High-Precision Approximate Homomorphic Encryption by Error Variance Minimization [⋆]

Yongwoo Lee[1], Joon-Woo Lee[1], Young-Sik Kim[2], HyungChul Kang[3], and 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. The recent development of machine learning and cloud computing arises a new privacy problem; how can one outsource computation on confidential data? Homomorphic encryption (HE) is a solution for that as it allows computation on encrypted data without decryption. The Cheon-Kim-Kim-Song (CKKS) scheme (Asiacrypt '17) is one of the highlighted fully homomorphic encryption (FHE) schemes as it is efficient to deal with encrypted real numbers, which are the usual data type for many applications such as machine learning. This paper proposes a generally applicable method to achieve high-precision approximate FHE using the following two techniques. 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 in the CKKS scheme. For that, the error variance is minimized instead of the upper bound of error when we deal with encrypted data. Second, we propose a novel polynomial approximation method for the CKKS scheme from the same perspective of minimizing error variance. We especially apply the approximation method to the bootstrapping of the CKKS scheme, where we achieve the smaller error variance in the bootstrapping compared to the prior arts. The performance improvement of the proposed methods for the CKKS scheme is verified by implementation over HE libraries: HEAAN and SEAL. The implementation results show that the message precision of the CKKS scheme is improved by reordering homomorphic operations and using the proposed polynomial approximation. Specifically, the proposed method uses only depth 8, although the bootstrapping precision is increased by 1 bit compared to that of the previous method using depth 11. We also suggest a loose lower bound of bootstrapping error in the CKKS scheme and show that the proposed method's bootstrapping error is only 2.8 bits on average larger than the lower bound. Therefore, various applications' quality of services using the proposed CKKS scheme, such as privacy-preserving

[⋆⋆⋆] Y.-S. Kim is the corresponding author.

[⋆] Supported by Samsung Advanced Institute of Technology.

machine learning, can be improved without compromising performance and security.

Keywords: Bootstrapping· Cheon-Kim-Kim-Song (CKKS)· Cryptography· Fully homomorphic encryption (FHE)· Privacy-preserving machine learning (PPML)· Residue number system-CKKS (RNS-CKKS).

1 Introduction

The recent development of machine learning, cloud computing, and blockchain arises a new 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 on encrypted data without decryption. Using HE, one can process encrypted data without decryption or leaking information.

The Cheon-Kim-Kim-Song (CKKS) scheme is an approximated homomorphic encryption scheme [1] using ring-learning with error (RLWE). The CKKS scheme [1] 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 and regression. When we deal with arbitrary precision real numbers using other FHE schemes such as (B)FV [16, 17, 20] and BGV [18] schemes, the size of ciphertext has an exponential growth rate according to the level, where the level of ciphertext is defined by the maximum depth of operation that can be homomorphically evaluated without bootstrapping. However, the ciphertext size has a polynomial growth rate according to the level in the CKKS scheme.

The CKKS scheme provides the trade-off between the efficiency and accuracy of messages, where messages in the CKKS scheme contain errors, and the errors accumulate during homomorphic operations. To our best knowledge, research to the date has provided high-probability upper bounds for errors in encrypted data [1, 2, 7]. As the processing of messages in the CKKS scheme proceeds, the upper bound of errors in encrypted data increases, and thus it becomes a loose and useless bound.

In previous studies on the error management of the CKKS scheme, the probabilistic concept has been adopted to some extent. Error control in the CKKS scheme so far used the high-probability upper bounds of error [1, 2] or average precision of message [3]. The high-probability upper bounds are derived from the distribution of error, and the average precision of message provided in [1–4, 7] is the expectation of the absolute value of error. In other words, it is not unfamiliar to use probabilistic methods for error analysis.

