High-Precision and Low-Complexity Approximate Homomorphic Encryption by Error Variance Minimization

Yongwoo Lee[1], Joon-Woo Lee[1], Young-Sik Kim[2], HyungChul Kang[3], and Jong-Seon No[1]

[1] Department of Electrical and Computer Engineering, INMC, Seoul National University, Seoul, 08826, Korea
yongwool@ccl.snu.ac.kr, joonwoo3511@ccl.snu.ac.kr, jsno@snu.ac.kr
[2] Department of Information and Communication Engineering, Chosun University, Gwangju, 61452, Korea iamyskim@chosun.ac.kr
[3] Samsung Advanced Institute of Technology, Suwon-si, Gyeonggi-do, 16678, Korea
hc1803.kang@samsung.com

Abstract. In this paper, we propose a high-precision and low-complexity Cheon-Kim-Kim-Song (CKKS) scheme (Asiacrypt '17) by reducing error and improving its bootstrapping through error variance minimization. The proposed algorithm allows us to leave more levels after bootstrapping, and thus, with practical accuracy, we reduce the degree of ciphertext polynomial of the CKKS scheme from 2^{16} to 2^{15}. The sizes of the key and ciphertext are reduced by at least 1/4, which significantly reduces their computational complexity. Those are possible from the fact that a smaller scaling factor can be used with the proposed algorithms for the same precision since the following two contributions significantly reduce the error of the CKKS scheme, especially in bootstrapping. First, we apply the concept of signal-to-noise ratio (SNR) and propose a method of maximizing the SNR of encrypted data by reordering homomorphic operations. For that, the error variance is minimized instead of the upper bound of error when we deal with encrypted data. Especially, we propose lazy rescaling and lazy relinearization, which reduces computational complexity as well as error variance. Second, we propose a novel polynomial approximation specialized for the CKKS scheme from the same perspective of minimizing the error variance of the encrypted data. We mainly apply the approximation to the bootstrapping of the CKKS scheme, where we achieve a smaller bootstrap error compared to the prior arts. The performance improvement of the proposed algorithms for the CKKS scheme is verified by implementation over a well-known homomorphic encryption library, SEAL. Specifically, for very accurate bootstrapping, the proposed method achieves the same bootstrap error with only a depth of 8, whereas it requires a depth of 13 in the prior art.

Keywords: Bootstrapping· Cheon-Kim-Kim-Song (CKKS) scheme· Fully homomorphic encryption (FHE)· Privacy-preserving machine learning

Supported by Samsung Advanced Institute of Technology.
Y.-S. Kim is the corresponding author.

(PPML)· Simple encrypted arithmetic library (SEAL) · Signal-to-noise ratio (SNR).

1 Introduction

The recent development of cloud computing, machine learning, and blockchain raises a privacy problem; how can one write a smart contract on a public blockchain or outsource computation in machine learning for confidential data? The need for cryptographic primitives for such scenarios has been exploded, and there have been extensive studies. Homomorphic encryption (HE) is a specific class of encryption schemes that allows computation over encrypted data.

The CKKS scheme [10] is one of the highlighted fully homomorphic encryption (FHE) schemes as it is efficient to deal with real (or complex) numbers, which are the usual data type for many applications such as deep learning. When we deal with fixed-point real numbers using other FHE schemes such as (B)FV [4, 5, 13] and BGV [3] schemes, the size of ciphertext has an exponential growth rate according to the level, where the level of ciphertext is defined by the maximum depth of circuit that can be homomorphically evaluated without bootstrapping. However, the ciphertext size has a polynomial growth rate according to the level in the CKKS scheme.

The CKKS scheme provides the trade-off between the efficiency and accuracy of messages as encrypted data of the CKKS scheme has an error. Errors in encrypted data are propagated and added along with homomorphic operations. To our best knowledge, research to the date has provided high-probability upper bounds for errors [10, 7, 20, 2]. As the operations on encrypted data proceeds, the upper bound of errors in encrypted data becomes a loose and useless bound. Moreover, as attacks against CKKS have recently been proposed [9, 27, 25], reducing errors become more crucial when using the CKKS scheme to mitigate the risk of the attack. Therefore, there is a need for a new technique that can thoroughly manage errors in encrypted data. Once tightly managing the error, a compiler technique can be adopted to reduce the error and computing time.

1.1 Our Results

This paper significantly reduces the error in the encrypted data and computational complexity of the CKKS scheme by the following two contributions: i) homomorphic operation reordering and ii) new polynomial approximation algorithm and bootstrapping using it. As a result, we achieve less bootstrapping error with a smaller scaling factor, and thus we enlarge the levels after bootstrapping. Therefore, bootstrapping can be done while maintaining practical precision even if we use a parameter of $N = 2^{15}$ instead of $N = 2^{16}$, where N is the ciphertext polynomial degree. In contrast, in previous works with $N = 2^{15}$, only about four levels remain after bootstrapping, and thus we were forced to use $N = 2^{16}$.

By using $N = 2^{15}$ instead of $N = 2^{16}$, the size of the keys and ciphertexts are reduced by at least 1/4, and the time for all basic homomorphic operations

is also reduced. We can also significantly reduce the amount of computation and communication of both the server and client. For example, based on SEAL (v.3.5)[1], the size of the transmitted keys can be reduced from 16.49 GB to 1.63 GB, even if the client only generates power-of-two rotation keys to minimize client communication instead of the additional computation of the server.

Error Variance Minimization and Homomorphic Operation Reordering First, we propose a method of managing the variance of errors to maximize the signal-to-noise ratio (SNR) of the encrypted data rather than the high-probability upper bounds of error. The CKKS scheme can be considered as a noisy channel, and thus we adopt the methodologies from communication theory, the SNR. In the proposed method, one can tightly manage the error in the encrypted data and can effectively reorder homomorphic operations to maximize SNR. As a simple example, we can reorder the operations to increase the SNR of the encrypted 74-th Chebyshev polynomial by 1973 times. Specifically, we propose lazy rescaling and relinearization that reduce the error variance and the computation time. For example, in the bootstrapping of this paper, the 255th order polynomial is calculated. If lazy rescaling and linearization are applied, the computation time can be reduced to a level similar to the conventional method of calculating the 52nd order polynomial and the double-angle formula [2].

Depth-Optimal Bootstrapping by Novel Polynomial Approximation The second contribution is a new polynomial approximation specialized for the CKKS scheme in terms of error variance minimization. To our best knowledge, this is the first method to find the optimal approximate polynomial that minimizes not only the approximation error but also the error in the polynomial basis that is amplified by coefficients. We can improve the bootstrapping of the CKKS scheme using our polynomial approximation, and bootstrapping can be implemented with fewer errors and less depth. The implementation shows that the proposed method reduces the bootstrap error compared to the state-of-the-arts [23, 2]. Moreover, the proposed method resolves the problem that the approximate polynomials have large coefficients, which could only be solved using the double-angle formula previously. The proposed method allows us to use a direct approximation for the modulus reduction in the bootstrapping, and thus we can do bootstrapping with less depth. The implementation result shows that for a very accurate bootstrapping, the proposed method achieves the same bootstrap error with only a depth of 8, whereas it required a depth of 13 previously.

Also, we propose various methods for efficient bootstrapping. First, we propose a method that significantly reduces computation by using lazy rescaling and linearization with the baby-step giant-step (BSGS) algorithm [17, 23]. Further, we propose a novel coefficient-to-slot algorithm that does not consume the ciphertext level; thus, we can leave more levels after bootstrapping.

However, it is meaningful to use $N = 2^{15}$ for practical precision (16-bit fixed-point arithmetic) since $N = 2^{16}$ is too heavy for clients with less computational

[1] source code: https://github.com/microsoft/SEAL

power; clients do not want to make 16 GB of keys. Besides, the proposed method has a significant advantage in reducing the time for bootstrapping, even though bootstrapping is done more frequently. For instance, a single rotation takes 1.5 s and 19.2 s when $N = 2^{15}$ and $N = 2^{16}$ in our experiment, respectively, and moreover, if we use larger N, more rotations are required for a bootstrapping.

1.2 Related Works

Bootstrapping of the CKKS Scheme Recently, research on bootstrapping of the CKKS scheme is actively in progress. After the first bootstrapping was proposed in [7], the Chebyshev interpolation has been applied to the homomorphic evaluation of modulus reduction [6, 17]. Then, a technique for direct-approximation was proposed using least squares method [24] and Lagrange interpolation [19]. However, the magnitudes of approximate polynomial coefficients of those methods are too large. Recently, two papers regarding the bootstrapping of the CKKS scheme are accepted to Eurocrypt'21 [23, 2]. The algorithm to find minimax approximate polynomial and the use of arcsin to reduce approximation error of the modular reduction is presented in [23]. The bootstrapping for the non-sparse-key CKKS scheme is proposed, and the computation time for homomorphic linear transformations is significantly improved in [2]. We note that all of the preceding methods cannot make a valid direct approximation for the modulus reduction function.

Error Management of the CKKS Scheme Since a drawback of the CKKS scheme is that errors are accumulated, many studies have been conducted to reduce errors. Recently, Kim *et al.* proposed a new method of reducing errors in encrypted data of the CKKS scheme by doing rescaling before a ciphertext-ciphertext (non-scalar) multiplication and using different scaling factors at each level [20]. Error management for the CKKS scheme so far used the high-probability upper bounds of error [10, 7] or average precision of messages. The high-probability upper bounds are derived from the distribution of error, and the average precision of message provided in [10, 7, 6, 17, 20] is the expectation of the absolute value of error. As will be described later, the high-probability upper bound becomes very loose with just several successive homomorphic operations.

Cryptanalysis and Significance of Error Minimization From recent cryptanalysis of the CKKS scheme, the management of the error in encrypted data has become more vital. Recently, an attack to recover the secret key using the error pattern after decryption is proposed [25]. The solution to this attack is to add errors or perform rounding after decryption, and several libraries have been updated as such. Since errors are added after decryption, errors in the encrypted data should be reduced than before. Moreover, the CKKS scheme uses a sparse-key to keep errors small, but attacks on sparse-key learning with error (LWE) was proposed [9, 27]. The error variance of the CKKS scheme is proportional to the Hamming weight of the secret key, and thus the significance of error management has been increased.

1.3 Organization of This Work

The remainder of the paper is organized as follows. In Section 2, we provide the necessary notations and signal-to-noise perspective on error. The CKKS scheme and its bootstrapping algorithm are summarized in Section 3. A new method of minimizing error variance of encrypted data by using homomorphic operation reordering is proposed, and some examples, including the novel lazy rescaling and lazy relinearization, are given in Section 4. We provide a new method to find optimal direct approximate polynomials for the CKKS scheme in Section 5, focusing on bootstrapping with a modified baby-step giant-step algorithm for modulus reduction to reduce the computational complexity and error variance. The implementation results and comparison are given in Section 6. Finally, we conclude paper in Section 7 with remarks and possible future research directions.

2 Preliminaries

2.1 Basic Notation

Vectors are denoted in boldface, such as $\boldsymbol{v}$, and every vector is a column vector. Matrices are denoted by boldfaced capital letters, for example, $\mathbf{M}$. We denote the inner product of two vectors by $\langle \cdot, \cdot \rangle$ or simply $\cdot$. Let $\boldsymbol{u} \times \boldsymbol{v}$ denote the component-wise multiplication of two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$. When it is evident, x^2 denotes the multiplication of x and its complex conjugate, where $x \in \mathbb{C}$.

$x \leftarrow \mathcal{D}$ denotes the sampling x according to a distribution $\mathcal{D}$. When a set is used instead of distribution, x is sampled uniformly at random among the set elements. Random variables are denoted by capital letters such as X. $E[X]$ and $Var[X]$ denote the mean and variance of random variable X, respectively. Some capital letters may represent something other than a random variable, such as a constant but context-sensitive. Let $\Phi_M(X)$ be the M-th cyclotomic polynomial of degree N, and when M is a power of two, $M = 2N$, and $\Phi_M(X) = X^N + 1$. Let $\mathcal{R} = \mathbb{Z}/\langle \Phi_M(X) \rangle$ be the ring of integers of a number field $\mathcal{S} = \mathbb{Q}/\langle \Phi_M(X) \rangle$, where $\mathbb{Q}$ is the set of rational numbers and we write $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$.

2.2 Chebyshev Polynomials

The Chebyshev polynomial of the first kind, in short, the Chebyshev polynomial is defined by the recursive relation [26]

$$T_0(x) = 1, T_1(x) = x, \text{ and } T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x).$$

The Chebyshev polynomial of degree n has n distinct roots in the interval $[-1, 1]$ and all its extrema are also in $[-1, 1]$. Moreover, $\frac{1}{2^{n-1}}T_n(x)$ is the polynomial, whose maximal absolute value is minimal among monic polynomials of degree n and its absolute value is $\frac{1}{2^{n-1}}$. Hence, unlike the monomial basis, the Chebyshev basis's high-degree terms do not converge to zero or diverge to infinity as the degree increases. In addition to the above, the Chebyshev polynomial has desirable properties as a basis for an approximate polynomial.

