IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 14, 2021, accepted February 15, 2022, date of publication March 2, 2022, date of current version March 11, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3155882

Optimization of Homomorphic Comparison Algorithm on RNS-CKKS Scheme

EUNSANG LEE[1], JOON-WOO LEE[1], (Graduate Student Member, IEEE), YOUNG-SIK KIM[2], (Member, IEEE), AND JONG-SEON NO[1], (Fellow, IEEE)

[1]Department of Electrical and Computer Engineering, INMC, Seoul National University, Seoul 08826, South Korea
[2]Department of Information and Communication Engineering, Chosun University, Gwangju 61452, South Korea

Corresponding author: Young-Sik Kim (iamyskim@chosun.ac.kr)

This work was supported in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) Grant funded by the Ministry of Science, Information and Communication Technology (ICT) and Future Planning (MSIT) (Development of Highly Efficient post-quantum cryptography (PQC) Security and Performance Verification for Constrained Devices) (contribution: 80%) under Grant 2021-0-00400; and in part by the Brain Korea 21 Project (BK21) FOUR Program of the Education and Research Program for Future ICT Pioneers, Seoul National University, in 2022 (contribution: 20%).

ABSTRACT The sign function can be adopted to implement the comparison operation, max function, and rectified linear unit (ReLU) function in the Cheon–Kim–Kim–Song (CKKS) scheme; hence, several studies have been conducted to efficiently evaluate the sign function in the CKKS scheme. Recently, Lee *et al.* (IEEE Trans. Depend. Sec. Comp.) proposed a practically optimal approximation method for the sign function in the CKKS scheme using a composition of minimax approximate polynomials. In addition, Lee *et al.* proposed a polynomial-time algorithm that finds the degrees of component polynomials that minimize the number of non-scalar multiplications. However, homomorphic comparison/max/ReLU functions using Lee *et al.*'s approximation method have not been successfully implemented in the residue number system variant CKKS (RNS-CKKS) scheme. In addition, the degrees of component polynomials found by Lee *et al.*'s algorithm are not optimized for the RNS-CKKS scheme because the algorithm does not consider that the running time of non-scalar multiplication depends significantly on the ciphertext level in the RNS-CKKS scheme. In this study, we propose a fast algorithm for the inverse minimax approximation error, which is a subroutine required to find the optimal set of degrees of component polynomials. The proposed algorithm facilitates determining the optimal set of degrees of component polynomials with higher degrees than in the previous study. In addition, we propose a method to find the degrees of component polynomials optimized for the RNS-CKKS scheme using the proposed algorithm for the inverse minimax approximation error. We successfully implement the homomorphic comparison, max function, and ReLU function algorithms on the RNS-CKKS scheme with a low comparison failure rate ($< 2^{-15}$), and provide various parameter sets according to the precision parameter α. We reduce the depth consumption of the homomorphic comparison, max function, and ReLU function algorithms by one depth for several values of α. In addition, the numerical analysis demonstrates that the homomorphic comparison, max function, and ReLU function algorithms using the degrees of component polynomials found by the proposed algorithm reduce the running time by 6%, 7%, and 6% on average, respectively, compared with those using the degrees of component polynomials found by Lee *et al.*'s algorithm.

INDEX TERMS Cheon–Kim–Kim–Song (CKKS) scheme, fully homomorphic encryption (FHE), homomorphic comparison operation, minimax approximate polynomial, Remez algorithm, residue number system variant CKKS (RNS-CKKS) scheme.

I. INTRODUCTION

Homomorphic encryption (HE) is a cryptosystem that allows certain algebraic operations on encrypted data. A HE that allows all algebraic operations on encrypted data is known as fully homomorphic encryption (FHE). Gentry proposed the first FHE scheme using bootstrapping in [1]. FHE has garnered considerable attention in various applications, and its standardization process is in progress.

The associate editor coordinating the review of this manuscript and approving it for publication was Mehdi Sookhak.

The Cheon–Kim–Kim–Song (CKKS) [2] scheme, a representative FHE scheme, allows the addition and multiplication of real and complex numbers. Because data are usually represented by real numbers, the CKKS scheme, which can deal with real numbers, has garnered significant attention in several applications, such as machine learning [3]–[6]. Thus, several studies have been conducted to optimize the CKKS scheme [7]–[11]. Cheon *et al.* [7] proposed a residue number system variant CKKS scheme (RNS-CKKS). The running time of the RNS-CKKS scheme is ten times faster than that of the original CKKS scheme with one thread. In addition, the running time performance can be improved in a multicore environment because the RNS-CKKS scheme enables parallel computation. Thus, several HE libraries, such as SEAL [12], PALISADE [13], and Lattigo [14], are implemented using the RNS-CKKS scheme.

Although the CKKS scheme can virtually support all arithmetic operations on encrypted data, several applications require nonarithmetic operations. One of the core non-arithmetic operations is the comparison operation, denoted as $\mathrm{comp}(a, b)$, which outputs 1 if $a > b$, 1/2 if $a = b$, and 0 if $a < b$. This comparison operation is widely employed in various real-world applications, including machine learning algorithms, such as support vector machines, cluster analysis, and gradient boosting [15], [16]. The max function and rectified linear unit (ReLU) functions are other essential nonarithmetic operations that are widely adopted in deep learning applications [17], [18]. These three non-arithmetic operations can all be implemented using the sign function $\mathrm{sgn}(x)$; that is,

$$\begin{aligned}\mathrm{comp}(a, b) &= \frac{1}{2}(\mathrm{sgn}(a-b)+1),\\ \max(a, b) &= \frac{1}{2}(a+b+(a-b)\,\mathrm{sgn}(a-b)),\\ \mathrm{ReLU}(x) &= \frac{1}{2}(x+x\,\mathrm{sgn}(x)),\end{aligned}$$

where $\mathrm{sgn}(x) = x/|x|$ for $x \neq 0$ and 0 otherwise. Thus, several studies have been conducted to implement the sign function in the CKKS scheme efficiently [9], [19]. A method to approximate $\mathrm{sgn}(x)$ using the composition of component polynomials was proposed in [19], and it was proven that this method achieves optimal asymptotic complexity. In addition, the authors of [9] proposed a practically optimal method that approximates $\mathrm{sgn}(x)$ with the minimum number of non-scalar multiplications using a composition of minimax approximate polynomials.

Although the authors of [9] proposed a comparison operation algorithm with practically optimal performance on the CKKS scheme, there are several limitations in using the comparison operation for the RNS-CKKS scheme. First, because the rescaling error is relatively large in the RNS-CKKS scheme, unlike in the CKKS scheme, it is necessary to deal with this comparatively large rescaling error to achieve low approximation failure rates. Another issue is determining a set of degrees of component polynomials that provide better comparison operation performance. Although the authors of [9] also proposed a polynomial-time algorithm that determines the set of degrees that minimizes the number of non-scalar multiplications, this set of degrees is not optimized for the RNS-CKKS scheme, unlike the CKKS scheme. This is because the running time of non-scalar multiplication alters significantly with the current ciphertext level in the RNS-CKKS scheme. Thus, if we optimize the degrees of component polynomials by considering the running time of non-scalar multiplication according to the ciphertext level, the performance will be improved further.

A. OUR CONTRIBUTIONS

The contributions of this study are presented as follows.

1) For the first time, we successfully implement the homomorphic comparison, max function, and ReLU function algorithms using a composition of minimax approximate polynomials on the RNS-CKKS scheme with a low failure rate ($< 2^{-15}$), and provide proper parameter sets.
2) We improve the performance of an algorithm to determine the *inverse minimax approximation error*, which is a subroutine to determine the optimal set of degrees of component polynomials. In a previous study, the optimal set of degrees of component polynomials that minimizes the number of non-scalar multiplications was determined among degrees only up to 31 [9]; however, we determine the optimal set of degrees of component polynomials among degrees up to 63, using the improved algorithm for inverse minimax approximation error (see Algorithm 7). Consequently, the depth consumption of the homomorphic comparison operation (resp. max/ReLU function) is reduced by one depth when α is 9 or 14 (resp. when α is 16, 17, or 18), thereby enabling an additional multiplication operation. In addition, this improved algorithm for inverse minimax approximation error enables the identification of a set of degrees of component polynomials optimized for homomorphic comparison operation, max function, or ReLU function in the RNS-CKKS scheme (see Section IV). Our source code for determining the optimized degrees is available at `https://github.com/eslee3209/MinimaxComp_degrees`.
3) We propose a method to determine the set of degrees of component polynomials optimized for the homomorphic comparison, max function, and ReLU function on the RNS-CKKS scheme using the proposed fast algorithm for inverse minimax approximation error. Using the optimized set of degrees for the RNS-CKKS scheme, we reduce the running time of the homomorphic comparison, max function, and ReLU function algorithms by 6%, 7%, and 6%, respectively, compared with the previous work in [9] on the RNS-CKKS scheme library SEAL [12].

B. RELATED WORKS

Although it is not difficult to perform a comparison operation (or max/ReLU function) in bit-wise FHE, such as the fastest homomorphic encryption in the West (FHEW) [20] or fast fully homomorphic encryption over the torus (TFHE) [21], the comparison operation is very challenging in word-wise FHE, such as the CKKS scheme. Thus, several studies have been conducted on comparison operations in the CKKS scheme that adopts the evaluation of approximate polynomials [9], [19], [22]. Among them, the comparison operation proposed in [9] exhibits the best performance, and we improve the performance of [9].

Another comparison operation method for the CKKS scheme that uses FHEW/TFHE bootstrapping was recently studied [23]–[26]. Although this approach uses FHEW/TFHE bootstrapping, users can still employ efficient word-wise operations in the CKKS scheme. When a comparison operation is required, users switch the ciphertexts to FHEW/TFHE ciphertexts and perform a comparison operation using FHEW/TFHE bootstrapping. This comparison method that uses FHEW/TFHE bootstrapping can be less efficient than the proposed homomorphic comparison that fully uses CKKS packing in terms of the amortized running time (running time per comparison operation). However, this comparison method is still interesting research topic because this can have advantages in the case of large-precision comparison.

C. OUTLINE

The remainder of this paper is organized as follows. Section II describes the notations, RNS-CKKS scheme, scaling factor management technique, and homomorphic comparison operation using a minimax composite polynomial. In Section III, a fast algorithm for determining the inverse minimax approximation error is proposed. A new algorithm that determines the set of degrees of component polynomials optimized for the homomorphic comparison of the RNS-CKKS scheme is proposed in Section IV. The application of min/max and ReLU functions is presented in Section V. In Section VI, the numerical results for the homomorphic comparison, max function, and ReLU function algorithms that use the proposed set of degrees for the component polynomials are provided in the RNS-CKKS scheme library SEAL. Finally, concluding remarks are presented in Section VII.