The CKKS scheme can be considered as a noisy channel. Thus we adopt in this paper the methodologies from communication theory, which are the power ratio of the signal (message) and error. The signal power can be controlled by the scaling factor of the message, and we show how to minimize the noise power during approximate operations in the CKKS scheme. Since the errors in the CKKS system are additive, errors are assumed to be Gaussian distributed due to the central limit theorem. Moreover, as shown in the later chapter, the high-probability upper bound becomes quite loose after successive operations. Therefore, it is better to control the variance of errors rather than the high-probability upper bound of errors and keep them as tagged information for the ciphertext.

Since a drawback of the CKKS scheme is that errors are accumulated, many studies have been conducted to reduce errors. Recently, Kim et al. proposed a new method reducing errors in encrypted data of the CKKS scheme and its residue number system (RNS) variants using lazy rescaling and different scaling factors at each level [7]. Although the error was reduced in their paper, the high-probability upper bound was still used as a measure of error. Especially, error amplification during the bootstrapping in the CKKS scheme has been studied in many research. After the first bootstrapping method was proposed in [2], the Chebyshev interpolation method has been applied to the homomorphic evaluation of modulus reduction [4]. Then, a technique for direct-approximation was proposed in [5], and the algorithm of finding minimax approximate polynomial and inverse sine method was presented in [6]. A bootstrapping algorithm for RNS-CKKS is proposed in [8], but this scale-invariant polynomial evaluation method can be generalized to the result of [7].

In this paper, we propose a method of managing the variance of errors to maximize the signal-to-noise ratio (SNR) of the messages in the CKKS scheme rather than minimizing the high-probability upper bounds of error. There are mainly two contributions in this paper. First, to minimize the error variance of the encrypted data, a criterion for optimizing the order of homomorphic operations is proposed. In the proposed method, the error variance of the CKKS scheme is treated as a value to be minimized rather than the upper bound of error, and by doing so, one can manage the error in the encrypted data tightly. Hence, one can improve the stability of various applications that use approximate homomorphic encryption by reordering homomorphic operations. Besides, the accuracy of the resultant message can also be improved. The second contribution is the optimization of the approximate polynomials in terms of the error variance of the encrypted data of the CKKS scheme. 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. Thus we can improve the bootstrapping algorithm of the CKKS scheme using the proposed polynomial approximation method, where bootstrapping can be implemented with smaller errors and less depth consumption. It is shown in this paper that the proposed method reduces the magnitude of bootstrapping error compared to the previous work [6]. Moreover, the proposed method resolves the problem that the approximate polynomials have large coefficients, which could only be solved using the double-angle formula in the previous work. The proposed method makes it possible to use a direct-approximation for the modulus reduction. The comparison with the previous methods shows that we can improve the message precision after bootstrapping while reducing the level consumption for bootstrapping by using the proposed method. Specifically, the proposed method uses only depth 8, while the bootstrapping precision is increased by 1 bit compared to that of the previous methods using depth 11.

The remainder of the paper is organized as follows. In Section 2, we provide the necessary notations and communication theoretic perspective on error. The CKKS scheme and its bootstrapping algorithm are summarized in Section 3. A newly proposed method of optimizing error variance of encrypted data and its analysis are given in Section 4. We provide a new method to find optimal approximate polynomials for the CKKS scheme in Section 5, focusing on bootstrapping. 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 $\boldsymbol{x}$ and every vector is a column vector. Matrices are denoted by boldfaced capital letters, for example, $\mathbf{A}$. 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 $\boldsymbol{v}$. Matrix multiplication is denoted by $\cdot$ or can be omitted when it is unnecessary. 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, it means that x is sampled uniformly at random among the set elements. Random variables are denoted by capital letters such as X. $E[X]$ and $Var[X]$ denote the mean and variance of random variable X, respectively. Some capital letters may represent something other than a random variable, such as a constant, but this is context-sensitive.

