Minimax Approximation of Sign Function by Composite Polynomial for Homomorphic Comparison

Eunsang Lee, Joon-Woo Lee, *Graduate Student Member, IEEE*, Jong-Seon No, *Fellow, IEEE*, and Young-Sik Kim, *Member, IEEE*

Abstract—The comparison operation for two numbers is one of the most frequently used operations in several applications, including deep learning. As such, lots of research has been conducted with the goal of efficiently evaluating the comparison operation in homomorphic encryption schemes. Recently, Cheon *et al.* (Asiacrypt 2020) proposed new comparison methods that approximated the sign function on homomorphically encrypted data using composite polynomials and proved that these methods had optimal asymptotic complexity. In this article, we propose a practically optimal method that approximates the sign function using compositions of minimax approximation polynomials. We prove that this approximation method is optimal with respect to depth consumption and the number of non-scalar multiplications. In addition, we propose a polynomial-time algorithm that determines the optimal composition of minimax approximation polynomials for the proposed homomorphic comparison operation using dynamic programming. The numerical analysis demonstrates that when minimizing runtime, the proposed comparison operation reduces the runtime by approximately 45 percent on average when compared to the previous algorithm. Likewise, when minimizing depth consumption, the proposed algorithm reduces the runtime by approximately 41 percent on average. In addition, when high precision in the comparison operation is required, the previous algorithm does not achieve 128-bit security, while the proposed algorithm does due to its small depth consumption.

Index Terms—Cheon–Kim–Kim–Song (CKKS) scheme, fully homomorphic encryption, homomorphic comparison operation, minimax approximation polynomial, Remez algorithm, sign function

1 Introduction

Homomorphic encryption (HE) is a type of encryption scheme that allows algebraic operations to be performed on encrypted data. Before Gentry's seminal work [1] in 2009, HE schemes were able to perform only a few specific operations on the encrypted data. In his work, Gentry developed the first fully homomorphic encryption (FHE) scheme, which allows all algebraic operations on the encrypted data without restriction. Accordingly, FHE has attracted significant attention in various applications, and its standardization process is currently being developed.

FHE schemes are classified as bitwise or word-wise. Word-wise FHE, such as Brakerski/Fan–Vercauteren [2] and Cheon–Kim–Kim–Song (CKKS) [3], allows for the addition and multiplication of an encrypted array over $\mathbb{C}$ or $\mathbb{Z}_p$ for a positive integer $p > 2$. All other operations in word-wise FHE are performed by composing these two basic operations. In contrast, the basic operations of a bitwise FHE scheme, such as fast fully homomorphic encryption over the torus [4], are logic gates such as NAND gates. Recently, word-wise FHE has been widely used in several applications such as deep learning [5], [6].

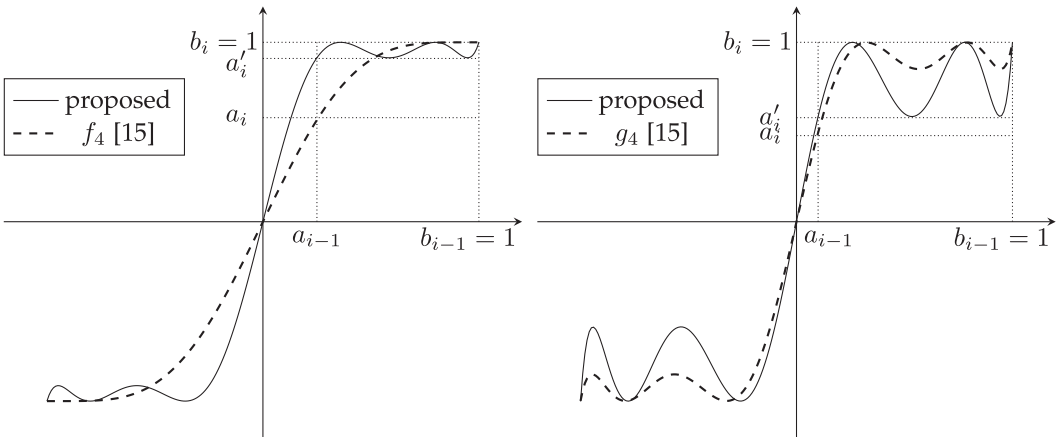
Although word-wise FHE can support virtually all arithmetic operations on encrypted data, several applications require non-arithmetic operations. One of the most important non-arithmetic operations is the comparison operation, which is denoted as $\mathrm{comp}(u, v)$ and outputs 1 if $u > v$, $1/2$ if $u = v$, and 0 if $u < v$. This operation is widely used in various real-world applications, including machine learning algorithms such as support-vector machines, cluster analysis, and gradient boosting [7], [8].

The max function $\max(u, v)$ is another essential non-arithmetic operation and is particularly important in deep learning because it is essentially the same as the max pooling operation that is widely used in deep learning. Several deep learning models, such as AlexNet [9], VGGNet [10], GoogleNet [11], Inception [12], and ResNet [13], use the max pooling operation. In addition, sorting algorithms that rely on the max function are used as subroutines of various other algorithms. Thus, the efficient implementation of the max function on encrypted data (homomorphic max function) is important, as it has a direct impact on the performance of the max pooling operation in privacy-preserving deep learning models and sorting algorithms on encrypted data (homomorphic sorting algorithms) [14].

Several studies have been conducted to efficiently implement the comparison operation on encrypted data

- *Eunsang Lee, Joon-Woo Lee, and Jong-Seon No are with the Department of Electrical and Computer Engineering, INMC, Seoul National University, Seoul 08826, South Korea. E-mail: {eslee3209, joonwoo3511}@ccl.snu.ac.kr, jsno@snu.ac.kr.*
- *Young-Sik Kim is with the Department of Information and Communication Engineering, Chosun University, Gwangju 61452, South Korea. E-mail: iamyskim@chosun.ac.kr.*

Manuscript received 9 Nov. 2020; revised 11 July 2021; accepted 12 Aug. 2021. Date of publication 18 Aug. 2021; date of current version 11 Nov. 2022. (Corresponding author: Joon-Woo Lee)
Digital Object Identifier no. 10.1109/TDSC.2021.3105111



(a) Comparison with f_4 (b) Comparison with g_4

Fig. 1. Comparison between the proposed minimax approximation polynomial and the previous polynomials f_4 and g_4 in [15].

(homomorphic comparison operation) and homomorphic max function. The comparison operation and max function can be easily implemented using the sign function; that is, $\mathrm{comp}(u, v) = \frac{1}{2}(\mathrm{sgn}(u - v) + 1)$, and $\max(u, v) = \frac{(u+v)+(u-v)\,\mathrm{sgn}(u-v)}{2}$, where $\mathrm{sgn}(x) = x/|x|$ for $x \neq 0$, and 0 otherwise. Thus, we focus on implementing $\mathrm{sgn}(x)$ in this study, as doing so leads directly to implementing $\mathrm{comp}(u, v)$ and $\max(u, v)$.

Because $\mathrm{sgn}(x)$ is not a polynomial, it is impossible to implement it exactly in FHE. Thus, it is necessary to evaluate a polynomial $p(x)$ that approximates $\mathrm{sgn}(x)$. Since $\mathrm{sgn}(x)$ is discontinuous at $x = 0$, the small neighborhood $(-\epsilon, \epsilon)$ around zero is not considered when measuring the difference between $p(x)$ and $\mathrm{sgn}(x)$. Specifically, it is required that

$$|p(x) - \mathrm{sgn}(x)| \leq 2^{1-\alpha} \text{ for } x \in [-1, -\epsilon] \cup [\epsilon, 1], \tag{1}$$

which implies that $\frac{1}{2}(p(u - v) + 1)$ is an approximate value of $\mathrm{comp}(u, v)$ within $2^{-\alpha}$ error for $u, v \in [0, 1]$ satisfying $|u - v| \geq \epsilon$. Here, ϵ and α determine the input and output precision of the comparison operation, respectively. When determining an approximate polynomial, it is best to reduce the polynomial degree as much as possible in order to minimize the runtime and depth consumption of the operation.

A method to approximate $\mathrm{sgn}(x)$ using a composite polynomial was recently proposed in [15], and it was proved that this method achieved optimal asymptotic complexity. However, the functions f_n and g_n used in [15] do not ensure approximation optimality, and thus finding better composite polynomials that approximate $\mathrm{sgn}(x)$ is an important area of research.

Let $p = p_k \circ p_{k-1} \circ \cdots \circ p_1$ be a composite polynomial that approximates $\mathrm{sgn}(x)$. Since $\mathrm{sgn}(x)$ is an odd function, it is natural to approximate $\mathrm{sgn}(x)$ using a composition of polynomials with odd-degree terms. Thus, we assume that $p_1, p_2, \ldots, p_k$ are polynomials with odd-degree terms. Because of symmetry it is sufficient to consider only the case where $x > 0$ to verify that the composite polynomial p satisfies the error condition in (1). Let $p_1([\epsilon, 1]) = [a_1, b_1], p_2 \circ p_1([\epsilon, 1]) = [a_2, b_2], \ldots, p_k \circ \cdots \circ p_1([\epsilon, 1]) = [a_k, b_k]$. Each component polynomial p_i can be seen as performing the task of mapping a given interval $[a_{i-1}, b_{i-1}]$ onto a smaller interval $[a_i, b_i]$. We should guarantee that $[a_k, b_k] \in [1 - 2^{1-\alpha}, 1 + 2^{1-\alpha}]$, thereby satisfying the error condition in (1). The difficulty in approximating $\mathrm{sgn}(x)$ is that the interval $[\epsilon, 1]$, whose size is nearly one, must ultimately be mapped onto a tiny interval $[1 - 2^{1-\alpha}, 1 + 2^{1-\alpha}]$, whose size is $2^{2-\alpha}$. Our key observation is that finding polynomials that efficiently reduce the given intervals is essential for an efficient comparison operation implementation. As such, the natural question arises:

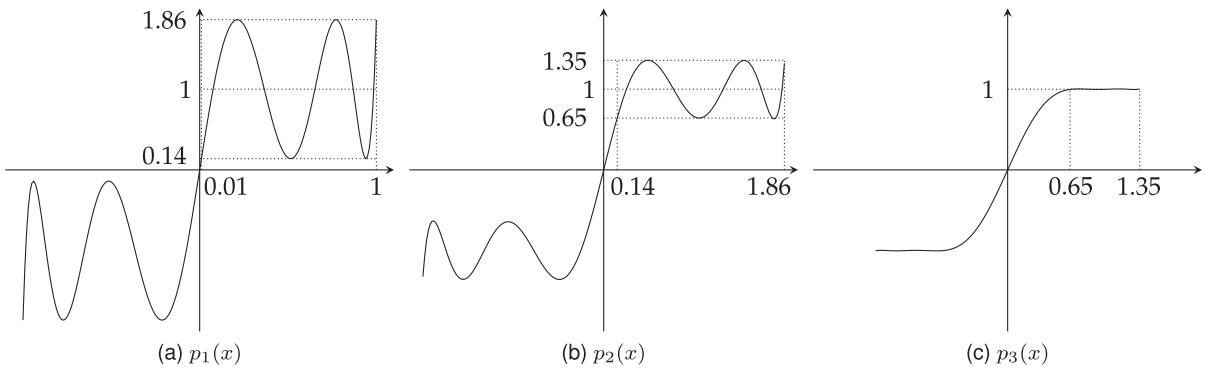
"Given an interval $[a_{i-1}, b_{i-1}]$ and an integer d_i, what is the polynomial p_i with odd-degree terms of at most degree d_i that maps this interval onto the smallest interval?"

1.1 Our Results

1.1.1 Minimax Composite Polynomial

First, for $[a_i, b_i] = p_i([a_{i-1}, b_{i-1}])$, we consider the case where b_i should be one. Then, the answer to the above question is the minimax approximation polynomial of $c\,\mathrm{sgn}(x)$ (for some constant $c < 1$ such that $b_i = 1$) with degree at most d_i on $[-b_{i-1}, -a_{i-1}] \cup [a_{i-1}, b_{i-1}]$. Thus, we come up with the idea of using a composite of minimax approximation polynomials, called *minimax composite polynomial*, where each component polynomial p_i is the minimax approximation polynomial defined on $[-b_{i-1}, -a_{i-1}] \cup [a_{i-1}, b_{i-1}] = p_{i-1} \circ \cdots \circ p_1([-1, -\epsilon] \cup [\epsilon, 1])$ with degree at most d_i.

The two functions f_n and g_n used in [15] are less efficient than those used in the proposed method. First, since f_n is not the minimax approximation polynomial, it reduces the size of the interval less efficiently. In addition, although g_n is the minimax approximation polynomial of $c\,\mathrm{sgn}(x)$ for some $c < 1$, it is defined on a different domain than $[-b_{i-1}, -a_{i-1}] \cup [a_{i-1}, b_{i-1}]$, which also results in some inefficiency. Fig. 1 shows that the size of the interval is reduced more efficiently when the proposed minimax approximation polynomial is used on $[-b_{i-1}, -a_{i-1}] \cup [a_{i-1}, b_{i-1}]$ than when f_n or g_n is used (that is, $[a'_i, 1] \subset [a_i, 1]$). In this study, the range $[a_i, b_i] = p_i([a_{i-1}, b_{i-1}])$ is actually in the form of $[1 - \tau_i, 1 + \tau_i]$ for $i \geq 2$, and not in the form where $b_i = 1$, as shown in Fig. 1. However, it is possible to arbitrarily change the range of a component polynomial by scaling, while maintaining the performance of the homomorphic comparison operation. Specifically, by using $cp_i(x)$ instead of $p_i(x)$



(a) $p_1(x)$ (b) $p_2(x)$ (c) $p_3(x)$

Fig. 2. The component polynomials of an example of the proposed minimax composite polynomial $p = p_3 \circ p_2 \circ p_1$ on $D = [-1, -0.01] \cup [0.01, 1]$ for $\{d_1, d_2, d_3\} = \{9, 9, 9\}$.

and $p_{i+1}(x/c)$ instead of $p_{i+1}(x)$ for some $c > 0$, we can scale the range $[a_i, b_i]$ by c without changing the degrees of $p_i(x)$ and $p_{i+1}(x)$. Thus, although minimax approximation polynomials whose ranges are centered at one are used in this study, the minimax approximation polynomial in Fig. 1 is scaled so that $b_i = 1$ for comparison with f_n and g_n in [15].