2.3 Signal-to-Noise Ratio Perspective of the CKKS Scheme

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

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

$$\mathrm{SNR} = \frac{E[S^2]}{E[N^2]},$$

where S and N denote the signal (message) and noise (error), respectively. As the signal and noise must be measured at the same or equivalent points in a system, the ratio of power is identical to the ratio of energy (or the second moment), $\frac{E[S^2]}{E[N^2]}$. As shown in the definition, a signal with high SNR has better quality.

A simple way to increase SNR is to increase signal power, but it is not easy in real systems due to regulatory or physical constraints. The CKKS system is also the same; a larger scaling factor should be multiplied to the message to increase the message power, but if one uses a larger scaling factor, the ciphertext level decreases, or larger parameters should be used to be secure under the RLWE problem. Hence, to increase SNR, it is essential to reduce the noise power in the CKKS scheme rather than increase the signal power.

Error estimation of the CKKS scheme so far has been focused on the high-probability upper bound of the error after several operations, and the upper bound was tracked using the upper bound of the message [10, 7]. As the homomorphic operation continues, the bound becomes quite loose, and its statistical significance may fade. In this paper, we propose methods to reduce the power (or energy) of error. We note that when the mean of error is zero, the energy of error is the same as its variance. Therefore, the energy and variance of errors are abused from now on if its mean is zero.

3 The CKKS Scheme and Its Bootstrapping

3.1 The RNS-CKKS Scheme

This subsection briefly introduces the RNS-CKKS scheme [17, 8, 20]. The CKKS scheme and its RNS variants provide homomorphic operations on encrypted complex number with errors, which is done by canonical embedding and its inverse. Recall that canonical embedding Emb of $a(X) \in \mathbb{Q}/\langle \Phi_M(X) \rangle$ into $\mathbb{C}^N$ is the vector of the evaluation values a at the roots of $\Phi_M(X)$ and Emb^{-1} denotes its inverse. Let π denote a natural projection from $\mathbb{H} = \{(z_j)_{j \in \mathbb{Z}_M^*} : z_j = \overline{z_{-j}}\}$ to $\mathbb{C}^{N/2}$, where $\mathbb{Z}_M^*$ is the multiplicative group of integer modulo M. The encoding ($\mathbb{C}^{N/2} \to \mathcal{R}$) and decoding are defined as follows.

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

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

where Δ is the scaling factor and $\lfloor \cdot \rceil_{\mathsf{Emb}(\mathcal{R})}$ denotes the discretization into an element of $\mathsf{Emb}(\mathcal{R})$.

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

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

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

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

The RNS-CKKS scheme performs all operations in RNS. The ciphertext modulus $Q_l = q \cdot \prod_{i=1}^{l} p_i$ is used, where p_i's are chosen as primes that satisfy $p_i = 1 \pmod{2N}$ to support efficient number theoretic transform. These prime numbers are also chosen such that $p \approx p_i$ for the initial scaling factor p. We note that $Q_0 = q$ is much greater than p as the final message's coefficients should not be greater than the ciphertext modulus q. For bootstrapping, we choose different moduli, such as $Q'_{L_0+L_1} = q \cdot \prod_{i=1}^{L_0} p_i \prod_{j=1}^{L_1} p'_j$, where $p'_j \approx q$ and $Q'_{L_0+L_1} \approx Q_L$.

The CKKS scheme is defined with the following operations.

– $\mathsf{KeyGen}(1^\lambda)$:
  - Given the security parameter λ, we choose a power-of-two M, an integer h, an integer P, a real number σ, and a maximum ciphertext modulus Q, such that $Q \geq Q_L$.
  - Sample the following values: $s \leftarrow \mathcal{HWT}(h), a \leftarrow \mathcal{R}_{Q_L}, e \leftarrow \mathcal{DG}(\sigma^2)$.
  - Set the secret key and the public key as

$$\mathsf{sk} \coloneqq (1, s), \mathsf{pk} \coloneqq (b, a) \in \mathcal{R}_{Q_L}^2,$$

respectively, where $b = -as + e \pmod{Q_L}$.

To take advantage of RNS, we use the hybrid key switching technique proposed in [17]. First, for predefined dnum, a small integer such as 4, we define partial products $\left\{\tilde{Q}_j\right\}_{0 \leq j < \mathsf{dnum}} = \left\{\prod_{i=j\alpha}^{(j+1)\alpha-1} p_i\right\}_{0 \leq j < \mathsf{dnum}}$, where $\alpha = (L+1)/\mathsf{dnum}$. For level l and $\mathsf{dnum}' = \lceil (l+1)/\alpha \rceil$, we define [17]

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

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

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

$$\langle \mathcal{WD}_l(a), \mathcal{PW}_l(b) \rangle = a \cdot b \,(\mathrm{mod}\ Q_l)\,.$$

Then, the operations in the RNS-CKKS scheme are defined as follows:

- $\mathsf{KSGen}_{\mathsf{sk}}(s')$: For auxiliary modulus $P = \prod_{i=0}^{k} p'_i \approx \max_j \tilde{Q}_j$, sample $a'_k \leftarrow \mathcal{R}_{PQ_L}$ and $e'_k \leftarrow \mathcal{DG}(\sigma^2)$. Output the switching key

$$\mathsf{swk} := (\mathsf{swk}_0, \mathsf{swk}_1) = (\{b'_k\}_{k=0}^{\mathsf{dnum}'-1}, \{a'_k\}_{k=0}^{\mathsf{dnum}'-1}) \in \mathcal{R}_{PQ_L}^{2\times \mathsf{dnum}'},$$

  where $b'_k = -a'_k s + e'_k + P \cdot \mathcal{PW}(s')_k \,(\mathrm{mod}\ PQ_L)$.
  - Set the evaluation key as $\mathsf{evk} := \mathsf{KSGen}_{\mathsf{sk}}(s^2)$.
- $\mathsf{Enc}_{\mathsf{pk}}(m)$: Sample $v \leftarrow \mathcal{ZO}(0.5)$ and $e_0, e_1 \leftarrow \mathcal{DG}(\sigma^2)$. Output $\boldsymbol{c} = v \cdot \mathsf{pk} + (m + e_0, e_1) \,(\mathrm{mod}\ Q_L)$.
- $\mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{c})$: Output $\bar{m} = \langle \boldsymbol{c}, \mathsf{sk} \rangle$.
- $\mathsf{Add}(\boldsymbol{c}_1, \boldsymbol{c}_2)$: For $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{R}_{Q_l}^2$, output $\boldsymbol{c}_{\mathsf{add}} = \boldsymbol{c}_1 + \boldsymbol{c}_2 \,(\mathrm{mod}\ Q_l)$.
- $\mathsf{Mult}(\boldsymbol{c}_1, \boldsymbol{c}_2)$: For $\boldsymbol{c}_1 = (b_1, a_1)$ and $\boldsymbol{c}_2 = (b_2, a_2) \in \mathcal{R}_{Q_l}^2$, return

$$\boldsymbol{c}_{\mathsf{mult}} = (d_0, d_1, d_2) := (b_1 b_2, a_1 b_2 + a_2 b_1, a_1 a_2) \,(\mathrm{mod}\ Q_l)\,.$$

- $\mathsf{RL}_{\mathsf{evk}}(d_0, d_1, d_2)$: For a three-tuple ciphertext (d_0, d_1, d_2) correspond to secret key $(1, s, s^2)$, return $(d_0, d_1) + \mathsf{KS}_{\mathsf{evk}}((0, d_2))$.
  We note that $\mathsf{Mult}_{\mathsf{evk}}$ consists of two stages: multiplication and relinearization $(\mathsf{KS}_{\mathsf{evk}}((0, d_2)))$.
- $\mathsf{cAdd}(\boldsymbol{c}_1, \boldsymbol{a}; \Delta)$: For a $\boldsymbol{a} \in \mathbb{C}^{N/2}$ and a scaling factor Δ, output $\boldsymbol{c}_{\mathsf{cadd}} \leftarrow \boldsymbol{c} + (\mathsf{Ecd}(\boldsymbol{a}; \Delta), 0)$.
- $\mathsf{cMult}(\boldsymbol{c}_1, \boldsymbol{a}; \Delta)$: For a $\boldsymbol{a} \in \mathbb{C}^{N/2}$ and a scaling factor Δ, output $\boldsymbol{c}_{\mathsf{cmult}} \leftarrow \mathsf{Ecd}(\boldsymbol{a}; \Delta) \cdot \boldsymbol{c}$.
- $\mathsf{RS}(\boldsymbol{c})$: For $\boldsymbol{c} \in \mathcal{R}_{Q_l}^2$, output $\boldsymbol{c}_{\mathsf{RS}} = \lfloor p_l^{-1} \boldsymbol{c} \rceil \,(\mathrm{mod}\ q_{l-1})$.
- $\mathsf{KS}_{\mathsf{swk}}(\boldsymbol{c})$: For $\boldsymbol{c} = (c_0, c_1) \in \mathcal{R}_{Q_l}^2$ and $\mathsf{swk} := (\mathsf{swk}_0, \mathsf{swk}_1)$, output

$$\boldsymbol{c}_{\mathsf{KS}} = \left(c_0 + \left\lfloor \frac{\langle \mathcal{WD}_l(c_1), \mathsf{swk}_0 \rangle}{P} \right\rceil, \left\lfloor \frac{\langle \mathcal{WD}_l(c_1), \mathsf{swk}_1 \rangle}{P} \right\rceil \right) (\mathrm{mod}\ Q_l)\,.$$

The key-switching techniques are used to provide various operations such as complex conjugate and rotation. Key switching consists of three steps: i) DECOMPOSE: generate $\mathcal{WD}_l(c_1)$, ii) MULTSUM: calculate $\langle \mathcal{WD}_l(c_1), \mathsf{swk}_1 \rangle$, and iii) MODDOWN. DECOMPOSE and MODDOWN are the most time-consuming [2].

To remove the error introduced by approximate scaling factors, one can use different scaling factors for each level as given in [20], or we can use the scale-invariant method proposed in [2] for polynomial evaluation. It is noted that Full RNS-HEAAN libraries are ($\mathsf{dnum} = 1$)-case and SEAL is ($\mathsf{dnum} = L + 1$)-case. We are using SEAL in the implementation as it is advantageous at the ciphertext level. We note that this paper aims at all the variants of the CKKS scheme.

3.2 Bootstrapping of the CKKS Scheme

There are extensive studies for bootstrapping of the CKKS scheme [7, 6, 17, 24, 23, 2]. The bootstrapping consists of the following four steps: ModRaise, CoeffToSlot, EvalMod, and SlotToCoeff.

Modulus Raising (ModRaise) ModRaise is raises the ciphertext modulus to a larger modulus. Let $\boldsymbol{c}$ be the ciphertext satisfying $m(X) = [\langle \boldsymbol{c}, \mathsf{sk} \rangle]_q$. Then, $t(X) = \langle \boldsymbol{c}, \mathsf{sk} \rangle \pmod{X^N + 1}$ is of the from $t(X) = qI(X) + m(X)$ for $I(X) \in \mathcal{R}$ with a high-probability bound $\|I(X)\|_\infty < K = \mathcal{O}(\sqrt{h})$. The following procedure aims to calculate the remaining coefficients of $t(X)$ when dividing by q.

Homomorphic Evaluation of Encoding (CoeffToSlot) Approximate homomorphic operations are performed in plaintext slots, but we need component-wise operations on coefficients. Thus, to deal with $t(X)$, we have to put polynomial coefficients in plaintext slots. In CoeffToSlot step, the $\mathsf{Emb}^{-1} \circ \pi^{-1}$ is performed homomorphically using matrix multiplication [7], or FFT-like hybrid method [6]. Then, we have two ciphertexts encrypting $\boldsymbol{z}'_0 = (t_0, \ldots, t_{\frac{N}{2}-1})$ and $\boldsymbol{z}'_1 = (t_{\frac{N}{2}}, \ldots, t_{N-1})$ (or equivalently, $(t_0 + i \cdot t_{\frac{N}{2}}, \ldots, t_{\frac{N}{2}-1} + i \cdot t_{N-1})$), where t_j denotes the j-th coefficient of $t(X)$. The matrix multiplication is composed of three steps [7]: i) rotate ciphertexts ii) multiply diagonal components of matrix to the rotated ciphertexts and iii) sum up the ciphertexts.