2.2 Chebyshev Polynomials

The Chebyshev interpolation is a well-known polynomial interpolation method that uses the Chebyshev polynomials as a basis of the interpolation polynomial. The Chebyshev polynomial of the first kind, in short, the Chebyshev polynomial is defined by the recursive relation [33]

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

The Chebyshev polynomial of degree n has n distinct roots in the interval $[-1, 1]$ and all its extrema are also in $[-1, 1]$. Moreover, $\frac{1}{2^{n-1}}T_n(x)$ is the polynomial, whose maximal absolute value is minimal among monic polynomials of degree n

and its absolute value is $\frac{1}{2^{n-1}}$. Hence, unlike the monomial basis, the value of the Chebyshev basis's high-degree terms do not converge to zero or diverse 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 Communication Perspective of the CKKS Scheme

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

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

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

An easy way to increase SNR is to increase signal power, but it is not easy in a real-world system due to regulation or physical constraints. That is the same for the CKKS scheme. A larger scaling factor should be multiplied to the message to increase the power of the message. However, if one uses a larger scaling factor, the ciphertext level decreases, or larger parameters should be used to keep the encryption secure under the RLWE problem. Also, usually in the RNS-CKKS scheme, the scaling factor is limited to 64 bits for efficient implementation. Hence, to increase SNR, it is essential to reduce the power of noise in the CKKS scheme rather than increase the signal's power.

The CKKS scheme trades off the efficiency of computation and accuracy of the message, and improving the accuracy will make the CKKS scheme more reliable. Error estimation of the CKKS scheme so far has been focused on the high-probability upper bound of the error, and the upper bound was tracked using the upper bound of the message [1, 2]. As the homomorphic operation continues, the error 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, hereinafter, the energy and variance of errors are abused if its mean is zero.

3 The CKKS Scheme and Its Bootstrapping

3.1 The CKKS Scheme

This section briefly introduces the CKKS scheme [1] and its RNS variant, the RNS-CKKS scheme [4, 9]. For a positive integer M, let $\Phi_M(X)$ be the M-th cyclotomic polynomial of degree N, where M is a power of two, $M = 2N$, and $\Phi_M(X) = X^N + 1$. Let $\mathcal{R} = \mathbb{Z}/\langle \Phi_M(X) \rangle$ 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 $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$.

The CKKS scheme [1] and its RNS variants [4, 7, 9] provide homomorphic operations on encrypted real number data with errors. This is done by canonical embedding and its inverse. Recall that canonical embedding Emb of $a(X) \in \mathbb{Q}/\langle \Phi_M(X) \rangle$ into $\mathbb{C}^N$ is the vector of the evaluation values a at the roots of $\Phi_M(X)$ and Emb^{-1} 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 procedure returns
$$m(X) = \mathsf{Emb}^{-1}\left(\left\lfloor \Delta \cdot \pi^{-1}(\boldsymbol{z}) \right\rceil_{\mathsf{Emb}(\mathcal{R})} \right) \in \mathcal{R},$$
where Δ is the scaling factor and $\left\lfloor \pi^{-1}(\boldsymbol{z}) \right\rceil_{\mathsf{Emb}(\mathcal{R})}$ denotes the discretization of $\pi^{-1}(\boldsymbol{z})$ into an element of $\mathsf{Emb}(\mathcal{R})$.

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

The infinity norm of $\mathsf{Emb}(a)$ for $a(X) \in \mathcal{R}$ is called the canonical embedding norm of a, denoted by $\|a\|_\infty^{\mathsf{can}} = \|\mathsf{Emb}(a)\|_\infty$. Refer [1] for the property of the canonical embedding norm.

Adopting notations in [1] and [13], we define three distributions as follows. For a real number $\sigma > 0$, $\mathcal{DG}(\sigma^2)$ denotes the distribution of vectors in $\mathbb{Z}^N$, whose entries are sampled independently from the discrete Gaussian distribution of variance σ^2. $\mathcal{HWT}(h)$ is the set of signed binary vectors in $\{0, \pm 1\}^N$ with Hamming weight h and $\mathcal{ZO}(\rho)$ denotes the distribution of vectors from $\{0, \pm 1\}^N$ with probability $\rho/2$ for each of ± 1 and probability of being zero $1 - \rho$. Suppose that we have ciphertexts of level l for $0 \le l \le L$, where level l means the maximum number of possible multiplications before bootstrapping. For convenience, we fix a power-of-two base $p > 0$ and a power-of-two modulus q and let $q_l = q \cdot p^l$. The base integer p is usually equivalent to the scaling factor Δ.