Now, the formal definition of the proposed *minimax composite polynomial* is described. We denote the minimax approximation polynomial of $\mathrm{sgn}(x)$ on domain D with degree at most n by $\mathsf{MP}(D; n)$. Then, for $D = [-b, -a] \cup [a, b]$ and set of integers $\{d_i\}_{1 \le i \le k}$, a composite polynomial $p_k \circ p_{k-1} \circ \cdots \circ p_1$ is called a *minimax composite polynomial on D for $\{d_i\}_{1 \le i \le k}$* if the followings are satisfied:

- $p_1 = \mathsf{MP}(D; d_1)$
- $p_i = \mathsf{MP}(p_{i-1} \circ p_{i-2} \circ \cdots \circ p_1(D); d_i)$ for $2 \le i \le k$.

Fig. 2 shows the component polynomials of an example minimax composite polynomial $p = p_3 \circ p_2 \circ p_1$ on $D = [-1, -0.01] \cup [0.01, 1]$ for $\{d_1, d_2, d_3\} = \{9, 9, 9\}$.

1.1.2 Finding Optimal Degrees Using Dynamic Programming

With our method defined, our goal now is to find the optimal set of degrees. Specifically, for a given depth consumption D, we find the set of degrees such that the minimax composite polynomial requires the minimum number of non-scalar multiplications while consuming depth D.

First, we may consider performing brute-force searching through all combinations of the composition numbers and the degrees of the component polynomials to find the optimal set of degrees. However, for the upper bound on the number of polynomials $\bar{k}$ and that of the degrees $\bar{d}$, this brute-force search requires $O(\bar{d}^{\bar{k}})$ time, which is impractical when high-precision polynomial approximations are required.

Instead, we propose a fast method to find the optimal set of degrees using dynamic programming. In Section 3.3, we prove that our method determines the set of degrees that is optimal with respect to depth consumption and the number of non-scalar multiplications. With upper bounds on the number of non-scalar multiplications, depth consumption, and component polynomial degrees given by $\bar{m}$, $\bar{n}$, and $\bar{d}$, respectively, the time complexity of the proposed algorithm is $O(\bar{m}\bar{n}\bar{d})$. This linear-time algorithm is much faster than the brute-force approach, and with it, we can obtain the optimal set of degrees for a range of approximation precisions (see Table 2).

1.1.3 Improved Performance of Homomorphic Comparison Operation

When the Homomorphic Encryption for Arithmetic of Approximate Numbers (HEAAN) library [3] is used, the proposed homomorphic comparison operation reduces the runtime by approximately 45 percent on average if the runtime is minimized and 41 percent if the depth consumption is minimized, as compared to the previous algorithm `CompG` (referred to as `NewCompG` in [15]). In addition, the proposed homomorphic comparison operation algorithm reduces the ciphertext modulus bit consumption by approximately 41 or 45 percent, respectively, if the runtime or depth consumption is minimized. It is also noteworthy that when the approximation precision parameter α is 20, the proposed homomorphic comparison operation achieves 128-bit security owing to its small depth consumption whereas the operation in [15] does not.

1.1.4 Improved Performance of Homomorphic Max Function and Homomorphic Sorting

From $\max(u, v) = \frac{(u+v)+(u-v)\,\mathrm{sgn}(u-v)}{2}$, the homomorphic max function can be easily implemented using the proposed method of approximating $\mathrm{sgn}(x)$. When run to minimize runtime, this implementation reduces the runtime by approximately 41 percent on average and the ciphertext modulus bit consumption by approximately 33 percent on average, as compared to the previous algorithm `MaxG` (referred to as `NewMaxG` in [15]). When run to minimize depth consumption, this implementation reduces the runtime and modulus bit consumption by approximately 35 and 45 percent, respectively.

Furthermore, we implement the homomorphic sorting of three and four numbers using the proposed homomorphic max function. The new algorithm is found to be approximately two times as fast as the previous homomorphic sorting algorithm. In addition, the previous algorithm does not achieve 128-bit security whereas the proposed algorithm does due to its small depth consumption.

1.2 Related Works

Previous research on the determination of polynomials that approximate $\mathrm{sgn}(x)$ or $\mathrm{comp}(u, v)$ in FHE has been

conducted. An analytic method to approximate the sign function using Fourier series was proposed in [16]. In [17], the sign function was approximated using the relation $\tanh(jx) = \frac{e^{jx}-e^{-jx}}{e^{jx}+e^{-jx}} \simeq \mathrm{sgn}(x)$ for large $j > 0$. Recently, an iterative algorithm was proposed in [18] that performs a homomorphic comparison using the equation $\lim_{j\to\infty} \frac{u^j}{u^j+v^j} = \mathrm{comp}(u, v)$, where the inverse operation can be performed using the Goldschmidt division algorithm [19]. However, the use of the inverse operation results in inefficient computation. More recently, in [15], the homomorphic comparison operation was approximated using composite polynomials with a smaller depth consumption and fewer non-scalar multiplications than in previous methods.

1.3 Outline

The remainder of this paper is organized as follows. Section 2 presents the preliminaries regarding the concepts of FHE, the comparison operation in FHE, approximation theory, and the algorithms for minimax approximation. In Section 3, a new method for approximating the sign function using minimax composite polynomials is proposed, and it is proved that the proposed approximation method is optimal. In addition, a polynomial-time algorithm to obtain the optimal minimax composite polynomial is proposed using dynamic programming, and its performance is compared with the previous best algorithm. In Section 4, the proposed method of approximating the sign function is applied to the homomorphic max function and homomorphic sorting. In Section 5, numerical results for the proposed homomorphic comparison operation, homomorphic max function, and homomorphic sorting algorithm are provided. Finally, concluding remarks are given in Section 6.

2 Preliminaries

2.1 Fully Homomorphic Encryption

FHE schemes are classified as bitwise or word-wise. The basic operations of the former are logical, whereas the basic operations of the latter are algebraic, namely addition and multiplication. In this study, we focus only on word-wise FHE, and the term FHE refers only to word-wise FHE. The definition of FHE is as follows:

Definition 1. *An FHE scheme is a set of five polynomial-time algorithms that satisfy the following:*

- $\mathsf{KeyGen}(\lambda) \to (\mathsf{pk}, \mathsf{sk}, \mathsf{evk})$*;* KeyGen *takes a security parameter* λ *as input, and outputs a public key* pk*, a secret key* sk*, and an evaluation key* evk*.*
- $\mathsf{Enc}(\mu, \mathsf{pk}) \to \mathsf{ct}$*;* Enc *takes a message* μ *and a public key* pk *as input, and outputs a ciphertext* ct *of* μ*.*
- $\mathsf{Dec}(\mathsf{ct}, \mathsf{sk}) \to \mu$ *or* $\perp$ *;* Dec *takes a ciphertext* ct *and a secret key* sk *as input, and outputs the plaintext* μ *of* ct*. If the decryption fails,* Dec *outputs a special symbol* $\perp$ *.*
- $\mathsf{Add}(\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{evk})$*;* Add *takes ciphertexts* ct_1 *and* ct_2 *of messages* μ_1 *and* μ_2*, respectively, and an evaluation key* evk *as input, and outputs a ciphertext* $\mathsf{ct}_{\mathsf{add}}$ *of message* $\mu_1 + \mu_2$*.*
- $\mathsf{Mult}(\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{evk})$*;* Mult *takes ciphertexts* ct_1 *and* ct_2 *of messages* μ_1 *and* μ_2*, respectively, and an evaluation key* evk *as input, and outputs a ciphertext* $\mathsf{ct}_{\mathsf{mult}}$ *of message* $\mu_1 \cdot \mu_2$*.*

An approximate FHE scheme deals with approximate values instead of exact values. In this case, Dec outputs an approximate value of μ. In addition, Add and Mult output ciphertexts that are approximations of the true ciphertexts of $\mu_1 + \mu_2$ and $\mu_1 \cdot \mu_2$, respectively.

2.2 CKKS Scheme

In our numerical analysis, we use the CKKS scheme, which is a representative word-wise approximate FHE scheme that supports the evaluation of encrypted real or complex data. The ciphertext modulus of bit-length ℓ is denoted by $q_\ell = 2^\ell$ for $1 \le \ell \le L$, where L is the bit length of the initial ciphertext modulus. Let $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$, where N is a power-of-two integer. Let χ_{key}, χ_{err}, and χ_{enc} be the key distribution, the error distribution, and the distribution for encryption over $\mathcal{R}$, respectively, chosen according to security requirements. We set the key distribution $\chi_{\mathsf{key}} = \mathcal{HWT}_N(256)$, which samples an element uniformly at random from $\mathcal{R}$ with ternary coefficients that have 256 nonzero values. Let $U(\mathcal{R}_q)$ denote the uniform distribution over $\mathcal{R}_q$. There is a field isomorphism $\bar{\tau} : \mathbb{R}[X]/(X^N + 1) \to \mathbb{C}^{N/2}$. This isomorphism is used for decoding, and its inverse isomorphism $\bar{\tau}^{-1}$ is used for encoding. The CKKS scheme is described as follows:

- $\mathsf{KeyGen}(\lambda)$: Sample $\bar{s} \leftarrow \chi_{\mathsf{key}}$, $\bar{e}, \bar{e}' \leftarrow \chi_{\mathsf{err}}$, $\bar{a} \leftarrow U(\mathcal{R}_{q_L})$, and $\bar{a}' \leftarrow U(\mathcal{R}_{q_L^2})$, and set $\mathsf{sk} \leftarrow (1, \bar{s})$, $\mathsf{pk} \leftarrow (-\bar{a} \cdot \bar{s} + \bar{e}, \bar{a}) \in \mathcal{R}_{q_L}^2$, and $\mathsf{evk} \leftarrow (-\bar{a}' \cdot \bar{s} + \bar{e}' + q_L \cdot \bar{s}^2, \bar{a}') \in \mathcal{R}_{q_L^2}^2$.
- $\mathsf{Enc}(\mathbf{z}; \mathsf{pk}, \Delta)$: For a plaintext vector $\mathbf{z} = (z_0, \ldots, z_{N/2-1}) \in \mathbb{C}^{N/2}$, a public key pk, and a scaling factor Δ, encode $\mathbf{z}$ by $\bar{m} \leftarrow \lfloor \Delta \cdot \bar{\tau}^{-1}(\mathbf{z}) \rceil \in \mathcal{R}$, where $\lfloor \cdot \rceil$ denotes the rounding operation. Then, sample $\bar{v} \leftarrow \chi_{\mathsf{enc}}$ and $\bar{e}_0, \bar{e}_1 \leftarrow \chi_{\mathsf{err}}$, and output the ciphertext $\mathsf{ct} = \lceil \bar{v} \cdot \mathsf{pk} + (\bar{m} + \bar{e}_0, \bar{e}_1) \rceil_{q_L}$.
- $\mathsf{Dec}(\mathsf{ct}; \mathsf{sk}, \Delta)$: For a ciphertext $\mathsf{ct} = (\bar{c}_0, \bar{c}_1) \in \mathcal{R}_{q_\ell}^2$, compute the plaintext polynomial $\bar{m}' \leftarrow \lceil \bar{c}_0 + \bar{c}_1 \cdot \bar{s} \rceil_{q_\ell}$, and output the plaintext vector $\mathbf{z}' \leftarrow \Delta^{-1} \cdot \bar{\tau}(\bar{m}') \in \mathbb{C}^{N/2}$.
- $\mathsf{Add}(\mathsf{ct}, \mathsf{ct}')$: Output $\mathsf{ct}_{\mathsf{add}} \leftarrow \lceil \mathsf{ct} + \mathsf{ct}' \rceil_{q_\ell}$.
- $\mathsf{cMult}(\mathsf{ct}, \mathbf{a}; \Delta)$: For a ciphertext $\mathsf{ct} = (\bar{c}_0, \bar{c}_1)$ and a constant vector $\mathbf{a} = (a_0, \ldots, a_{N/2-1}) \in \mathbb{C}^{N/2}$, output $\mathsf{ct}' \leftarrow \lceil \Delta^{-1} \lfloor \Delta \cdot \bar{\tau}^{-1}(\mathbf{a}) \rceil \cdot \mathsf{ct} \rceil_{q_\ell/\Delta}$
- $\mathsf{Mult}(\mathsf{ct}, \mathsf{ct}'; \mathsf{evk}, \Delta)$: For $\mathsf{ct} = (\bar{c}_0, \bar{c}_1)$ and $\mathsf{ct}' = (\bar{c}_0', \bar{c}_1')$, compute $(\bar{d}_0, \bar{d}_1, \bar{d}_2) = (\bar{c}_0\bar{c}_0', \bar{c}_0\bar{c}_1' + \bar{c}_0'\bar{c}_1, \bar{c}_1\bar{c}_1')$ and $\mathsf{ct}'_{\mathsf{mult}} \leftarrow \lceil (\bar{d}_0, \bar{d}_1) + \lfloor q_L^{-1} \cdot \bar{d}_2 \cdot \mathsf{evk} \rceil \rceil_{q_\ell}$. Then, output $\mathsf{ct}_{\mathsf{mult}} \leftarrow \lceil \lfloor \Delta^{-1} \cdot \mathsf{ct}'_{\mathsf{mult}} \rceil \rceil_{q_\ell/\Delta}$.

The CKKS scheme has two types of multiplication: scalar multiplication cMult and non-scalar multiplication Mult. Because non-scalar multiplications require significantly more runtime, we focus on reducing the number of non-scalar rather than scalar multiplications in this study.

It should be noted that an attack method on the CKKS scheme has recently been proposed [26]. It was shown in [26] that the CKKS scheme could be broken if the secret key owner shared the decryption results with others. Then, an attack prevention method was proposed in [27]. This method adds a Gaussian error at the end of the decryption.

Procedures other than the decryption do not need to be modified, and thus the proposed homomorphic comparison operation in this study can be used without change as it utilizes only the addition and multiplication.