Evaluation of the Approximate Modulus Reduction (EvalMod) In the EvalMod step, an approximate evaluation of modulus reduction function of t_i's is performed. As additions and multiplications cannot represent the modulus reduction function, an approximate polynomial for this function is used. For approximation, it is desirable to control the message size to ensure $m_i \leq \epsilon \cdot q$ for a small ϵ. Previous works *et al.* approximated the modulus reduction function as $\frac{q}{2\pi} \sin\left(\frac{2\pi t}{q}\right)$ [7, 6, 17, 2]. However, in these approaches, the sine function is used, and thus there is still the fundamental approximation error, that is,

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

Then, direct-approximation methods were proposed in [24, 19], but their coefficients are too large and amplify errors of polynomial basis. A composition with inverse sine function that offers a trade-off between the precision and the remaining homomorphic capacity is proposed in [23] to remove the fundamental approximation error between the sine function and the modulus reduction.

Homomorphic Evaluation of Decoding (SlotToCoeff) SlotToCoeff is the inverse operation of CoeffToSlot. Since the matrix elements do not have to be precise as in CoeffToSlot, we can use a smaller modulus.

Table 1. Experimental result of probability mass function of I when $h = 192$.

i	$\Pr_I(i)$	i	$\Pr_I(i)$	i	$\Pr_I(i)$	i	$\Pr_I(i)$	i	$\Pr_I(i)$	i	$\Pr_I(i)$
0	$9.94{\cdot}10^{-2}$	±4	$6.05{\cdot}10^{-2}$	±8	$1.36{\cdot}10^{-2}$	±12	$1.12{\cdot}10^{-3}$	±16	$3.34{\cdot}10^{-5}$	±20	$3.40{\cdot}10^{-7}$
±1	$9.64{\cdot}10^{-2}$	±5	$4.58{\cdot}10^{-2}$	±9	$8.02{\cdot}10^{-3}$	±13	$5.15{\cdot}10^{-4}$	±17	$1.16{\cdot}10^{-5}$	±21	$9.41{\cdot}10^{-8}$
±2	$8.78{\cdot}10^{-2}$	±6	$3.25{\cdot}10^{-2}$	±10	$4.44{\cdot}10^{-3}$	±14	$2.20{\cdot}10^{-4}$	±18	$3.84{\cdot}10^{-6}$		
±3	$7.52{\cdot}10^{-2}$	±7	$2.17{\cdot}10^{-2}$	±11	$2.30{\cdot}10^{-3}$	±15	$8.84{\cdot}10^{-5}$	±19	$1.20{\cdot}10^{-6}$		

Table 2. Probability of Irwin-Hall distribution, $X = \sum_{k=1}^{h+1} U_k$, where U_k's are independent and identically distributed $U[0,1]$ and $h = 192$. $\Pr_X(i)$ denotes the probability that $\Pr\left(\frac{h+1}{2} + i - \frac{1}{2} \leq X \leq \frac{h+1}{2} + i + \frac{1}{2}\right)$.

i	$\Pr_I(i)$	i	$\Pr_I(i)$	i	$\Pr_I(i)$	i	$\Pr_I(i)$	i	$\Pr_I(i)$	i	$\Pr_I(i)$
0	$9.91{\cdot}10^{-2}$	±4	$6.05{\cdot}10^{-2}$	±8	$1.37{\cdot}10^{-2}$	±12	$1.15{\cdot}10^{-3}$	±16	$3.46{\cdot}10^{-5}$	±20	$3.71{\cdot}10^{-7}$
±1	$9.61{\cdot}10^{-2}$	±5	$4.58{\cdot}10^{-2}$	±9	$8.11{\cdot}10^{-3}$	±13	$5.26{\cdot}10^{-4}$	±17	$1.23{\cdot}10^{-5}$	±21	$1.01{\cdot}10^{-7}$
±2	$8.76{\cdot}10^{-2}$	±6	$3.26{\cdot}10^{-2}$	±10	$4.50{\cdot}10^{-3}$	±14	$2.27{\cdot}10^{-4}$	±18	$4.09{\cdot}10^{-6}$		
±3	$7.51{\cdot}10^{-2}$	±7	$2.18{\cdot}10^{-2}$	±11	$2.34{\cdot}10^{-3}$	±15	$9.15{\cdot}10^{-5}$	±19	$1.27{\cdot}10^{-6}$		

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

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

$$t(X) = q \cdot I(X) + m(X) = \langle \boldsymbol{c}, \mathsf{sk} \rangle \left(\mathrm{mod}\ X^N + 1\right).$$

sk has Hamming weight h and each coefficient of a ciphertext (c_0, c_1) is an element of $\mathbb{Z}_q$ and thus, each coefficient of $\langle \boldsymbol{c}, \mathsf{sk} \rangle = c_0 + c_1 s$ is considered as a sum of $(h+1)$ elements in $[-q/2, q/2)$. Therefore, $I(X) = \left\lfloor \frac{1}{q} \langle \boldsymbol{c}, \mathsf{sk} \rangle \right\rceil$ is upper bounded by $\frac{1}{2}(h+1)$. In practice, a heuristic assumption is used, where a high-probability upper bound $K = O(\sqrt{h})$ for $\|I\|_\infty$ exists.

The high-probability upper bound K of $\|I\|_\infty$ is us used so far, but EvalMod outputs a useless value when the input is not in the desired domain, i.e., $\|I\|_\infty > K$, resulting in the bootstrapping failure. For example, the probability that $\|I\|_\infty \geq K$ is about 2^{-22}, when $(h, K) = (64, 128), (128, 17)$ and $(192, 21)$.

We can numerically obtain the distribution of I or analytically calculate its distribution through heuristic assumption given as follows. From the RLWE assumption, each coefficient of c_0 and c_1 can be considered as distributed uniformly at random; in other words, coefficients of t follows a distribution similar to the well-known Irwin–Hall distribution. Table 1 is probability density function of I, obtained numerically using SEAL. Table 2 is the probability of I, analytically calculated by using Irwin-Hall distribution. In is shown that our probability analysis using the Irwin-Hall distribution and the experimental results agree. We note here that a probabilistic approach is widely used in the error estimation and bootstrapping of the CKKS scheme, and thus it is reasonable to reduce the error of the CKKS scheme and its bootstrapping through a stochastic approach.

4 Optimization of Error Variance in the Encrypted Data

This section provides a new criterion for the quality of the encrypted data of the CKKS scheme, that is, SNR. Measuring the quality of the encrypted message by SNR is the main idea, which is natural and widely used in communication systems. In subsection 4.3, we present three cases that can reduce error and computation time by homomorphic operation reordering: finding polynomial basis, lazy rescaling and relinearization, and successive multiplication. In the proposed simple examples, the error can be reduced by up to 1/1973 when calculating polynomials, lazy rescaling, and relinearization reduce the EvalMod time by 42%, and successive multiplication reduces the error by $1/2^{22}$.

We can assume the following statements:

i) The mean of error is zero.
ii) The message and error are statistically independent.

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

4.1 Tagged Information for Ciphertext

We propose new tagged information for the full ciphertext of the CKKS scheme to tightly manage the errors in encrypted data. The tagged information is introduced in [10], and it is used to estimate the magnitude of the error. The tagged information so far comprises a level l, an upper bound $v \in \mathbb{R}$ of the message, and a high-probability upper bound $B \in \mathbb{R}$ of error. However, as the homomorphic operation continues, the upper bound becomes exponentially loose.

We take a simple example of how the upper bound becomes loose. In [10], 6σ is the high-probability upper bound of the Gaussian error with variance σ^2, where $Pr(|X| > 6\sigma)$ is $2^{-27.8}$. Assume that there are 100 distinct fresh ciphertexts and let σ_o^2 be the error variance. Previously, the error upper bound of a sum of two ciphertexts with error upper bounds B_1 and B_2, was given as $B_1 + B_2$ [10, 20]. Then, the upper bound of the error in the sum of the hundred ciphertexts is $600\sigma_o$. However, the ciphertexts are independent, and the error of the sum is Gaussian error with a variance of $100\sigma_o^2$. Therefore, the probability of the upper bound, $600\sigma_o$, is $Pr(|e| > 600\sigma_o) \approx 2^{-2602.1}$, which is quite loose, where e is the summation of error. In conclusion, the previous upper bound is too loose to obtain useful information of the error after successive homomorphic operations.

A real-world application such as deep learning requires a lot more computation than this, and thus the resultant upper bound is much looser than

the above example. Thus, instead of using the upper bounds, we propose to use the variance of messages and errors as the tagged information. In other words, three tuples are the tagged information, which consist of message means, $\{E[\pi(\mathsf{Emb}(m))_i]\}_{i=0,\cdots n-1}$, message variances, $\{Var[\pi(\mathsf{Emb}(m))_i]\}_{i=0,\cdots n-1}$, and error variances, $\{Var[\pi(\mathsf{Emb}(e))_i]\}_{i=0,\cdots n-1}$, denoted by $\boldsymbol{\mu}_{\mathsf{m}} \in \mathbb{C}^n$ and $\boldsymbol{v}_{\mathsf{m}}, \boldsymbol{v}_{\mathsf{e}} \in \mathbb{R}^n$, respectively, where $\mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{c}) = m + e$ and n is number of slots. Hence, the full ciphertext is given as

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

where l and Δ are the level and the scaling factor of a ciphertext, respectively. If each slot value follows the identical distribution, the tagged information can be replaced as scalar values $\mu_{\mathsf{m}} \in \mathbb{C}$ and $v_{\mathsf{m}}, v_{\mathsf{e}} \in \mathbb{R}$.

The distribution of messages and errors after operations depends on the messages' actual distribution and the dependencies between messages. However, it is challenging to know the exact correlation between the messages and their distribution after several homomorphic operations. It is shown through the implementation in Section 6 that errors in the encrypted data can strictly be managed.

Worst Case Assumption One might argue that a high-probability upper bound and the minimax approximation should be used as someone other than the data owner does not know the message distribution. However, it is reasonable to assume that the computing party knows minimal information about the distribution of the message, the mean and variance, for the following reasons. First of all, in previous studies, stochastic assumptions about the distribution of messages were used naturally, e.g., the message is within the high-probability upper bound. Second, in many applications such as deep learning, control of the distributions of intermediate node values is crucial. For example, the input and the intermediate values are usually normalized or standardized, which is vital for improving the accuracy and speed of neural network training [14, 1]. Finally, some information about the message distribution is known regardless of security, such as the integer part of plaintext coefficient after ModRaise.

If one does not even want to provide the mean and variance of the message, the server can assume the worst-case that the slot values $\boldsymbol{z} \in \mathbb{Z}^n$ follow centered Gaussian distribution with a variance they are in $[-B, B]$ with high-probability, for a given high-probability upper bound B. In other words, one should let the error variance $(B/6)^2$ and try to minimize the error variance, not the high-probability upper bound, when reordering operations. In the experimental results in Section 6, even though the worst-case scenario is used, it is shown that the error value in the proposed method is smaller than that of the prior arts [17, 23].

4.2 Error in Homomorphic Operations of the CKKS Scheme

The error analysis in homomorphic operations is shown in this subsection. The following lemmas are based on the lemmas in [10, 7, 20]. The difference is that previous works used the variances to obtain the high-probability upper bounds, but we focus on the variance of errors. As mentioned earlier, it is impossible to

accurately know the distribution of random variables that have been subjected to multiple calculations. Nevertheless, the management using approximated variances is much tighter than that of the high-probability upper bound. By using the scaling factor control in [20] and scale-invariant method in [2], encryption error and the error in the RNS-CKKS scheme introduced by approximate scaling factor is removed; hence, we ignore them.