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

- $\mathsf{KeyGen}(1^\lambda)$:
  - Given the security parameter λ, we choose a power-of-two M, an integer h, an integer P, a real number σ, and a maximum ciphertext modulus Q, such that $Q \geq q_L$.
  - 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

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

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

$$\mathsf{swk} := (b', a') \in \mathcal{R}_{Pq_L}^2,$$

  where $b' = -a's + e' + Ps' \pmod{Pq_L}$.
  - Set the evaluation key as $\mathsf{evk} := \mathsf{KSGen}_{\mathsf{sk}}(s^2)$.
- $\mathsf{Enc}_{\mathsf{pk}}(m)$:
  Sample $v \leftarrow \mathcal{ZO}(0.5)$ and $e_0, e_1 \leftarrow \mathcal{DG}(\sigma^2)$.
  Output $\boldsymbol{c} = v \cdot \mathsf{pk} + (m + e_0, e_1) \pmod{q_L}$.
- $\mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{c})$:
  Output $\bar{m} = \langle \boldsymbol{c}, \mathsf{sk} \rangle$.
- $\mathsf{Add}(\boldsymbol{c}_1, \boldsymbol{c}_2)$:
  For $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{R}_{q_l}^2$, output

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

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

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

  Output

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

  where $\lfloor \cdot \rceil$ denotes the rounding operation.
- $\mathsf{cAdd}(\boldsymbol{c}_1, \boldsymbol{a}; \Delta)$:
  For a $\boldsymbol{a}\mathbb{C}^{N/2}$ and a scaling factor Δ, output

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

- $\mathsf{cMult}(\boldsymbol{c}_1, \boldsymbol{a}; \Delta)$:
  For a $\boldsymbol{a}\mathbb{C}^{N/2}$ and a scaling factor Δ, output

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

– $\mathsf{RS}_{l\to l'}(\boldsymbol{c})$:
For $\boldsymbol{c} \in \mathcal{R}_{q_l}^2$, output

$$\boldsymbol{c}_{\mathsf{RS}} = \left\lfloor \frac{q_{l'}}{q_l}\boldsymbol{c} \right\rceil \pmod{q_{l'}}.$$

The subscript is omitted when $l' = l - 1$.

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

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

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

There are computationally more efficient variants of the CKKS scheme, namely the RNS-CKKS scheme in [4, 9], and the basic operations supported therein are similar. Hence, it is worth noting that this paper's proposed methods aim for all the variants of the CKKS scheme and the original CKKS scheme.

3.2 CKKS Scheme in RNS

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

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

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

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

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

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

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

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

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

where $b'_k = -a'_k s + e'_k + P \cdot \mathcal{PW}(s')_k \pmod{Pq_L}$.
  - Set the evaluation key as $\mathsf{evk} := \mathsf{KSGen}_{\mathsf{sk}}(s^2)$.

– $\mathsf{RS}(\boldsymbol{c})$:
For $\boldsymbol{c} \in \mathcal{R}_{q_l}^2$, output

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

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

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

To remove the approximation error introduced by approximate rescaling, one can use different scaling factor for each level as given in [7].

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

3.3 Bootstrapping of the CKKS Scheme

There are several studies for bootstrapping of the CKKS scheme [2–6]. The bootstrapping consists of the following four steps: ModRaise, CoeffToSlot, EvalMod, and SlotToCoeff.