2.3 Comparison Operation in Fully Homomorphic Encryption

Because FHE schemes do not support non-arithmetic operations such as comparison, an approximation of the comparison operation should be performed using additions and multiplications. For a, b such that $0 < a < b$, we denote $[-b, -a] \cup [a, b]$ by $\tilde{R}_{a,b}$. Particularly, if $a = 1 - \tau$ and $b = 1 + \tau$ for some $\tau \in (0, 1)$, we denote $\tilde{R}_{1-\tau,1+\tau}$ by R_τ. The comparison operation and the sign function are defined as

$$\mathrm{comp}(u, v) = \begin{cases} 1 & \text{if } u > v \\ 1/2 & \text{if } u = v \\ 0 & \text{if } u < v \end{cases}, \quad \mathrm{sgn}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}.$$

From $\mathrm{comp}(u, v) = \frac{\mathrm{sgn}(u-v)+1}{2}$, the polynomial approximation of $\mathrm{comp}(u, v)$ is easily obtained from that of $\mathrm{sgn}(x)$. As such, we only focus on the polynomial approximation of $\mathrm{sgn}(x)$. Definition 2 below quantifies how close a polynomial that approximates $\mathrm{sgn}(x)$ is to $\mathrm{sgn}(x)$. Because $\mathrm{sgn}(x)$ is discontinuous at $x = 0$, it cannot be approximated near zero. Definition 2 implies that the approximation error is ensured to be below $2^{-\alpha}$ only for $\epsilon \le |x| \le 1$.

Definition 2 ([15]). *For $\alpha > 0$ and $0 < \epsilon < 1$, a (composite) polynomial p is said to be (α, ϵ)-close to $\mathrm{sgn}(x)$ over $[-1, 1]$ if p satisfies the following:*

$$||p(x) - \mathrm{sgn}(x)||_{\infty, \tilde{R}_{\epsilon,1}} \le 2^{-\alpha},$$

where $||\cdot||_{\infty,D}$ denotes the infinity norm over the domain D.

For precision parameters α and ϵ, a polynomial $\tilde{p}(u, v)$ that approximates the comparison operation should satisfy the following error condition:

$$|\tilde{p}(u, v) - \mathrm{comp}(u, v)| \le 2^{-\alpha} \quad \text{for any } u, v \in [0, 1] \text{ satisfying } |u - v| \ge \epsilon. \tag{2}$$

Since $\mathrm{comp}(u, v) = \frac{\mathrm{sgn}(u-v)+1}{2}$, if a polynomial $p(x)$ approximating $\mathrm{sgn}(x)$ is $(\alpha - 1, \epsilon)$-close, then $\tilde{p}(u, v) = \frac{p(u-v)+1}{2}$ satisfies the comparison operation error condition in (2). Thus, our method finds $(\alpha - 1, \epsilon)$-close composite polynomials that approximate $\mathrm{sgn}(x)$.

Because bootstrapping is time-consuming, minimizing the depth consumption for the homomorphic comparison operation is important. As such, we approximate $\mathrm{sgn}(x)$ by polynomials that minimize depth consumption as well as the number of non-scalar multiplications.

2.4 Approximation Theory

Herein, certain concepts from approximation theory are introduced.

Definition 3. *Let D be a closed subset of $[a, b]$ and f be a continuous function on D. For a degree bound n, a polynomial p is said to be the minimax approximation polynomial of f on D with degree at most n if p minimizes $max_D||p(x) - f(x)||_\infty$ among all the polynomials with degree at most n.*

It is known that for any continuous function f on D, the minimax approximation polynomial with degree at most n uniquely exists [20]. In addition, the unique minimax approximation polynomial can be obtained using the improved multi-interval Remez algorithm (see Section 2.5). We set $f(x) = \mathrm{sgn}(x)$, and we are only concerned with cases in which D is the union of two symmetric closed intervals, $\tilde{R}_{a,b}$ in this study. $\mathsf{MP}(D; n)$ denotes the minimax approximation polynomial of $\mathrm{sgn}(x)$ on D with degree at most n. In addition, for the minimax approximation polynomial $p(x) = \mathsf{MP}(D; n)$, the minimax approximation error $\max_D||p(x) - \mathrm{sgn}(x)||_\infty$ is denoted by $\mathsf{ME}(D; n)$.

We refer to the definition of Haar's condition [20] for a set of functions, which provides a generalized definition of polynomial bases. It is well known that the power basis and Chebyshev polynomial basis satisfy Haar's condition. Thus, if an argument deals with a set of functions that satisfy the Haar's condition, it naturally includes the cases of power basis and Chebyshev polynomial basis.

Definition 4 (Haar's condition and generalized polynomials [20]). *A set of functions $\{\phi_1, \phi_2, \ldots, \phi_n\}$ satisfies the Haar's condition if each ϕ_i is continuous and the determinant*

$$D[x_1, \ldots, x_n] = \begin{vmatrix} \phi_1(x_1) & \cdots & \phi_n(x_1) \\ \vdots & \ddots & \vdots \\ \phi_1(x_n) & \cdots & \phi_n(x_n) \end{vmatrix},$$

is not zero for any n distinct points $x_1, \ldots, x_n$. A linear combination of $\{\phi_1, \phi_2, \ldots, \phi_n\}$ is referred to as a generalized polynomial.

The following theorem and lemmas are required for some of the proofs presented in Section 3.

Theorem 1 (Chebyshev alternation theorem [20]). *Let D be a closed subset of $[a, b]$, and $\{\phi_1, \phi_2, \ldots, \phi_n\}$ be a set of continuous functions on $[a, b]$ that satisfy the Haar's condition. A polynomial $p = \sum_i c_i\phi_i$ is the minimax approximation polynomial of any given continuous function f on D if and only if there are $n + 1$ elements $x_0 < \cdots < x_n$ in D such that*

$$r(x_i) = -r(x_{i-1}) = \pm \sup_{x \in D} |r(x)|, 1 \le i \le n, \tag{3}$$

for the error function $r = f - p$ restricted on D.

Remark 1. The condition in (3) is called the equioscillation condition. Let D be $[-b, -a] \cup [a, b]$. As $r(x_i) = \pm\sup_{x \in D}|r(x)|$ for $0 \le i \le n$, it follows that $r(x)$ should have extreme points at x_i for $0 \le i \le n$. Thus, $p'(x_i) = 0$ and $x_i \in (-b, -a) \cup (a, b)$, or $x_i \in \{-b, -a, a, b\}$.

Lemma 1 (Generalized de La Vallee Poussin theorem [21]). *Let $\{\phi_1, \phi_2, \ldots, \phi_n\}$ be a set of continuous functions on $[a, b]$ that satisfy Haar's condition. Let D be a closed subset of $[a, b]$ and $f(x)$ be a continuous function on D. Let $x_i, 0 \le i \le n$ be $n + 1$ consecutive points in D and $p(x)$ be a generalized polynomial such that $p - f$ has alternately positive and negative values at $x_i, 0 \le i \le n$. Further, let $p^*(x)$ be a minimax approximation polynomial of f on D, and let $e(f)$ be the corresponding approximation error of $p^*(x)$. Then,*

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

Lemma 2 ([22]). *If $f(x)$ is an odd function, the minimax approximation polynomial of $f(x)$ with degree at most n is also an odd function.*

2.5 Algorithms for Minimax Approximation

The Remez algorithm [23] in Algorithm 1 obtains the minimax approximation polynomial of a continuous function on an interval. First, it initializes reference points $\{x_1, \ldots, x_{n+1}\}$, which will converge to the extreme points of the minimax approximation polynomial. Then, it determines a polynomial $p(x)$ with basis $\{\phi_1, \ldots, \phi_n\}$ that satisfies $p(x_i) - f(x_i) = (-1)^i E, 1 \leq i \leq n+1$ for some $E > 0$; that is, it determines coefficients $c_1, \ldots, c_n$ that satisfy $c_1\phi_1(x_i) + \cdots + c_n\phi_n(x_i) - f(x_i) = (-1)^i E, 1 \leq i \leq n+1$. These equations form a system of linear equations consisting of $n+1$ equations and $n+1$ unknowns (coefficients $c_1, \ldots, c_n$ and E). This system is guaranteed not to be singular by Haar's condition, and thus the polynomial $p(x)$ can be uniquely determined.

Next, the algorithm finds n zeros $z_1, \ldots, z_n$ of $p(x) - f(x)$, where $x_i < z_i < x_{i+1}$, as well as $n+1$ extreme points $y_1, \ldots, y_{n+1}$ of $p(x) - f(x)$ such that $y_i \in [z_{i-1}, z_i]$, where $z_0 = a$ and $z_{n+1} = b$. If the extreme points satisfy the equioscillation condition in (3), the Remez algorithm outputs $p(x)$ as the minimax approximation polynomial. Otherwise, it replaces the reference points with these extreme points and proceeds with the above steps again. It was proved in [23] that this polynomial $p(x)$ always converged to the minimax approximation polynomial.

Algorithm 1. Remez Algorithm [21]

Input: A basis $\{\phi_1, \ldots, \phi_n\}$, a domain $[a, b]$, an approximation parameter γ, and a continuous function f on $[a, b]$
Output: The minimax approximation polynomial p of f
1: Choose $x_1, \ldots, x_{n+1} \in [a, b]$, where $x_1 < \cdots < x_{n+1}$;
2: Find the polynomial $p(x)$ in terms of $\{\phi_1, \ldots, \phi_n\}$ such that $p(x_i) - f(x_i) = (-1)^i E, 1 \leq i \leq n+1$ for some E;
3: Divide the domain $[a, b]$ into $n+1$ sections $[z_{i-1}, z_i], i = 1, \ldots, n+1$.
$z_1, \ldots, z_n$ are the zeros of $p(x) - f(x)$, where $x_i < z_i < x_{i+1}$, and $z_0 = a, z_{n+1} = b$;
4: Find the maximum or minimum point of $p - f$ for each section when $p(x_i) - f(x_i)$ has a positive or negative value, respectively. These points $y_1, \ldots, y_{n+1}$ are called extreme points;
5: $\epsilon_{\max} \leftarrow \max_{1 \leq i \leq n+1} |p(y_i) - f(y_i)|$;
6: $\epsilon_{\min} \leftarrow \min_{1 \leq i \leq n+1} |p(y_i) - f(y_i)|$;
7: **if** $(\epsilon_{\max} - \epsilon_{\min})/\epsilon_{\min} < \gamma$ **then**
8: Return $p(x)$;
9: **else**
10: Replace x_i with y_i for all i. Go to line 2;
11: **end**

Recently, Lee *et al.* [21] proposed the improved multi-interval Remez algorithm in Algorithm 2 that determined the minimax approximation polynomial on multiple intervals and proved that this algorithm always obtained the minimax approximation polynomial of any piecewise continuous function. In this study, we obtain the minimax approximation polynomial of the sign function using this algorithm.

Algorithm 2. Improved Multi-Interval Remez Algorithm [21]

Input: A basis $\{\phi_1, \ldots, \phi_n\}$, an approximation parameter γ, an input domain $D = \bigcup_{i=1}^{l} [a_i, b_i] \subset \mathbb{R}$, and a continuous function f on D
Output: The minimax approximation polynomial p of f
1: Choose $x_1, \ldots, x_{n+1} \in D$, where $x_1 < \cdots < x_{n+1}$;
2: Find the polynomial $p(x)$ in terms of $\{\phi_1, \ldots, \phi_n\}$ such that $p(x_i) - f(x_i) = (-1)^i E, 1 \leq i \leq n+1$ for some E;
3: Collect all the extreme and boundary points of $p - f$ on D such that $\mu(x)(p(x) - f(x)) \geq |E|$ and put them in a set B;
4: Find $n+1$ extreme points $y_1 < y_2 < \cdots < y_{n+1}$ in B that satisfy the alternating condition and maximum absolute sum condition;
5: $\epsilon_{\max} \leftarrow \max_{1 \leq i \leq n+1} |p(y_i) - f(y_i)|$;
6: $\epsilon_{\min} \leftarrow \min_{1 \leq i \leq n+1} |p(y_i) - f(y_i)|$;
7: **if** $(\epsilon_{\max} - \epsilon_{\min})/\epsilon_{\min} < \gamma$ **then**
8: Return $p(x)$;
9: **else**
10: Replace x_i with y_i for all i. Go to line 2
11: **end**

The improved multi-interval Remez algorithm is similar to the Remez algorithm. However, unlike in the Remez algorithm, there may be more than $n+1$ extreme points in the improved algorithm. Thus, $n+1$ extreme points $y_1, \ldots, y_{n+1}$ should be chosen among all candidate extreme points. The concavity function $\mu(x)$ is required to do so. It is defined as

$$\mu(x) = \begin{cases} 1, & p(x) - f(x) \text{ is concave at } x \text{ on } D \\ -1, & p(x) - f(x) \text{ is convex at } x \text{ on } D \\ 0, & \text{otherwise.} \end{cases}$$

The $n+1$ extreme points in Algorithm 2 are selected based on the following three criteria:

(i) *Local extreme value condition.* $\min_i \mu(y_i)(p(y_i) - f(y_i)) \geq E$, where E is the value obtained when line 2 in Algorithm 2 is performed.
(ii) *Alternating condition.* $\mu(y_i) \cdot \mu(y_{i+1}) = -1$ for $i = 1, \ldots, n$.
(iii) *Maximum absolute sum condition.* $\sum_{i=1}^{n+1} |p(y_i) - f(y_i)|$ is maximized for all candidate sets of extreme points satisfying the local extreme value and the alternating conditions.

The improved multi-interval Remez algorithm operates with n basis functions $\{\phi_1, \ldots, \phi_n\}$. The minimax approximation polynomial $p(x)$ is represented in terms of the basis functions as $p(x) = \sum_{i=1}^{n} c_i\phi_i(x)$, where the coefficients c_i are determined by the improved multi-interval Remez algorithm. Although the simplest basis is the power basis $\{1, x, x^2, \ldots, x^{n-1}\}$, when the sign function is approximated using this basis, the magnitudes of the coefficients c_i are often unstable (i.e., excessively small or large values),

resulting in greater numerical errors. Thus, the Chebyshev polynomials are used as basis functions in this study. The Chebyshev polynomials T_i on $[-1, 1]$ are defined by the following recursion:

$$T_0(x) = 1$$
$$T_1(x) = x$$
$$T_i(x) = 2xT_{i-1}(x) - T_{i-2}(x) \text{ for } i \geq 2.$$

If the sign function should be approximated on a domain larger than $[-1, 1]$, then scaled Chebyshev polynomials $\tilde{T}_i(x) = T_i(x/w)$ for some $w > 1$ should be used instead of T_i for all i.