II. PRELIMINARIES

A. NOTATION

Let $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ be polynomial rings, where N is a power-of-two integer. Let $\mathcal{C} = \{q_0, q_1, \cdots, q_{\ell-1}\}$ be the set of positive coprime integers. Then, for $a \in \mathbb{Z}_Q$, where $\mathbb{Z}_Q$ is the set of integers modulo Q and $Q = \prod_{i=0}^{\ell-1} q_i$, we denote the RNS representation of a with respect to $\mathcal{C}$ as $[a]_{\mathcal{C}} = ([a]_{q_0}, \cdots, [a]_{q_{\ell-1}}) \in \mathbb{Z}_{q_0} \times \cdots \times \mathbb{Z}_{q_{\ell-1}}$. For the set of real numbers $\mathbb{R}$ and set of complex numbers $\mathbb{C}$, the field isomorphism $\bar{\tau} : \mathbb{R}[X]/(X^N + 1) \to \mathbb{C}^{N/2}$ is defined as $\bar{\tau} : r(X) \mapsto (r(\bar{\zeta}^{5^j}))_{0 \le j < N/2}$, where $\bar{\zeta} = \exp(-\pi i/N)$ is the $(2N)$th root of unity in $\mathbb{C}$. $\mathcal{HWT}_N(h)$ is the set of signed binary vectors in $\{0, \pm 1\}^N$ with a Hamming weight h. For $0 < a, b \in \mathbb{R}$, we denote $[-b, -a] \cup [a, b]$ as $\tilde{R}_{a,b}$. In particular, if $a = 1 - \tau$ and $b = 1 + \tau$ for some $\tau \in (0, 1)$, then $\tilde{R}_{a,b} = \tilde{R}_{1-\tau,1+\tau}$ is denoted by R_τ. $|\{(n_1, n_2, \cdots, n_i); S(n_1, \cdots, n_i)\}|$ denotes the number of tuples $(n_1, \cdots, n_i)$, such that statement $S(n_1, \cdots, n_i)$ is true. $\alpha_{\max}, \ell_{\max}, m_{\max}, n_{\max}$, and $t_{\max}$ denote the upper bound of the precision α, ciphertext level, number of non-scalar multiplications, depth consumption, and running time, respectively. These values should be sufficiently large; thus, we set $\alpha_{\max} = 20$, $\ell_{\max} = 30$, $m_{\max} = 70$, $n_{\max} = 40$, and $t_{\max} = 240$ in this study. $d_{\max}$ denotes the upper bound of the degrees of the component polynomials, and $d_{\max}$ of 31 or 63 is used in this study.

B. RNS-CKKS SCHEME

Before describing the RNS-CKKS scheme, some basic operations for the RNS are presented. Let $\mathcal{B} = \{p_0, \cdots, p_{k-1}\}$, $\mathcal{C} = \{q_0, \cdots, q_{\ell-1}\}$, and $\mathcal{D} = \{p_0, \cdots, p_{k-1}, q_0, \cdots, q_{\ell-1}\}$, where p_i and q_j are the distinct primes.

- $\texttt{Conv}_{\mathcal{C}\to\mathcal{B}}$: For $[a]_{\mathcal{C}} = (a^{(0)}, a^{(1)}, \cdots, a^{(\ell-1)}) \in \mathbb{Z}_{q_0} \times \cdots \times \mathbb{Z}_{q_{\ell-1}}$, output

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

  where $\hat{q}_j = \prod_{j' \neq j} q_{j'} \in \mathbb{Z}$. This algorithm over integers $\texttt{Conv}_{\mathcal{C}\to\mathcal{B}}(\cdot) : \prod_{j=0}^{\ell-1} \mathbb{Z}_{q_j} \to \prod_{i=0}^{k-1} \mathbb{Z}_{p_i}$ can be extended to an algorithm over polynomial rings as $\texttt{Conv}_{\mathcal{C}\to\mathcal{B}}(\cdot) : \prod_{j=0}^{\ell-1} \mathcal{R}_{q_j} \to \prod_{i=0}^{k-1} \mathcal{R}_{p_i}$ by applying it coefficient-wise.
- $\texttt{ModUp}_{\mathcal{C}\to\mathcal{D}}$: For $[a]_{\mathcal{C}} \in \prod_{j=0}^{\ell-1} \mathcal{R}_{q_j}$, output

$$(\texttt{Conv}_{\mathcal{C}\to\mathcal{B}}([a]_{\mathcal{C}}), [a]_{\mathcal{C}}) \in \prod_{i=0}^{k-1} \mathcal{R}_{p_i} \times \prod_{j=0}^{\ell-1} \mathcal{R}_{q_j}.$$

- $\texttt{ModDown}_{\mathcal{D}\to\mathcal{C}}$: For $([a]_{\mathcal{B}}, [b]_{\mathcal{C}}) \in \prod_{i=0}^{k-1} \mathcal{R}_{p_i} \times \prod_{j=0}^{\ell-1} \mathcal{R}_{q_j}$, output

$$([b]_{\mathcal{C}} - \texttt{Conv}_{\mathcal{B}\to\mathcal{C}}([a]_{\mathcal{B}})) \cdot [P^{-1}]_{\mathcal{C}},$$

  where $P = \prod_{i=0}^{k-1} p_i$.

The basic algorithms in the RNS-CKKS scheme are described as follows:

- $\texttt{Setup}(\lambda; \Delta, L)$: For a security parameter λ, scaling factor Δ, and the number of levels L (also called the maximum level), we set some parameters. The polynomial degree N of $\mathcal{R}$ is chosen such that the number of levels L can be supported by security λ. A secret key distribution χ_{key}, error distribution χ_{err} over $\mathcal{R}$, and encryption key distribution χ_{enc} are chosen according to the security parameter λ. Bases with prime numbers $\mathcal{B} = \{p_0, p_1, \cdots, p_{k-1}\}$ and

$\mathcal{C} = \{q_0, q_1, \cdots, q_L\}$ are selected such that $p_i \equiv 1 \bmod 2N$ for $0 \le i \le k-1$ and $q_j \equiv 1 \bmod 2N$ for $0 \le j \le L$. q_0 is usually set close to 2^{60} and $q_j - \Delta$ is as small as possible for $1 \le j \le L$. All prime numbers are distinct. We assume that $\mathcal{D} = \mathcal{B} \cup \mathcal{C}$. Let $\mathcal{C}_\ell = \{q_0, q_1, \cdots, q_\ell\}$, and $\mathcal{D}_\ell = \mathcal{B} \cup \mathcal{C}_\ell$ for $0 \le \ell \le L$. Let $P = \prod_{i=0}^{k-1} p_i$ and $Q = \prod_{j=0}^{L} q_j$. Let $\hat{p}_i = \prod_{0 \le i' \le k-1, i' \ne i} p_{i'}$ for $0 \le i \le k-1$ and $\hat{q}_{\ell,j} = \prod_{0 \le j' \le \ell, j' \ne j} q_{j'}$ for $0 \le j \le \ell \le L$. The following numbers are then computed:

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

– $\mathtt{Ecd}(\mathbf{z}; \Delta)$: For $\mathbf{z} \in \mathbb{C}^{N/2}$, output $\lfloor \bar{\tau}^{-1}(\Delta \mathbf{z}) \rceil \in \mathcal{R}$.
– $\mathtt{Dcd}(m; \Delta)$: For $m \in \mathcal{R}$, output $\Delta^{-1} \cdot \bar{\tau}(m) \in \mathbb{C}^{N/2}$.
– $\mathtt{KSGen}(s_1, s_2)$: This algorithm generates the switching key for switching the secret key s_1 to s_2. First, sample $(a'^{(0)}, \cdots, a'^{(k+L)}) \leftarrow U(\prod_{i=0}^{k-1} \mathcal{R}_{p_i} \times \prod_{j=0}^{L} \mathcal{R}_{q_j})$ and $e' \leftarrow \chi_{\mathsf{err}}$. Then, for the given $s_1, s_2 \in \mathcal{R}$, output the switching key $\mathsf{swk} = (\mathsf{swk}^{(0)} = (b'^{(0)}, a'^{(0)}), \cdots, \mathsf{swk}^{(k+L)} = (b'^{(k+L)}, a'^{(k+L)})) \in \prod_{i=0}^{k-1} \mathcal{R}_{p_i}^2 \times \prod_{j=0}^{L} \mathcal{R}_{q_j}^2$, where $b'^{(i)} \leftarrow -a'^{(i)} \cdot s_2 + e' \bmod p_i$ for $0 \le i \le k-1$ and $b'^{(k+j)} \leftarrow -a'^{(k+j)} \cdot s_2 + [P]_{q_j} \cdot s_1 + e' \bmod q_j$ for $0 \le j \le L$.
– $\mathtt{KeyGen}(\lambda)$: This algorithm generates the secret, public, and evaluation keys. First, sample $s \leftarrow \chi_{\mathsf{key}}$ and set the secret key $\mathsf{sk} \leftarrow (1, s)$. Sample $(a^{(0)}, \cdots, a^{(L)}) \leftarrow U(\prod_{j=0}^{L} \mathcal{R}_{q_j})$ and $e \leftarrow \chi_{\mathsf{err}}$. Then, the public key is $\mathsf{pk} \leftarrow (\mathsf{pk}^{(j)} = (b^{(j)}, a^{(j)}) \in \mathcal{R}_{q_j}^2)_{0 \le j \le L}$, where $b^{(j)} \leftarrow -a^{(j)} \cdot s + e \bmod q_j$ for $0 \le j \le L$. The evaluation key is $\mathsf{evk} \leftarrow \mathtt{KSGen}(s^2, s)$.
– $\mathtt{Enc}(\mathbf{z}; \mathsf{pk}, \Delta)$: For a message slot $\mathbf{z} \in \mathbb{C}^{N/2}$, compute $m = \mathtt{Ecd}(\mathbf{z}; \Delta)$. Then, sample $v \leftarrow \chi_{\mathsf{enc}}$ and $e_0, e_1 \leftarrow \chi_{\mathsf{err}}$. The ciphertext is $\mathsf{ct} = (\mathsf{ct}^{(j)})_{0 \le j \le L} \in \prod_{j=0}^{L} \mathcal{R}_{q_j}^2$, where $\mathsf{ct}^{(j)} \leftarrow v \cdot \mathsf{pk}^{(j)} + (m + e_0, e_1) \bmod q_j$ for $0 \le j \le L$.
– $\mathtt{Dec}(\mathsf{ct}; \mathsf{sk}, \Delta)$: For a ciphertext $\mathsf{ct} = (\mathsf{ct}^{(j)})_{0 \le j \le \ell} \in \prod_{j=0}^{\ell} \mathcal{R}_{q_j}^2$, obtain $\tilde{m} = \langle \mathsf{ct}^{(0)}, \mathsf{sk} \rangle \bmod q_0$. Then, output $\mathbf{z} = \mathtt{Dcd}(\tilde{m}; \Delta)$.
– $\mathtt{Add}(\mathsf{ct}_1, \mathsf{ct}_2)$: For two ciphertexts $\mathsf{ct}_r = (\mathsf{ct}_r^{(j)})_{0 \le j \le \ell}$ for $r = 1, 2$, output the ciphertext $\mathsf{ct}_{\mathsf{add}} = (\mathsf{ct}_{\mathsf{add}}^{(j)})_{0 \le j \le \ell}$, where $\mathsf{ct}_{\mathsf{add}}^{(j)} \leftarrow \mathsf{ct}_1^{(j)} + \mathsf{ct}_2^{(j)} \bmod q_j$ for $0 \le j \le \ell$.
– $\mathtt{Mult}(\mathsf{ct}_1, \mathsf{ct}_2; \mathsf{evk})$: For two ciphertexts $\mathsf{ct}_r = (\mathsf{ct}_r^{(j)} = (c_{r0}^{(j)}, c_{r1}^{(j)}))_{0 \le j \le \ell}$ for $r = 1, 2$, compute the following:
  - $d_0^{(j)} = c_{00}^{(j)} c_{10}^{(j)} \bmod q_j$, $d_1^{(j)} = c_{00}^{(j)} c_{11}^{(j)} + c_{01}^{(j)} c_{10}^{(j)} \bmod q_j$, and $d_2^{(j)} = c_{01}^{(j)} c_{11}^{(j)} \bmod q_j$ for $0 \le j \le \ell$.
  - $\mathtt{ModUp}_{C_\ell \to D_\ell}(d_2^{(0)}, \cdots, d_2^{(\ell)}) = (\tilde{d}_2^{(0)}, \cdots, \tilde{d}_2^{(k-1)}, d_2^{(0)}, \cdots, d_2^{(\ell)})$.
  - $\tilde{ct} = (\tilde{ct}^{(k+\ell)} = (\tilde{c}_0^{(j)}, \tilde{c}_1^{(j)}))_{0 \le j \le k+\ell}$, where $\tilde{ct}^{(i)} = \tilde{d}_2^{(i)} \cdot \mathsf{evk}^{(i)} \bmod p_i$ and $\tilde{ct}^{(k+j)} = d_2^{(j)} \cdot \mathsf{evk}^{(k+j)} \bmod q_j$ for $0 \le i \le k-1$ and $0 \le j \le \ell$.
  - $(\hat{c}_r^{(0)}, \cdots, \hat{c}_r^{(\ell)}) = \mathtt{ModDown}_{D_\ell \to C_\ell}(\tilde{c}_r^{(0)}, \cdots, \tilde{c}_r^{(k+\ell)})$ for $r = 0, 1$.
  - $\mathsf{ct}_{\mathsf{mult}} = (\mathsf{ct}_{\mathsf{mult}}^{(j)})_{0 \le j \le \ell}$, where $\mathsf{ct}_{\mathsf{mult}}^{(j)} = (\hat{c}_0^{(j)} + d_0^{(j)}, \hat{c}_1^{(j)} + d_1^{(j)}) \bmod q_j$ for $0 \le j \le \ell$.

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

In this study, we set the key distribution $\chi_{\mathsf{key}} = \mathcal{HWT}_N(256)$, which samples an element in $\mathcal{R}$ with ternary coefficients that have 256 nonzero values uniformly at random.

C. SCALING FACTOR MANAGEMENT

A technique for eliminating the large rescaling error in the RNS-CKKS scheme was proposed in [27], where different scaling factors at different levels were utilized instead of the same scaling factor for each level. If the maximum level is L and the ciphertext modulus for level i is q_i, then the scaling factor for each level is set as follows: $\Delta_L = q_L$ and $\Delta_i = \Delta_{i+1}^2 / q_{i+1}$ for $i = 0, \cdots, L-1$.

When two ciphertexts at the same level are multiplied homomorphically, an approximate rescaling error is not introduced. Then, we consider when two ciphertexts are at different levels, levels i and j, such that $i > j$. In this case, the moduli $q_i, q_{i-1}, \cdots, q_{j+1}$ in the first ciphertext are dropped, and the first ciphertext is then multiplied by a constant $\lfloor \frac{\Delta_j q_{j+1}}{\Delta_i} \rceil$. Subsequently, we rescale the first ciphertext using q_{j+1}. Because both ciphertexts are now at the same level, conventional homomorphic multiplication can be performed. The approximate rescaling error decreased in this manner.

D. HOMOMORPHIC COMPARISON OPERATION USING MINIMAX COMPOSITE POLYNOMIAL

In this study, the required depth consumption and number of non-scalar multiplications for evaluating a polynomial of degree d with odd-degree terms using the odd baby-step giant-step algorithm and optimal level consumption technique are denoted by $\mathsf{dep}(d)$ and $\mathsf{mult}(d)$, respectively. The values of $\mathsf{dep}(d)$ and $\mathsf{mult}(d)$ for odd degrees d of up to 63 are presented in Table 1.

The minimax approximate polynomial of degree at most d on D for $\mathrm{sgn}(x)$ is denoted by $\mathsf{MP}(D; d)$. In addition, for the minimax approximate polynomial $p(x) = \mathsf{MP}(D; d)$, the minimax approximation error $\max_D \|p(x) - \mathrm{sgn}(x)\|_\infty$ is denoted by $\mathsf{ME}(D; d)$. It is known that for any continuous function f on D, the minimax approximate polynomial of degree d at most on D is unique [29]. Furthermore, the minimax approximate polynomial can be obtained using the improved multi-interval Remez algorithm [30].

For a domain $D = \tilde{R}_{a,b}$ and set of odd integers $\{d_i\}_{1 \le i \le k}$, a composite polynomial $p_k \circ \cdots \circ p_1$ is called a *minimax composite polynomial* on D for $\{d_i\}_{1 \le i \le k}$, denoted by $\mathsf{MCP}(D; \{d_i\}_{1 \le i \le k})$ if the followings are satisfied:

- $p_1 = \mathsf{MP}(D; d_1)$

TABLE 1. Required depth consumption and the number of non-scalar multiplications for evaluating polynomials of degree d with odd-degree terms using the optimal level consumption technique [10] and the odd baby-step giant-step algorithm [28].

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
33	6	11
35	6	11
37	6	11
39	6	11
41	6	12
43	6	12
45	6	13
47	6	13
49	6	14
51	6	14
53	6	14
55	6	14
57	6	15
59	6	15
61	6	16
63	6	17

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

Because $\mathsf{ME}(R_\tau; d)$ is a strictly increasing function of τ, its inverse function exists, which is called the *inverse minimax approximation error*, and is denoted by $\mathsf{IME}(\tau'; d)$. Thus, for $\tau \in (0, 1)$ and $d \in \mathbb{N}$, $\mathsf{IME}(\tau'; d)$ is equal to the value $\tau' \in (0, 1)$ that satisfies $\mathsf{ME}(R_{\tau'}; d) = \tau$. An approximate value of $\mathsf{IME}(\tau'; d)$ can be obtained using binary search, as indicated in Algorithm 1 [9].

The comparison operation is denoted as

$$\mathrm{comp}(a, b) = \begin{cases} 1 & \text{if } a > b \\ 1/2 & \text{if } a = b \\ 0 & \text{if } a < b. \end{cases}$$

The procedure for obtaining an approximate value of $\mathrm{comp}(a, b)$ for the given precision parameters α, ϵ and inputs $a, b \in [0, 1]$ is summarized as follows:

1) Obtain the minimum depth consumption M_{dep} from `ComputeMinDep` algorithm in Algorithm 3.
2) Choose depth consumption $D(\ge M_{\mathrm{dep}})$ and obtain the optimal set of degrees M_{degs} from `ComputeMinMultDegs` algorithm in Algorithm 4.
3) For some appropriate margin η, perform the homomorphic comparison algorithm `MinimaxComp` in Algorithm 5 using M_{degs}.

Algorithm 1: `AppIMEbinary`$(\tau'; d, \bar{i})$ [9]

Input: Target maximum error τ', an odd degree d, and iteration number $\bar{i}$
Output: An approximate value of $\mathsf{IME}(\tau'; d)$

$\mathsf{min} \leftarrow 2^{-21}$ and $\mathsf{max} \leftarrow 1 - 2^{-21}$
while $\bar{i} > 0$ **do**
  if $\mathsf{ME}(R_{(\mathsf{min}+\mathsf{max})/2}; d) < \tau'$ **then**
    $\mathsf{min} \leftarrow \frac{\mathsf{min}+\mathsf{max}}{2}$
  else
    $\mathsf{max} \leftarrow \frac{\mathsf{min}+\mathsf{max}}{2}$
  end
  $\bar{i} \leftarrow \bar{i} - 1$
end
return $\frac{\mathsf{min}+\mathsf{max}}{2}$

Algorithm 2: `ComputehG`(τ) [9]

Input: An input τ and an odd maximum degree $d_{\max}$
Output: 2-dimensional tables $\tilde{h}$ and $\tilde{G}$

Generate 2-dimensional tables $\tilde{h}$ and $\tilde{G}$, both of which have size of $(m_{\max} + 1) \times (n_{\max} + 1)$.
for $m \leftarrow 0$ **to** $m_{\max}$ **do**
  for $n \leftarrow 0$ **to** $n_{\max}$ **do**
    if $m \le 1$ or $n \le 1$ **then**
      $\tilde{h}(m, n) \leftarrow \tau$
      $\tilde{G}(m, n) \leftarrow \phi$
    else
      $j \leftarrow \underset{\substack{1 \le i \le \frac{d_{\max}-1}{2} \\ \mathsf{mult}(2i+1) \le m \\ \mathsf{dep}(2i+1) \le n}}{\mathrm{argmax}} \mathsf{IME}(\tilde{h}(m - \mathsf{mult}(2i + 1), n - \mathsf{dep}(2i + 1)); 2i + 1)$
      $\tilde{h}(m, n) \leftarrow \mathsf{IME}(\tilde{h}(m - \mathsf{mult}(2j + 1), n - \mathsf{dep}(2j + 1)); 2j + 1)$
      $\tilde{G}(m, n) \leftarrow \{2j+1\} \cup \tilde{G}(m - \mathsf{mult}(2j+1), n - \mathsf{dep}(2j+1))$
    end
  end
end

`ComputeMinDep` and `ComputeMinMultDegs` algorithms use `ComputehG` algorithm as a subroutine, and `MinimaxComp` algorithm uses `ComputeMinMultDegs` algorithm as a subroutine. Subsequently, the output of the `MinimaxComp` algorithm, $\tilde{p}(a, b) = \frac{p_k \circ p_{k-1} \circ \cdots \circ p_1(a-b)+1}{2}$ satisfies the following comparison operation error condition.