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

Putting Polynomial Coefficients in Plaintext Slots Approximate homomorphic operations are performed in plaintext slots. Thus, to deal with $t(X)$, we have to put polynomial coefficients in plaintext slots. In CoeffToSlot step, the $\mathsf{Emb}^{-1} \circ \pi^{-1}$ is performed homomorphically using matrix multiplication [2] or FFT-like operations or a hybrid method of both [3]. Then, we have two ciphertexts encrypting $\boldsymbol{z}'_0 = (t_0, \ldots, t_{\frac{N}{2}-1})$ and $\boldsymbol{z}'_1 = (t_{\frac{N}{2}}, \ldots, t_{N-1})$ (or combined using imaginary, e.g., $(t_0 + i \cdot t_{\frac{N}{2}}, \ldots, t_{\frac{N}{2}-1} + i \cdot t_{N-1})$), where t_j denotes the j-th coefficient of $t(X)$.

Evaluation of the Approximate Modulus Reduction 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 instead. For approximation, it is desirable to control the size of the message so that we can ensure $m_i \leq \epsilon \cdot q$ for a small ϵ, where m_i is a coefficient of the message polynomial $m(X)$. At first, Cheon et al. approximated the modulus reduction function as $\frac{q}{2\pi} \sin\left(\frac{2\pi t}{q}\right)$ and used an approximate polynomial for sine function using Taylor series expansion of exponential function in [2]. Hence there exists fundamental error between the approximate polynomial and modulus reduction function, that is, the difference of sine function and modulus reduction function, which is upper bounded by

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

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

After that, the method of improving polynomial approximation using Chebyshev interpolation was proposed [3]. By selecting optimized nodes for Chebyshev interpolation, Han et al. significantly improved the error performance of the approximation in the bootstrapping of the CKKS scheme [4]. However, in both approaches, the sine function is used, and thus there is still the fundamental approximation error. Then, a direct-approximation method using a discretization of the target function and the least square method is proposed in [5]. A composition with inverse sine function is proposed in [6] to remove the fundamental approximation error between the sine function and the modulus reduction. In [6], an approximation algorithm that finds the minimax approximate polynomial, namely the modified Remez algorithm, is used.

Switching Back to the Coefficient Representation SlotToCoeff is the inverse operation of CoeffToSlot.

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

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

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

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

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

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

4 Optimization of Error Variance in the Encrypted Data

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

We can assume the following statements:

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

The first assumption is straightforward and clear. If the mean of error is not zero, one can simply subtract the mean value to reduce the error. In general, the second one is also true. When we deal with approximate polynomial, the approximation error is dependent on the message. However, the approximation error is usually small compared to the message, and the covariance is negligible. From these two assumptions, the variance of error introduced by multiplication can be obtained. Moreover, from the central limit theorem, the sum of independent random variables can be approximated to a Gaussian distribution. Now, since the power of noise and the variance of error are the same, 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 for ciphertext is introduced in [1], and it is used to estimate the magnitude of the error. The tagged information is composed of a level l, where $0 \leq l \leq L$, an upper bound $v \in \mathbb{R}$ of the message, and a high-probability upper bound $B \in \mathbb{R}$ of error. The upper bound is informative when there are few homomorphic operations. However, as the homomorphic operation continues, the upper bound becomes exponentially loose and thus useless.

We take a simple example of how the upper bound becomes loose. In [1], 6σ is used as the high-probability upper bound of the error that follows Gaussian distribution with variance σ^2. The probability that $Pr(|X| > 6\sigma)$ is $2^{-27.8}$. Previously, the error upper bound of a ciphertext, which was the sum of two ciphertexts with error upper bounds B_1 and B_2, was specified as $B_1 + B_2$ [1, 7]. Assume that there are 100 distinct fresh ciphertexts and let σ_o^2 be the error variance. Then the previous tagged information states that the upper bound of error is $6\sigma_o$. Hence, the upper bound of the error in the encrypted data, which is the sum of the hundred ciphertexts, is $600\sigma_o$. However, the ciphertexts are independent, and its error follows the Gaussian distribution with a variance of $100\sigma_o^2$. Therefore, the probability of the given upper bound, $600\sigma_o$, is $Pr(|e| > 600\sigma_o) \approx 2^{-2602.1}$, which is quite loose, where e is the summation of error. The previous upper bound is too loose to obtain useful information on the error. A real application such as deep learning requires a lot more computation than this, and thus the high-probability upper bound of error becomes looser than this example. In other words, managing the upper limit of errors is practically futile.