3 Approximation of Sign Function Using Optimal Composition of Minimax Approximation Polynomials

3.1 New Approximation Method for Sign Function Using Composition of Minimax Approximation Polynomials

In [15], $\mathrm{sgn}(x)$ was approximated using self composition of the polynomial f_n on $[-1, 1]$, where $f_n(x) = \sum_{i=0}^{n} \frac{1}{4^i}\binom{2i}{i} x(1 - x^2)^i$. As n and the number of compositions s_n increase, the composite polynomial $f_n^{(s_n)}$ better approximates $\mathrm{sgn}(x)$. In addition, by using an acceleration polynomial g_n alongside f_n, the efficiency of the composite polynomial was further improved with a smaller number of required compositions. Nevertheless, because the polynomials f_n and g_n do not ensure approximation optimality, there is room left open for an improved approximation method.

In this study, we construct composite polynomials using component polynomials p_i, which are different from the polynomials f_n and g_n used in [15]. Additionally, our method does not use the repeated composition of each p_i. As $\mathrm{sgn}(x)$ is an odd function, it is natural to approximate $\mathrm{sgn}(x)$ using a composition of polynomials with odd-degree terms. Let $p_k \circ p_{k-1} \circ \cdots \circ p_1$ be a composition of polynomials with odd-degree terms approximating $\mathrm{sgn}(x)$ on $[-1, -\epsilon] \cup [\epsilon, 1]$. To prove that the composite polynomial $p_k \circ p_{k-1} \circ \cdots \circ p_1$ is $(\alpha - 1, \epsilon)$-close, it suffices, due to symmetry, to consider only the case where $x > 0$. Let $[a_0, b_0] = [\epsilon, 1]$, $p_1([a_0, b_0]) = [a_1, b_1]$, $p_2([a_1, b_1]) = [a_2, b_2], \ldots, p_k([a_{k-1}, b_{k-1}]) = [a_k, b_k]$. We note that $p_k \circ p_{k-1} \circ \cdots \circ p_1$ is $(\alpha - 1, \epsilon)$-close if and only if $[a_k, b_k] \subseteq [1 - 2^{1-\alpha}, 1 + 2^{1-\alpha}]$. As $[a_k, b_k]$ must be a very small interval, each p_i on the domain $[a_{i-1}, b_{i-1}]$ should have the smallest range possible. The key observation is that if minimax approximation polynomials are used in the composition, the range $[a_i, b_i]$ reduces rapidly as i increases. Thus, we use a composition of minimax approximation polynomials that can be obtained by the improved multi-interval Remez algorithm.

In [15], the coefficients of approximate polynomials are rounded to $\frac{j}{2^i}$ for some integers i and j, and multiplication by coefficients is implemented by adding a ciphertext j times and then removing i least significant bits. Thus, the depth consumption due to scalar multiplications is not significant in [15]. On the other hand, we implement multiplication by coefficients using scalar multiplication, and thus we should consider the depth consumption due to both scalar and non-scalar multiplications. The approximate polynomials are evaluated with minimum depth consumption using the odd baby-step giant-step algorithm [24] and the optimal level consumption technique of [25], which consider the depth consumption due to both scalar and non-scalar multiplications. For an odd integer d, the two functions $\mathsf{dep}(d)$ and $\mathsf{mult}(d)$ denote the depth consumption and number of non-scalar multiplications, respectively, required to evaluate a polynomial of degree d with odd-degree terms using the above algorithm and technique. Table 1 shows the values of $\mathsf{dep}(d)$ and $\mathsf{mult}(d)$ for odd degrees d up to 31.

TABLE 1
Required Depth Consumption and Number of Non-Scalar Multiplications for Evaluating Polynomials of Degree d With Odd-Degree Terms Using the Odd Baby-Step Giant-Step Algorithm [24] and the Optimal Level Consumption Technique [25]

polynomial degree d	depth consumption $\mathsf{dep}(d)$	multiplications $\mathsf{mult}(d)$
3	2	2
5	3	3
7	3	5
9	4	5
11	4	6
13	4	7
15	4	8
17	5	8
19	5	8
21	5	9
23	5	9
25	5	10
27	5	10
29	5	11
31	5	12

In this study, our goal is to determine an $(\alpha - 1, \epsilon)$-close polynomial $p(x)$. However, for a concise description of Sections 3.2 and 3.3, it is also necessary to consider the following definition.

Definition 5 ([15]). *For $\alpha > 0$ and $0 < \delta < 1$, a (composite) polynomial $p(x)$ is said to be (α, δ)-two-sided-close to $\mathrm{sgn}(x)$ if it satisfies the following:*

$$||p(x) - \mathrm{sgn}(x)||_{\infty, R_\delta} \leq 2^{-\alpha},$$

where $||\cdot||_{\infty, D}$ denotes the infinity norm over the domain D.

Our objective is to determine an $(\alpha - 1, \epsilon)$-close composite polynomial $p_k \circ \cdots \circ p_1$ that minimizes $\sum_{i=1}^{k} \mathsf{mult}(\deg(p_i))$ as well as $\sum_{i=1}^{k} \mathsf{dep}(\deg(p_i))$. The following lemma implies that determining an (α, δ)-two-sided-close composite polynomial and determining an (α, ϵ)-close composite polynomial are equivalent for $\delta = \frac{1-\epsilon}{1+\epsilon}$.

Lemma 3. *Let $\{\tilde{p}_i\}_{1 \leq i \leq k}$ be a set of polynomials with odd-degree terms such that $\tilde{p}_1(x) = p_1(\frac{1+\epsilon}{2}x)$ and $\tilde{p}_i(x) = p_i(x)$, $2 \leq i \leq$*

k. Then, for $\delta = \frac{1-\epsilon}{1+\epsilon}$, $p_k \circ p_{k-1} \circ \cdots \circ p_1$ *is* $(\alpha-1,\epsilon)$*-close if and only if* $\tilde{p}_k \circ \tilde{p}_{k-1} \circ \cdots \circ \tilde{p}_1$ *is* $(\alpha-1,\delta)$*-two-sided-close.*

Proof. Let $p_k \circ p_{k-1} \circ \cdots \circ p_1$ be an $(\alpha-1,\epsilon)$-close composition of polynomials with odd-degree terms. As such, it suffices to consider the case of $x > 0$. Then, $p_k \circ p_{k-1} \circ \cdots \circ p_1(x) \in [1-2^{-(\alpha-1)}, 1+2^{-(\alpha-1)}]$ for $\epsilon \le x \le 1$. Let $x' = \frac{2}{1+\epsilon}x$. $\epsilon \le x \le 1$ corresponds to $1-\delta \le x' \le 1+\delta$. Then, $\tilde{p}_k \circ \tilde{p}_{k-1} \circ \cdots \circ \tilde{p}_1(x') = p_k \circ p_{k-1} \circ \cdots \circ p_1(x) \in [1-2^{-(\alpha-1)}, 1+2^{-(\alpha-1)}]$ for $1-\delta \le x' \le 1+\delta$. Thus, $\tilde{p}_k \circ \tilde{p}_{k-1} \circ \cdots \circ \tilde{p}_1$ is $(\alpha-1,\delta)$-two-sided-close.

Conversely, let $\tilde{p}_k \circ \tilde{p}_{k-1} \circ \cdots \circ \tilde{p}_1(x') \in [1-2^{-(\alpha-1)}, 1+2^{-(\alpha-1)}]$ for $1-\delta \le x' \le 1+\delta$. Let $x = \frac{1+\epsilon}{2}x'$. $1-\delta \le x' \le 1+\delta$ corresponds to $\epsilon \le x \le 1$. Then, $p_k \circ p_{k-1} \circ \cdots \circ p_1(x) = \tilde{p}_k \circ \tilde{p}_{k-1} \circ \cdots \circ \tilde{p}_1(x') \in [1-2^{-(\alpha-1)}, 1+2^{-(\alpha-1)}]$ for $\epsilon \le x \le 1$, which implies that $p_k \circ p_{k-1} \circ \cdots \circ p_1$ is $(\alpha-1,\epsilon)$-close. Thus, the lemma is proved. □

We note that as $\deg(p_i) = \deg(\tilde{p}_i), 1 \le i \le k$ in Lemma 3, we have

$$\sum_{i=1}^{k} \mathsf{mult}(\deg(p_i)) = \sum_{i=1}^{k} \mathsf{mult}(\deg(\tilde{p}_i)),$$
$$\sum_{i=1}^{k} \mathsf{dep}(\deg(p_i)) = \sum_{i=1}^{k} \mathsf{dep}(\deg(\tilde{p}_i)).$$

It follows that the following two algorithms are equivalent:

(i) An algorithm that determines the $(\alpha-1,\epsilon)$-close composite polynomial $p_k \circ \cdots \circ p_1$ that minimizes depth consumption and the number of non-scalar multiplications.

(ii) An algorithm that determines the $(\alpha-1,\delta)$-two-sided-close composite polynomial $\tilde{p}_k \circ \tilde{p}_{k-1} \circ \cdots \circ \tilde{p}_1$ that minimizes depth consumption and the number of non-scalar multiplications, where $\delta = \frac{1-\epsilon}{1+\epsilon}$.

Thus, we henceforth focus on the latter. In fact, δ is just a temporary precision parameter that is introduced for a concise description. When users perform the proposed algorithms in Algorithms 4, 5, 6, and 7 they should use ϵ rather than δ.

The *minimax composite polynomial*, which is the core of the proposed homomorphic comparison method, is now defined as follows. The principle of the proposed method is to use the minimax composite polynomial to approximate the sign function.

Definition 6. *For* $a, b \in \mathbb{R}$ *such that* $0 < a < b$*, let* $D = \tilde{R}_{a,b}$*. For* $k \in \mathbb{N}$*, let* $\{d_i\}_{1\le i\le k}$ *be a set of odd integers. Then, a composite polynomial* $p_k \circ p_{k-1} \circ \cdots \circ p_1$ *is called a minimax composite polynomial on* D *for* $\{d_i\}_{1\le i\le k}$ *if the followings are satisfied:*

- $p_1 = \mathsf{MP}(D; d_1)$
- $p_i = \mathsf{MP}(p_{i-1} \circ p_{i-2} \circ \cdots \circ p_1(D); d_i), 2 \le i \le k.$

The minimax composite polynomial on D *for* $\{d_i\}_{1\le i\le k}$ *is denoted by* $\mathsf{MCP}(D; \{d_i\}_{1\le i\le k})$*. In addition, for the composite polynomial* $p(x) = \mathsf{MCP}(D; \{d_i\}_{1\le i\le k})$*, the approximation error* $\max_D ||p(x) - \mathrm{sgn}(x)||_\infty$ *is denoted by* $\mathsf{MCE}(D; \{d_i\}_{1\le i\le k})$.

For $1 \le i \le k$, let τ_i be the minimax approximation error of p_i, that is, $\tau_i = \mathsf{ME}(p_{i-1} \circ p_{i-2} \circ \cdots \circ p_1(D); d_i)$. We note that $p_i \circ p_{i-1} \circ \cdots \circ p_1(D) = R_{\tau_i}$ for $1 \le i \le k$ by Theorem 1. In fact, τ_i decreases as i increases. For a minimax composite polynomial $p_k \circ \cdots \circ p_1(x) = \mathsf{MCP}(D; \{d_i\}_{1\le i\le k})$, each component polynomial p_i is a polynomial with odd-degree terms by Lemma 2. If $\tau_k \le 2^{-(\alpha-1)}$, then the minimax composite polynomial on R_δ becomes $(\alpha-1,\delta)$-two-sided-close. The principle problem is to determine the set of degrees $\{d_i\}_{1\le i\le k}$ for which $\mathsf{MCP}(R_\delta, \{d_i\}_{1\le i\le k})$ minimizes the depth consumption and number of non-scalar multiplications among all $(\alpha-1,\delta)$-two-sided-close composite polynomials.

3.2 Optimality of Approximation of the Sign Function by a Minimax Composite Polynomial

In this section, we prove that the approximation of $\mathrm{sgn}(x)$ using our minimax composite polynomial is the optimal approximation with respect to the number of non-scalar multiplications and depth consumption. That is, we prove that for any given $(\alpha-1,\delta)$-two-sided-close composite polynomial that approximates $\mathrm{sgn}(x)$, there is another $(\alpha-1,\delta)$-two-sided-close minimax composite polynomial for some degrees $\{d_i\}_{1\le i\le k}$ that requires a smaller or equal depth consumption and number of non-scalar multiplications. The following definition and lemmas are required for the proof.

Definition 7. *Let* $\{p_i\}_{1\le i\le k}$ *be a set of polynomials.* $p_k \circ p_{k-1} \circ \cdots \circ p_1$ *is called a 1-centered-range composite polynomial on* R_δ *if* $\{p_i\}_{1\le i\le k}$ *is a set of polynomials with odd-degree terms, and there exists a set* $\{\tau_i\}_{1\le i\le k}$ *such that* $p_1([1-\delta, 1+\delta]) = [1-\tau_1, 1+\tau_1]$ *and* $p_i([1-\tau_{i-1}, 1+\tau_{i-1}]) = [1-\tau_i, 1+\tau_i]$ *for* $2 \le i \le k$.

Lemma 4. *Let* $d \in \mathbb{N}$*, and let* $a_1, b_1, a_2, b_2 \in \mathbb{R}$ *satisfy* $0 < a_1 < b_1$ *and* $0 < a_2 < b_2$*. If* $[a_2, b_2] \subseteq [a_1, b_1]$*, then* $\mathsf{ME}(\tilde{R}_{a_2,b_2}; d) \le \mathsf{ME}(\tilde{R}_{a_1,b_1}; d)$.

Proof. Let $e_1 = \mathsf{ME}(\tilde{R}_{a_1,b_1}; d)$ and $e_2 = \mathsf{ME}(\tilde{R}_{a_2,b_2}; d)$. Let e_1' be the maximum approximation error when $\mathsf{MP}(\tilde{R}_{a_1,b_1}; d)$ approximates $\mathrm{sgn}(x)$ on $\tilde{R}_{a_2,b_2}$. Then, we have $e_1 \ge e_1'$. By the definition of the minimax approximation polynomial, among all polynomials that approximate $\mathrm{sgn}(x)$ on $\tilde{R}_{a_2,b_2}$ and have degree less than or equal to d, $\mathsf{ME}(\tilde{R}_{a_2,b_2}; d)$ has the smallest maximum approximation error. As the degree of $\mathsf{ME}(\tilde{R}_{a_1,b_1}; d)$ is smaller than or equal to d, we have that $e_2 \le e_1' \le e_1$, thereby proving the lemma. □

Lemma 5. *Let* $\tilde{p}_k \circ \tilde{p}_{k-1} \circ \cdots \circ \tilde{p}_1$ *be an* $(\alpha-1,\delta)$*-two-sided-close 1-centered-range composite polynomial on* R_δ*. Then, there exists a set of degrees* $\{d_i\}_{1\le i\le k}$ *such that* $\mathsf{MCP}(R_\delta; \{d_i\}_{1\le i\le k})$ *is* $(\alpha-1,\delta)$*-two-sided-close and* $d_i = \deg(\tilde{p}_i)$ *for all* $i, 1 \le i \le k$.