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

The set of degrees, $M_{\mathrm{degs}} = \{d_1, \cdots, d_k\}$, obtained from the `ComputeMinMultDegs` algorithm satisfies $\deg(p_i) = d_i, 1 \le i \le k$. M_{degs} is the optimal set of degrees,

Algorithm 3: `ComputeMinDep`(α, ϵ) [9]

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 4: `ComputeMinMultDegs`(α, ϵ, D) [9]

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

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

Algorithm 5: `MinimaxComp`$(a, b; \alpha, \epsilon, D, \eta)$ [9]

Input: Inputs $a, b \in (0, 1)$, precision parameters α and ϵ, depth consumption D, and margin η
Output: Approximate value of comp(a, b)

$M_{\text{degs}} = \{d_1, d_2, \cdots, d_k\} \leftarrow$ `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 $\frac{p_k \circ p_{k-1} \circ \cdots \circ p_1(a-b)+1}{2}$

such that the homomorphic comparison operation minimizes the number of non-scalar multiplications and satisfies the comparison operation error condition in (1) for the given depth consumption D.

III. FAST ALGORITHM FOR INVERSE MINIMAX APPROXIMATION ERROR

The `ComputehG` algorithm in Algorithm 2 [9] should be performed to obtain the optimal set of degrees from the `ComputeMinMultDegs` algorithm. To apply the `ComputehG` algorithm, the *inverse minimax approximation error* $\mathsf{IME}(\tau'; d)$ should be computed many times. That is, the `AppIMEbinary` algorithm [9] determining an approximate value of $\mathsf{IME}(\tau'; d)$ should be called many times. However, a single call of the `AppIMEbinary` algorithm also requires multiple computations of $\mathsf{ME}(R_\tau; d)$, that is, several calls for the improved multi-interval Remez algorithm [30]. Accordingly, the `ComputehG` algorithm requires a significant running time. Specifically, the number of computations for $\mathsf{IME}(\tau'; d)$ in `ComputehG` for each precision parameter α is provided as follows:

$$\begin{aligned}
&\sum_{m=0}^{m_{\max}} \sum_{n=0}^{n_{\max}} \left| \left\{ i;\, \mathsf{mult}(2i+1) \leq m,\, \mathsf{dep}(2i+1) \leq n,\right.\right. \\
&\qquad \left.\left. 1 \leq i \leq \frac{d_{\max}-1}{2} \right\} \right| \\
&= \left| \left\{ (m, n, i);\, 0 \leq m \leq m_{\max},\, 0 \leq n \leq n_{\max},\right.\right. \\
&\qquad \mathsf{mult}(2i+1) \leq m,\, \mathsf{dep}(2i+1) \leq n, \\
&\qquad \left.\left. 1 \leq i \leq \frac{d_{\max}-1}{2} \right\} \right| \\
&= \sum_{i=1}^{\frac{d_{\max}-1}{2}} \left| \left\{ (m, n);\, \mathsf{mult}(2i+1) \leq m \leq m_{\max},\right.\right. \\
&\qquad \left.\left. \mathsf{dep}(2i+1) \leq n \leq n_{\max} \right\} \right| \\
&= \sum_{i=1}^{\frac{d_{\max}-1}{2}} (m_{\max} - \mathsf{mult}(2i+1) + 1)(n_{\max} - \mathsf{dep}(2i+1) + 1) \\
&= \frac{d_{\max}-1}{2}(m_{\max}+1)(n_{\max}+1) \\
&\qquad -(m_{\max}+1) \sum_{i=1}^{\frac{d_{\max}-1}{2}} \mathsf{dep}(2i+1) \\
&\qquad -(n_{\max}+1) \sum_{i=1}^{\frac{d_{\max}-1}{2}} \mathsf{mult}(2i+1) \\
&\qquad + \sum_{i=1}^{\frac{d_{\max}-1}{2}} \mathsf{mult}(2i+1)\mathsf{dep}(2i+1).
\end{aligned}$$

If we set $d_{\max} = 31$, $m_{\max} = 70$, and $n_{\max} = 40$, as in [9], the number of computations for $\mathsf{IME}(\tau'; d)$ is

$$\begin{aligned}
&15(m_{\max}+1)(n_{\max}+1) - (m_{\max}+1) \cdot 64 \\
&\qquad -(n_{\max}+1) \cdot 113 + 64 \cdot 113 \\
&= 15 \cdot 71 \cdot 41 - 71 \cdot 64 - 41 \cdot 113 + 64 \cdot 113
\end{aligned}$$

$= 41,720$.

If we want to adopt $d_{\max} = 63$, the number of computations for $\mathsf{IME}(\tau'; d)$ is 117, 675.

To obtain a precise approximate value of $\mathsf{IME}(\tau'; d)$ using the `AppIMEbinary` algorithm, we iterate at least ten times. Then, the expected number of computations for $\mathsf{ME}(R_\tau; d)$ in `ComputehG` is at least $117,675 \times 10 = 1,176,750$. Notably, this is the case for only one value of the precision parameter α, where the input τ of the `ComputehG` algorithm is $2^{1-\alpha}$. To perform the `ComputehG` algorithm for α ranging from 4 to 20, approximately $1,176,750 \times 17 = 20,004,750$ calls for $\mathsf{ME}(R_\tau; d)$ are required. Because of the large number of calls for $\mathsf{ME}(R_\tau; d)$, a value of $d_{\max}$ larger than 31 could not be used in [9], failing to improve the performance of homomorphic comparison operations using higher degrees. Thus, it is desirable to study how to efficiently determine the approximate value of $\mathsf{IME}(\tau'; d)$.

A. PROPOSED ALGORITHM FOR INVERSE MINIMAX APPROXIMATION ERROR

We propose a fast method to determine the approximate value of $\mathsf{IME}(\tau'; d)$, which enables the use of a value of $d_{\max}$ larger than 31. The procedure for the proposed method is as follows.

1) Sample the values of τ at moderate intervals.
2) Compute the values of $\mathsf{ME}(R_\tau; d)$ for the sampled τ.
3) For $\tau' \in (0, 1)$, obtain an approximate value of $\mathsf{IME}(\tau'; d)$ by interpolation using the computed sample values of $\mathsf{ME}(R_\tau; d)$.

For $\alpha_{\max}$, which is the upper-bound of α, we consider sampling τ between $2^{-\alpha_{\max}-1}$ and $1-2^{-\alpha_{\max}-1}$. If τ is close to zero or one, sampling should be very dense; however, densely sampling the entire range between $2^{-\alpha_{\max}-1}$ and $1-2^{-\alpha_{\max}-1}$ requires a large number of samples. Thus, we propose to sample densely when τ is close to zero or one, and sparsely otherwise.

Specifically, we first sample t uniformly between $-\alpha_{\max}-1$ and -1. Furthermore, we compute $\mathsf{ME}(R_{2^t}; d)$ for the sampled values of t. In addition, we sample t uniformly between 1 and $\alpha_{\max}+1$ and compute $\mathsf{ME}(R_{1-2^{-t}}; d)$ for the sampled values of t. By interpolating these samples, we can achieve a precise approximation of $\mathsf{IME}(\tau'; d)$ using a smaller number of samples. Precision $\bar{n}$ determines how frequently the values of t are sampled, and we set $\bar{n} = 10$. For a given maximum degree, $d_{\max}$, and precision, $\bar{n}$, the `StoreME` algorithm in Algorithm 6 stores 2^t (resp. $1-2^{-t}$) in a two-dimensional table $\tilde{X}$, and $\mathsf{ME}(R_{2^t}; d)$ (resp. $\mathsf{ME}(R_{1-2^{-t}}; d)$) in a two-dimensional table $\tilde{Y}$ for all degrees $d, 3 \le d \le d_{\max}$. For $\bar{n} = 10$ and $\alpha_{\max} = 20$, the number of calls for $\mathsf{ME}(R_\tau; d)$ is 6, 030 for $d_{\max} = 31$ and 12, 462 for $d_{\max} = 63$.

The `AppIME` algorithm in Algorithm 7 outputs an approximate value of $\mathsf{IME}(\tau'; d)$ using tables $\tilde{X}$ and $\tilde{Y}$ obtained from the `StoreME` algorithm. Here, several calls for the `AppIME` algorithm require only one computation of tables $\tilde{X}$ and $\tilde{Y}$, that is, one execution of the `StoreME` algorithm. Thus, `StoreME` is performed only once for various precision parameters, α.

Algorithm 6: `StoreME`$(d_{\max}, \bar{n})$

Input: Maximum degree $d_{\max}$ and precision $\bar{n}$
Output: 2-dimensional tables $\tilde{X}$ and $\tilde{Y}$

Generate 2-dimensional tables $\tilde{X}$ and $\tilde{Y}$, both of which have size of $\frac{d_{\max}-1}{2} \times 2(\bar{n}\alpha_{\max}+1)$
for $i \leftarrow 0$ **to** $\frac{d_{\max}-3}{2}$ **do**
  for $j \leftarrow 0$ **to** $\bar{n}\alpha_{\max}$ **do**
    $\tau \leftarrow 2^{-\alpha_{\max}-1+j/\bar{n}}$
    $\tilde{X}(i,j) \leftarrow \tau$
    $\tilde{Y}(i,j) \leftarrow \mathsf{ME}(R_\tau; 2i+3)$
  end
  for $j \leftarrow 0$ **to** $\bar{n}\alpha_{\max}$ **do**
    $\tau \leftarrow 1 - 2^{-(1+j/\bar{n})}$
    $\tilde{X}(i, j+\bar{n}\alpha_{\max}+1) \leftarrow \tau$
    $\tilde{Y}(i, j+\bar{n}\alpha_{\max}+1) \leftarrow \mathsf{ME}(R_\tau; 2i+3)$
  end
end
return $\tilde{X}$ and $\tilde{Y}$

B. RUNNING TIME OF THE PROPOSED ALGORITHM

We compare the running time of the `ComputehG` algorithm using the previous algorithm for the inverse minimax approximation error with that using the proposed algorithm. Numerical analysis is conducted on a Linux PC with an Intel Core i7-10700 CPU at 2.90 GHz with one thread. One call for the improved multi-interval Remez algorithm takes approximately 1.4 s on average when $d_{\max} = 31$, and 4.9 s on average when $d_{\max} = 63$. The expected running time of `ComputehG` can then be obtained. Table 2 presents the expected running time of the `ComputehG` algorithm for α in the range 4–20 using the previous and proposed algorithms for the inverse minimax approximation error. As presented in Table 2, the proposed `AppIME` algorithm requires significantly less running time than the previous `AppIMEbinary` algorithm, which enables the execution of the `ComputehG` algorithm for $d_{\max} = 63$. Although the running time of 17 h might still seem to be large, it should be noted that this process only needs to be performed once because its goal is to determine the optimal set of degrees.