Lemma 4.1 (RNS rescaling). *Let* $(\boldsymbol{c}, l, \Delta, \boldsymbol{\mu}_m, \boldsymbol{v}_m, \boldsymbol{v}_e)$ *be an encryption of the encoded message* $m(X) \in \mathcal{R}$ *of* $\boldsymbol{z} \in \mathbb{C}^{N/2}$*. Then*

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

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

Lemma 4.2 (RNS key-switching). *Let* $(\boldsymbol{c}, l, \Delta, \boldsymbol{\mu}_m, \boldsymbol{v}_m, \boldsymbol{v}_e)$ *be a ciphertext with respect to a secret key* $\mathsf{sk}' = (1, s')$ *and let* $\mathsf{swk} \leftarrow \mathsf{KSGen}_{\mathsf{sk}}(s')$*, where* $\mathsf{sk} = (1, s)$*. Then* $\left(\boldsymbol{c}', l, \Delta, \boldsymbol{\mu}_m, \boldsymbol{v}_m, \boldsymbol{v}_e + \frac{1}{P^2}\boldsymbol{v}_{ks} + \boldsymbol{v}_{scale}\right)$ *is a valid ciphertext with respect to* sk *for the same message, where* $\boldsymbol{c}' \leftarrow \mathsf{KS}_{\mathsf{swk}}(\boldsymbol{c})$ *and* $\boldsymbol{v}_{ks} = \frac{dnum \cdot p^{\alpha}}{12}\sigma N$*.*

Lemma 4.3 (Addition and multiplication). *Let* $\left(\boldsymbol{c}_i, l, \Delta_i, \boldsymbol{\mu}_{m,i}, \boldsymbol{v}_{m,i}, \boldsymbol{v}_{e,i}\right)$ *be two independent encryptions of the encoded messages* $m_i(X)$ *of values* $\boldsymbol{z}_i \in \mathbb{C}^{N/2}$ *for* $i = 1, 2$*, and let* $\boldsymbol{c}_{add} \leftarrow \mathsf{Add}(\boldsymbol{c}_1, \boldsymbol{c}_2)$ *and* $\boldsymbol{c}_{mult} \leftarrow \mathsf{Mult}_{evk}(\boldsymbol{c}_1, \boldsymbol{c}_2)$*. Then,*

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

and

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

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

Lemma 4.4 (Addition and multiplication by constant). *Let* $(\boldsymbol{c}, l, \Delta, \boldsymbol{\mu}_m, \boldsymbol{v}_m, \boldsymbol{v}_e)$ *be an encryption of the encoded message* $m(X)$ *of* $\boldsymbol{z} \in \mathbb{C}^{N/2}$*. For a constant tuple* $\boldsymbol{a} \in \mathbb{C}^{N/2}$*, let* $\boldsymbol{c}_{cadd} \leftarrow \mathsf{cAdd}(\boldsymbol{c}_1, \boldsymbol{a}; \Delta)$ *and* $\boldsymbol{c}_{cmult} \leftarrow \mathsf{cMult}(\boldsymbol{c}_1, \boldsymbol{a}; \Delta')$*, where* $\boldsymbol{c}_{cadd}$ *and* $\boldsymbol{c}_{cmult}$ *correspond to the constant multiplication and addition with constant* $\boldsymbol{a}$*, scaled by* Δ'*, respectively. Then,*

$$(\boldsymbol{c}_{cadd}, l, \Delta, \boldsymbol{\mu}_m + \Delta\boldsymbol{a}, \boldsymbol{v}_m, \boldsymbol{v}_e)$$

and

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

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

When the sparse packing method [7] is applied, N in the above lemmas can be replaced by $2n$ when there are n slots.

Table 3. Second moment of $T_i(x)$ when x follows the Gaussian distribution with zero mean and variance σ^2 for $\sigma = 1/6$ and $1/24$, so x is highly probable to be in $[-1, 1]$.

	i	0	1	2	3	4	5	6	7
$E[T_i(x)^2]$	$\sigma = 1/6$	1.00	0.028	0.898	0.200	0.704	0.376	0.569	0.465
	$\sigma = 1/24$	1.00	0.002	0.993	0.015	0.973	0.042	0.941	0.078

4.3 Reordering Homomorphic Operations

In this subsection, we show some rules and examples for homomorphic operation reordering. Using Lemmas 4.1 to 4.4, one can reorder homomorphic operations to minimize the error variance. The main advantage of reordering homomorphic operations is that the errors in the encrypted data are reduced without compromising security and time. Considering the error propagation in the CKKS scheme, a small difference in building blocks has an enormous impact on the error variance as the depth of the circuit increases. It is worth noting that the most interesting application of the CKKS scheme, deep learning has a deep depth.

Polynomial Basis With Smaller Magnitude of Error Here, we present to reduce the error in the encrypted polynomial basis by operation reordering. The implementation shows that, for example, the error variance for T_{74} is reduced to $1/1973$ by using the proposed method. As will be mentioned later, the error in the polynomial basis, namely the basis error, is amplified by the evaluated polynomial coefficients. Therefore, it is crucial to reduce the basis error for polynomial evaluation. Polynomials are frequently used not only for bootstrapping, but also in various applications using HE [21, 11, 12], for example, an activation function of a neural network.

As the Chebyshev polynomial basis will be used in the later sections, we use here the Chebyshev polynomial basis as an example. For depth and simplicity, $T_n(x)$ is usually obtained by as follows:

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

where k is the greatest integer satisfying $2^k < n$. Let $\boldsymbol{c}_i$ be a ciphertext of message $T_i(x)$ with scaling factor Δ, and it contains error e_i Then, the error in $\boldsymbol{c}_{i+j}$ obtained by $\boldsymbol{c}_{i+j} = 2\mathsf{Mult}(\boldsymbol{c}_i, \boldsymbol{c}_j) - \boldsymbol{c}_{|i-j|}$ is

$$(2T_i(x)e_j + 2T_j(x)e_i)\Delta + 2e_ie_j - e_{|i-j|} \tag{1}$$

by Lemma 4.3. We can see that the dominant term of error variance in (1) is

$$Var[T_i(x)e_j] = E[T_i(x)^2]Var[e_j]. \tag{2}$$

Since T_i's are not independent, calculating the exact error distribution of encrypted T_i's is challenging, but roughly according to (2) and Table 3, we can see that the error multiplied by the even term tends to remain, and the error multiplied by the odd term tends to decrease. As a simple example, it is assumed

in Table 3 that the input messages follow the Gaussian distribution with zero means, and it is shown that $E[T_i(x)^2]$ is close to one when i is an even number for low-degree polynomials. Meanwhile, $E[T_i(x)]$ is zero and $Var[T_i(x)]$ is a small value when i is an odd number. Thus, the error is large when multiplication of even terms is performed when calculating the Chebyshev basis. Therefore, when n is even, T_n should be calculated by $T_n(x) = 2T_{2^k-1}(x) \cdot T_{n+1-2^k} - T_{2^{k+1}-n-2}$ rather than $T_n(x) = 2T_{2^k}(x) \cdot T_{n-2^k} - T_{2^{k+1}-n}$. Also, it is noted that, for the above reasons, the power-of-two polynomials have a large basis error.

In practice, basis errors in the lower-degree polynomial are crucial in homomorphic polynomial evaluation because the evaluation algorithms such as the BSGS algorithm mostly utilize the low-degree terms. Most importantly, the higher degree terms are obtained from lower degree terms, and the error propagates along with homomorphic operations. Moreover, in bootstrapping, the input distribution is much more concentrated in the center than that of Table 3 and thus, the proposed reordering method is quite efficient.

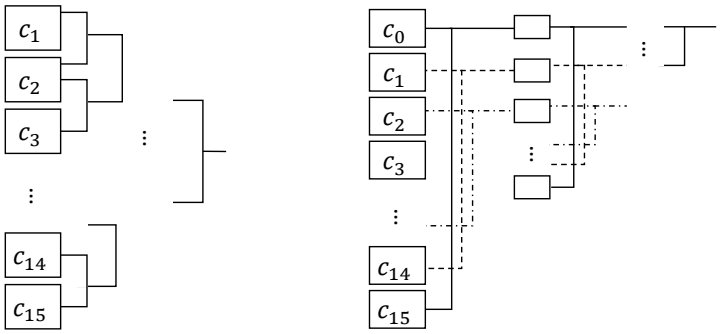
Lazy Rescaling and Lazy Relinearization We propose to do the rescaling (RS) and relinearization ($\mathsf{RL_{evk}}$) as lazy as possible. This reduces the error and computation time introduced by RS and $\mathsf{RL_{evk}}$. Since RS and $\mathsf{RL_{evk}}$ are performed after plaintext-ciphertext (scalar) multiplications and additions, more ring multiplication and addition operations are required. However, since the computation time of RS and $\mathsf{RL_{evk}}$, and ring addition and multiplication differs by orders of magnitude, the total computation time is much improved. For example, by using this method, we can roughly reduce the computation time for EvalMod by 42 percent.

In Kim *et al.*'s method [20], the scaling factor of a ciphertext is a value around p^2, and it is rescaled right before a multiplication, not after a multiplication. We generalize their method as to do RS as lazy as possible. In other words, the ciphertext can have any scaling factor, such as $\approx p^3$ or p^4, and the RS is done when it is necessary. The RS is done before non-scalar multiplication by the following rules: i) the scaling factor of at least one of the two ciphertexts must be $\approx p$, ii) ciphertext modulus of two ciphertexts should be the same.

Moreover, we propose to do the most time-consuming part of the multiplication, $\mathsf{RL_{evk}}$, as lazy as possible, namely the lazy relinearization. Specifically, after a non-scalar multiplication, the ciphertext is kept in three-tuple (d_0, d_1, d_2) corresponding to the secret key $(1, s, s^2)$. Then, addition and scalar multiplication are done without $\mathsf{RL_{evk}}$. Right before a non-scalar multiplication, it is linearized.

By the lazy rescaling and relinearization, we might reduce the computation introduced by RS and $\mathsf{RL_{evk}}$, preventing the amplification of error. The main idea is that RS and $\mathsf{RL_{evk}}$ introduce error and have a distributive property. Thus, we reduce the error amplification and the number of RS and $\mathsf{RL_{evk}}$ by doing it lazily.

If $\mathsf{RL_{evk}}$ is not done before addition or scalar multiplication, the amount of ring addition and multiplication increases. However, since these operations are much lighter than $\mathsf{RL_{evk}}$, it is generally beneficial to delay $\mathsf{RL_{evk}}$ unless we have to make extensive additions. For example, when performing an approximate



(a) Multiply in order of magnitude of messages with error variance $2^{44.4}$

(b) Multiply the larger and smaller ones first with error variance $2^{21.5}$

Fig. 1. Two different methods of obtaining $\prod_{i=0}^{15} \boldsymbol{c}_i$.

polynomial of order 255 in bootstrapping, this method can reduce $\mathsf{RL}_{\mathsf{evk}}$ and RS by 37 and 86 percent, respectively, while the number of polynomial multiplication and addition are increased by 16 and 19 percent, respectively. As a result, the total computation time is reduced by 42 percent when $N = 2^{15}$ is used on SEAL.

Successive Multiplication of Ciphertexts with Distinct Magnitudes of Messages When multiplying many ciphertexts, it is easy to see that the error can be reduced by pairing the largest and smallest values and multiplying them first. We assume there are 16 ciphertexts with level l as

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

for $i = 0, \ldots, 15$, where $\Delta = 2^{30}$ and $n = 2^{14}$. We compare two ways to obtain the multiplication $\prod_{i=0}^{15} \boldsymbol{c}_i$ in Fig. 1. The resultant message and time are the same, but the errors variances are $2^{44.4}$ for Fig. 1(a) and $2^{21.5}$ for the other.

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

i) Mean and variance of the message should be considered when we find the polynomial basis.
ii) Rescaling and relinearization should be done as lazy as possible.
iii) The error can be reduced by pairing the largest and smallest values and multiplying them first when successively multiplying ciphertexts.

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

5 Optimal Approximate Polynomial and Bootstrapping of the CKKS Scheme

The CKKS scheme supports addition and multiplication, and thus only polynomials can be evaluated. Hence, approximate polynomials are used to replace non-polynomial functions, such as ReLU in machine learning and min/max, in real-world applications [12, 18]. This subsection proposes a new method to find the optimal approximate polynomial, specially designed for the CKKS scheme considering the basis error.

5.1 Polynomial Basis Error and Polynomial Evaluation in the CKKS Scheme

Let $\{\phi_0(x), \phi_1(x), \ldots, \phi_n(x)\}$ denote a polynomial basis of degree n. When a polynomial $f(x) = \sum c_i \phi_i(x)$ is evaluated homomorphically, it is expected that the result is $f(x)+e$ for a small error e. In the CKKS scheme, there exists error in encrypted data, and thus, each $\phi_i(x)$ contains independent $e_{\mathsf{b},i}$ due to rounding and encryption errors, namely the basis error. Thus, the output is given as

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

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

In conclusion, the magnitude of c_i's should be controlled when we find an approximate polynomial of any non-polynomial functions. High-degree approximate polynomials have large coefficients in general. There have been series of studies in approximate polynomials in the CKKS scheme [10, 6, 17, 2, 23, 19, 24], but the error amplified by coefficients were not considered in the previous studies.