Lemma 6. *Let* $p_k \circ p_{k-1} \circ \cdots \circ p_1$ *be an* $(\alpha-1,\delta)$*-two-sided-close composition of polynomials with odd-degree terms. Then, there is an* $(\alpha-1,\delta)$*-two-sided-close 1-centered-range composite polynomial* $\tilde{p}_k \circ \tilde{p}_{k-1} \circ \cdots \circ \tilde{p}_1$ *on* R_δ *such that* $\deg(\tilde{p}_i) = \deg(p_i)$ *for* $1 \le i \le k$.

The proofs of Lemmas 5 and 6 can be found in Appendices A and B, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TDSC.2021.3105111, respectively. The optimality of

the proposed method of using a minimax composite polynomial is now proved in Theorem 2.

Theorem 2. *Let $p_k \circ p_{k-1} \circ \cdots \circ p_1$ be an $(\alpha-1,\delta)$-two-sided-close composition of polynomials with odd-degree terms. Then, there exists a set of degrees $\{d_i\}_{1\le i\le k}$ such that $\mathsf{MCP}(R_\delta;\{d_i\}_{1\le i\le k})$ is $(\alpha-1,\delta)$-two-sided-close and $d_i = \deg(p_i)$ for all $i, 1 \le i \le k$.*

Proof. Let $p_k \circ p_{k-1} \circ \cdots \circ p_1$ be an $(\alpha-1,\delta)$-two-sided-close composition of polynomials with odd-degree terms. By Lemma 6, there is an $(\alpha-1,\delta)$-two-sided-close 1-centered-range composite polynomial $\tilde{p}_k \circ \tilde{p}_{k-1} \circ \cdots \circ \tilde{p}_1$ on R_δ such that $\deg(\tilde{p}_i) = \deg(p_i)$, $1 \le i \le k$. In addition, by Lemma 5, there is a set of degrees $\{d_i\}_{1\le i\le k}$ such that $\mathsf{MCP}(R_\delta;\{d_i\}_{1\le i\le k})$ is $(\alpha-1,\delta)$-two-sided-close and $d_i = \deg(\tilde{p}_i)$ for all $i, 1 \le i \le k$. Therefore, $d_i = \deg(\tilde{p}_i) = \deg(p_i)$ for all $i, 1 \le i \le k$, thereby proving the theorem. □

3.3 Achieving A Polynomial-Time Algorithm for the New Approximation Method Using Dynamic Programming

From Section 3.2, we know that the method of approximating $\mathrm{sgn}(x)$ using a minimax composite polynomial is the optimal approximation method. Now, for a given depth consumption D, we wish to find a set of degrees $\{d_i\}_{1\le i\le k}$ such that the minimax composite polynomial for $\{d_i\}_{1\le i\le k}$ consumes depth D, is $(\alpha-1,\delta)$-two-sided-close, and requires a smaller or equal number of non-scalar multiplications than any other $(\alpha-1,\delta)$-two-sided-close composite polynomial that consumes depth D. Such a set of degrees is said to be optimal, and in this section we propose a fast method to find such a set.

First, we may consider a naive method to determine the optimal degrees among all sets of degrees by brute-force searching through all combinations of the composition number k and the degrees of the component polynomials $d_1,\ldots,d_k$. However, for an upper bound on the composition number $\bar{k}$ and on the degrees of component polynomials $\bar{d}$, this search requires $O(\bar{d}^{\bar{k}})$ time, which is not practical for large α. Fortunately, dynamic programming allows us to determine the optimal set of degrees in polynomial time, for which we propose an algorithm. Before describing that algorithm, we present a lemma and definitions of $\mathsf{IME}(\tau';d)$, $h_\tau(m,n)$, and $G_\tau(m,n)$.

Lemma 7. *For a fixed odd $d \in \mathbb{N}$, $\mathsf{ME}(R_\tau;d)$ is a strictly increasing continuous function of τ on (0,1).*

The proof of Lemma 7 is given in Appendix C, available in the online supplemental material.

Because $\mathsf{ME}(R_\tau;d)$ is a strictly increasing function of τ, its inverse function exists. We denote the inverse function by $\mathsf{IME}(\tau';d)$. That is, for $d \in \mathbb{N}$ and $\tau \in (0,1)$, $\mathsf{IME}(\tau';d)$ is equal to a value $\tau' \in (0,1)$ such that $\mathsf{ME}(R_{\tau'};d) = \tau$. The approximate value of $\mathsf{IME}(\tau';d)$ can be obtained by a binary search using the improved multi-interval Remez algorithm.

For $m,n \ge 0$ and $\tau \in (0,1)$, $h_\tau(m,n)$ and $G_\tau(m,n)$ are defined by the following recursions:

$$h_\tau(m,n) = \begin{cases} \tau & \text{if } m \le 1 \text{ or } n \le 1 \\ \mathsf{IME}(h_\tau(m-\mathsf{mult}(2j_{m,n}+1), \\ n-\mathsf{dep}(2j_{m,n}+1));2j_{m,n}+1) & \text{otherwise,} \end{cases}$$

$$G_\tau(m,n) = \begin{cases} \phi & \text{if } m \le 1 \text{ or } n \le 1 \\ \{2j_{m,n}+1\} \cup G_\tau(m-\mathsf{mult}(2j_{m,n}+1), \\ n-\mathsf{dep}(2j_{m,n}+1)) & \text{otherwise,} \end{cases}$$

where $j_{m,n} =$ $\mathrm{argmax}_{1\le i;\ \mathsf{mult}(2i+1)\le m;\ \mathsf{dep}(2i+1)\le n} \mathsf{IME}(h_\tau(m-\mathsf{mult}(2i+1), n-\mathsf{dep}(2i+1));2i+1)$.

In the definition of $G_\tau(m,n)$, $\{2j_{m,n}+1\} \cup G_\tau(m-\mathsf{mult}(2j_{m,n}+1), n-\mathsf{dep}(2j_{m,n}+1))$ is the insertion of $2j_{m,n}+1$ into the ordered set $G_\tau(m-\mathsf{mult}(2j_{m,n}+1), n-\mathsf{dep}(2j_{m,n}+1))$ as the first component. Then, a theorem for $h_\tau(m,n)$ and $G_\tau(m,n)$ is presented as follows.

Theorem 3. *Let $m,n \in \mathbb{Z}$ such that $m \ge 2$ and $n \ge 2$. Then, $h_\tau(m,n)$ is the maximum value of $\tau' \in (0,1)$ such that there exists a set of degrees $\{d_i\}_{1\le i\le k}$ that satisfies $\mathsf{MCE}(R_{\tau'};\{d_i\}_{1\le i\le k}) \le \tau$, $\sum_{i=1}^{k}\mathsf{mult}(d_i) \le m$, and $\sum_{i=1}^{k}\mathsf{dep}(d_i) \le n$. In addition, for $G_\tau(m,n) = \{d'_i\}_{1\le i\le k'}$, we have $\mathsf{MCE}(R_{h_\tau(m,n)};\{d'_i\}_{1\le i\le k'}) \le \tau$, $\sum_{i=1}^{k'}\mathsf{mult}(d'_i) \le m$, and $\sum_{i=1}^{k'}\mathsf{dep}(d'_i) \le n$.*

The proof of Theorem 3 is given in Appendix D, available in the online supplemental material.

For $0 \le m \le m_{\max}$ and $0 \le n \le n_{\max}$, the values of $h_\tau(m,n)$ and $G_\tau(m,n)$ are recursively computed by the `ComputehG` algorithm in Algorithm 3. That is, the two dimensional tables $\tilde{h}$ and $\tilde{G}$ that satisfy $\tilde{h}(m,n) = h_\tau(m,n)$ and $\tilde{G}(m,n) = G_\tau(m,n)$ for $0 \le m \le m_{\max}$ and $0 \le n \le n_{\max}$ are obtained from the `ComputehG` algorithm. In this study, only minimax approximation polynomials with degree at most 31 are used because using polynomials of higher degree may cause more numerical errors.

Algorithm 3. `ComputehG`(τ)

Input: τ
Output: 2-dimensional tables $\tilde{h}$ and $\tilde{G}$
1: Generate 2-dimensional tables $\tilde{h}$ and $\tilde{G}$, both of which have size of $(m_{\max}+1)\times(n_{\max}+1)$.
2: **for** $m \leftarrow 0$ **to** $m_{\max}$ **do**
3: **for** $n \leftarrow 0$ **to** $n_{\max}$ **do**
4: **if** $m \le 1$ or $n \le 1$ **then**
5: $\tilde{h}(m,n) \leftarrow \tau$
6: $\tilde{G}(m,n) \leftarrow \phi$
7: **else**
8: $j \leftarrow \mathrm{argmax}_{\substack{1\le i \\ \mathsf{mult}(2i+1)\le m \\ \mathsf{dep}(2i+1)\le n}} \mathsf{IME}(\tilde{h}(m-\mathsf{mult}(2i+1), n-\mathsf{dep}(2i+1));2i+1)$
9: $\tilde{h}(m,n) \leftarrow \mathsf{IME}(\tilde{h}(m-\mathsf{mult}(2j+1), n-\mathsf{dep}(2j+1));2j+1)$
10: $\tilde{G}(m,n) \leftarrow \{2j+1\} \cup \tilde{G}(m-\mathsf{mult}(2j+1), n-\mathsf{dep}(2j+1))$
11: **end**
12: **end**
13: **end**

The algorithms `ComputeMinDep` in Algorithm 4 and `ComputeMinMultDegs` in Algorithm 5 are now introduced. They use the values of $h_\tau(m,n)$ and $G_\tau(m,n)$ obtained from the `ComputehG` algorithm. First, `ComputeMinDep` determines the minimum required depth

consumption M_{dep}. The user then chooses a depth consumption $D(\geq M_{\text{dep}})$ to use, and `ComputeMinMultDegs` determines the minimum number of non-scalar multiplications M_{mult} and corresponding optimal set of degrees M_{degs}. If the algorithms fail to obtain M_{dep} or M_{mult} because $m_{\max}$ and $n_{\max}$ are not large enough, they return an error symbol $\perp$. As such, $m_{\max}$ and $n_{\max}$ should be set large enough for these algorithms to obtain accurate outputs, and we experimentally confirm that these algorithms obtain accurate outputs when $m_{\max} = 70$ and $n_{\max} = 40$ for $\alpha \leq 20$.

The procedure to obtain the optimal minimax composite polynomial using dynamic programming is summarized as follows:

(i) $h_\tau(m,n)$ and $G_\tau(m,n)$ are computed recursively using the `ComputehG` algorithm.

(ii) For a depth consumption D chosen by the user, M_{degs} is obtained from the `ComputeMinMultDegs` algorithm using the values of $h_\tau(m,n)$ and $G_\tau(m,n)$.

(iii) The component minimax approximation polynomials p_i with degrees $(M_{\text{degs}})_i$ for $i, 1 \leq i \leq |M_{\text{degs}}|$ are determined using the improved multi-interval Remez algorithm.

Algorithm 4. `ComputeMinDep` (α, ϵ)

Input: Precision parameters α and ϵ
Output: Minimum depth consumption M_{dep}

$\tilde{h}, \tilde{G} \leftarrow$ `ComputehG`$(2^{1-\alpha})$
for $i \leftarrow 0$ **to** $n_{\max}$ **do**
  if $\tilde{h}(m_{\max}, i) \geq \delta = \frac{1-\epsilon}{1+\epsilon}$ **then**
    $M_{\text{dep}} \leftarrow i$
    return M_{dep}
  end
  if $i = n_{\max}$ **then**
    return $\perp$
  end
end

Algorithm 5. `ComputeMinMultDegs`(α, ϵ, D)

Input: Precision parameters α and ϵ, and depth consumption D
Output: Minimum number of multiplications M_{mult} and optimal set of degrees M_{degs}

1: $\tilde{h}, \tilde{G} \leftarrow$ `ComputehG`$(2^{1-\alpha})$
2: **for** $j \leftarrow 0$ **to** $m_{\max}$ **do**
3: **if** $\tilde{h}(j, D) \geq \delta = \frac{1-\epsilon}{1+\epsilon}$ **then**
4: $M_{\text{mult}} \leftarrow j$
5: Go to line 11
6: **end**
7: **if** $j = m_{\max}$ **then**
8: **return** $\perp$
9: **end**
10: **end**
11: $M_{\text{degs}} \leftarrow \tilde{G}(M_{\text{mult}}, D)$ // M_{degs}: ordered set
12: **return** M_{mult} and M_{degs}

Theorem 4 shows that M_{dep} obtained from `ComputeMinDep` is the minimum depth consumption that satisfies the given precision requirements. For M_{mult} and M_{degs} obtained from `ComputeMinMultDegs`, Theorem 5 shows that M_{mult} is the minimum number of non-scalar multiplications when the depth consumption is D, and M_{degs} is the corresponding optimal set of degrees.

