling number zero in [24], compared to the modular reduction function. That is, let $p_1(x)$ be the optimal minimax approximate polynomial of the normod function and let $q_1(x)$ be the approximate polynomial obtained by Han et al.'s method with scaling number zero when the half-width of approximation region is 2^{-10}. Then, $\max_{x\in D} |p_1(x) - \mathsf{normod}(x)|$ and $\max_{x\in D} |q_1(x) - \mathsf{normod}(x)|$ are compared in Fig. 1(a). Note that while the minimax approximation error of the approximate polynomial of the modular reduction function decreases steadily as the degree of the approximate polynomial increases, the minimax approximation error of the previous method does not decrease when the degree is larger than 76 because of the approximation error between the modular reduction function and the sine/cosine function.

Fig. 1(b) shows the minimax approximation error of the composition of the approximate polynomial of the scaled cosine function with scaling number two derived by the modified Remez algorithm and two double angle formulas, and the minimax approximation error of method in [24], compared to the cosine function. That is, let $p_2(x)$ be the optimal minimax approximate polynomial of $\cos\left(\frac{\pi}{2}(x-1/4)\right)$ and let $q_2(x)$ be the approximate polynomial obtained by Han et al.'s method with scaling number two when the half-width of approximation region is 2^{-3}. If $r(x) = 2x^2 - 1$, then $\max_{x\in D} |r \circ r \circ p_2(x) - \sin(2\pi x)|$ and $\max_{x\in D} |r \circ r \circ q_2(x) - \sin(2\pi x)|$ are compared in Fig. 1(b). The proposed method improves the minimax approximation error by 2.3 bits on average, and by 5 bits at maximum for the same degree of the approximate polynomial. This improvement leads to a reduction of 1∼2 degrees for the given minimax approximation error.

4 Message Precision in Bootstrapping with Modified Remez Algorithm of RNS-CKKS Scheme

4.1 Analysis in Rescaling Operation

The rescaling operation, which is required after the homomorphic scalar multiplication or the homomorphic non-scalar multiplication, causes approximation error in the RNS-CKKS scheme because the message is divided by the prime number used as one of the RNS modulus rather than by the exact scaling factor during the rescaling operation. Let Δ be the scaling factor and q_i be the prime number divided during the rescaling operation. Then, the rescaled message becomes Δ/q_i times the intended message. To the best of our knowledge, there is no paper analyzing how much this message precision is related to the scaling factor, and thus we address the relation between them in this section.

The conditions for choosing moduli in the RNS-CKKS scheme are as follows. First, they should satisfy $q_i \equiv 1 \mod 2N$ to enable the number-theoretic transform. Second, they have to be as close as possible to the scaling factor to minimize the rescaling errors. To satisfy these conditions, we start with $\Delta + 1$ and then add or subtract $2N$ with determining whether each number is a prime number and choose the prime numbers nearest to Δ as moduli.

In this process, there are two parameters that affect the rescaling error Δ/q_i: the degree of the polynomial modulus N and the scaling factor Δ. Δ affects the density of the prime number. The prime number theorem [3], a well-known theorem in the analytic number theory, states that the density of the prime numbers is roughly proportional to $1/\log\Delta$ near Δ. In other words, if the integer we want to determine whether it is a prime number is ℓ-bit number, the probability that a number is a prime number is proportional to $\frac{1}{\ell}$. N determines the density of the number meeting the first condition above. Assuming that the probability of a prime number meeting the first condition depends purely on the prime density in that area near the number, the value of $|q_i - \Delta|$ is roughly proportional to $N\log\Delta$. Then, the distance from Δ/q_i to 1 will be proportional to $\frac{N\log\Delta}{\Delta}$. Therefore, the smaller N and the larger the scaling factor, the smaller the rescaling error.