Whereas the `ComputehG` algorithm in Algorithm 2 can only be performed for $d_{\max} \le 31$ in [9], we apply the `ComputehG` algorithm for $d_{\max} \le 63$ using the proposed fast `AppIME` algorithm in Algorithm 7. Table 3 presents the optimal sets of degrees, M_{degs}, and the corresponding minimum depth consumption, $D_{\min}$, for $d_{\max} = 31$ and $d_{\max} = 63$. From Table 3, it can be observed that depth consumption is reduced by one when α is nine or 14. That is, for $\alpha = 9$ or $\alpha = 14$, a high $d_{\max}$ enables one more non-scalar multiplication per homomorphic comparison operation in the FHE setting, where the available number of operations is very limited per bootstrapping. Furthermore, the proposed

Algorithm 7: $\texttt{AppIME}(\tau, d; d_{\max}, \bar{n})$

Input: Target maximum error τ, degree d, the odd maximum degree $d_{\max}$, and precision $\bar{n}$
Output: An approximate value of $\mathsf{IME}(\tau; d)$

$\tilde{X}, \tilde{Y} \leftarrow \texttt{StoreME}(d_{\max}, \bar{n})$
for $j \leftarrow 0$ **to** $2\bar{n}\alpha_{\max} + 1$ **do**
  $x_j \leftarrow \tilde{X}(\frac{d-3}{2}, j)$
  $y_j \leftarrow \tilde{Y}(\frac{d-3}{2}, j)$
end
for $j \leftarrow 1$ **to** $2\bar{n}\alpha_{\max} + 1$ **do**
  if $y_j \geq \tau$ **then**
    if $x_{j-1} < 0.5$ and $x_j \geq 0.5$ **then**
      return x_j
    else if $y_{j-1} < 0.5$ and $y_j \geq 0.5$ **then**
      return x_j
    else if $x_j < 0.5$ and $y_j < 0.5$ **then**
      $p \leftarrow \log x_{j-1} + (\tau - \log y_{j-1}) \frac{\log x_j - \log x_{j-1}}{\log y_j - \log y_{j-1}}$
      return 2^p
    else if $x_{j-1} \geq 0.5$ and $y_j < 0.5$ **then**
      $p \leftarrow -\log(1 - x_{j-1})$
      $-(\tau - \log y_{j-1}) \frac{\log(1-x_j) - \log(1-x_{j-1})}{\log y_j - \log y_{j-1}}$
      return $1 - 2^{-p}$
    else if $y_{j-1} \geq 0.5$ **then**
      $p \leftarrow -\log(1 - x_{j-1})$
      $+(\tau + \log(1 - y_{j-1})) \frac{\log(1-x_j) - \log(1-x_{j-1})}{\log(1-y_j) - \log(1-y_{j-1})}$
      return $1 - 2^{-p}$
    else
      return $\perp$
  end
end
return $\perp$

TABLE 2. Expected running time of $\texttt{ComputehG}$ algorithm for α in the range 4–20 using the previous and proposed algorithms for inverse minimax approximation error.

	$d_{\max}$	previous method using AppIMEbinary	proposed method using AppIME
expected number of calls for $\mathsf{ME}(R_\tau; d)$	31	7,092,400	6,030
	63	20,004,750	12,462
expected running time of ComputehG for α from 4 to 20	31	2,758 hours	2 hours
	63	27,228 hours	17 hours

AppIME algorithm enables the identification of a set of degrees optimized for the RNS-CKKS scheme, as described in Section IV.

IV. DETERMINING DEGREES OF COMPONENT POLYNOMIALS OPTIMIZED FOR THE RNS-CKKS SCHEME

Unlike the previous study on the homomorphic comparison operation in the CKKS scheme [9], we study the homomorphic comparison operation in the RNS-CKKS scheme as well, and there are additional aspects to be considered. Unlike the CKKS scheme, the RNS-CKKS scheme has a relatively large rescaling error with the risk of a high failure rate in the homomorphic comparison operation using a minimax composite polynomial [9]. We replace all additions and multiplications required for polynomial evaluation with additions and multiplications that use the scaling factor management technique. It can be observed in Section VI that a low failure rate is achieved using this technique and appropriate parameter sets.

TABLE 3. Optimal sets of degrees M_{degs} and corresponding minimum depth consumption for homomorphic comparison operation for $d_{\max} = 31$ and $d_{\max} = 63$. $D_{\min}$ denotes the minimum depth consumption for the homomorphic comparison operation.

α	the optimal set of degrees from $\texttt{ComputeMinMultDegs}(\alpha, 2^{-\alpha}, D_{\min})$			
	maximum degree $d_{\max} = 31$		maximum degree $d_{\max} = 63$	
	degrees	depth	degrees	depth
4	{27}	5	{27}	5
5	{7,13}	7	{7,13}	7
6	{15,15}	8	{15,15}	8
7	{7,7,13}	10	{7,7,13}	10
8	{7,15,15}	11	{7,15,15}	11
9	{7,7,7,13}	13	{55,55}	**12**
10	{7,7,13,15}	14	{7,7,13,15}	14
11	{7,15,15,15}	15	{7,15,15,15}	15
12	{15,15,15,15}	16	{15,15,15,15}	16
13	{15,15,15,31}	17	{15,15,15,31}	17
14	{7,7,15,15,27}	19	{55,59,59}	**18**

Another difference exists between the homomorphic comparison operation in the CKKS scheme and that in the RNS-CKKS scheme. For a given depth consumption D, the set of degrees M_{degs} that minimizes the number of non-scalar multiplications can be obtained using the $\texttt{ComputeMinMultDegs}$ algorithm. Because the computation time of a non-scalar multiplication does not depend significantly on the current ciphertext modulus in the CKKS scheme, minimizing the number of non-scalar multiplications corresponds to minimizing the running time. However, because the computation time of non-scalar multiplication depends significantly on the current level in the RNS-CKKS scheme, minimizing the number of non-scalar multiplications does not always correspond to minimizing the running time.

Fig. 1 illustrates the computation time of an example polynomial of degree seven, according to the current level on the RNS-CKKS scheme library SEAL [12]. From Fig. 1, it can be observed that the computation time of a polynomial tends to increase quadratically according to the maximum level. For example, we consider two ordered sets $M_{\text{degs}} = \{7, 7, 31, 31\}$ and $M_{\text{degs}} = \{31, 31, 7, 7\}$. In the CKKS scheme, the computation time of the homomorphic comparison operation using two sets of degrees will be approximately the same. However, the homomorphic comparison operation using the former is faster in the RNS-CKKS scheme because a high degree polynomial is computed at a lower level. Our core idea is to determine the set of degrees that minimizes the running time rather than the number of non-scalar multiplications. Specifically, we modify the previous $\texttt{ComputehG}$ and $\texttt{ComputeMinMultDegs}$ algorithms so that the modified algorithms can determine the set of degrees M_{degs} that minimizes the running time.

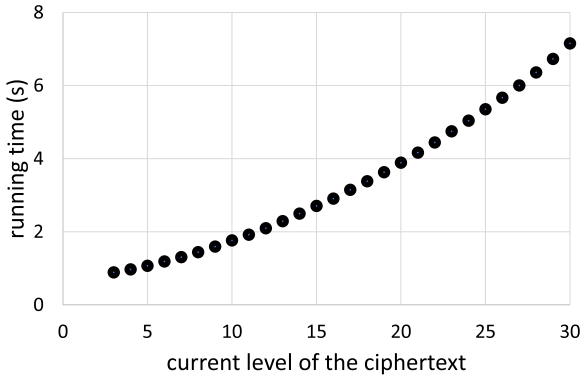

FIGURE 1. Running time of a polynomial of degree seven according to the current level of the ciphertext in the RNS-CKKS scheme library SEAL [12].

Here, we define $C_\ell(i, \ell')$ for $0 \le i \le \frac{d_{\max}-3}{2}$, $0 \le \ell \le \ell_{\max}$, $0 \le \ell' \le \ell_{\max}$. First, we set the maximum level to ℓ. Then, starting from level $\ell'(\le \ell)$, any polynomial of degree $2i+3$ is evaluated using the optimal level consumption technique [10] and an odd baby-step giant-step algorithm [28]. If t is the running time of the polynomial evaluation in milliseconds, we define $C_\ell(i, \ell')$ as $C_\ell(i, \ell') = \lfloor \frac{t}{100} \rceil$. Here, if $\ell < \ell'$ or $\ell' < \lceil \log_2(2i+3) \rceil$, polynomial evaluation is infeasible; thus, $C_\ell(i, \ell')$ is set to a sufficiently large value of 100, 000 in this case. We obtain the values of $C_\ell(i, \ell')$ by performing polynomial evaluation on encrypted data. This computation is done on Intel Core i7-10700 CPU at 2.90 GHz in single thread with an Ubuntu 20.04 LTS distribution. Subsequently, $u_{\tau,L}(m, n)$ and $V_{\tau,L}(m, n)$ are defined recursively using the values of $C_\ell(i, \ell')$ as follows:

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

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

where

$$j_{m,n} = \underset{\substack{1 \le i \\ C_L(i-1,n) \le m \\ \mathsf{dep}(2i+1) \le n}}{\operatorname{argmax}} \mathsf{IME}(u_{\tau,L}(m - C_L(i-1, n), \\ n - \mathsf{dep}(2i+1)); 2i+1).$$

$u_{\tau,L}(m, n)$ implies the maximum value of $\tau' \in (0, 1)$, such that there exists a set of degrees $\{d_i\}_{1 \le i \le k}$ that satisfy the following:

$$\mathsf{MCE}(R_{\tau'}; \{d_i\}_{1 \le i \le k}) \le \tau$$
$$\sum_{i=1}^{k} \mathsf{dep}(d_i) \le n$$

Algorithm 8: `ComputeuV`$(\tau; L)$

Input: τ, maximum level L
Output: 2-dimensional tables $\tilde{u}$ and $\tilde{V}$