Theorem 4. *Let M_{dep} be the output value of Algorithm 4 for inputs α and ϵ. Then, $M_{\text{dep}} \leq \sum_{i=1}^{k} \mathsf{dep}(\deg(p_i))$ for any $(\alpha-1,\epsilon)$-close composition of polynomials with odd-degree terms $p_k \circ p_{k-1} \circ \cdots \circ p_1$.*

Proof. Let $p_k \circ p_{k-1} \circ \cdots \circ p_1$ be an $(\alpha-1,\epsilon)$-close composition of polynomials with odd-degree terms. Let $\sum_{i=1}^{k} \mathsf{mult}(\deg(p_i)) = m$ and $\sum_{i=1}^{k} \mathsf{dep}(\deg(p_i)) = n$. From Lemma 3, there exists an $(\alpha-1,\delta)$-two-sided-close composition of polynomials with odd-degree terms $\tilde{p}_k \circ \cdots \circ \tilde{p}_1$ such that $\deg(\tilde{p}_i) = \deg(p_i)$ for all $i, 1 \leq i \leq k$. In addition, from Theorem 2, there is a set of degrees $\{d_i\}_{1\leq i\leq k}$ such that $\mathsf{MCP}(R_\delta; \{d_i\}_{1\leq i\leq k})$ is $(\alpha-1,\delta)$-two-sided-close and $d_i = \deg(\tilde{p}_i)$ for all $i, 1 \leq i \leq k$. We assume that $n < M_{\text{dep}}$. Then, we have $\sum_{i=1}^{k} \mathsf{dep}(d_i) = \sum_{i=1}^{k} \mathsf{dep}(\deg(\tilde{p}_i)) = \sum_{i=1}^{k} \mathsf{dep}(\deg(p_i)) = n < M_{\text{dep}}$. Because $n < M_{\text{dep}}$, and M_{dep} is the smallest i that satisfies $h_{2^{1-\alpha}}(m_{\max}, i) \geq \delta$, we have $h_{2^{1-\alpha}}(m_{\max}, n) < \delta$. This leads to a contradiction because there exists a set of degrees $\{d_i\}_{1\leq i\leq k}$ such that $\mathsf{MCE}(R_\delta; \{d_i\}_{1\leq i\leq k}) \leq 2^{1-\alpha}$, $\sum_{i=1}^{k} \mathsf{mult}(d_i) \leq m_{\max}$, and $\sum_{i=1}^{k} \mathsf{dep}(d_i) \leq n$. Thus, we have $M_{\text{dep}} \leq n = \sum_{i=1}^{k} \mathsf{dep}(\deg(p_i))$. □

Theorem 5. *Let M_{mult} and M_{degs} be the output values of Algorithm 5 for depth consumption D and precision parameters α and ϵ. Then, $M_{\text{mult}} \leq \sum_{i=1}^{k} \mathsf{mult}(\deg(p_i))$ for any $(\alpha-1,\epsilon)$-close composition of polynomials with odd-degree terms $p_k \circ p_{k-1} \circ \cdots \circ p_1$ satisfying $\sum_{i=1}^{k} \mathsf{dep}(\deg(p_i)) = D$.*

Proof. Let $p_k \circ p_{k-1} \circ \cdots \circ p_1$ be an $(\alpha-1,\epsilon)$-close composition of polynomials with odd-degree terms. Let $\sum_{i=1}^{k} \mathsf{mult}(\deg(p_i)) = m$ and $\sum_{i=1}^{k} \mathsf{dep}(\deg(p_i)) = D$. From Lemma 3, there exists an $(\alpha-1,\delta)$-two-sided-close composition of polynomials with odd-degree terms $\tilde{p}_k \circ \cdots \circ \tilde{p}_1$ such that $\deg(\tilde{p}_i) = \deg(p_i)$ for all $i, 1 \leq i \leq k$. In addition, from Theorem 2, there is a set of degrees $\{d_i\}_{1\leq i\leq k}$ such that $\mathsf{MCP}(R_\delta; \{d_i\}_{1\leq i\leq k})$ is $(\alpha-1,\delta)$-two-sided-close and $d_i = \deg(\tilde{p}_i)$ for all $i, 1 \leq i \leq k$. We assume that $m < M_{\text{mult}}$. Then we have $\sum_{i=1}^{k} \mathsf{mult}(d_i) = \sum_{i=1}^{k} \mathsf{mult}(\deg(\tilde{p}_i)) = \sum_{i=1}^{k} \mathsf{mult}(\deg(p_i)) = m < M_{\text{mult}}$ and $\sum_{i=1}^{k} \mathsf{dep}(d_i) = \sum_{i=1}^{k} \mathsf{mult}(\deg(\tilde{p}_i)) = \sum_{i=1}^{k} \mathsf{mult}(\deg(p_i)) = D$. Because $m < M_{\text{mult}}$, and M_{mult} is the smallest j that satisfies $h_{2^{1-\alpha}}(j, D) \geq \delta$, we have $h_{2^{1-\alpha}}(m, D) < \delta$. This leads to a contradiction because there exists a set of degrees $\{d_i\}_{1\leq i\leq k}$ such that $\mathsf{MCE}(R_\delta; \{d_i\}_{1\leq i\leq k}) \leq 2^{1-\alpha}$, $\sum_{i=1}^{k} \mathsf{mult}(d_i) \leq m$, and $\sum_{i=1}^{k} \mathsf{dep}(d_i) \leq D$. Thus, we have $M_{\text{mult}} \leq m = \sum_{i=1}^{k} \mathsf{mult}(\deg(p_i))$. □

The `MinimaxComp` algorithm, which outputs an approximate value of $\text{comp}(u,v)$, is now proposed in Algorithm 6. It uses the optimal set of degrees for α, ϵ, and D obtained from `ComputeMinMultDegs`. The error between the output of `MinimaxComp` and $\text{comp}(u,v)$ is bounded by $2^{-\alpha}$ for any $u, v \in [0,1]$ satisfying $|u-v| \geq \epsilon$. Here, ϵ and α are precision parameters for the input and output, respectively, that users of the homomorphic comparison operation can choose.

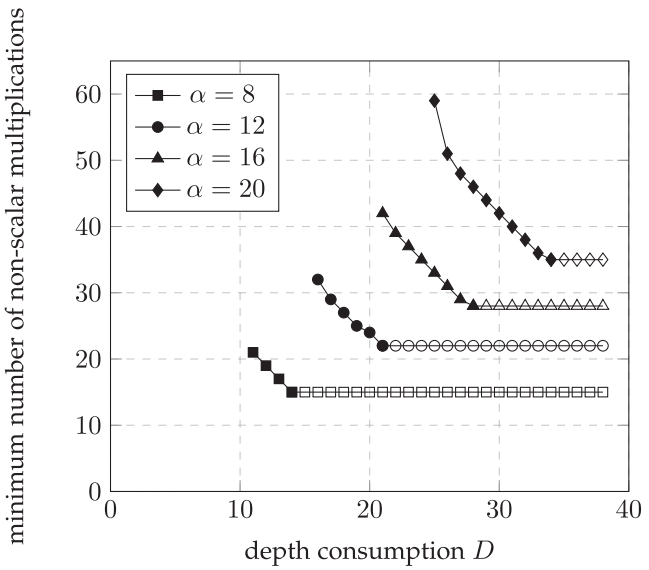


Fig. 3. The minimum number of non-scalar multiplications of minimax composite polynomials for homomorphic comparison operation according to depth consumption D.

3.4 Optimal Composite Polynomials for Homomorphic Comparison and Their Performance

Herein, using dynamic programming and the `ComputeMinMultDegs` algorithm, the optimal set of degrees and the corresponding number of non-scalar multiplications for a given depth consumption D are determined. Fig. 3 shows the minimum number of non-scalar multiplications of minimax composite polynomials required for the homomorphic comparison operation of depth consumption D. It is apparent that there is a tradeoff between the required depth consumption and the number of non-scalar multiplications. An empty mark implies that the operation never needs to be used with those parameters because another set of parameters uses the same number of non-scalar multiplications but with a smaller depth consumption.

Algorithm 6. `MinimaxComp`$(u, v; \alpha, \epsilon, D)$

Input: Inputs $u, v \in (0, 1)$, precision parameters α and ϵ, and depth consumption D
Output: Approximate value of $\mathrm{comp}(u, v)$

$M_{\mathrm{degs}} = \{d_1, d_2, \ldots, d_k\} \leftarrow$ `ComputeMinMultDegs`(α, ϵ, D)
$p_1 \leftarrow \mathsf{MP}(\tilde{R}_{1-\epsilon,1}; d_1)$
$\tau_1 \leftarrow \mathsf{ME}(\tilde{R}_{1-\epsilon,1}; d_1)$
for $i \leftarrow 2$ **to** k **do**
  $p_i \leftarrow \mathsf{MP}(R_{\tau_{i-1}}; d_i)$
  $\tau_i \leftarrow \mathsf{ME}(R_{\tau_{i-1}}; d_i)$
end
return $\frac{p_k \circ p_{k-1} \circ \cdots \circ p_1(u-v)+1}{2}$

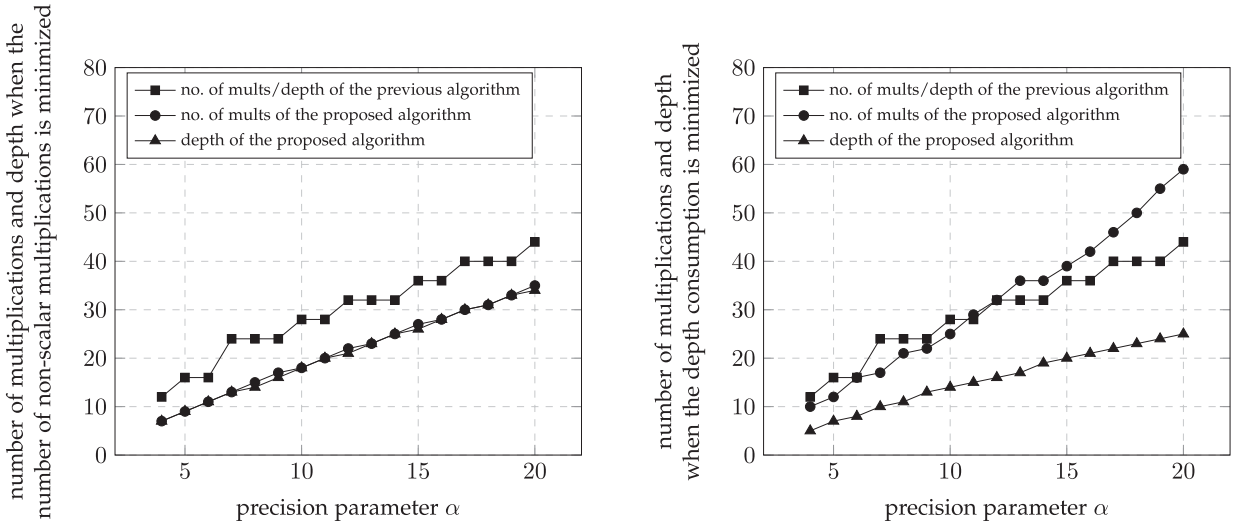
We analyze the performance of several parameterizations in which either the number of non-scalar multiplications or the depth consumption is minimized. Table 2 shows the ordered sets of degrees M_{degs} that optimizes the comparison operation for either the number of non-scalar multiplications or the depth consumption at various output precisions α. The minimum depth consumption and the minimum number of required non-scalar multiplications are computed using `ComputeMinDep` and `ComputeMinMultDegs`. Other options that are not shown in Table 2 are given in Appendix E, available in the online supplemental material.

TABLE 2
Optimal Set of Degrees M_{degs} Obtained from `ComputeMinMultDegs` Algorithm Indicating the Degrees of the Component Minimax Approximation Polynomials When Depth Consumption or the Number of Non-Scalar Multiplications is Minimized

α	ordered set of degrees of the optimal component minimax approximation polynomials M_{degs}	
	minimize multiplications	minimize depth
4	$\{3,3,5\}$	$\{27\}$
5	$\{5,5,5\}$	$\{7,13\}$
6	$\{3,5,5,5\}$	$\{15,15\}$
7	$\{3,3,5,5,5\}$	$\{7,7,13\}$
8	$\{3,3,5,5,9\}$	$\{7,15,15\}$
9	$\{5,5,5,5,9\}$	$\{7,7,7,13\}$
10	$\{5,5,5,5,5,5\}$	$\{7,7,13,15\}$
11	$\{3,5,5,5,5,5,5\}$	$\{7,15,15,15\}$
12	$\{3,5,5,5,5,5,9\}$	$\{15,15,15,15\}$
13	$\{3,5,5,5,5,5,5,5\}$	$\{15,15,15,31\}$
14	$\{3,3,5,5,5,5,5,5,5\}$	$\{7,7,15,15,27\}$
15	$\{3,3,5,5,5,5,5,5,9\}$	$\{7,15,15,15,27\}$
16	$\{3,3,5,5,5,5,5,5,5,5\}$	$\{15,15,15,15,27\}$
17	$\{5,5,5,5,5,5,5,5,5,5\}$	$\{15,15,15,29,29\}$
18	$\{3,3,5,5,5,5,5,5,5,5,5\}$	$\{15,15,29,29,31\}$
19	$\{5,5,5,5,5,5,5,5,5,5,5\}$	$\{15,29,31,31,31\}$
20	$\{5,5,5,5,5,5,5,5,5,5,9\}$	$\{29,31,31,31,31\}$

For an integer $n = 4$ and precision parameters α and ϵ, the previous best comparison algorithm `CompG` (referred to as `NewCompG` in [15]) had lower bounds for depth consumption and the number of non-scalar multiplications of $4(\lceil \frac{1}{\log 0.98 c_n^2} \cdot \log(2/\epsilon) \rceil + \lceil \frac{1}{\log(n+1)} \cdot \log(\alpha - 2) \rceil)$, where $c_n = \frac{2n+1}{4^n}\binom{2n}{n}$. This lower bound, which can be obtained from Lemmas 1 and 3 in [15], is very close to that obtained experimentally. In this study, the performances of the previous and proposed homomorphic comparison algorithms are both analyzed when $\epsilon = 2^{-\alpha}$.

Fig. 4a shows a comparison of the number of required non-scalar multiplications and depth consumption between the previous homomorphic comparison operation algorithm and the proposed algorithm when the number of non-scalar multiplications is minimized. We note that the previous algorithm has the same number of non-scalar multiplications and depth consumption. This is because the degrees of all of the component polynomials used in [15] are nine, and the required number of non-scalar multiplications and depth consumption of polynomials of degree nine using the polynomial evaluation method in [15] are both four. The minimum number of required non-scalar multiplications and the corresponding depth consumption for the proposed algorithm are reduced by approximately 30 and 31 percent, respectively, on average, compared with the corresponding values for the previous algorithm. In this case, though the proposed algorithm is aimed at minimizing the number of non-scalar multiplications, the corresponding depth consumption is also reduced.



(a) When the number of non-scalar multiplications is minimized (b) When the depth consumption is minimized

Fig. 4. Comparison of the minimum number of non-scalar multiplications and depth consumption between the previous and the proposed algorithms when the number of non-scalar multiplications or the depth consumption is minimized.