To check that this analysis is practically meaningful, we perform the scalar multiplication by 1.0 and then rescale the ciphertext to examine the rescaling error. The left table in Table 2 shows the average approximation error for various scaling factors when the polynomial modulus degree N is set to be 2^{17}, and the right table shows the average approximation error for various polynomial modulus degree N when the scaling factor is set to be 60. The number of slots is set to be 2^{14} and we compute the average of the approximation error of each slot. We can see that the rescaling error is inversely proportional to the scaling factor and proportional to N. Since the numerical rescaling error follows the theoretical relation above, we can see that the rescaling error is predominant compared to other errors.

Table 2: Approximation error for various scaling factors and various polynomial modulus degree when the homomorphic scalar multiplication with 1.0 is performed

scaling factor Δ	approximation error	message precision (bits)
40	9.93×10^{-8}	23.3
45	4.10×10^{-9}	27.9
50	1.31×10^{-10}	32.8
55	3.73×10^{-12}	38.0
60	1.27×10^{-13}	42.8

polynomial modulus degree N	approximation error	message precision (bits)
15	4.49×10^{-14}	44.3
16	6.46×10^{-14}	43.8
17	1.27×10^{-13}	42.8
18	3.20×10^{-13}	41.5
19	5.33×10^{-13}	40.7

We can observe two trade-offs with respect to the precision in the RNS-CKKS scheme. The first trade-off is between the number of levels available and the message precision. With a fixed security parameter, the upper bound of the product of all RNS moduli is given according to the polynomial modulus degree N of the ring and this bound increases as N increases. If the maximum number of levels is not sufficient, we have to increase the value of N, which reduces the message precision. Therefore, when increasing the value of N to increase the number of levels available, the resulting decrease in message precision should be taken into account.

The second trade-off is between the scaling factor and the message precision. This trade-off has a greater effect than the previous trade-off. Because the rescaling error is proportional to $\frac{\log \Delta}{\Delta}$, the message precision is improved almost proportionally to the scaling factor. Therefore, in order to ensure sufficient message precision, the large scaling factor should be used.

4.2 Bit Length Difference between Default Scaling Factor and Bootstrapping Scaling Factor

The bit length difference between the default scaling factor Δ and the bootstrapping scaling factor Δ_{boot}, which will be denoted as $\delta_{\mathsf{diff}} = \log \Delta_{\mathsf{boot}} - \log \Delta$, is closely related to the message precision. The value of δ_{diff} is usually chosen as

10 bits to lower the difference between the modular reduction function and the sine/cosine function since the half-width of each interval in the approximation region is $2^{-\delta_{\mathsf{diff}}}$. Two factors affect the message precision due to the value of δ_{diff}.

The first factor is that δ_{diff} determines the maximum value of the default scaling factor Δ. If we use the 64-bit integer data type, $\log \Delta_{\mathsf{boot}}$ has to be less than 61, and thus the maximum value of $\log \Delta$ is $61 - \delta_{\mathsf{diff}}$. As mentioned in the previous subsection, the scaling factor has a direct effect on rescaling precision. If we use large δ_{diff}, the maximum message precision for the intended operations may not be sufficiently high.

The second factor is that it also causes loss to the message precision of the bootstrapping itself. In the bootstrapping procedure, we need to divide the message by $2^{\delta_{\mathsf{diff}}}$ so that it can be included in the approximation region, and multiply $2^{\delta_{\mathsf{diff}}}$ at the end of the bootstrapping. If the precision error until multiplying $2^{\delta_{\mathsf{diff}}}$ is e, the final error becomes $2^{\delta_{\mathsf{diff}}}e$. e cannot be reduced below a certain error value because of the rescaling error dealt in the previous subsection. If we denote this lower bound as $e_b = 2^{-\delta_b}$, the message precision will be $\delta_b - \delta_{\mathsf{diff}}$.

Because δ_{diff} has a significant effect on both the message precision of the bootstrapping and the message precision of the intended operation in the application, it is desirable to reduce the δ_{diff} to prevent this precision loss. However, the difference between the sine/cosine function and the modular reduction function is somewhat dominant, and this difference becomes more dominant as δ_{diff} increases. This effect is numerically shown in the next subsection.