Generate 2-dimensional tables $\tilde{u}$ and $\tilde{V}$, both of which have size of $(t_{\max}+1) \times (n_{\max}+1)$
for $m \leftarrow 0$ **to** $t_{\max}$ **do**
  for $n \leftarrow 0$ **to** $n_{\max}$ **do**
    if $m \le 1$ or $n \le 1$ **then**
      $\tilde{u}(m, n) \leftarrow \tau$
      $\tilde{V}(m, n) \leftarrow \phi$
    else
      $j \leftarrow \underset{\substack{1 \le i \\ C_L(i-1,n) \le m \\ \mathsf{dep}(2i+1) \le n}}{\operatorname{argmax}} \mathsf{IME}(\tilde{u}(m - C_L(i-1, n), n - \mathsf{dep}(2i+1)); 2i+1)$
      $\tilde{u}(m, n) \leftarrow \mathsf{IME}(\tilde{u}(m - C_L(j-1, n), n - \mathsf{dep}(2j+1)); 2j+1)$
      $\tilde{V}(m, n) \leftarrow \{2j+1\} \cup \tilde{V}(m - C_L(j-1, n), n - \mathsf{dep}(2j+1))$
    end
  end
end

$$\sum_{i=1}^{k} C_L(\frac{d_i-3}{2}, L - \sum_{j=1}^{i-1} \mathsf{dep}(d_j)) \le m,$$

where $\sum_{i=1}^{k} C_L(\frac{d_i-3}{2}, L - \sum_{j=1}^{i-1} \mathsf{dep}(d_j))$ denotes the running time of the homomorphic comparison operation using a set of degrees $M_{\text{degs}} = \{d_i\}_{1 \le i \le k}$. For $V_{\tau,L}(m, n) = \{d'_i\}_{1 \le i \le k'}$, $\mathsf{MCE}(R_{u_{\tau,L}(m,n)}; \{d'_i\}_{1 \le i \le k'}) \le \tau$, $\sum_{i=1}^{k'} \mathsf{dep}(d'_i) \le n$, and $\sum_{i=1}^{k} C_L(\frac{d_i-3}{2}, L - \sum_{j=1}^{i-1} \mathsf{dep}(d_j)) \le m$.

The `ComputeuV` algorithm in Algorithm 8 outputs two-dimensional tables $\tilde{u}$ and $\tilde{V}$ that store the values of $u_{\tau,L}$ and $V_{\tau,L}$, respectively. This `ComputuV` algorithm requires several computations for $\mathsf{IME}(\tau'; d)$. However, these computations can be performed rapidly using the proposed `AppIME` algorithm.

Then, the `ComputeMinTimeDegs` algorithm in Algorithm 9 outputs the minimum running time M_{time} (in 100 ms) and optimal set of degrees M_{degs} using the two tables $\tilde{u}$ and $\tilde{V}$ obtained from the `ComputeuV` algorithm.

We now propose the homomorphic comparison algorithm `OptMinimaxComp` in Algorithm 10, which uses the `ComputeMinTimeDegs` algorithm. This is a modified version of the previous `MinimaxComp` algorithm in Algorithm 5 [9], minimizing the running time of the RNS-CKKS scheme for a given depth consumption D.

V. APPLICATION TO MIN/MAX AND ReLU FUNCTION

In this section, we apply the methods of improving the homomorphic comparison operation proposed in Sections III and IV to the max and ReLU functions. First, the max function is an important operation employed in several applications, including deep learning. The max function is

Algorithm 9: `ComputeMinTimeDegs`(α, ϵ, D)

Input: Precision parameters α and ϵ, and depth consumption D
Output: Minimum running time M_{time} and the optimal set of degrees M_{degs}

$\tilde{u}, \tilde{V} \leftarrow$ `ComputeuV`$(2^{1-\alpha}; D)$
for $j \leftarrow 0$ **to** $t_{\max}$ **do**
  if $\tilde{u}(j, D) \geq \delta = \frac{1-\epsilon}{1+\epsilon}$ **then**
    $M_{\text{time}} \leftarrow j$
    Go to line 11
  end
  if $j = t_{\max}$ **then**
    return $\perp$
  end
end
$M_{\text{degs}} \leftarrow \tilde{V}(M_{\text{time}}, D)$; // M_{degs}: ordered set
return M_{time} and M_{degs}

Algorithm 10: `OptMinimaxComp`$(a, b; \alpha, \epsilon, D, \eta)$

Input: Inputs $a, b \in (0, 1)$, precision parameters α and ϵ, depth consumption D, and margin η
Output: Approximate value of comp(a, b)

$M_{\text{degs}} = \{d_1, d_2, \cdots, d_k\} \leftarrow$ `ComputeMinTimeDegs`(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 $\frac{p_k \circ p_{k-1} \circ \cdots \circ p_1(a-b)+1}{2}$

easily implemented using the sign function, that is,

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

Thus, the approximate polynomial for the max function $\tilde{p}(a, b)$ can be obtained from the approximate polynomial for the sign function $p(x)$ as

$$\tilde{p}(a, b) = \frac{(a+b) + (a-b)p(a-b)}{2}.$$

Then, $\tilde{p}(a, b)$ should satisfy the following max function error condition for precision parameter α:

$$|\tilde{p}(a, b) - \max(a, b)| \leq 2^{-\alpha} \text{ for any } a, b \in [0, 1]. \quad (2)$$

Because we have $\min(a, b) = a + b - \max(a, b)$, the approximate polynomial for the min. function $\hat{p}(a, b)$ can also be easily obtained, that is, $\hat{p}(a, b) = a + b - \tilde{p}(a, b)$.

The previous homomorphic max function `MinimaxMax` in [9] adopts the set of degrees of component polynomials obtained by executing the `ComputeMinMultDegs` algorithm in Algorithm 4 for inputs α, $\zeta \cdot 2^{-\alpha}$, and $D - 1$, where ζ is the max function factor that can be determined experimentally. The proposed algorithm in Algorithm 11 improves

Algorithm 11: `OptMinimaxMax`$(a, b; \alpha, \zeta, D, \eta)$

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

$M_{\text{degs}} = \{d_1, d_2, \cdots, d_k\} \leftarrow$ `ComputeMinTimeDegs`$(\alpha, \zeta \cdot 2^{-\alpha}, D-1)$
$p_1 \leftarrow \mathsf{MP}(\tilde{R}_{1-\epsilon,1}; d_1)$
$\tau_1 \leftarrow \mathsf{ME}(\tilde{R}_{1-\epsilon,1}; d_1) + \eta$
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 $\frac{(a-b)p_k \circ p_{k-1} \circ \cdots \circ p_1(a-b)+(a+b)}{2}$

on the previous `MinimaxMax` algorithm, and we obtain the set of degrees using the `ComputeMinTimeDegs` algorithm instead of the `ComputeMinMultDegs` algorithm.

In addition, the authors of [17] proposed a method to precisely approximate the ReLU function using the approximation of the sign function. This precise approximation of the ReLU function is necessary to evaluate the pretrained convolutional neural networks on FHE. The ReLU and sign functions have the following relationship

$$\mathrm{ReLU}(x) = \frac{x + x\,\mathrm{sgn}(x)}{2}.$$

Thus, the approximate polynomial $r(x)$ for the ReLU function can be implemented using the approximate polynomial $p(x)$ for the sign function, as follows:

$$r(x) = \frac{x + xp(x)}{2}. \quad (3)$$

Then, $r(x)$ should satisfy the following ReLU function error condition for precision parameter α:

$$|r(x) - \mathrm{ReLU}(x)| \leq 2^{-\alpha} \text{ for any } x \in [-1, 1]. \quad (4)$$

The previous ReLU function algorithm that uses equation (3) can be described as Algorithm 12, which we refer to as `MinimaxReLU`. The proposed ReLU function algorithm in Algorithm 13 improves on the previous ReLU function algorithm `MinimaxReLU`, and we obtain the set of degrees using the `ComputeMinTimeDegs` algorithm instead of the `ComputeMinMultDegs` algorithm. It should be noted that the ReLU function algorithm uses the same value of the max function factor ζ as the max function algorithm for a given precision parameter α.

As explained in Section III, the proposed `AppIME` algorithm makes it possible to apply the `ComputehG` algorithm for $d_{\max} = 63$, which enables us to obtain a better set of degrees of component polynomials using `ComputeMinMultDegs`. Table 4 presents the optimal set of degrees M_{degs} for max/ReLU functions and corresponding minimum depth consumption $D_{\min}$ for $d_{\max} = 31$ and $d_{\min} = 63$. From Table 4, it can be observed that the depth consumption is reduced by one when α is 16, 17,

Algorithm 12: `MinimaxReLU`$(x; \alpha, \zeta, D, \eta)$ [17]

Input: Inputs $x \in [-1, 1]$, precision parameter α, max function factor ζ, depth consumption D, and margin η

Output: Approximate value of ReLU(x)

$M_{\text{degs}} = \{d_1, d_2, \cdots, d_k\} \leftarrow$ `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 $\frac{xp_k \circ p_{k-1} \circ \cdots \circ p_1(x) + x}{2}$

Algorithm 13: `OptMinimaxReLU`$(x; \alpha, \zeta, D, \eta)$

Input: Inputs $x \in [-1, 1]$, precision parameter α, max function factor ζ, depth consumption D, and margin η

Output: Approximate value of ReLU(x)

$M_{\text{degs}} = \{d_1, d_2, \cdots, d_k\} \leftarrow$ `ComputeMinTimeDegs`$(\alpha, \zeta \cdot 2^{-\alpha}, D-1)$

$p_1 \leftarrow \mathsf{MP}(\tilde{R}_{1-\epsilon,1}; d_1)$

$\tau_1 \leftarrow \mathsf{ME}(\tilde{R}_{1-\epsilon,1}; d_1) + \eta$

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 $\frac{xp_k \circ p_{k-1} \circ \cdots \circ p_1(x) + x}{2}$

TABLE 4. Optimal set of degrees M_{degs} and corresponding minimum depth consumption for the homomorphic max/ReLU function for $d_{\text{max}} = 31$ and $d_{\text{max}} = 63$. D_{min} is the minimum depth consumption for the homomorphic max and ReLU functions.