5.2 Variance-Minimizing Polynomial Approximation

The approximate polynomial can be optimized by minimizing the variance of the approximation error, rather the minimax approximate polynomial [23, 17]. As mentioned, the basis error is amplified by coefficients of the approximate polynomial. Thus, the magnitude of its coefficients should not be large values and using the generalized least squares method, the optimal coefficients vector $\boldsymbol{c}^*$ of the approximate polynomial is obtained as

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} \left(Var[e_{\mathsf{aprx}}] + \sum w_i \boldsymbol{c}_i^2 \right) \tag{3}$$

for weight constants w_i, where w_i's are determined by basis error and e_{aprx} is the approximation error. We call the proposed approximate polynomial obtained by (3) as the *error variance-minimizing approximate polynomial*, and there exists an analytic solution.

Significantly, the variance-minimizing approximate polynomial is more efficient than the minimax approximate polynomial for the CKKS bootstrapping. Considering SlotToCoeff, the error in the j-th slot is given as $e(\zeta_M^j) = \sum_{i=0}^{N-1} e_i \cdot \zeta_M^{ji}$, which is the sum of independent and identically distributed random variables, $e_i \cdot \zeta_M^{ji}$. Hence, the upper bound provided by the minimax polynomial is quite loose, as described previously, and it does not minimize the bootstrap error. Instead, minimizing the error variance of each coefficient minimizes errors in slot values after bootstrapping. This implies that minimizing the variance of the approximate error is optimal to reduce the bootstrap error of the CKKS scheme. The error variance-minimizing approximate polynomial is described in detail by taking bootstrapping as a specific example in the next subsection.

5.3 Optimal Approximate Polynomial for Bootstrapping and Magnitude of Its Coefficients

The most depth-consuming and noisy part of the CKKS bootstrapping is EvalMod. In this subsection, we show how to find the optimal approximate polynomial for EvalMod in a variance-minimizing manner. We focus on the direct-approximation of modulus reduction rather than trigonometric function to minimize the bootstrap error and depth. However, we note that the proposed error variance-minimizing method can be applied to arbitrary functions. Nevertheless, since the proposed method has a lower depth, it is possible to perform EvalMod faster by the parallelization method described later.

By scaling the modulus reduction function by $\frac{1}{q}$, we define $f_{\mathsf{mod}}(t) = t - i$ if $t \in I_i$, that is, $f_{\mathsf{mod}} : \bigcup_{i=-K+1}^{K-1} I_i \to [-\epsilon, \epsilon]$, where $I_i = [i - \epsilon, i + \epsilon]$ and i is an integer such that $|i| < K$. Here, ϵ denotes the ratio of the maximum coefficient of the message polynomial and the ciphertext modulus, that is, $|m_i|/q \leq \epsilon$, where m_i denotes a coefficient of $m(X)$. Let T be the random variable of input t of $f_{\mathsf{mod}}(t)$. Then, $T = R + I$, where R is the random variable of the rational part of t, r, and I is the random variable of i, for $t \in I_i$. It should be noted that $\Pr_T(t) = \Pr_I(i) \cdot \Pr_R(r)$ is satisfied for $t = r + i$ as i and r are independent and $\bigcup_i I_i = [-\epsilon, \epsilon] \times \{0, \pm 1, \ldots, \pm(K-1)\}$, where $\Pr_T, \Pr_I$, and $\Pr_R$ are probability mass functions or probability density functions of T, I, and R, respectively.

The approximation error for t is given as

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

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

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

where the mean of e_{aprx} is zero by Lemma 5.1. It is noted that the integral can be directly calculated or replaced by the sum of discretized values as in [24].

Lemma 5.1. *There exists an approximate polynomial of degree n, $p(t)$, such that minimizes $Var[f(t)-p(t)]$ for a function f and satisfies $E[f(t)-p(t)]=0$.*

Proof. Let $p'(t)$ be a polynomial of degree n that minimizes $Var[f(t)-p'(t)]$, and $E[f(t)-p'(t)]=\mu$. polynomial $p(t)=p'(t)+\mu$ always satisfies

$$Var[f(t)-p(t)] = E[(f(t)-p'(t))^2]-\mu^2 = Var[f(t)-p'(t)],$$

where $E[f(t)-p(t)]=0$. □

In the CKKS scheme, large coefficients amplify the basis error, and thus an approximate polynomial with small coefficients is required. Let the approximate polynomial $p(t)=\sum_{k=0}^{n} c_k\phi_k(t)$, and then we find $\boldsymbol{c}^*$ such that

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} \left(Var[e_{\mathsf{aprx}}] + \sum_{i=0}^{n} w_i c_i^2 \right), \tag{4}$$

and the solution satisfies

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

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

It is noted that the variance of error in each basis $\phi_i(t)$, namely basis error, may differ by i. Hence, a precise adjustment of the magnitude of polynomial coefficients can also be made using multiple weight constants, w_i's. Theorem 5.1 states that we can find the approximate polynomial for $p(t)$ efficiently; the computation time of solving this system of linear equations is the same as that of finding an interpolation polynomial for given points, which is faster than the modified Remez algorithm [23].

Theorem 5.1. *There exists a polynomial-time algorithm that finds $\boldsymbol{c}=(c_0,\ldots,c_n)$ satisfying*

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

Proof. From Lemma 5.1, we have $E[e_{\mathsf{aprx}}]=0$, and thus it is satisfied that $Var[e_{\mathsf{aprx}}]=E[e_{\mathsf{aprx}}^2]=E[f_{\mathsf{mod}}(t)^2]+2E[f_{\mathsf{mod}}(t)\cdot p(t)]+E[p(t)^2]$. By substituting $p(t)=\sum_{k=0}^{n} c_k\phi_k(t)$, we have

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

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

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

$$(\mathbf{T} + w\mathbf{I}) \cdot \boldsymbol{c} = \boldsymbol{y}, \tag{5}$$

where

$$\mathbf{T} = \begin{bmatrix} E[\phi_0\phi_0] & E[\phi_0\phi_1] & \dots & E[\phi_0\phi_n] \\ E[\phi_1\phi_0] & E[\phi_1\phi_1] & \dots & \vdots \\ \vdots & & \ddots & \vdots \\ E[\phi_n\phi_0] & E[\phi_n\phi_1] & \dots & E[\phi_n\phi_n] \end{bmatrix} \text{ and } \boldsymbol{y} = \begin{bmatrix} E[f_{\mathsf{mod}}\phi_0] \\ E[f_{\mathsf{mod}}\phi_1] \\ \vdots \\ E[f_{\mathsf{mod}}\phi_n] \end{bmatrix}.$$

As $E[\phi_i\phi_j]$ and $E[f_{\mathsf{mod}}\phi_i]$ are integral of polynomials, easily calculated. Also, the equation can be simplified by the linear transformation from monomial basis to ϕ, and thus, the approximation of other functions is also easy. □

5.4 Reducing Complexity and Error Using Odd Function

When the approximate polynomial is an odd function, we can save time finding and homomorphically evaluating the approximate polynomial. Moreover, by omitting the even-degree terms, we can reduce the approximate and basis error.

Variance-Minimizing Polynomial for an Odd Function This subsection shows that the variance-minimizing polynomial of an odd function is an odd function to utilize that $f_{\mathsf{mod}}(t)$ is an odd function. Using an odd function, we can reduce the error and computation time to find and evaluate the polynomial. First of all, when both $f_{\mathsf{mod}}(t)$ and the approximate polynomial, the sizes of vector and matrix in (5) become half and 1/4, respectively, as we only care about the odd-degree coefficients. Second, when obtaining each element of (5), we only need to integrate over the positive domain. Next, the number of operations to evaluate the approximate polynomial can also be reduced by omitting even-order terms when using the odd-BSGS algorithm in Algorithm 1. Finally, the amplified basis error is also reduced as only half of the terms are added. In the following sections, odd approximate polynomials are obtained and implemented.

Theorem 5.2 shows that when the target function of polynomial approximation such as $f_{\mathsf{mod}}(t)$ is odd and the probability density function is even, the error variance-minimizing approximate polynomial is also an odd function.

Theorem 5.2. *If* $\Pr_T(t)$ *is an even function and* $f(t)$ *is an odd functions, the error variance-minimizing approximate polynomial for* $f(t)$ *is an odd function.*

Odd Baby-Step Giant-Step Algorithm In this subsection, we propose to use an algorithm to efficiently evaluate odd polynomials over the CKKS scheme in Algorithm 1, namely the odd-BSGS algorithm. By omitting unnecessary even-order terms, we can reduce the number of non-scalar multiplication. For example, when we evaluate a polynomial of degree 255 by using ordinary BSGS algorithm in [17], 35 non-scalar multiplications are required; however, when we use the odd-BSGS algorithm [23], 30 non-scalar multiplications are required. Moreover,

Algorithm 1 Odd-BSGS Algorithm + Lazy Relinearization

Instance: A ciphertext $\boldsymbol{c}$ of t, the BSGS coefficients $\boldsymbol{d} = (d_{0,1}, d_{0,3}, \ldots, d_{2^l-1,k-1})$.
Output: A ciphertext encrypting $p(t)$.

1: Let l be the smallest integer satisfying $2^l k > n$ for an even number k.
2: **procedure** SetUp($\boldsymbol{c}, l, k$) ▷ Do not rescale or relinearize $\boldsymbol{c}_i$'s if unnecessary.
3: $\boldsymbol{c}_{\mathsf{cheb}\,i} \leftarrow$ encryption of $T_i(t)$ ▷ **only for all odd** $i < k$ and i's to find such c_i's.
4: $\boldsymbol{c}_{\mathsf{cheb}\,2^i k} \leftarrow$ encryption of $T_{2^i k}(t)$ ▷ for $0 \le i < l$.
5: **end procedure**
6: **procedure** BabyStep($\boldsymbol{b}, \boldsymbol{c}_i$'s, l, k) ▷ No rescaling here
7: $\boldsymbol{c}_{\mathsf{poly}_i^0} \leftarrow \sum_{j\in\{1,3,\ldots,k-1\}} d_{i,j}\boldsymbol{c}_{\mathsf{cheb}\,j}(t)$ ▷ baby polynomials.
8: **end procedure**
9: **procedure** GiantStep(p_i^0's, l, k)
10: $\boldsymbol{c}_{\mathsf{poly}_i^{j+1}} \leftarrow \boldsymbol{c}_{\mathsf{poly}_{2i}^{j}} + \boldsymbol{c}_{\mathsf{poly}_{2i+1}^{j}} \cdot \boldsymbol{c}_{2^j k}$
▷ $\boldsymbol{c}_{\mathsf{poly}_{2i}^{j}}$: not linearized, rescaled / $\boldsymbol{c}_{2^j k}, \boldsymbol{c}_{\mathsf{poly}_{2i+1}^{j}}$: linearized, rescaled for mult.
11: Recursively, calculate $\boldsymbol{c}_{\mathsf{poly}_0^{l}}$
12: **return** $\boldsymbol{c}_{\mathsf{poly}_0^{l}}$
13: **end procedure**

by the lazy relinearization method, the number of $\mathsf{RL}_{\mathsf{evk}}$ is reduced to 22. (For depth optimal evaluation [2], three more $\mathsf{RL}_{\mathsf{evk}}$ is required.) This is impressive because in conventional bootstrapping [2], we do 20 $\mathsf{RL}_{\mathsf{evk}}$ to compute the 52nd order polynomial and twice of the double-angle formula, even though it has 2^7 times greater bootstrap error variance than our results.

The odd-BSGS algorithm with lazy rescaling and relinearization is given in Algorithm 1. We note that the methods in [2, 20] should be applied for optimal depth and scale-invariant evaluation, but we omitted it for the sake of brevity. Also, Mult, cMult, and RS are replaced by $\cdot$ or comments. The BSGS coefficients are pre-computed for optimal parameters k and l, which minimizes the number of $\mathsf{RL}_{\mathsf{evk}}$, where $2^l \cdot k \ge deg(p(t))$. The basic blocks of the odd-BSGS algorithm are polynomials of degree less than k, namely baby polynomials, $p_i^0(t) = \sum_{j\in\{1,3,\ldots,k-1\}} d_{i,j}T_j(t)$ for $i = 0, 1, \ldots, 2^l - 1$. For example, when $l = 2$, we have

$$p(t) = p_0^2(t) = p_0^1(t) + p_1^1(t) \cdot T_{2k}(t), \tag{6}$$

where $p_i^1(t) = p_{2i}^0(t) + p_{2i+1}^0(t) \cdot T_k(t)$. It is shown that the coefficients of $p_i^0(t)$'s amplify the basis error, and thus the basis error of degree $\le k$ is crucial.