4.3 Numerical Analysis of Message Precision in Bootstrapping with Modified Remez Algorithm in `FullRNS-HEAAN` Library

Since the previous researches for the bootstrapping of the RNS-CKKS scheme did not deal deeply with its message precision, we numerically analyze the message precision for the bootstrapping with modified Remez algorithm of the RNS-CKKS scheme by changing several parameters: the degree d of the approximate polynomial of the scaled cosine function, the bit-length difference δ_{diff} between the default scaling factor and the bootstrapping scaling factor, the bootstrapping scaling factor Δ_{boot}, and the number of the slots.

Numerical analysis in this section is conducted in PC with AMD Ryzen Threadripper 1950X 16-core CPU @ 3.40GHz, and the `FullRNS-HEAAN` library [1] is used. The double angle formula for the cosine function is assumed to be used twice. The modified Remez algorithm is used to obtain the optimal minimax approximate polynomial in all simulations, rather than polynomial approximation methods in the previous papers [9,12,24]. The polynomial modulus degree is set to be 2^{17}, and the maximum modulus for the ciphertext is set to be 2^{2250}, which satisfies the 128-bit security as in [15]. When the 61-bit scaling factor is used, the maximum level available is 36. The CoeffToSlot and SlotToCoeff procedures in [9] are used in all simulations, and the consumed level of depth in each procedure is fixed to be two.

Degree of Approximate Polynomial: Table 3 shows the message precision of the bootstrapping with the modified Remez algorithm when the degree of the approximate polynomial for the scaled cosine function is changed. The value of $\log \delta_{\mathsf{diff}}$ is 10, $\log \Delta_{\mathsf{boot}}$ is 61, and the number of the slots is 2^{10} in this simulation. The approximation error means the minimax error of the approximate polynomial for the scaled cosine function, and the bootstrapping error means the average error for each slot when the bootstrapping is performed with the library. When the scaling factor is changed from the bootstrapping scaling factor to the default scaling factor, the message and its error are multiplied by δ_{diff}. We show both the bootstrapping error before changing the scaling factor and that after changing the scaling factor. Although the approximation error continues to decrease as the degree of the approximate polynomial increases, the bootstrapping error does not decrease below a certain value. This bound is caused by either the difference between the modular reduction function and the cosine function or the approximate rescaling error depending on the situation. Thus, we cannot raise the message precision infinitely by using high degree approximate polynomial. The actual lower bound of the bootstrapping error before changing the scaling factor is denoted by e_{min} in this section, and then the lower bound of the bootstrapping error after changing the scaling factor is $e_{\mathsf{min}}\delta_{\mathsf{diff}}$.

Table 3: Message precision of the bootstrapping with the modified Remez algorithm for various degrees of the approximate polynomials

degree of approx. poly.	approximation error by the optimal minimax polynomial	bootstrapping error		message precision (bits)
		before changing scaling factor e_{min}	after changing scaling factor $e_{\mathsf{min}}\delta_{\mathsf{diff}}$	
27	2.16×10^{-6}	5.52×10^{-5}	5.66×10^{-2}	4.14
30	4.51×10^{-9}	1.05×10^{-7}	1.08×10^{-4}	13.2
33	1.41×10^{-10}	1.85×10^{-8}	1.89×10^{-5}	15.7
36	4.19×10^{-13}	1.69×10^{-8}	1.73×10^{-5}	15.8
39	5.24×10^{-15}	6.25×10^{-9}	6.40×10^{-6}	17.3
42	1.785×10^{-17}	6.25×10^{-9}	6.40×10^{-6}	17.3

Value of δ_{diff}: Table 4 shows the maximum message precision of the bootstrapping with modified Remez algorithm for various δ_{diff}. The degree of the approximate polynomials for each case is set to be large enough to reach the minimum approximate error e_{min}, and the scaling factor and the number of slots are fixed to be 61 and 2^{10}, respectively.