Fig. 4b shows a comparison of the number of required non-scalar multiplications and depth consumption between the previous algorithm and the proposed homomorphic comparison algorithm when depth consumption is minimized. The minimum depth consumption for the proposed algorithms is reduced by approximately 48 percent on average, and the corresponding number of non-scalar multiplications is increased by approximately 4 percent on average. Although the number of non-scalar multiplications for the proposed algorithm is slightly larger than that for the previous algorithm, the proposed algorithm would require less runtime when bootstrapping is required because bootstrapping, due to the large depth consumption, requires much more runtime than non-scalar multiplication operations.

3.5 Proposed Homomorphic Comparison Operation Using Margin

The proposed homomorphic comparison operation in Algorithm 6 satisfies the comparison operation error condition if there are no approximation errors other than those caused by the polynomial approximation itself. However, if Algorithm 6 is used in the approximate CKKS scheme, the difference between $p_i \circ \cdots \circ p_1(x)$ and $\mathrm{sgn}(x)$ for some $i, 1 \le i \le k$ and $x \in [\epsilon, 1]$ can be greater than the minimax approximation error τ_i of p_i due to approximation errors caused by operations of the CKKS scheme. This means that $p_i \circ \cdots \circ p_1(x)$ may not fall into $[1-\tau_i, 1+\tau_i]$, which is the domain of the next minimax approximation polynomial p_{i+1}, causing the homomorphic comparison operation to fail. Thus, we propose an improved algorithm, Algorithm 7. This algorithm adds the use of margins to Algorithm 6 to make it resistant to errors caused by the CKKS scheme. Algorithm 7 is used instead of Algorithm 6 in the numerical analysis in Section 5.

As the value of margin η becomes larger, the homomorphic comparison operation becomes more resistant to approximation errors caused by the CKKS scheme. Thus, η is set as large as possible among all values for which $\tau_k \le 2^{1-\alpha}$, implying that the homomorphic comparison operation using margin satisfies the comparison operation error condition.

Algorithm 7. `MinimaxComp` $(u, v; \alpha, \epsilon, D, \eta)$ (Using Margin)

Input: Inputs $u, v \in (0, 1)$, precision parameters α and ϵ, depth consumption D, and margin η
Output: Approximate value of $\mathrm{comp}(u, v)$

$M_{\mathrm{degs}} = \{d_1, d_2, \ldots, d_k\} \leftarrow$ `ComputeMinMultDegs`(α, ϵ, D)
$p_1 \leftarrow \mathsf{MP}(\tilde{R}_{1-\epsilon,1}; d_1)$
$\tau_1 \leftarrow \mathsf{ME}(\tilde{R}_{1-\epsilon,1}; d_1) + \eta$
for $i \leftarrow 2$ **to** k **do**
  $p_i \leftarrow \mathsf{MP}(R_{\tau_{i-1}}; d_i)$
  $\tau_i \leftarrow \mathsf{ME}(R_{\tau_{i-1}}; d_i) + \eta$
end
return $\frac{p_k \circ p_{k-1} \circ \cdots \circ p_1(u-v)+1}{2}$

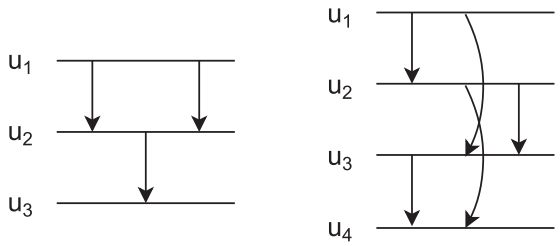
4 Application to Min/Max and Sorting

The max function is useful in a variety of applications, such as the deep learning max pooling operation and sorting algorithms. It is easily implemented using the sign function, that is,

$$\max(u, v) = \frac{(u+v) + (u-v)\,\mathrm{sgn}(u-v)}{2}.$$

Thus, the proposed method of determining polynomials that approximate $\mathrm{sgn}(x)$ can also be used to determine polynomials that approximate the max function. Additionally, the min function can be easily implemented using the max function, that is, $\min(u, v) = u + v - \max(u, v)$.

For some polynomial $p(x)$ that approximates $\mathrm{sgn}(x)$, the error between $\frac{(u+v)+(u-v)p(u-v)}{2}$ and $\max(u, v)$ should be bounded by $2^{-\alpha}$ for any $u, v \in [0, 1]$. The `MinimaxMax` algorithm in Algorithm 8 is the proposed homomorphic max function that uses the optimal set of degrees M_{degs} obtained from `ComputeMinMultDegs`. It should be noted that the inputs of `ComputeMinMultDegs` to perform the proposed



(a) Three numbers (b) Four numbers

Fig. 5. Diagrams representing the sorting algorithms for three and four numbers. Each arrow from u_i to u_j indicates the sorting of the data u_i and u_j. After the sorting, $\max(u_i, u_j)$ and $\min(u_i, u_j)$ are at the end and start points of the arrow, respectively.

homomorphic max function are different from those to perform the proposed homomorphic comparison operation. First, $\zeta \cdot 2^{-\alpha}$ for some max function factor $\zeta > 1$ is used instead of ϵ. In addition, depth consumption $D-1$ is used instead of depth consumption D because another depth is required to compute $\frac{(u-v)p_k \circ p_{k-1} \circ \cdots \circ p_1(u-v)+(u+v)}{2}$ in the homomorphic max function. The larger the value of ζ, the smaller the depth consumption and the number of non-scalar multiplications that this homomorphic max function requires. We experimentally set the value of ζ as large as possible among all values of ζ for which the homomorphic max function satisfies the error condition.

Algorithm 8. `MinimaxMax` $(u, v; \alpha, \zeta, D, \eta)$ (Using Margin)

Input: Inputs $u, v \in (0, 1)$, precision parameter α, max function factor ζ, depth consumption D, and margin η
Output: Approximate value of $\max(u, v)$

$M_{\text{degs}} = \{d_1, d_2, \ldots, d_k\} \leftarrow$ `ComputeMinMultDegs`$(\alpha, \zeta \cdot 2^{-\alpha}, D-1)$
$p_1 \leftarrow \mathsf{MP}(\tilde{R}_{1-\epsilon,1}; d_1)$
$\tau_1 \leftarrow \mathsf{ME}(\tilde{R}_{1-\epsilon,1}; d_1) + \eta$
for $i \leftarrow 2$ **to** k **do**
  $p_i \leftarrow \mathsf{MP}(R_{\tau_{i-1}}; d_i)$
  $\tau_i \leftarrow \mathsf{ME}(R_{\tau_{i-1}}; d_i) + \eta$
end
return $\frac{(u-v)p_k \circ p_{k-1} \circ \cdots \circ p_1(u-v)+(u+v)}{2}$

The proposed homomorphic max function algorithm can also be used for sorting. In this study, we analyze the performance of sorting algorithms for three numbers and four numbers using the proposed homomorphic max function. Fig. 5 offers a diagrammatic representation of these sorting algorithms.

5 Numerical Results

In this section, numerical results regarding the proposed `MinimaxComp`, `MinimaxMax`, and homomorphic sorting algorithms are presented. The performances of the proposed algorithms are evaluated and compared with those of the previous algorithms in [15]. The numerical analysis is conducted using the CKKS scheme library HEAAN [3] on a Linux PC with an Intel Core i7-10700 CPU at 2.90GHz with 8 threads.

5.1 Parameter Setting

The precision parameters, ϵ and α, determine the input and output precisions of the homomorphic comparison operation, respectively. Throughout our analysis, we set $\epsilon = 2^{-\alpha}$, implying that the input and output of the homomorphic comparison operation have the same bit precision. Unlike the homomorphic comparison operation, there is only one precision parameter α that determines the precision in the homomorphic max function. We set $N = 2^{17}$, and then initial ciphertext modulus can be set up to $q_L = 2^{1700}$ to achieve 128-bit security as estimated by Albrecht's LWE estimator [28]. The script for security estimation can be found in [15]. We simultaneously perform the homomorphic comparison operation or homomorphic max function for $N/2$ tuples of real numbers. Thus, the amortized runtime is computed by dividing the total runtime by $N/2$.

5.1.1 Scaling Values and Margins

If the power basis is used for the polynomial approximation of the sign function, the coefficients often become excessively small or large, potentially reducing the accuracy of the homomorphic comparison operation. Accordingly, we use the scaled Chebyshev polynomials $\tilde{T}_i(t) = T_i(t/w)$ for some scaling value $w \geq 1$ as basis polynomials. The scaled Chebyshev polynomials are evaluated using the following recursion:

$$\begin{aligned}
&\tilde{T}_0(x) = 1 \\
&\tilde{T}_1(x) = x/w \\
&\tilde{T}_{i+j}(x) = 2\tilde{T}_i(x)\tilde{T}_j(x) - \tilde{T}_{i-j}(x) \text{ for } i \geq j \geq 1.
\end{aligned}$$

We experimentally obtain the scaling values w and margins η (for use in Algorithms 7 and 8) as shown in Table 3.

5.1.2 Initial Modulus

When we execute the homomorphic comparison operation, we set the initial ciphertext modulus q_L such that the final modulus is $\log \Delta + 10$ bits long as in [15]. For a composite polynomial $p_k \circ p_{k-1} \circ \cdots \circ p_1$, where $\deg(p_i) = d_i$, the depth $\sum_{i=1}^{k} \mathsf{dep}(d_i)$ is consumed. Additional depth consumption is also required because of the multiplications by $1/w$ in computing the scaled Chebyshev polynomial $\tilde{T}_1(x)$ and by $1/2$ in computing $\frac{p_k \circ p_{k-1} \circ \cdots \circ p_1(u-v)+1}{2}$ in Algorithm 6. However, for a set of scaling values $\{w_1, w_2, \ldots, w_k\}$, if we multiply the coefficients of p_i by $1/w_{i+1}$ for $1 \leq i \leq k-1$ and those of p_k by $1/2$, then no additional depth consumption is required. Then, the initial ciphertext modulus bit length $\log q_L$ is given by

$$\log q_L = \log \Delta \cdot (\mathsf{dep}(d_1) + \cdots + \mathsf{dep}(d_k)) + \log \Delta + 10.$$

Similarly, when we execute the homomorphic max function, the initial ciphertext modulus bit length is

$$\log q_L = \log \Delta \cdot (\mathsf{dep}(d_1) + \cdots + \mathsf{dep}(d_k)) + 2\log \Delta + 10,$$

which is $\log \Delta$ bits larger than in the homomorphic comparison operation because the max function requires one more depth in computing $\frac{(u-v)p_k \circ p_{k-1} \circ \cdots \circ p_1(u-v)+(u+v)}{2}$ in Algorithm 8.

5.1.3 Scaling Factor

Increasing the scaling factor Δ increases the accuracy of the homomorphic comparison operation and homomorphic

TABLE 3
Set of Scaling Values and Margins for the Proposed Homomorphic Comparison Operation and Homomorphic Max Function

α	proposed homomorphic comparison operation MinimaxComp				proposed homomorphic max function MinimaxMax			
	minimize runtime		minimize depth		minimize runtime		minimize depth	
	scaling values w	margin η	scaling values w	margin η	scaling values w	margin η	scaling values w	margin η
8	{1,2,1.6}	2^{-12}	{1,2,1.6}	2^{-12}	{1,2,1.6}	2^{-9}	{1,1.6}	$2^{-10.5}$
12	{1,2,2,2,1.6}	2^{-15}	{1,2,2,1.6}	2^{-14}	{1,2,2,1.6}	2^{-13}	{1,2,1.7}	2^{-13}
16	{1,2,2,2,2,1.7}	2^{-18}	{1,2,2,2,1.6}	2^{-17}	{1,2,2,2,2,1.6}	2^{-16}	{1,2,2,2,1.6}	2^{-17}
20	{1,2,2,2,2,2,2,2,1.6}	2^{-21}	{1,2,2,2,1.6}	2^{-22}	{1,2,2,2,2,1.6}	$2^{-20.5}$	{1,2,2,2,1.6}	2^{-20}

TABLE 4
Failure Rate of the Proposed Algorithm MinimaxComp on HEAAN According to the Scaling Factor for Various α When Runtime and Depth Consumption Are Minimized

$\log \Delta$	failure rate of the proposed algorithm MinimaxComp							
	minimize runtime				minimize depth consumption			
	α							
	8	12	16	20	8	12	16	20
32	$< 10^{-6}$	$< 10^{-6}$	0.004653	0.522744	$< 10^{-6}$	$< 10^{-6}$	0.009860	0.974903
34	$< 10^{-6}$	$< 10^{-6}$	0.000075	0.202683	$< 10^{-6}$	$< 10^{-6}$	0.000801	0.390946
36	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	0.125566	$< 10^{-6}$	$< 10^{-6}$	0.000004	0.171154
38	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	0.000921	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	0.006707
40	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	0.000019	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	0.000655
42	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	0.000020
44	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$

TABLE 5
Optimal Set of Degrees for the Proposed Homomorphic Comparison Operation and Homomorphic Max Function When Runtime and Depth Consumption Are Minimized

α	optimal set of degrees					
	proposed homomorphic comparison operation MinimaxComp		proposed homomorphic max function MinimaxMax			
	minimize runtime	minimize depth	ζ	minimize runtime	minimize depth
8	{7,15,15}	{7,15,15}	12	{3,5,9}	{7,15}
12	{7,7,7,13,13}	{15,15,15,15}	15	{5,7,7,15}	{7,15,27}
16	{7,7,7,13,13,27}	{15,15,15,15,27}	19	{5,5,5,7,7,15}	{7,7,7,15,15}
20	{5,5,5,7,7,7,7,7,27}	{29,31,31,31,31}	21	{7,7,7,13,13,27}	{15,15,15,15,27}

max function. These functions are said to fail for an input if the output does not satisfy the corresponding error condition. We perform the proposed algorithms for 2^{20} input tuples for each α and record the number of failures. The failure rate of the algorithm is thus the number of failures divided by the total number of inputs tuples. As such, a rate of less than 10^{-6} indicates that no failure is recorded for a given parameter set. Table 4 presents the failure rate of the proposed MinimaxComp algorithm for various scaling factors and precisions when either runtime or depth consumption is minimized. We note that if the scaling factor is larger than a certain threshold, the number of failures is zero. In particular, when α is 8, 12, or 16, a value of 40 for $\log\ \Delta$ results in no errors, whereas a value of 44 is required when α is 20.

5.1.4 Optimal Degrees

In this numerical analysis, the initial modulus is determined according to the depth consumption and therefore the runtime is affected by not just the number of non-scalar multiplications but also by depth consumption. As such, minimizing the number of non-scalar multiplications does not necessarily correspond with runtime minimization, and we perform numerical analyses for both the cases of minimized runtime and minimized depth consumption. Table 5 details the optimal set of degrees for the proposed homomorphic comparison operation and homomorphic max function when either the runtime or the depth consumption is minimized.