α	ζ	the optimal set of degrees from `ComputeMinMultDegs`$(\alpha, \zeta \cdot 2^{-\alpha}, D_{\min}-1)$			
		maximum degree $d_{\max} = 31$		maximum degree $d_{\max} = 63$	
		degrees	depth	degrees	depth
4	5	{5}	4	{5}	4
5	5	{13}	5	{13}	5
6	10	{3,7}	6	{3,7}	6
7	11	{7,7}	7	{7,7}	7
8	12	{7,15}	8	{7,15}	8
9	13	{15,15}	9	{15,15}	9
10	13	{7,7,13}	11	{7,7,13}	11
11	15	{7,7,27}	12	{7,7,27}	12
12	15	{7,15,27}	13	{7,15,27}	13
13	16	{15,15,27}	14	{15,15,27}	14
14	17	{15,27,29}	15	{15,27,29}	15
15	17	{29,29,29}	16	{15,27,59}	16
16	19	{7,7,7,15,15}	18	{27,29,59}	**17**
17	19	{7,7,15,15,15}	19	{29,59,59}	**18**
18	19	{7,15,15,15,15}	20	{59,59,61}	**19**

or 18, enabling one more non-scalar multiplication per homomorphic max or ReLU function.

Furthermore, the proposed `OptMinimaxMax` and `OptMinimaxReLU` algorithms minimize the running time of the RNS-CKKS scheme for a given depth consumption D using the `ComputeMinTimeDegs` algorithm instead of the `ComputeMinMultDegs` algorithm.

VI. NUMERICAL RESULTS

In this section, numerical results of the proposed `OptMinimaxComp`, `OptMinimaxMax`, and `OptMinimaxReLU` algorithms in Algorithms 10, 11, and 13, respectively, are presented. The performances of the `OptMinimaxComp`, `OptMinimaxMax`, and `OptMinimaxReLU` algorithms using the proposed `ComputeMinTimeDegs` algorithm are evaluated and compared with those of the `MinimaxComp`, `MinimaxMax`, and `MinimaxReLU` algorithms using the previous `ComputeMinMultDegs` algorithm. Numerical analyses are conducted using the representative RNS-CKKS scheme library SEAL [12] on an Intel Core i7-10700 CPU at 2.90 GHz in a single thread with an Ubuntu 20.04 LTS distribution.

A. PARAMETER SETTING

Precision parameters, ϵ and α, are the input and output precisions of the homomorphic comparison algorithm `MinimaxComp` or `OptMinimaxComp` in the RNS-CKKS scheme. We set $\epsilon = 2^{-\alpha}$, which implies that the input and output precisions are the same. By contrast, the homomorphic max function algorithms, `MinimaxMax` and `OptMinimaxMax`, and the homomorphic ReLU function algorithms, `MinimaxReLU` and `OptMinimaxReLU`, use only the input precision parameter α. We set $N = 2^{16}$. `MinimaxComp`, `MinimaxMax`, `MinimaxReLU`, `OptMinimaxComp`, `OptMinimaxMax`, and `OptMinimaxReLU` are performed simultaneously for $N/2$ tuples of real numbers. The amortized running time is then obtained by dividing the running time by $N/2$.

1) SCALING VALUES AND MARGINS

We adopt the scaled Chebyshev polynomials $\tilde{T}_i(t) = T_i(t/w)$ for a scaling value $w > 1$ as the basis polynomials. The scaled Chebyshev polynomials are computed using the following recursion:

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

The scaling values, w, and margins, η, are obtained experimentally. The obtained scaling values and margins used in our numerical analyses of the homomorphic comparison operation and homomorphic max/ReLU functions are presented in Tables 5 and 6, respectively.

2) SCALING FACTOR

If the output of the homomorphic comparison operation or homomorphic max function for one input tuple for two real numbers a and b does not satisfy the comparison operation error condition in (1) or the max function error condition in (2), then it is considered to have failed. In addition, if the

TABLE 5. Set of scaling values and margins for the previous homomorphic comparison algorithm `MinimaxComp` and the proposed algorithm `OptMinimaxComp`.

α	depth	previous homomorphic comparison algorithm MinimaxComp		proposed homomorphic comparison algorithm OptMinimaxComp	
		scaling value w	margin η	scaling value w	margin η
8	11	{1,2,1.7}	2^{-12}	{1,2,1.8}	2^{-26}
12	16	{1,2,2,1.6}	2^{-16}	{1,2,2,1.7}	2^{-16}
	17	{1,2,2,2,1.6}	2^{-16}	{1,2,2,2,1.7}	2^{-16}
	18	{1,2,2,2,2,1.6}	2^{-16}	{1,2,2,2,1.7}	2^{-16}
16	21	{1,2,2,2,1.6}	2^{-20}	{1,2,2,2,1.8}	2^{-24}
	22	{1,2,2,2,2,1.6}	2^{-20}	{1,2,2,2,2,1.8}	2^{-22}
	23	{1,2,2,2,2,2,1.6}	2^{-20}	{1,2,2,2,2,2,1.6}	2^{-20}
20	25	{1,2,2,2,1.6}	2^{-22}	{1,2,2,2,1.7}	2^{-24}
	26	{1,2,2,2,2,1.6}	2^{-23}	{1,2,2,2,2,1.7}	2^{-26}
	27	{1,2,2,2,2,2,2,1.6}	2^{-22}	{1,2,2,2,2,2,1.7}	2^{-29}
	28	{1,2,2,2,2,2,2,2,1.6}	2^{-22}	{1,2,2,2,2,2,2,1.6}	2^{-26}

TABLE 6. Set of scaling values and margins for the previous homomorphic max/ReLU function algorithms `MinimaxMax`/`MinimaxReLU` and the proposed algorithms `OptMinimaxMax`/`OptMinimaxReLU`.

α	ζ	depth	previous homomorphic max/ReLU function algorithms MinimaxMax/MinimaxReLU		proposed homomorphic max/ReLU function algorithms OptMinimaxMax/OptMinimaxReLU	
			scaling value w	margin η	scaling value w	margin η
8	12	9	{1,2,1.6}	2^{-12}	{1,1.7}	2^{-12}
12	15	14	{1,2,2,1.6}	2^{-16}	{1,2,2,1.6}	2^{-16}
16	19	18	{1,2,2,2,1.6}	2^{-20}	{1,2,2,2,1.6}	2^{-20}
		19	{1,2,2,2,2,1.6}	2^{-20}	{1,2,2,2,1.6}	2^{-20}
20	21	22	{1,2,2,2,1.6}	2^{-24}	{1,2,2,2,1.8]	2^{-24}
		23	{1,2,2,2,2,1.6}	2^{-24}	{1,2,2,2,2,1.6}	2^{-24}
		24	{1,2,2,2,2,2,1.6}	2^{-24}	{1,2,2,2,2,2,1.6}	2^{-24}

TABLE 7. Comparison between the running time (amortized running time) of the previous homomorphic comparison algorithm `MinimaxComp` and that of the proposed algorithm `OptMinimaxComp` in the RNS-CKKS scheme.

α	depth	previous homomorphic comparison algorithm MinimaxComp		proposed homomorphic comparison algorithm OptMinimaxComp	
		degrees	running time	degrees	running time
8	11	{7,15,15}	4.21 s (0.12 ms)	{3,15,31}	4.08 s (0.12 ms)
12	16	{15,15,15,15}	9.92 s (0.30 ms)	{7,15,15,31}	9.61 s (0.29 ms)
	17	{7,7,7,13,13}	9.91 s (0.30 ms)	{3,3,13,15,31}	9.40 s (0.28 ms)
	18	{3,5,7,7,7,13}	9.67 s (0.29 ms)	{5,5,5,15,29}	9.12 s (0.27 ms)
16	21	{15,15,15,15,27}	18.45 s (0.56 ms)	{7,13,15,15,59}	17.49 s (0.53 ms)
	22	{7,7,7,13,13,27}	18.23 s (0.55 ms)	{5,3,7,15,15,59}	16.79 s (0.51 ms)
	23	{5,7,7,7,7,13,13}	18.08 s (0.55 ms)	{5,3,5,7,7,15,31}	16.73 s (0.51 ms)
20	25	{29,31,31,31,31}	32.04 s (0.97 ms)	{15,15,31,59,63}	30.73 s (0.93 ms)
	26	{13,15,15,15,27,27}	29.59 s (0.90 ms)	{5,15,15,15,29,61}	28.27 s (0.86 ms)
	27	{7,7,7,7,7,7,15,27}	29.19 s (0.89 ms)	{5,5,7,15,15,15,59}	27.18 s (0.82 ms)
	28	{5,7,7,7,7,7,7,7,15}	29.13 s (0.88 ms)	{5,5,5,7,7,15,15,31}	26.77 s (0.81 ms)

output of the homomorphic ReLU function for one real input number x does not satisfy the ReLU function error condition in (4), it is said to have failed. The homomorphic comparison operation, max function, or ReLU function is performed for 2^{15} inputs for each α, and the number of failures is obtained. The failure rate is the number of failures divided by the total number of inputs, 2^{15}. We set the scaling factor to be sufficiently large such that the homomorphic comparison operation, max function, or ReLU function does not fail in any slot; the failure rate is said to be less than 2^{-15} in this case. We set the scaling factor $\Delta = 2^{50}$ in all our numerical analyses, and the number of failures is zero in all numerical results.

3) BASES WITH PRIME NUMBERS

Bases with prime numbers $\mathcal{B} = \{p_0, p_1, \cdots, p_{k-1}\}$ and $\mathcal{C} = \{q_0, q_1, \cdots, q_L\}$ should be selected. We set $k = 1$ and $p_0 \approx 2^{60}$. In the numerical analysis of the homomorphic comparison operation that consumes depth D, we set the maximum level $L = D$. We set $q_0 \approx 2^{60}$ and $q_j \approx \Delta$ to $1 \le j \le L$.

B. PERFORMANCE OF THE PROPOSED HOMOMORPHIC COMPARISON ALGORITHM

The previous homomorphic comparison operation uses a set of degrees M_{degs} from `ComputeMinMultDegs` for $d_{\max} = 31$. By contrast, the proposed homomorphic comparison operation obtains M_{degs} for $d_{\max} = 63$ from `ComputeMinTimeDegs`. Depth consumption D should satisfy $D \ge M_{\text{dep}}$, where M_{dep} is the minimum depth consumption obtained from the `ComputeMinDep` algorithm. The set of degrees and running times (amortized running times) of the previous homomorphic comparison algorithm `MinimaxComp` and the proposed algorithm

TABLE 8. Comparison between the running times (amortized running times) of the previous homomorphic max/ReLU function algorithms `MinimaxMax`/`MinimaxReLU` and that of the proposed algorithms `OptMinimaxMax`/`OptMinimaxReLU` in the RNS-CKKS scheme.