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

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

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

The distribution of messages and errors after homomorphic operations varies depending on the actual distribution of messages and the dependency between messages. However, it is difficult to know the exact correlation of the messages, and their distribution after several homomorphic operations is quite complicated. Therefore, while managing the mean and variance values, the message and error can be roughly treated as independent Gaussian distribution. It is shown through the implementation in Section 6 that errors in the encrypted data can strictly be managed in this way too.

4.2 Worst Case Assumption

One might argue that an upper bound and minimax approximation should be used as it is not appropriate to assume that someone other than the data owner knows the mean and variance of the message, $\boldsymbol{\mu}_{\mathsf{m}}$ and $\boldsymbol{v}_{\mathsf{m}}$. However, it is entirely reasonable to assume that someone other than the data owner knows minimal information about the distribution of the message, the mean and variance, for the following reasons. First, the previously used measure of error, the high-probability upper bound, is also related to the distribution. Second, in many applications such as deep learning, control of the distributions of intermediate node values is crucial. For example, the input is usually normalized or standardized, and many methods to normalize the intermediate values are widely used [26]-[29], which is crucial for the accuracy and speed up the training of neural networks. Finally, some information about the message distribution is known regardless of security, such as the message distribution after ModRaise in bootstrapping.

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

4.3 Error in Homomorphic Operations of the CKKS Scheme

The error analysis in each operation, such as encoding, encryption, addition, multiplication, and key switching, is shown in this subsection. It should be noted that the proposed error variance minimization method can be applied for all the variants of the CKKS scheme, such as the original CKKS scheme [1], RNS variants [4, 9], and the reduced-error variants [7]. The only difference for the above variants is the variance of errors.

The following lemmas are basically based on the lemmas in [1, 2, 7]. The difference is that the lemmas so far have focused on high-probability upper bounds of errors, but in the proposed method, we focus on the variance of errors in encrypted data because the upper bounds become loose after successive homomorphic operations. Error variances in encrypted data for the original CKKS scheme and the RNS-CKKS scheme are given in the following lemmas. In the RNS-CKKS scheme, the rescaling factors q_i's are different for each level, and thus the errors in rescaling are different from that of the original CKKS scheme.

Lemma 1 (Encoding and encryption). *Given a secret key* sk *with Hamming weight* h*, we have the following variance of encryption noise in the CKKS scheme:*

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

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

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

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

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

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

$\square$

Lemma 2 (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* $\boldsymbol{z} \in \mathbb{C}^{N/2}$*, where* l *is its level. Then*

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

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

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

Lemma 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*

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

Then,

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

and

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

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

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

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

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

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

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

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

□

Lemma 4 (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* $sk' = (1, s')$ *and let* $swk \leftarrow KSGen_{sk}(s')$, *where* $sk = (1, s)$. *Then*

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

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

Proof. The proof is similar to that of Lemma 3 and thus, it is omitted. □

Lemma 5 (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*

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

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

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

and

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

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

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

The following two lemmas are slightly modified from the original lemmas in [4]. It is noted that in the RNS rescaling, the error introduced by approximate scaling factor is eliminated by managing the exact scaling factor after rescaling, Δ/p_l.

Lemma 6 (RNS rescaling). *For the RNS-CKKS scheme, 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*

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

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

Proof. The proof is the same as that of the original CKKS scheme except for the scaling factor. In RNS-CKKS, the scaling factor is slightly different depending on the operations done to the ciphertext, and thus when adding different ciphertexts, an error occurs according to the ratio of p_l and p in the process of forcibly treating the scaling factor as p. The methods to remove such error was proposed in [7, 8]. □

Lemma 7 (RNS key-switching). *For the RNS-CKKS scheme, 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 $\left(\boldsymbol{c}', l, \Delta, \boldsymbol{\mu}_m, \boldsymbol{v}_m, \boldsymbol{v}_e + \frac{1}{P^2}\boldsymbol{v}_{\mathsf{ksrns}} + \boldsymbol{v}_{\mathsf{scale}}\right)$ is a valid ciphertext with respect to a secret key sk for the same message, where $\boldsymbol{c}' \leftarrow \mathsf{KS}_{\mathsf{swk}}(\boldsymbol{c})$ and $\boldsymbol{v}_{\mathsf{ksrns}} = \frac{\mathsf{dnum} \cdot p^{\alpha}}{12}\sigma N$.