The bootstrapping error after changing the scaling factor is $e_{\mathsf{min}}\delta_{\mathsf{diff}}$. As δ_{diff} decreases, e_{min} increases rapidly so that the $e_{\mathsf{min}}\delta_{\mathsf{diff}}$ grows. This is because the difference between the modular reduction function and the cosine function becomes larger when the approximation region is enlarged. We can naively expect that the bootstrapping error can be decreased infinitely when $\log \delta_{\mathsf{diff}}$ is increased, because $|\epsilon - \sin \epsilon| = O(\epsilon^3)$. However, if the δ_{diff} is larger than 10, the value of e_{min} does not decrease, and thus $e_{\mathsf{min}}\delta_{\mathsf{diff}}$ increases. This lower bound of e_{min} is caused by the approximate rescaling error and the homomorphic linear transform in the bootstrapping. This bound of e_{min} is related to the bootstrapping scaling factor Δ_{boot} and the number of slots, which will be dealt with in the following paragraphs.

Table 4: Message precision of the bootstrapping with the modified Remez algorithm for various values of $\log \delta_{\mathsf{diff}}$

$\log \delta_{\mathsf{diff}}$	bootstrapping error		message precision (bits)
	before changing scaling factor e_{min}	after changing scaling factor $e_{\mathsf{min}}\delta_{\mathsf{diff}}$	
3	1.13×10^{-3}	9.02×10^{-3}	6.79
4	1.42×10^{-4}	2.27×10^{-3}	8.78
6	2.22×10^{-6}	1.42×10^{-4}	12.8
8	3.51×10^{-8}	8.99×10^{-6}	16.8
9	7.37×10^{-9}	3.77×10^{-6}	18.0
10	6.25×10^{-9}	6.40×10^{-6}	17.2
11	6.24×10^{-9}	1.28×10^{-5}	16.3

Bootstrapping Scaling Factor: Table 5 shows the maximum message precision for various bootstrapping scaling factors when the number of slots is 2^{10}. The degree of the approximate polynomial and the value of δ_{diff} are set to reach the lower bound of e_{min} for each bootstrapping scaling factor and to minimize the value of $e_{\mathsf{min}}\delta_{\mathsf{diff}}$, which determines the actual message precision of the bootstrapping. The second column in Table 5 shows the lower bound of e_{min}, the third column shows the value of δ_{diff} which minimizes $e_{\mathsf{min}}\delta_{\mathsf{diff}}$, and the last column shows the maximum message precision with corresponding bootstrapping scaling factor.

Since the scaling factor affects the approximate rescaling error as we stated in Subsection 4.1, the maximum message precision of the bootstrapping in the RNS-CKKS scheme decreases as the bootstrapping scaling factor decreases. There are many homomorphic multiplications with approximate rescaling in the process of the bootstrapping, and thus the minimum e_{min} is much larger than the approximate rescaling error. This means that we have to use as large a bootstrapping scaling factor as possible when we need precise bootstrapping. Since the bootstrapping scaling factor is related to the multiplicative depth, this gives the trade-off between the depth and the precision.

Number of Slots: Table 6 shows the maximum message precision for various numbers of slots when the bootstrapping scaling factor is 61, the maximum

Table 5: Maximum message precision of the bootstrapping with the modified Remez algorithm for various bootstrapping scaling factors

$\log \Delta_{\mathsf{boot}}$	$\log \delta_{\mathsf{diff}}$	bootstrapping error		message precision (bits)
		before changing scaling factor e_{min}	after changing scaling factor $e_{\mathsf{min}}\delta_{\mathsf{diff}}$	
50	5	2.16×10^{-5}	6.90×10^{-4}	10.5
53	6	2.76×10^{-6}	1.77×10^{-4}	12.5
57	7	3.21×10^{-7}	4.11×10^{-5}	14.6
61	9	7.37×10^{-9}	3.77×10^{-6}	18.0

Table 6: Maximum message precision of the bootstrapping with modified Remez algorithm for various numbers of slots