Moreover, we should reduce the magnitude of $\boldsymbol{d}$. Let $\boldsymbol{c} = (c_1, c_3, \ldots, c_n)$ be a vector of coefficients for Chebyshev basis, in other words, $p(t) = \sum c_i T_i(t)$. Then, $\boldsymbol{c}$ and $\boldsymbol{d}$ have the following linearity:

$$\boldsymbol{c} = \mathbf{L} \cdot \boldsymbol{d} = \begin{bmatrix}\mathbf{A}_{2^{l-1}k}\end{bmatrix} \cdot \begin{bmatrix}\mathbf{A}_{2^{l-2}k} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{2^{l-2}k}\end{bmatrix} \cdots \begin{bmatrix}\mathbf{A}_k & & \\ & \ddots & \\ & & \mathbf{A}_k\end{bmatrix} \cdot \boldsymbol{d}, \tag{7}$$

where

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

$\mathbf{I}_{k/2}$ is the $k/2 \times k/2$ identity matrix, and $\mathbf{J}_{k/2}$ is the $k/2 \times k/2$ exchange matrix.

5.5 Generalization of Weight Constants and Numerical Method

In this subsection, we generalize the amplified basis error term $\sum_{i=0}^{n} w_i \boldsymbol{c}_i^2$ and find the optimal approximate polynomial for odd-BSGS algorithm. The numerical method to select the weight constant is also proposed.

Generalization of Weight Constant Let $v_{\mathsf{b},i}$ be the variance of basis error for the encryption of $T_i(x)$. Then, the basis errors are multiplied by d_i and the amplified error is given as $\sum_{i=0}^{n} d_i^2 v_{\mathsf{b},i}$. Considering the approximation error, it is noted that a large scaling factor $\Delta_{\mathsf{bs}} = O(q)$ is multiplied to the result of EvalMod [7, 24]. For the sake of brevity, we let $\Delta_{\mathsf{bs}} = q$; by letting its scaling factor q, the slot values after CoeffToSlot become m_i/q. Hence, the approximation error is given as $q^2 \cdot Var[e_{\mathsf{aprx}}]$.

Let E_p be a function of $\boldsymbol{d}$, which is the variance of basis error amplified by coefficients $\boldsymbol{d} = (d_{0,1}, d_{0,3}, \ldots, d_{2^{l}-1,k-1})$. We simplify E_p by a heuristic assumption that T_i's are independent and the encryptions of $T_k(t), \ldots, T_{2^{l-1}k}(t)$ have small error. Let $\hat{T}_i$ be the product of all $T_{2^j k}$'s multiplied to p_i^0 in the giant step, for example, $\hat{T}_0 = 1$ and $\hat{T}_3 = T_k T_{2k}$ in (6). Considering the error multiplied by $d_{i,j}$, $e_j \cdot \hat{T}_i$ is the dominant term as T_i has zero mean and a small variance for small and odd integer i as in Table 3. Thus, it can say that $E_p \approx \sum_i \sum_j d_{i,j}^2 E[\hat{T}_i^2] v_{\mathsf{b},j}$, a quadratic function of $\boldsymbol{d}$. In other words, we have $E_p = \boldsymbol{d}^{\intercal} \mathbf{H} \boldsymbol{d}$, where $\mathbf{H}$ is a diagonal matrix that $\mathbf{H}_{ki+j,ki+j} = E[\hat{T}_i^2] v_{\mathsf{b},j}$. Thus, (4) is generalized as

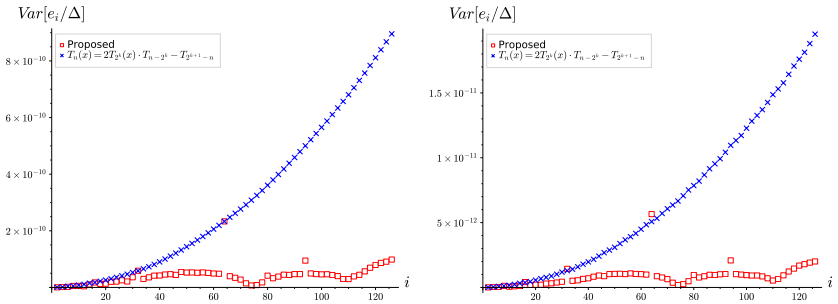
$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} \left(Var[e_{\mathsf{aprx}}] + \frac{1}{q^2} E_p \right).$$

Equation (7) gives us that the optimal coefficient $\boldsymbol{d}^*$ satisfies

$$\left(\mathbf{L}^{\intercal} \mathbf{T} \mathbf{L} + \frac{1}{q^2} \mathbf{H} \right) \boldsymbol{d}^* = \mathbf{L}^{\intercal} \boldsymbol{y}.$$

Numerical Method of Finding Optimal Approximate Polynomial Instead of finding E_p, a simple numerical method can also be used. In practice, the numerical method shows good error performance in the implementation in Subsection 6.2. We can let $w_i = w$ for all i and find w numerically. When w increases, the magnitude of coefficients decreases and $Var[e_{\mathsf{aprx}}]$ increases, and thus its sum is a convex function of w. The magnitude of the basis errors that are amplified by coefficients $\boldsymbol{d}$ has the order of the rounding error whose variance is $\frac{N(h+1)}{12}$. After multiplying $\|\boldsymbol{d}\|_2$ with the variance, it is added to $q^2 \cdot Var[e_{\mathsf{aprx}}]$. In other words, we adjust w to minimize

$$Var[e_{\mathsf{aprx}}] + w \cdot \|\boldsymbol{d}\|_2^2, \tag{8}$$



(a) Result when $\epsilon = 2^{-10}$ and $q = 2^{40}$ (b) Result when $\epsilon = 2^{-5}$ and $q = 2^{45}$

Fig. 2. Variance of basis error in $T_i(t)$ for even i using HEAAN (a) and SEAL (b) with various parameters, where $h = 64$.

where $w \approx \frac{N(h+1)}{12 \cdot q^2}$. The odd-BSGS coefficients $\boldsymbol{d}$, which minimizes (8), satisfies

$$(\mathbf{L}^\intercal \mathbf{T} \mathbf{L} + w\mathbf{I})\boldsymbol{d} = \mathbf{L}^\intercal \boldsymbol{y}.$$

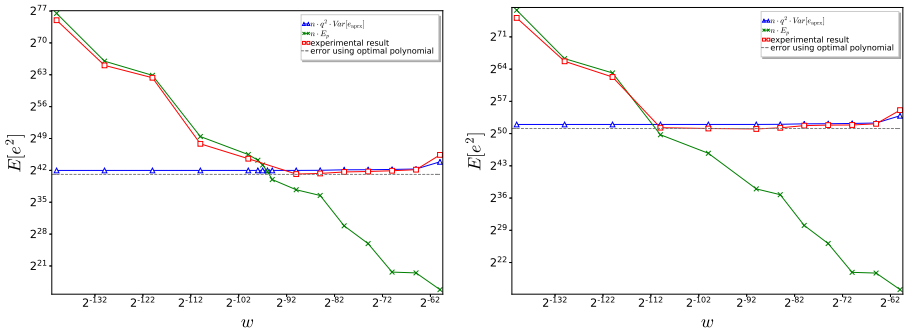
We can fine-tune w by a numerical method of performing bootstrapping and measure the bootstrap error variance, and then adjust w. Once we decide on $\boldsymbol{d}$, it becomes just part of the implementation; one can even hard-wire it.

6 Implementation of the Proposed Method and Performance Comparison

The proposed methods are implemented on the well-known libraries of the CKKS schemes, HEAAN and SEAL. We compare the experimental error of the Chebyshev polynomial with and without the proposed operation reordering. Finding error variance-minimizing approximate polynomial is implemented by SageMath, and the bootstrapping using the proposed approximate polynomial is implemented on SEAL. In this section, several implementation results and comparisons for the previous bootstrapping algorithms are also presented.

6.1 Basis Error Variance Minimization

In this subsection, we show how to find a small-error Chebyshev polynomial basis. Fig. 2 shows the variance of error in $T_i(t)$ for even i's, where t is the output value of CoeffToSlot. Square mark and × mark legends are the results with and without operation reordering, respectively. In other words, the proposed method uses $T_i = 2 \cdot T_{2^k-1} \cdot T_{i-2^k+1} - T_{2^{k+1}-2-i}$ to calculate encryption of T_i for even numbers i. It can be seen that the basis error is significantly improved by reordering operations to find ciphertext of $T_i(t)$. In particular, for $T_{74}(t)$, the variance of error for the proposed method becomes smaller by $1/1973$ compared to that without minimization. As shown in Subsection 4.3, multiplication of two even-degree terms should be avoided when we calculate the even-degree terms.



(a) Theoretical variance of errors and experimental result when $p = 2^{40}$

(b) Theoretical variance of errors and experimental result when $p = 2^{45}$

Fig. 3. Theoretical mean-square of the approximation error, amplified basis error, and experimental results, implemented in HEAAN. Polynomials of degree 81 are used.

6.2 Weight Constant and Minimum Approximate Error Variance

In Subsection 5.5, we discussed analytic solution and numerical method for variance-minimizing approximate polynomial. In this subsection, these methods are implemented and verified. We confirm that the numerical method finds a polynomial that is very close to and has a slightly larger error than that of the optimal one, and $w \approx \frac{(h+1)2n}{12}$.

The experimental results are shown in Fig. 3 with parameter $N = 2^{16}, h = 64$, and $n = 2^3$. The blue lines with triangular legend show the error by polynomial approximation as $n \cdot q^2 \cdot Var[e_{\mathsf{aprx}}]$. The green lines with × mark legend show the amplified basis errors as $n \cdot E_p$, and the red lines with square legend are for the mean square of errors obtained by experiments. The reason for multiplying the above result by n is because in SlotToCoeff, the element size of the matrix $\mathbf{U}$ is 1, and n vectors multiplied by this value are summed. The gray dot line is the variance of bootstrap error achieved by the analytic solution of the variance-minimizing approximate polynomial of the same degree, which is the lower bound of bootstrap error. For the worst-case assumption, we assume that m is distributed uniformly at random.

In Fig. 3, the sum of blue lines with triangular legend and green lines with × mark legend meets the red lines with the square legend. In other words, it shows that the theoretical derivation and experimental results are agreed upon. It can also be seen that it is possible to obtain an approximate polynomial with a small error with the proposed numerical method, but the error is slightly larger than that of the analytical solution. It is noted that the variance of the rescaling error is $\frac{(h+1)2n}{12}$, and the optimal w is close to it.

6.3 Comparison of the Proposed Bootstrapping and Prior Arts

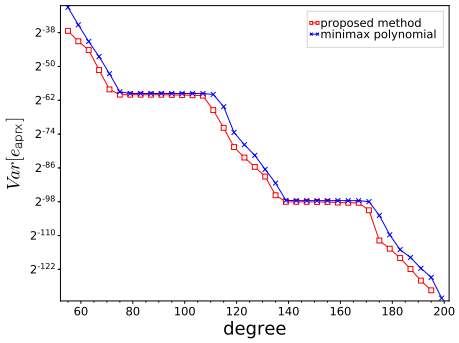


Fig. 4. Comparison of the minimum achievable variance of approximation error of the proposed method and that of the minimax polynomial, when $\epsilon = 2^{-10}$.

Comparison of Approximate Error Variance We compare the error variance of the proposed method and the minimax approximate polynomial [23] in Fig. 4. We deliberately set a harsh condition to our method in the experiment. In other words, for finding polynomials and measuring variance, we use distinct distributions of messages: uniform and Gaussian distributions, respectively. However, Fig. 4 shows that despite the harsh condition, our method achieves less error. Therefore, we can see that the probability distribution of I is a dominant term of approximate error variance. As the distribution of I is given regardless of security or input message as in Table 1, we can always improve bootstrapping through variance-minimizing approximate polynomial.

Experimental Result of Bootstrap Error The experimental results of bootstrap errors, implemented on SEAL, using various methods are compared in Table 4. In this table, the proposed variance-minimizing polynomial directly approximates $f_{\mathsf{mod}}(t)$, and the previous methods approximate the sine function and use the double-angle formula. For a very high precision achieved by a method in [23], $\frac{1}{2\pi}\arcsin(t)$ is evaluated, which consumes at least two more levels. The scale-invariant evaluation [20, 2] is also applied to the previous methods.