5.2 Performance of the Proposed Homomorphic Comparison Algorithm MinimaxComp

We compare the performance of the proposed homomorphic comparison algorithm MinimaxComp in Algorithm 7 with that of the previous algorithm CompG [15] using the scaling values w and margins η in presented Table 3. Table 6 shows the runtime (and amortized runtime) of CompG and

TABLE 6
Runtime (Amortized Runtime) and Ciphertext Modulus Bit Consumption of CompG (Cheon *et al.* [15]) and the Proposed MinimaxComp on HEAAN for Various α; an Asterisk (*) Indicates That the Parameter Set Does Not Achieve 128-bit Security Because of the Large $\log q_L \geq 1700$

α	$\log \Delta$	CompG (Cheon *et al.* [15])		proposed algorithm MinimaxComp			
				minimize runtime		minimize depth	
		runtime	$\log(q_L/q_f)$	runtime	$\log(q_L/q_f)$	runtime	$\log(q_L/q_f)$
8	40	15 s (0.22 ms)	844	8 s (0.12 ms)	440	8s (0.12 ms)	440
12	40	28 s (0.42 ms)	1184	15 s (0.22 ms)	680	16s (0.24 ms)	640
16	40	45 s (0.68 ms)	1524	24 s (0.36 ms)	880	25s (0.38 ms)	840
20	44	62 s* (0.94 ms)	1854*	38 s (0.57 ms)	1276	43s (0.65 ms)	1100

TABLE 7
Runtime (Amortized Runtime) and Ciphertext Modulus Bit Consumption of MaxG (Cheon *et al.* [15]) and the Proposed MinimaxMax on HEAAN for Various α

α	$\log \Delta$	MaxG (Cheon *et al.* [15])		proposed algorithm MinimaxMax			
				minimize runtime		minimize depth	
		runtime	$\log(q_L/q_f)$	runtime	$\log(q_L/q_f)$	runtime	$\log(q_L/q_f)$
8	40	8 s (0.12 ms)	548	4 s (0.06 ms)	400	5s (0.07 ms)	320
12	40	16 s (0.24 ms)	885	10 s (0.15 ms)	560	11s (0.16 ms)	520
16	40	29 s (0.44 ms)	1225	16 s (0.24 ms)	800	17s (0.25 ms)	720
20	44	41 s (0.62 ms)	1527	28 s (0.42 ms)	1012	29s (0.44 ms)	968

MinimaxComp on HEAAN for various values of α. The proposed algorithm MinimaxComp reduces the runtime by approximately 45 percent on average when optimized for runtime and by 41 percent when optimized for depth consumption, as compared to the previous algorithm CompG.

In addition, Table 6 presents the ciphertext modulus bit consumption of CompG and MinimaxComp on HEAAN for various values of α. Let q_L and q_f be the moduli of the initial ciphertext and the final ciphertext, respectively, after the homomorphic comparison operation. Then, the ciphertext modulus bit consumption $\log(\frac{q_L}{q_f})$ measures the number of ciphertext bits lost during the operation. The data show that MinimaxComp reduces bit consumption by approximately 41 percent on average when the runtime is minimized and by 45 percent when the depth consumption is minimized, as compared to the previous algorithm CompG. It should also be noted that when the precision parameter α is 20, the previous algorithm does not achieve 128-bit security, while the proposed algorithm does due to its small depth consumption.

We note that MinimaxComp requires slightly more runtime when the depth consumption is minimized than when runtime is minimized. However, less ciphertext modulus bit consumption is required in the former case, resulting in less frequent bootstrapping and hence considerably reduced runtime for FHE applications that use bootstrapping.

5.3 Performance of the Proposed Homomorphic Max Function Algorithm MinimaxMax

We compare the performance of the proposed homomorphic max function MinimaxMax in Algorithm 8 with that of the previous algorithm MaxG (referred to as NewMaxG in [15]) using the scaling values w and margins η presented in Table 3. Table 7 presents the runtime (and amortized runtime) and the ciphertext modulus bit consumption of MaxG and MinimaxMax on HEAAN for various values of α. The data show that when we minimize the runtime, MinimaxMax reduces the runtime by approximately 41 percent on average and the ciphertext modulus bit consumption by approximately 33 percent on average, compared to the previous algorithm MaxG. Additionally, when we minimize the depth consumption, the runtime and bit consumption are reduced by 35 and 40 percent, respectively.

5.4 Performance of the Homomorphic Sorting Algorithm Using MinimaxMax

We compare the performance of homomorphic sorting algorithms for three and four numbers using the proposed max function algorithm MinimaxMax with the performance of homomorphic sorting algorithms using the previous algorithm MaxG. In these tests, we set $\alpha = 12$. As in the numerical analyses of the homomorphic comparison operation and max function, we set the initial ciphertext modulus q_L such that the final modulus bit length is $\log \Delta + 10$ after the completion of the sorting algorithm.

Table 8 presents the runtime (and amortized runtime) and the ciphertext modulus bit consumption of the sorting algorithm when we use MaxG from [15] or the proposed MinimaxMax on HEAAN. The data show that the sorting algorithm using MinimaxMax is approximately two times as fast as the sorting algorithm when MaxG is used. In addition, ciphertext modulus bit consumption is reduced by 41 percent. It should be noted that the sorting algorithm that uses the previous max function algorithm does not achieve

TABLE 8
Runtime (Amortized Runtime) and the Ciphertext Modulus Bit Consumption of Sorting When We Use `MaxG` (Cheon *et al.* [15]) and the Proposed `MinimaxMax` on HEAAN; an Asterisk (*) Indicates That the Parameter Set Does Not Achieve 128-Bit Security Because of the Large $\log q_L \geq 1700$

		MaxG	MinimaxMax (minimize depth)
sorting of three numbers	runtime	140 s* (2.13 ms)	76 s (1.15 ms)
	$\log(q_L/q_f)$	2655*	1560
sorting of four numbers	runtime	252 s* (3.84 ms)	136 s (2.07 ms)
	$\log(q_L/q_f)$	2655*	1560

128-bit security, whereas the sorting algorithm that uses the proposed max function algorithm does due to its small depth consumption.

6 Conclusion

We proposed the optimal method of approximating $\mathrm{sgn}(x)$ using minimax composite polynomials for use in the homomorphic comparison operation. Its principle is to determine the set of degrees for which the minimax composite polynomial is optimal with respect to the number of non-scalar multiplications and the depth consumption among all minimax composite polynomials that satisfy the comparison operation error condition. We proved that for any given composition of polynomials with odd-degree terms that satisfies the comparison operation error condition, there exists a minimax composite polynomial for some set of degrees that satisfies the error condition and requires a smaller or equal depth consumption and number of non-scalar multiplications. As a brute-force search requires considerable runtime for high precision approximations, we proposed polynomial-time algorithms that obtained the optimal minimax composite polynomials using dynamic programming.

Our numerical analysis demonstrated that if runtime was to be minimized, the proposed homomorphic comparison algorithm reduced runtime by approximately 45 percent on average compared to the previous algorithm when the HEAAN library was used. When depth consumption was to be minimized, the algorithm reduced runtime by approximately 41 percent on average. In addition, the proposed homomorphic max function algorithm reduced runtime by approximately 41 and 35 percent on average compared to the previous algorithm when runtime and depth consumption, respectively, were to be minimized. Finally, the homomorphic sorting algorithm using the proposed homomorphic max function algorithm was approximately two times as fast as the homomorphic sorting algorithm using the previous homomorphic max function algorithm.

Acknowledgments

This work was supported by the Samsung Research Funding and Incubation Center of Samsung Electronics under Project SRFC-IT1801-08. We would like to thank anonymous reviewers for their valuable suggestions and comments that helped improve the quality of this paper.

References

[1] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, 2019, pp. 169–178.
[2] J. Fan and F. Vercautern, "Somewhat practical fully homomorphic encryption," Cryptol. ePrint Arch., *Tech. Rep. 2012/144*, 2012. [Online]. Available: https://eprint.iacr.org/2012/144.
[3] J. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2017, pp. 409–437.
[4] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachene, "TFHE: Fast fully homomorphic encryption over the torus," *J. Cryptol.*, vol. 33, no. 1, pp. 34–91, 2020.
[5] R. Gilad-Bachrach , N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 201–210.
[6] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *Proc. 27th USENIX Secur. Symp.*, 2018, pp. 1651–1669.
[7] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[8] J. A. Hartigan and M. A. Wong, "A k-means clustering algorithm," *J. Roy. Statist. Soc. Series C Appl. Statist.*, vol. 28, no. 1, pp. 100–108, 1979.
[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Advances Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: https://arxiv.org/abs/1409.1556.
[11] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.
[12] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2818–2826.
[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[14] N. Emmadi, P. Gauravaram, H. Narumanchi, and H. Syed, "Updates on sorting of fully homomorphic encrypted data," in *Proc. Int. Conf. Cloud Comput. Res. Innov.*, 2015, pp. 19–24.
[15] J. Cheon, D. Kim, and D. Kim, "Efficient homomorphic comparison methods with optimal complexity," in *Proc. Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2020, pp. 221–256.
[16] C. Boura, N. Gama, and M. Georgieva, "Chimera: Combining ring-LWE-based fully homomorphic encryption schemes," *J. Math. Cryptol.*, vol. 14, no. 1, pp. 316–338, 2020.
[17] D. Chialva and A. Dooms, "Conditionals in homomorphic encryption and machine learning applications," Cryptol. ePrint Arch., Tech. Rep. 2018/1032, 2018. [Online]. Available: https://eprint.iacr.org/2018/1032
[18] J. Cheon, D. Kim, D. Kim, H. Lee, and K. Lee, "Numerical method for comparison on homomorphically encrypted numbers," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2019, pp. 415–445.
[19] R. E. Goldschmidt, "Applications of division by convergence," PhD dissertation, Dept. Elect. Eng., Massachusetts Inst. Technol., Cambridge, MA, 1964.
[20] E. W. Cheney, *Introduction to Approximation Theory*. Cambridge, U.K.: McGraw-Hill, 1966.
[21] J.-W. Lee, E. Lee, Y. Lee, Y.-S. Kim, and J.-S. No, "High-precision bootstrapping of RNS-CKKS homomorphic encryption using optimal minimax polynomial approximation and inverse sine function," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2021, pp. 618–647.
[22] Y. Lee, J.-W. Lee, Y.-S. Kim, and J.-S. No, "Near-optimal polynomial for modulus reduction using l2-norm for approximate homomorphic encryption," *IEEE Access*, vol. 8, pp. 144 321–144 330, 2020.
[23] E. Y. Remez, "Sur la determination des polynomes d'approximation de degre donnee," *Comm. Soc. Math. Kharkov*, vol. 10, no. 196, pp. 41–63, 1934.

[24] J.-W. Lee, E. Lee, Y. Lee, Y.-S. Kim, and J.-S. No, "Optimal minimax polynomial approximation of modular reduction for bootstrapping of approximate homomorphic encryption," Cryptol. ePrint Arch., Tech. Rep. 2020/552/20200803:084202, 2020.
[25] J. Bossuat, C. Mouchet, J. Troncoso-Pastoriza , and J. Hubaux, "Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2021, pp. 587–617.
[26] B. Li and D. Micciancio, "On the security of homomorphic encryption on approximate numbers," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2021, pp. 648–677.
[27] J. Cheon, S. Hong, and D. Kim, "Remark on the security of CKKS scheme in practice," Cryptol. ePrint Arch., Tech. Rep. 2020/1581, 2020.
[28] M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of learning with errors," *J. Math. Cryptol.*, vol. 9, no. 3, pp. 169–203, 2015.

Eunsang Lee received the BS and PhD degrees in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2014 and 2020, respectively. He is now a postdoctoral researcher with the Institute of New Media and Communications, Seoul National University. His research interests include homomorphic encryption and lattice-based cryptography.

Joon-Woo Lee (Graduate Student Member, IEEE) received the BS degree in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2016, where he is currently working toward the PhD degree. His research interests include homomorphic encryption and lattice-based cryptography.

Jong-Seon No (Fellow, IEEE) received the BS and MSEE degrees in electronics engineering from Seoul National University, Seoul, South Korea, in 1981 and 1984, respectively, and the PhD degree in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 1988. He was a senior MTS with Hughes Network Systems, from 1988 to 1990. He was an associate professor with the Department of Electronic Engineering, Konkuk University, Seoul, from 1990 to 1999. He joined the faculty of the Department of Electrical and Computer Engineering, Seoul National University, in 1999, where he is currently a professor. His research interests include error-correcting codes, cryptography, sequences, LDPC codes, interference alignment, and wireless communication systems. He became a member of the National Academy of Engineering of Korea (NAEK), in 2015, where he served as the Division Chair of Electrical, Electronic, and Information Engineering from 2019 to 2020. He was a recipient of IEEE Information Theory Society Chapter of the Year Award, in 2007. From 1996 to 2008, he served as the Founding Chair of the Seoul Chapter of IEEE Information Theory Society. He was the General Chair of Sequence and Their Applications 2004 (SETA 2004), Seoul. He also served as the General co-chair of the International Symposium on Information Theory and Its Applications 2006 (ISITA 2006) and the International Symposium on Information Theory, 2009 (ISIT 2009), Seoul. He served as the co-editor-in-chief for the IEEE Journal of Communications and Networks, from 2012 to 2013.

Young-Sik Kim (Member, IEEE) received the BS, MS, and PhD degrees in electrical engineering and computer science from Seoul National University, in 2001, 2003, and 2007, respectively. He joined the Semiconductor Division, Samsung Electronics, where he worked in the research and development of security hardware IPs for various embedded systems, including modular exponentiation hardware accelerator (called Tornado 2MX2) for RSA and elliptic-curve cryptography in smart-card products and mobile application processors of Samsung Electronics, until 2010. He is currently a professor with Chosun University, Gwangju, South Korea. He is also a Submitter for two candidate algorithms (McNie and pqsigRM) in the first round for the NIST Post Quantum Cryptography Standardization. His research interests include post-quantum cryptography, IoT security, physical-layer security, data hiding, channel coding, and signal design. He is selected as one of 2025's 100 Best Technology Leaders (for Crypto-Systems) by the National Academy of Engineering of Korea.