$\log n$	degree of approx. poly.	$\log \delta_{\mathsf{diff}}$	bootstrapping error		message precision (bits)	remaining depth level	running time (s)
			before changing scaling factor e_{min}	after changing scaling factor $e_{\mathsf{min}}\delta_{\mathsf{diff}}$			
3	37	10	8.42×10^{-10}	8.62×10^{-7}	20.1	18	55.4
5	37	10	1.09×10^{-9}	1.12×10^{-6}	19.8	18	67.8
8	37	9	5.19×10^{-9}	2.66×10^{-6}	18.5	18	100.0
10	37	9	7.37×10^{-9}	3.77×10^{-6}	18.0	18	120.0
12	37	9	1.28×10^{-8}	6.53×10^{-6}	17.2	18	192.2
14	36	8	4.08×10^{-8}	1.04×10^{-5}	16.5	18	293.2

scaling factor. The degree of the approximate polynomials and δ_{diff} are set to the same as Table 5. The remaining level is computed with the bootstrapping scaling factor, but the actual level can be more than them because the default scaling factor is less than the bootstrapping scaling factor. The error analysis in [12] shows that the bootstrapping error is proportional to $\sqrt{n}$ by SlotToCoeff step, where n is the number of slots. The result of Table 6 corresponds to this error analysis. This gives the trade-off between the number of slots and the message precision. Note that all precision is less than 23-bit precision, which is the precision of the single-precision floating point number system. Thus, the message precision of the bootstrapping with the modified Remez algorithm is not that big enough to be used in reliable homomorphic computations. We will solve this problem in the next section by using the composite function method with the inverse sine function.

5 Improvement of Message Precision by Composite Function Approximation of Modular Reduction Function

At first glance, it seems to be the best method to use the optimal minimax approximate polynomials for the modular reduction function. However, we can see that some of the coefficients of the optimal minimax approximate polynomials with regard to the Chebyshev basis are so large that the amplified approximate errors by these coefficients totally distort the messages in the ciphertext. On the other hand, the coefficients of the optimal minimax approximate polynomial of the scaled sine/cosine functions with more than one scale number are small enough not to distort the messages. Thus, the approximation of the modular reduction function by the sine/cosine function is essential for the correctness in the RNS-CKKS scheme.

When we adhere to the approximation by the scaled sine/cosine function, the difference of the modular reduction function and the sine/cosine function is a crucial obstacle, which is mentioned as an important open problem in Han et al.'s paper [24]. This difference is sharply increased as the approximation region of the modular reduction function becomes longer, and this prevents us from reducing δ_{diff}.

5.1 Composite Function Approximation of Modular Reduction Function by Inverse Sine Function

We propose a simple and novel method for solving this problem, which is called the composite function approximation method. In short, we compose the optimal minimax approximate polynomial of the sine/cosine function and the approximate polynomial of the inverse sine function. It is easy to check that if we have two functions f ad g for $0 < \epsilon < \frac{1}{4}$ as

$$f : \bigcup_{k=-\infty}^{\infty} [2\pi(k-\epsilon), 2\pi(k+\epsilon)] \to [-\sin 2\pi\epsilon, \sin 2\pi\epsilon], \quad f(x) = \sin x$$

$$g : [-\sin 2\pi\epsilon, \sin 2\pi\epsilon] \to [-2\pi\epsilon, 2\pi\epsilon], \quad g(x) = \arcsin x,$$

then the following equation holds as

$$x - 2\pi \cdot \mathsf{round}\left(\frac{x}{2\pi}\right) = (g \circ f)(x), \quad x \in \bigcup_{k=-\infty}^{\infty} [2\pi(k-\epsilon), 2\pi(k+\epsilon)].$$

If we substitute $t = \frac{x}{2\pi}$, then we have

$$\mathsf{normod}(t) = \frac{1}{2\pi}(g \circ f)(2\pi t), \quad t \in \bigcup_{k=-\infty}^{\infty} [k-\epsilon, k+\epsilon]. \tag{16}$$

If we approximate both f and g with the optimal minimax approximate polynomials derived by the modified Remez algorithm, we can approximate the modular reduction function with any small approximate error by the composition of f and g. Note that $g(x)$ can be approximated very well with some approximate polynomials of small degree since the domain of $g(x)$ is only one interval. Indeed, the cosine approximation with double-angle formula in [24] can be regarded as the special case of the proposed composite function approximation, in that they approximate $g(x)$ with x, that is, the identity function. Note that the cosine function in [24] is merely a parallel shift of the sine function. Thus, it is said that they approximate the sine function instead of the cosine function.