In Table 4, the first four rows are with $h = 64$ and $n = 2^3$ for a very accurate bootstrapping, and we can see that the proposed method has a much less bootstrap error compared to the prior arts with the same depth. In contrast, the previous methods have limited precision or require more depth of bootstrapping to evaluate arcsin. The rests of the rows are implementation results for fine-tuned parameter sets with a bootstrap error variance at most $2^{-27.58}$ and the maximum message coefficients 16. This corresponds to the 16-bit fixed-point precision: 4-bit above and 12-bit below the decimal point. As the bottleneck of error in the CKKS scheme is bootstrapping, we determine the precision of the system through the variance of bootstrap error. [15] states that fixed-point arithmetic with 4-bit precision above and 12-bit precision below the decimal point is enough to train a neural network. The variance of error by stochastic rounding

Table 4. Comparison of the variance of bootstrap error of the proposed variance-minimizing polynomial and prior arts. The error variance is obtained by 2^{16} samples of experiments implemented on SEAL. The last four rows are fine-tuned parameters set for 16-bit precision. The proposed method and [2] achieve the similar precision even though the proposed method uses an 8 times smaller q. Thus, more room for level.

algorithm	h	N	n	$\log p$	$\log q$	$\log \epsilon$	EvalMod $\cos$ r $\sin^{-1}$	depth	#relin	$Var[e]$	precision
[17]	64	2^{16}	2^3	50	60	-10	52 2 -	**8**	20	$2^{-41.90}$	**0+19.65**
[23]		2^{16}	2^3	50	53	-3	52 2 17	**13**	-	$2^{-68.32}$	**0+32.86**
proposed	64	2^{16}	2^3	50	58	-10	f_{mod}: 111	**7**	17*	$2^{-38.73}$	**0+18.07**
			2^3	50	58	-8	f_{mod}: 255	**8**	25*	$2^{-68.30}$	**0+32.86**
[2] + [23]	192	2^{16}	2^{14}	57	50	-3	75 2 31	14	-	$2^{-73.43}$	0+38.51
[2]	192	2^{16}	2^{14}	39	53	-10	52 2 -	8	20	$2^{-29.56}$	4+12.99
			2^{10}	35	49	-10	52 2 -	8	20	$2^{-29.65}$	4+13.03
proposed	192	2^{16}	2^{14}	36	49	-9	f_{mod}: 255	8	25*	$2^{-28.22}$	4+12.32
			2^{10}	32	45	-9	f_{mod}: 255	8	25*	$2^{-28.83}$	4+12.62
[2]	192	$\underline{2^{15}}$	2^{14}	**39**	**53**	-10	52 2 -	8	20	$2^{-29.41}$	4+12.91
			2^{10}	**35**	**49**	-10	52 2 -	8	20	$2^{-29.64}$	4+13.02
proposed	192	$\underline{2^{15}}$	2^{14}	**36**	**49**	-9	f_{mod}: 255	8	25*	$2^{-28.52}$	4+12.47
			2^{10}	**32**	**45**	-9	f_{mod}: 255	8	25*	$2^{-29.01}$	4+12.71

*lazy relinearization applied

Table 5. Fine-tuned parameters for 16-bit accuracy of the proposed method and prior arts, where $N = 2^{15}$

algorithm	n	$\log Q_L P$	$\log P$	$\log p$	$\log q$	precision	CtS	Mod	StC	rem. level
[2]	2^{14}	767	54	39	53	4+12.91	53	424	44	4
	2^{10}		50	35	49	4+13.02	49	392	40	6
proposed	2^{14}	767	50	36	49	4+12.47	49	392	44	6
	2^{10}		46	32	45	4+12.71	45	360	40	8
proposed + new CtS	2^{14}	767	50	36	49	4+12.47	0	392	37	8
	2^{10}		46	32	45	4+12.71	0	360	40	10

with 12-bit precision below decimal point is given as $\frac{1}{12}(2^{-12})^2 = 2^{-27.58}$. It is shown that the proposed method achieves the error variance of $2^{-29.01}$ with scaling factor $\approx 2^{32}$ and $\epsilon = 2^{-9}$, while the previous methods require scaling factor $\approx 2^{35}$ and $\epsilon = 2^{-10}$. Therefore, previously, it was challenging to use $N = 2^{15}$, as at most 4 to 6 levels remain after bootstrapping; we improve this by the proposed variance-minimizing approximate polynomial and operation reordering. At 16-bit precision, [23] was excluded because performing arcsin is a waste of level.

Table 5 shows the fine-tuned parameter sets, their precision, and the remaining level. CtS, Mod, and StC correspond to CoeffToSlot, EvalMod, and SlotToCoeff, respectively, in the table. Using the proposed method leaves much more levels after bootstrap: $4 \to 6$ (50%) for $n = 2^{10}$, $6 \to 8$ (33%) for full slots. Hence, we can significantly reduce the required number of bootstrapping. If there are many levels, the linear transformations could be performed faster by consuming additional levels [17, 16], but it is assumed here that such a method is not used. However, as the parameter size itself has been reduced from $N = 2^{16}$ to 2^{15}, bootstrapping is performed quickly without this hybrid method,

Table 6. Computation time and key size by N in SEAL , tested in single core, Intel Xeon Silver 4210 CPU @ 2.20GHz, 64 GB ram. This experiment is for full-level ciphertexts with parameter $p = 2^{36}, q = 2^{45}$, and $h = 192$, with 128-bit security. The result may differ by ciphertext level.

	$\mathsf{SKGen}_{\mathsf{sk}}$	RS	$\mathsf{RL}_{\mathsf{evk}}$	rotation	Add	cMult	swk	pk
$N = 2^{15}$	30,628 ms	28 ms	322 ms	1,547 ms	1.51 ms	18.58 ms	111 MB	6.5 MB
$N = 2^{16}$	333,096 ms	144 ms	3,417 ms	19,294 ms	8.57 ms	23.01 ms	1054 MB	27.6 MB

and moreover, it is more suitable for parallelism. We note that $n = 2^{14}$ is a nice parameter as one channel of 128×128 images fits in one ciphertext.

In Table 6, the comparison of time and key sizes by N is given. The table shows that by reducing N from 2^{16} to 2^{15}, we can significantly reduce the computation time and key sizes; the time for rotation is reduced by $1/12$, and the key-switching key size is reduced by $1/10$. It is worth noting that when N is larger, more key-switching keys are required, and more rotations are required for bootstrapping even though the slot size is the same. Therefore, the advantage of memory, communication, and time by using $N = 2^{15}$ outweigh the benefit of the number of levels by using $N = 2^{16}$. The advantage is even more crucial for the clients as they have limited computational resources. We note that Han and Ki's method reduces the key sizes, but it trades the ciphertext level [17].

It is worth noting that all the previous techniques, such as the double-angle formula and composition with arcsin, can also be utilized along with variance-minimizing approximate polynomial. Although the variance-minimizing approximate polynomial gives us a nice approximation of f_{mod}, it is observed that its performance gets worse at a higher degree. This is because it is challenging to approximate with the limited magnitude of coefficients. Using the double-angle formula, we can narrow the approximation domain, and trigonometric functions are suitable to approximate, and thus the use of the double-angle formula and the inverse sine function is useful for a very accurate bootstrapping. However, instead of previous approximations, we should use the variance-minimizing approximation for a more accurate approximation of the trigonometric functions as the proposed variance-minimizing approximation improves the approximation by using the distribution of I. It is advantageous to approximate more accurately where the probability is high. Moreover, it reduces the basis error and the approximate error at the same time.

One might argue that the proposed bootstrapping may fail because the approximation error is too large where the probability is very low. In fact, it is not; by experiments, we check that the maximum approximation error of the error variance-minimizing polynomial is also small.

Further Level Saving With New SlotToCoeff In Table 5, one or two level gain is achieved through new SlotToCoeff in the last two rows. This section presents a new SlotToCoeff algorithm that does not consume a ciphertext level; the proposed SlotToCoeff trades the remaining level and computation

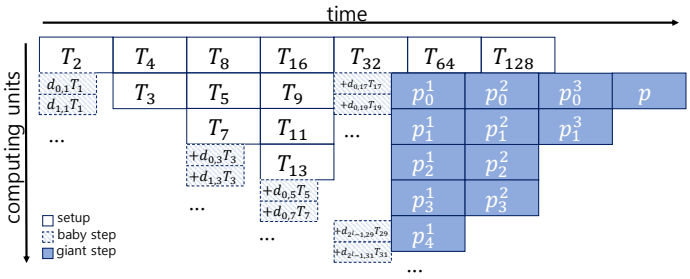


Fig. 5. Parallelization of Algorithm 1 for 255th degree polynomial, $k = 32$, and $l = 3$.

time. In ordinary ModRaise [7], the ciphertext modulus is raised to $Q_{L_0+L_1}$, but we propose to change the ciphertext modulus to $P'Q_{L_0+L_1}$ for $P' \leq P$.

For a ciphertext whose modulus is greater than $Q_{L_0+L_1}$, key switching is not allowed, and thus we cannot rotate it. However, we still can perform scalar multiplication, rescaling, addition, and coefficient permutation (without key switching). The following procedure performs CoeffToSlot, but the ciphertext modulus after it is $Q_{L_0+L_1}$, not $Q_{L_0+L_1-1}$. First, perform rotations on the ciphertext, and then multiply the corresponding diagonal terms of the matrix $\mathbf{U}^{-1}$ with scaling factor P', and then rescale it. Then, we can and should perform key switching for rotation. The key switching is composed of *Decompose*, *MultSum*, and *ModDown* [2]; we change the order of *ModDown* and ciphertext addition. In other words, we decompose the ciphertexts whose coefficients are permuted and multiplied by the diagonal components of $\mathbf{U}^{-1}$, perform *MultSum*, and then add them all together. Then, only one *ModDown* is required for the proposed CoeffToSlot and the memory consumption is $O(1)$. However, since matrix-BSGS [2] cannot be used in this method, it still requires much computation.

Parallelism and The Proposed Method In the proposed method, instead of evaluating consecutive low-degree polynomials, we compute the high-degree polynomial once by the odd-BSGS algorithm. Although the proposed lazy rescaling and relinearization significantly reduce the computation, we briefly introduce the parallelization of the BSGS algorithm as it can be applied along with Algorithm 1 and further improve the performance. In HE applications, the server generally has a significant computational resource, and thus we can effectively utilize the resources by parallelization. Thus, the advantages of parallel computing outweigh the disadvantages of a few additional multiplications. The proposed method has a lower depth than the method using arcsin, and thus despite more multiplications, it can be calculated faster through parallelization.

In Fig. 5, a diagram of parallelization of Algorithm 1 is given. The Chebyshev basis with the same depth can be computed in parallel as they have no dependency. Hence, we can obtain all the Chebyshev basis for the baby step before calculating $\boldsymbol{c}_{\mathsf{cheb}2^i k}$'s. When we use lazy rescaling, the baby step does not require rescaling or relinearization, and thus it is evaluated quickly. Moreover,

the low-degree terms of baby polynomials can be summed in advance before the high-degree terms so that the baby step can be done with the setup process in parallel. The giant step is done as follows: the first step is the product of the baby polynomial and $\boldsymbol{c}_{\mathsf{cheb}\,k}$, the next step is the product of the first step's output and $\boldsymbol{c}_{\mathsf{cheb}\,2k}$, and so on. Therefore, the first step can be combined with finding $\boldsymbol{c}_{\mathsf{cheb}\,2k}$. Each step of the giant step and computing a $\boldsymbol{c}_{\mathsf{cheb}\,2^i k}$ both require one non-scalar multiplication and one addition, and thus they take the same time.

7 Conclusion

In this paper, we introduced two novel methods to improve the precision and complexity of the CKKS scheme. First, SNR, a widely-used measure of error performance of noisy media such as communication systems, was adopted as a measure of message quality. To maximize the SNR of encrypted data, we proposed a method to minimize the variance of errors; we replaced the high-probability upper bound in the tagged information with the variance of errors for tight management of error. As a result, the homomorphic operations were effectively reordered to minimize the error variance. Second, we proposed a method to find the optimal approximate polynomial for the CKKS scheme in the same aspect of reducing the error variance. Especially, the newly proposed operation reordering and variance-minimizing approximate polynomial were applied to the bootstrapping of the CKKS scheme. The proposed two contributions allowed us to leave more levels after bootstrapping, and thus, we reduced the degree of ciphertext polynomial of the CKKS scheme from 2^{16} to 2^{15} with practical precision for deep learning. The sizes of the key and ciphertext are reduced by at least $1/4$, which significantly reduces their computational complexity.

From its implementation on SEAL, it is shown that the bootstrap error of the CKKS scheme was significantly improved. To our best knowledge, this is the first bootstrapping algorithm that contemplates various parameters, such as slot size, the error characteristics of the CKKS scheme, and the polynomial evaluation algorithm. Even though the number of non-scalar multiplication is slightly increased, we could reduce the computation time by the proposed lazy rescaling and relinearization. Moreover, the lower depth and error have two main advantages: the use of $N = 2^{15}$ and parallelization.

In this work, 16-bit fixed-point arithmetic with rounding was referenced as a practical parameter. However, the error shape is different in the CKKS scheme; random noise is added to the message. Thus, it is worth further researching how much error is tolerated when learning and inference in noisy environments like the CKKS scheme. From our proposed method, now there are two criteria to reorder homomorphic operations when we use the CKKS scheme: error variance reduction and computation time reduction. In addition to the examples in this paper, there are various methodologies to reorder homomorphic operations to minimize the error, and it will affect the error performance of the CKKS scheme significantly for deeper operations. In the field of optimizing compilers, there has

been much research on instruction reordering, such as peephole optimization [22], and we leave the adoption of compiler techniques as future work.