α	ζ	depth	previous homomorphic max/ReLU function algorithms MinimaxMax/MinimaxReLU			proposed homomorphic max/ReLU function algorithms OptMinimaxMax/OptMinimaxReLU		
			degrees	running time		degrees	running time	
				MinimaxMax	MinimaxReLU		OptMinimaxMax	OptMinimaxReLU
8	12	9	{3,7,7}	2.26 s (0.069 ms)	2.27s (0.069 ms)	{5,23}	2.14 s (0.065 ms)	2.17 s (0.066 ms)
12	15	14	{5,7,7,15}	5.71 s (0.17 ms)	5.74 s (0.17 ms)	{3,5,7,29}	5.53 s (0.16 ms)	5.55 s (0.16 ms)
16	19	18	{7,7,7,15,15}	11.46 s (0.34 ms)	11.37 s (0.34 ms)	{3,3,15,15,31}	10.88 s (0.33 ms)	10.95 s (0.33 ms)
		19	{3,7,7,7,7,13}	11.76 s (0.35 ms)	11.79 s (0.36 ms)	{5,5,7,13,29}	10.65 s (0.32 ms)	10.71 s (0.32 ms)
20	21	22	{15,15,15,15,27}	21.72 s (0.66 ms)	20.38 s (0.62 ms)	{7,13,15,15,59}	19.26 s (0.58 ms)	19.40 s (0.59 ms)
		23	{7,7,7,13,13,27}	19.79 s (0.60 ms)	19.81 s (0.60 ms)	{5,7,7,13,15,31}	18.59 s (0.56 ms)	18.57 s (0.56 ms)
		24	{5,7,7,7,7,7,23}	19.77 s (0.60 ms)	19.81 s (0.60 ms)	{3,5,5,7,7,15,31}	18.38 s (0.56 ms)	18.37 s (0.56 ms)

`OptMinimaxComp` are presented in Table 7. It can be observed that the proposed homomorphic comparison algorithm reduces the running time by 6% on average compared to the previous algorithm.

Increasing the depth consumption D sometimes increases running time. In this case, a depth consumption greater than D does not need to be used, and Table 7 does not include this case. Table 7 also does not include cases in which the previous and proposed algorithms use the same set of degrees, M_{degs}.

C. PERFORMANCE OF THE PROPOSED HOMOMORPHIC MAX/ReLU FUNCTION ALGORITHM

As in the numerical analysis of the homomorphic comparison operation, the proposed homomorphic max and ReLU function algorithms obtain M_{degs} from `ComputeMinTimeDegs` for $d_{\max} = 63$. The set of degrees and running times (amortized running times) of the previous homomorphic max/ReLU function algorithms `MinimaxMax`/`MinimaxReLU` and the proposed algorithms `OptMinimaxMax`/`OptMinimaxReLU` are presented in Table 8. It can be observed that the proposed homomorphic max and ReLU function algorithms reduce the running time by 7% and 6% on average, respectively, compared with the previous homomorphic max and ReLU function algorithms. As in the numerical analysis of the homomorphic comparison operation, Table 8 does not include cases when a larger depth increases the running time or when the previous and proposed algorithms use the same set of degrees M_{degs}.

VII. CONCLUSION

We implemented the optimized homomorphic comparison, max function, and ReLU function algorithms for the RNS-CKKS scheme using a composition of minimax approximate polynomials for the first time. We successfully implemented the algorithms on the RNS-CKKS scheme with a low failure rate ($< 2^{-15}$) and provided parameter sets according to the precision parameter α. In addition, we proposed a fast algorithm for the inverse minimax approximation error, which is a subroutine required to determine the optimal set of degrees. This algorithm enabled us to determine the optimal set of degrees for a higher maximum degree than that in the previous study. Finally, we proposed a method to determine the set of degrees that is optimized for the RNS-CKKS scheme using the proposed fast algorithm for inverse minimax approximation error. We reduced the depth consumption of the homomorphic comparison operation (resp. max/ReLU functions) by one depth when α is 9 or 14 (resp. when α is 16, 17, or 18). In addition, the numerical analysis demonstrated that the proposed homomorphic comparison, max function, and ReLU function algorithms reduced the running time by 6%, 7%, and 6% on average, respectively, compared with the previous algorithms.

REFERENCES

[1] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Symp. Theory Comput. (STOC)*, 2009, pp. 169–178.

[2] J. H. 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.

[3] F. Boemer, Y. Lao, R. Cammarota, and C. Wierzynski, "nGraph-HE: A graph compiler for deep learning on homomorphically encrypted data," in *Proc. 16th ACM Int. Conf. Comput. Frontiers*, 2019, pp. 3–13.

[4] F. Boemer, A. Costache, R. Cammarota, and C. Wierzynski, "nGraph-HE2: A high-throughput framework for neural network inference on encrypted data," in *Proc. 7th ACM Workshop Encrypted Comput. Appl. Homomorphic Cryptogr.*, 2019, pp. 45–56.

[5] X. Jiang, M. Kim, K. Lauter, and Y. Song, "Secure outsourced matrix computation and application to neural networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 1209–1222.

[6] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, "Fast homomorphic evaluation of deep discretized neural networks," in *Proc. Annu. Int. Cryptol. Conf.*, 2018, pp. 483–512.

[7] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "A full RNS variant of approximate homomorphic encryption," in *Proc. Int. Conf. Sel. Areas Cryptogr.*, 2018, pp. 347–368.

[8] 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. 144321–144330, 2020.

[9] E. Lee, J.-W. Lee, J.-S. No, and Y.-S. Kim, "Minimax approximation of sign function by composite polynomial for homomorphic comparison," *IEEE Trans. Dependable Secure Comput.*, early access, Aug. 18, 2021, doi: 10.1109/TDSC.2021.3105111.

[10] J.-P. Bossuat, C. Mouchet, J. Troncoso-Pastoriza, and J.-P. Hubaux, "Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2021, pp. 587–617.

[11] Y. Lee, J.-W. Lee, Y.-S. Kim, H. Kang, and J.-S. No, "High-precision approximate homomorphic encryption by error variance minimization," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn. (Eurocrypt)*, 2022.

[12] (Apr. 2020). *Microsoft SEAL (Release 3.5)*. Microsoft Research, Redmond, WA, USA. [Online]. Available: https://github.com/Microsoft/SEAL

[13] (Sep. 2020). *Lattice Cryptography Library (Release 1.10.4)*. [Online]. Available: https://palisade-crypto.org/

[14] (Jul. 2021). *Lattigo V2.2.0*. [Online]. Available: http://github.com/ldsec/lattigo
[15] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[16] J. A. Hartigan and M. A. Wong, "Ak-means clustering algorithm," *J. Roy. Stat. Society: Ser. C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.
[17] J. Lee, E. Lee, J.-W. Lee, Y. Kim, Y.-S. Kim, and J.-S. No, "Precise approximation of convolutional neural networks for homomorphically encrypted data," 2021, *arXiv:2105.10879*.
[18] J.-W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, E. Lee, J. Lee, D. Yoo, Y.-S. Kim, and J.-S. No, "Privacy-preserving machine learning with fully homomorphic encryption for deep neural network," 2021, *arXiv:2106.07229*.
[19] J. H. Cheon, D. Kim, and D. Kim, "Efficient homomorphic comparison methods with optimal complexity," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2020, pp. 221–256.
[20] L. Ducas and D. Micciancio, "FHEW: Bootstrapping homomorphic encryption in less than a second," in *Proc. Annual Int. Conf. Theory Appl. Cryptograph. Techn.*, 2015, pp. 617–640.
[21] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: Fast fully homomorphic encryption over the torus," *J. Cryptol.*, vol. 33, no. 1, pp. 34–91, 2020.
[22] J. H. Cheon, D. Kim, D. Kim, H. 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.
[23] I. Chillotti, D. Ligier, J.-B. Orfila, and S. Tap, "Improved programmable bootstrapping with larger precision and efficient arithmetic circuits for TFHE," Cryptol. ePrint Arch., IACR, Tech. Rep. 2021/729, 2021.
[24] K. Kluczniak and L. Schild, "FDFB: Full domain functional bootstrapping towards practical fully homomorphic encryption," 2021, *arXiv:2109.02731*.
[25] Z. Liu, D. Micciancio, and Y. Polyakov, "Large-precision homomorphic sign evaluation using FHEW/TFHE bootstrapping," Cryptol. ePrint Arch., IACR, Tech. Rep. 2021/1337, 2021.
[26] W.-J. Lu, Z. Huang, C. Hong, Y. Ma, and H. Qu, "PEGASUS: Bridging polynomial and non-polynomial evaluations in homomorphic encryption," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2021, pp. 1057–1073.
[27] A. Kim, A. Papadimitriou, and Y. Polyakov, "Approximate homomorphic encryption with reduced approximation error," Cryptol. ePrint Arch., IACR, Tech. Rep. 2020/1118, 2020.
[28] J.-W. Lee, E. Lee, Y.-W. Lee, and J.-S. No, "Optimal minimax polynomial approximation of modular reduction for bootstrapping of approximate homomorphic encryption," Cryptol. ePrint Arch., IACR, Tech. Rep. 2020/552/20200803:084202, 2020.
[29] E. W. Cheney, *Introduction to Approximation Theory*. Providence, RI, USA: AMS, 1966.
[30] 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. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2021, pp. 618–647.

EUNSANG LEE received the B.S. and Ph.D. degrees in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2014 and 2020, respectively. He is currently a Postdoctoral Researcher at the Institute of New Media and Communications, Seoul National University. His current research interests include homomorphic encryption and lattice-based cryptography.

JOON-WOO LEE (Graduate Student Member, IEEE) received the B.S. degree in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2016, where he is currently pursuing the Ph.D. degree. His current research interests include homomorphic encryption and lattice-based cryptography.

YOUNG-SIK KIM (Member, IEEE) received the B.S., M.S., and Ph.D. 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 smartcard products and mobile application processors with 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, the IoT security, physical-layer security, data hiding, channel coding, and signal design. He is selected as one of the 2025's 100 Best Technology Leaders (for Crypto-Systems) by the National Academy of Engineering of Korea.

JONG-SEON NO (Fellow, IEEE) received the B.S. and M.S.E.E. degrees in electronics engineering from Seoul National University, Seoul, South Korea, in 1981 and 1984, respectively, and the Ph.D. 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 an IEEE Fellow through the IEEE Information Theory Society, in 2012. He became a member of the National Academy of Engineering of Korea (NAEK), in 2015, where he served as the Division Chair for Electrical, Electronic, and Information Engineering, from 2019 to 2020. He was a recipient of the IEEE Information Theory Society Chapter of the Year Award in 2007. From 1996 to 2008, he served as the Founding Chair for the Seoul Chapter of the 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 for 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.