The sine function f was evaluated by composing the scaled cosine function and several double-angle formulas in [24]. If the number of the used double-angle formula is ℓ, then the functions h_1, h_2, and h_3 are defined as

$$h_1(x) = \cos\left(\frac{2\pi}{2^\ell}\left(x - \frac{1}{4}\right)\right),\ h_2(x) = 2x^2 - 1,\ h_3(x) = \frac{1}{2\pi}\arcsin(x).$$

Then, the normod function, which is equivalent to the modular reduction function, can be represented as

$$\mathsf{normod}(x) = h_3 \circ h_2^\ell \circ h_1(x).$$

Thus, if $\tilde{h}_1$ is the optimal minimax approximate polynomial of h_1 and $\tilde{h}_3$ is that of h_3, we can approximate normod function by the composition of several polynoimals as

$$\mathsf{normod}(x) \approx \tilde{h}_3 \circ h_2^\ell \circ \tilde{h}_1(x).$$

With this method, we can approximate the modular reduction function by the composition of several polynomials at arbitrary precision. This enables us to reduce δ_{diff} to 3 and reach the message precision of $\delta_b - \delta_{\mathsf{diff}}$, which is the best precision mentioned in the previous section. The next section shows that we can indeed reach this high precision in the `FullRNS-HEAAN` library.

5.2 Simulation Result

In this subsection, we demonstrate that the composite function method can improve the message precision in the RNS-CKKS scheme. The simulation environment is the same as the simulation in Section 4. Up to now, bootstrapping simulation has been performed with a single thread in previous works, but we perform the simulations with a multi-thread environment. Most of the current CPU's support multi-core and the homomorphic operations are likely to be performed on the high-performance servers with many cores used by the enterprise rather than on light devices for individuals. Thus, the performance of running time in a multi-core environment represents the actual running time performance well. Since the computations in each modulus in the RNS-CKKS scheme can be processed in parallel, we evaluate each running time by modifying the `FullRNS-HEAAN` library, which was designed to use only a single thread, to generate and use as many threads as the number of levels for all homomorphic operations.

Table 7 shows that the value of e_{min} with the composite function method does not change. The degrees of approximate polynomials of scaled cosine function and inverse sine function are set to minimize e_{min}, and these degrees are shown in Table 7. In contrast to the result in Table 4, all of the values of e_{min} in Table 7 are almost the same as the minimum value of e_{min} in Table 4 regardless of

δ_{diff}. Since e_{min} is fixed with the minimum value, the bootstrapping precision, which is determined by $e_{\mathsf{min}}\delta_{\mathsf{diff}}$, is increased as δ_{diff} decreases. Thus, the maximum message precision is increased by 5.8-7.6 bits, and becomes 22.4-27.8 bit precision, most of which are larger than 23-bit precision, the precision of the single-precision floating-point number.

Table 7: Maximum message precision of the bootstrapping with modified Remez algorithm and composite function method for various δ_{diff}

$\log \delta_{\mathsf{diff}}$	bootstrapping error		message precision (bits)
	before changing scaling factor e_{min}	after changing scaling factor $e_{\mathsf{min}}\delta_{\mathsf{diff}}$	
3	6.25×10^{-9}	5.00×10^{-8}	24.3
4	6.24×10^{-9}	9.99×10^{-8}	23.3
6	6.24×10^{-9}	3.99×10^{-7}	21.3
8	6.24×10^{-9}	1.60×10^{-6}	19.3
9	6.24×10^{-9}	3.19×10^{-6}	18.3

Table 8 shows the maximum precision of the bootstrapping with the modified Remez algorithm and composite function method for various slots. The δ_{diff} value and $\log \varDelta_{\mathsf{diff}}$ value are set to be 3 and 61, respectively. Even when we use 2^{12} slots, the message precision is larger than the 23-bit precision. Even the worst-case message precision in the bootstrapping with the modified Remez algorithm and the composite function method with many slots is larger than the best-case message precision in the bootstrapping without the composite function method with small slots. Thus, we make the bootstrapping of the RNS-CKKS scheme more reliable enough to be used in practical applications.