References

1. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint (2016)
2. Bossuat, J.P., Mouchet, C., Troncoso-Pastoriza, J., Hubaux, J.P.: Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. IACR Cryptol. ePrint Arch (2020)
3. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) Fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) **6**(3), 1–36 (2014)
4. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Proceedings of Annual International Cryptology Conference. pp. 505–524. Santa Barbara, CA, USA (2011)
5. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. SIAM Journal on Computing **43**(2), 831–871 (2014)
6. Chen, H., Chillotti, I., Song, Y.: Improved bootstrapping for approximate homomorphic encryption. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 34–54 (2019)
7. Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: Bootstrapping for approximate homomorphic encryption. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 360–384. Tel Aviv, Israel (2018)
8. Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: A full RNS variant of approximate homomorphic encryption. In: Proceedings of International Conference on Selected Areas in Cryptography. pp. 347–368. Calgary, Canada (2018)
9. Cheon, J.H., Hhan, M., Hong, S., Son, Y.: A hybrid of dual and meet-in-the-middle attack on sparse and ternary secret lwe. IEEE Access **7**, 89 497–506 (2019)
10. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Proceedings of International Conference on the Theory and Application of Cryptology and Information Security. pp. 409–437 (2017)
11. Cheon, J.H., Kim, D., Kim, Y., Song, Y.: Ensemble method for privacy-preserving logistic regression based on homomorphic encryption. IEEE Access **6**, 46 938–948 (2018)
12. Chou, E., Beal, J., Levy, D., Yeung, S., Haque, A., Fei-Fei, L.: Faster cryptonets: Leveraging sparsity for real-world encrypted inference. arXiv preprint (2018)
13. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptol. ePrint Arch. (2012)
14. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), http://www.deeplearningbook.org
15. Gupta, S., Agrawal, A., Gopalakrishnan, K., Narayanan, P.: Deep learning with limited numerical precision. In: International conference on machine learning. pp. 1737–1746. PMLR (2015)
16. Han, K., Han, M., Cheon, J.H.: Improved homomorphic discrete Fourier transforms and FHE bootstrapping. IEEE Access **7**, 57 361–370 (2019)
17. Han, K., Ki, D.: Better bootstrapping for approximate homomorphic encryption. In: Proceedings of Cryptographers' Track at the RSA Conference. pp. 364–390. San Francisco, CA, USA (2020)

18. Hesamifard, E., Takabi, H., Ghasemi, M.: Cryptodl: Deep neural networks over encrypted data. arXiv preprint (2017)
19. Jutla, C.S., Manohar, N.: Modular Lagrange interpolation of the mod function for bootstrapping for approximate HE. IACR Cryptol. ePrint Arch (2020)
20. Kim, A., Papadimitriou, A., Polyakov, Y.: Approximate homomorphic encryption with reduced approximation error. IACR Cryptol. ePrint Arch (2020)
21. Kim, M., Song, Y., Wang, S., Xia, Y., Jiang, X.: Secure logistic regression based on homomorphic encryption: Design and evaluation. JMIR medical informatics **6**(2), e19 (2018)
22. Lattner, C., Adve, V.: LLVM: a compilation framework for lifelong program analysis & transformation. In: Proceedings of International Symposium on Code Generation and Optimization. pp. 75–86. Palo Alto, CA, USA (2004)
23. Lee, J.W., Lee, E., Lee, Y., Kim, Y.S., No, J.S.: High-precision bootstrapping of RNS-CKKS homomorphic encryption using optimal minimax polynomial approxiamtion and inverse sine function. IACR Cryptol. ePrint Arch (2020)
24. Lee, Y., Lee, J.W., Kim, Y.S., No, J.S.: Near-optimal polynomial for modulus reduction using l2-norm for approximate homomorphic encryption. IEEE Access **8**, 144 321–330 (2020)
25. Li, B., Micciancio, D.: On the security of homomorphic encryption on approximate numbers. IACR Cryptol. ePrint Arch (2020)
26. Mason, J.C., Handscomb, D.C. (eds.): Chebyshev Polynomials, chap. Chebyshev interpolation, pp. 154–172. CRC Press, FL, USA (2002)
27. Son, Y., Cheon, J.H.: Revisiting the hybrid attack on sparse and ternary secret LWE. IACR Cryptol. ePrint Arch. (2019)

Appendices

A Comparison of Approximate Error Variance of the Proposed Method and Error Variance of Minimax Polynomial

We put additional comments on Fig. 4 in this appendix. The best-known approximation method for the CKKS bootstrapping so far is the modified Remez algorithm [23], which finds the minimax approximate polynomial. The minimax approach is reasonable when the input distribution is unknown. However, in the CKKS bootstrapping, the input distribution is partially known; the probability mass function of I is known regardless of security as in Table 1. Therefore, our variance-minimizing approximate polynomial is effective as it utilizes stochastic information. Finding the variance-minimizing approximate polynomial, we use the worst-case assumption that r is uniformly distributed. However, in the actual experiment of measuring error variance for the given approximate polynomials, we assume r follows the Gaussian distribution. In other words, we experiment in a harsh environment for our proposed method by using different message distributions for finding polynomial and calculating variance.

It is shown in Fig. 4 that the variance of the approximation error for the proposed method is less than that of minimax polynomial, despite the harsh condition. Therefore, we can see that the probability distribution of I is a dominant term of approximate error variance. As the distribution of I is given regardless of security or input message, we can always improve bootstrapping by using our method. As the magnitude of the approximate polynomial coefficients cannot be reduced in the modified Remez algorithm, the approximate polynomials for both methods are compared without controlling the magnitude of coefficients in Fig 4. In other words, the variance-minimizing approximation with $w = 0$ in (3) is used in Fig. 4. However, unlike the prior approaches, it is possible to reduce the approximate polynomial coefficients in the proposed method with slightly increased error variance. In contrast, the use of the double-angle formula is essential for previous methods, which results in more depth.

B Proofs of Lemmas

Proof (Theorem 5.2). Existence and uniqueness: Equation (4) is a quadratic polynomial for the coefficients $\boldsymbol{c}$, and thus there exists one and only solution.
Oddness: Let P_m be the subspace of the polynomial of degree at most m and $f_m(t)$ denote the unique element in P_m that is closest to $f(t)$ in the variance of difference. Then, $Var[-f(-t) - p(t)] + \sum w_i c_i^2$ is minimized when $p(t) =$

$-f_m(-t)$, because

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

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

□

Proof (Lemma 4.1). The rescaled ciphertext $\boldsymbol{c}_{\mathsf{RS}} \leftarrow \left\lfloor p_l^{-1}\boldsymbol{c} \right\rceil$ satisfies $\langle \boldsymbol{c}_{\mathsf{RS}}, \mathsf{sk}\rangle = p_l^{-1}(m+e) + e_{\mathsf{scale}}$, where $e_{\mathsf{scale}} = \langle \boldsymbol{\tau}, \mathsf{sk}\rangle$ and $\boldsymbol{\tau} = (\tau_0, \tau_1)$ is the rounding error vector. We can assume that the coefficients of polynomials τ_0 and τ_1 are distributed uniformly at random on $p_l^{-1}\mathbb{Z}_{p_l}$, and thus the variance of $\tau_0 + \tau_1 \cdot s$ is $N/12 + hN/12$. Therefore, the variance of e_{scale} is given as $\frac{1}{12}(h+1)N$.

In RNS-CKKS, the scaling factor is slightly different depending on the operations done to the ciphertext, and thus when adding different ciphertexts, an error occurs according to the ratio of p_l and p in the process of forcibly treating the scaling factor as p. The methods to remove such error was proposed in [20, 2]. □

Proof (Lemma 4.2). The key switching noise comes from the rounding terms $\boldsymbol{\tau}$ as in Lemma 4.1 and from the error terms e'_k in swk_0. The variance of error from $\boldsymbol{\tau}$ is $\boldsymbol{v}_{\mathsf{scale}}$. The other error is given as

$$\frac{\langle \mathcal{WD}_l(c_1), \{e'_k\}_{0\le k<\mathsf{dnum}-1}\rangle}{P}. \tag{9}$$

It can be assumed that the i-th component of $\mathcal{WD}_l(c_1)$ follows uniform distribution in $\tilde{Q}_i$. Then, its variance is $\frac{\tilde{Q}_i}{12}$ and the variance of each coefficient of e'_k is σ^2. Thus, the variance of error in (9) is derived as

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

□

Proof (Lemma 4.3). The addition is trivial. The ciphertext of $m_1(X) \cdot m_2(X)$

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

contains additional error $e'' = P^{-1} \cdot d_2 e'$ and the error by scaling. As $d_2 = a_1 a_2$ and from RLWE assumption, we can assume that d_2 is distributed uniformly

at random on $\mathcal{R}_{Q_l}$. Thus, the variance of Pe'' is derived as $Q_l^2 N/12 \cdot \sigma^2 N = \frac{1}{12} Q_l^2 \sigma^2 N^2$. The total error is given as

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

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

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

□

Proof (Lemma 4.4). The rounding during encoding introduces a rounding error. However, we could assume that the scaling factor is large enough so that there are no errors. Then, it is self-evident. □

C Example of Error Variance Reduction by Lazy Rescaling

For example, when one calculates encryption of $a\boldsymbol{x}^8$ by using $\boldsymbol{c}$, which is the encryption of $\boldsymbol{x}$ with scaling factor $\approx \Delta^2$, previously, the calculation was done as [20]

$$\begin{aligned} \boldsymbol{c}_2 &\leftarrow \mathsf{Mult}(\mathsf{RS}(\boldsymbol{c}), \mathsf{RS}(\boldsymbol{c})) \\ \boldsymbol{c}_4 &\leftarrow \mathsf{Mult}(\mathsf{RS}(\boldsymbol{c}_2), \mathsf{RS}(\boldsymbol{c}_2)) \\ \boldsymbol{c}_8 &\leftarrow \mathsf{Mult}(\mathsf{RS}(\boldsymbol{c}_4), \mathsf{RS}(\boldsymbol{c}_4)) \\ \boldsymbol{c}_{output} &\leftarrow \mathsf{cMult}(\mathsf{RS}(\boldsymbol{c}_8), a; \Delta), \end{aligned} \tag{10}$$

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

$$\begin{aligned} \boldsymbol{c}_a &\leftarrow \mathsf{cMult}(\mathsf{RS}\left(\boldsymbol{c}\right), a^{1/8}; \Delta) \\ \boldsymbol{c}_{2a} &\leftarrow \mathsf{Mult}(\mathsf{RS}(\boldsymbol{c}_a), \mathsf{RS}(\boldsymbol{c}_a)) \\ \boldsymbol{c}_{4a} &\leftarrow \mathsf{Mult}(\mathsf{RS}(\boldsymbol{c}_{2a}), \mathsf{RS}(\boldsymbol{c}_{2a})) \\ \boldsymbol{c}_{8a} &\leftarrow \mathsf{Mult}(\mathsf{RS}(\boldsymbol{c}_{4a}), \mathsf{RS}(\boldsymbol{c}_{4a})). \end{aligned} \tag{11}$$

When $a\boldsymbol{x}^8$ is obtained using (10), the error introduced by RS is amplified by a unlike (11). However, when a is an integer, it is not necessary to consume level; in other words, a is not scaled by Δ, and thus (10) may be advantageous in terms of depth. In that case, unless a is a prime number, depth and error can be reduced simultaneously by multiplying the factors of a in advance.

Equation (11) can be improved further by replacing the first line in (11) as

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

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

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

Then, from Lemma 4.1, the full ciphertext of $\mathsf{RS}(\boldsymbol{c})$ is expressed as

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

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

$$\left(\boldsymbol{c}_a, l-1, \Delta^2, a^{1/8}\boldsymbol{\mu}_{\mathsf{m}}, a^{1/4}\boldsymbol{v}_{\mathsf{m}}, a^{1/4}\boldsymbol{v}_{\mathsf{e}} + a^{1/4}\Delta^2\boldsymbol{v}_{\mathsf{scale}}\right). \tag{12}$$

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

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

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

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

We note that $\boldsymbol{v}_{\mathsf{scale}}$ (C) is negligible after a RS, but $a^{1/4}\Delta^2\boldsymbol{v}_{\mathsf{scale}}$ in (12) is not. In (11), the RS error introduced by $\mathsf{RS}(\boldsymbol{c})$ is amplified by a, but we can even rule out this by skipping RS.