Table 8: Maximum message precision of the bootstrapping with composite function method for various number of slots

$\log n$	deg. of app. poly. of cos.	deg. of app. poly. of inv. sine	bootstrapping error		message precision (bits)	remaining depth level	running time (s)	increasing ratio (%)
			before changing scaling factor e_{min}	after changing scaling factor $e_{\mathsf{min}}\delta_{\mathsf{diff}}$				
3	38	21	5.38×10^{-10}	4.30×10^{-9}	27.8	14	65.8	18.8
5	38	19	9.94×10^{-10}	7.95×10^{-9}	26.9	14	80.6	18.9
8	37	17	3.10×10^{-9}	2.48×10^{-8}	25.3	14	111.0	10.9
10	37	17	5.91×10^{-9}	4.72×10^{-8}	24.3	14	128.7	7.3
12	37	17	1.21×10^{-8}	9.71×10^{-8}	23.3	14	201.4	4.8
14	37	17	2.29×10^{-8}	1.83×10^{-7}	22.4	14	300.3	2.4

Since we have to evaluate the approximate polynomial for the inverse sine function, the running time is increased compared to the previous bootstrapping. This gives the trade-off between the running time and the message precision of the bootstrapping, but this trade-off can be considered as a practically meaningful trade-off. Since the CoeffToSlot and SlotToCoeff steps take more time when the number of slots is large, the increasing ratio of computation time is somewhat small in this case. These simulation results suggest that the RNS-CKKS scheme with the improved message precision bootstrapping can be practically used in applications with deep-depth operations.

6 Conclusion and Future Works

We first proposed the algorithm for obtaining the optimal minimax approximate polynomial for any continuous function on the union of the finite set, including the scaled cosine function on separate approximate regions. Then we analyzed the message precision of the bootstrapping with the modified Remez algorithm in RNS-CKKS and its maximum message precision is measured in the `FullRNS-HEAAN` library. We also proposed the composite function method with inverse sine function to improve the message precision of the bootstrapping significantly, and thus the improved message precision bootstrapping has the precision higher than the precision of the single-precision floating-point number system, even when lots of slots are used. Thus, the deep-depth operations in advanced applications, such as training a convolutional neural network for encrypted data, is needed to be implemented by the RNS-CKKS scheme with the improved message precision bootstrapping.

Since the `FullRNS-HEAAN` library is just for the proof-of-concept of the RNS-CKKS scheme, the bootstrapping for the practical homomorphic encryption library supporting the RNS-CKKS scheme, such as `SEAL` and `PALISADE`, will have better running time performance with similar message precision, which is important future work.

References

1. FullRNS-HEAAN. `https://github.com/KyoohyungHan/FullRNS-HEAAN` (Oct 2018)
2. PALISADE Lattice Cryptography Library (release 1.10.4). `https://palisade-crypto.org/` (Sep 2020)
3. Apostol, T.M.: *Introduction to analytic number theory*. Springer Science & Business Media (2013)
4. Boemer, F., Costache, A., Cammarota, R., Wierzynski, C.: nGraph-HE2: A high-throughput framework for neural network inference on encrypted data. In: *Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography*. pp. 45–56 (2019)
5. Boemer, F., Lao, Y., Cammarota, R., Wierzynski, C.: nGraph-HE: a graph compiler for deep learning on homomorphically encrypted data. In: *Proceedings of the 16th ACM International Conference on Computing Frontiers*. pp. 3–13 (2019)

6. Bourse, F., Minelli, M., Minihold, M., Paillier, P.: Fast homomorphic evaluation of deep discretized neural networks. In: *Advances in Cryptology-CRYPTO 2018*. pp. 483–512. Springer (2018)
7. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory* **6**(3), 13 (2014)
8. Brutzkus, A., Gilad-Bachrach, R., Elisha, O.: Low latency privacy preserving inference. In: *International Conference on Machine Learning*. pp. 812–821. PMLR (2019)
9. Chen, H., Chillotti, I., Song, Y.: Improved bootstrapping for approximate homomorphic encryption. In: *Advances in Cryptology-EUROCRYPT 2019*. pp. 34–54. Springer (2019)
10. Chen, Y., Luo, T., Liu, S., Zhang, S., He, L., Wang, J., Li, L., Chen, T., Xu, Z., Sun, N., et al.: Dadiannao: A machine-learning supercomputer. In: *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*. pp. 609–622. IEEE (2014)
11. Cheney, E.: *Introduction to approximation theory*. McGraw-Hill (1966)
12. Cheon, J., Han, K., Kim, A., Kim, M., Song, Y.: Bootstrapping for approximate homomorphic encryption. In: *Advances in Cryptology-EUROCRYPT 2018*. pp. 360–384. Springer (2018)
13. Cheon, J., Han, K., Kim, A., Kim, M., Song, Y.: A full rns variant of approximate homomorphic encryption. In: *International Conference on Selected Areas in Cryptography-SAC 2018*. pp. 347–368. Springer (2018)
14. Cheon, J., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: *Advances in Cryptology-ASIACRYPT 2017*. pp. 409–437. Springer (2017)
15. Cheon, J., Kim, D., Kim, D.: Efficient homomorphic comparison methods with optimal complexity. Cryptology ePrint Archive, Report 2019/1234 (2019)
16. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: fast fully homomorphic encryption over the torus. *Journal of Cryptology* **33**(1), 34–91 (2020)
17. Dathathri, R., Saarikivi, O., Chen, H., Laine, K., Lauter, K., Maleki, S., Musuvathi, M., Mytkowicz, T.: Chet: an optimizing compiler for fully-homomorphic neural-network inferencing. In: *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*. pp. 142–156 (2019)
18. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive p. Report 2012/144 (2012)
19. Filip, S.: A robust and scalable implementation of the Parks-McClellan algorithm for designing FIR filters. *ACM Transactions on Mathematical Software* **43**(1), 1–24 (2016)
20. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*. pp. 169–178 (2009)
21. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: *Advances in Cryptology-CRYPTO 2013*. pp. 75–92. Springer (2013)
22. Gupta, S., Agrawal, A., Gopalakrishnan, K., Narayanan, P.: Deep learning with limited numerical precision. In: *International Conference on Machine Learning*. pp. 1737–1746 (2015)
23. Gysel, P., Motamedi, M., Ghiasi, S.: Hardware-oriented approximation of convolutional neural networks. In: *International Conference on Learning Representation* (2016)

24. Han, K., Ki, D.: Better bootstrapping for approximate homomorphic encryption. In: *Cryptographers' Track at the RSA Conference*. pp. 364–390. Springer (2020)
25. Jiang, X., Kim, M., Lauter, K., Song, Y.: Secure outsourced matrix computation and application to neural networks. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. pp. 1209–1222 (2018)
26. McClellan, J., Parks, T.: A personal history of the Parks-McClellan algorithm. *IEEE Signal Processing Magazine* **22**(2), 82–86 (2005)
27. Powell, M.: *Approximation theory and methods*. Cambridge University Press (1981)
28. Remez, E.: Sur la détermination des polynômes d'approximation de degré donnée. *Communications of the Kharkov Mathematical Society* **10**(196), 41–63 (1934)
29. Rudin, W.: *Principles of mathematical analysis*, vol. 3. McGraw-Hill New York (1964)
30. Microsoft SEAL (release 3.5). `https://github.com/Microsoft/SEAL` (Apr 2020), microsoft Research, Redmond, WA.
31. Van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: *Advances in Cryptology-EUROCRYPT 2010*. pp. 24–43. Springer (2010)