Proof. The key switching noise comes from the rounding terms $\boldsymbol{\tau}$ as in Lemma 2 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

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

It can be assumed that the i-th component of $\mathcal{WD}_l(c_1)$ follows uniform distribution in $\tilde{q}_i$. Then, its variance is $\frac{\tilde{q}_i}{12}$ and the variance of each coefficient of e'_k is σ^2. Thus, the variance of error in (1) is derived as

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

□

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

4.4 Reordering Homomorphic Operations

As the proposed method enables tight management of errors in encrypted data, homomorphic operations can be effectively reordered to reduce errors. The main advantage of reordering homomorphic operations is that the errors in the encrypted data are reduced without compromising security and performance. Using Lemmas 1 to 7, one can reorder homomorphic operations to minimize the error variance. In this subsection, we show some operations patterns that can be reordered to reduce the error in encrypted data. Considering that the error increases cumulatively in the CKKS scheme, the little differences in the following examples greatly affect the error variance as the depth of operations advances. It is worth noting that the most beneficial application of the CKKS scheme, deep learning has a deep of operations, too.

In addition to the below examples, there are many methods to reorder homomorphic operations corresponding to the inputs and the operations themselves. Thus the reordering of homomorphic operations can be done in an on-the-fly manner. In the field of optimizing compilers, there has been many researche on instruction reordering [37], and we leave the adoption of compiler techniques as future work.

Polynomial Basis With Smaller Magnitude of Error In this subsection, we propose a way to reduce the error in the encrypted polynomial basis by reordering homomorphic operations. The error in each encrypted polynomial basis depends on the order of homomorphic operations obtaining it. Polynomials are frequently evaluated not only for bootstrapping [2]-[6] but also in various applications using HE [12, 22]-[30] as all the homomorphic operations are polynomial operations, except for the rotation and conjugation.

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

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

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

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

Let $\boldsymbol{c}_i$ be the ciphertext of message $T_i(x)$ with scaling factor Δ for $i = 0, \ldots, n$. Considering that $\boldsymbol{c}_i$ contains error e_i, the error in $\boldsymbol{c}_n$ obtained by $T_n(x) = 2T_k(x) \cdot T_{n-2^k}(x) - T_{2^{k+1}-n}(x)$ is $(2T_{2^k}(x)e_{n-2^k} + 2T_{n-2^k}(x)e_{2^k})\Delta + 2e_{2^k}e_{n-2^k} - e_{2^{k+1}-n}$ by Lemma 3, if we ignore the key-switching error for brevity. When the ciphertext is rescaled by Δ, the error becomes

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

where $e_{\mathsf{scale,rs}}$ is another scaling error as described in Lemma 2. It is noted that Δ is large enough so that $(2e_{2^k}e_{n-2^k} - e_{2^{k+1}-n})/\Delta$ can be ignored. Roughly, as $T_{n-2^k}(x)$ and e_{2^k} are independent and $E[e_{2^k}]$ is zero, the variance of error $T_{n-2^k}(x)e_{2^k}$ is given as

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

Since T_i's are not independent variables, calculating the exact error distribution in encrypted T_i's is not straightforward, but roughly according to (2) and Table 1, we can see that the error multiplied by the even term tends to remain and the error multiplied by the odd term tends to decrease. In Table 1, it is shown that $E[T_i(x)^2]$ is close to one when i is an even number. Meanwhile, $E[T_i(x)]$ is zero and $Var[T_i(x)]$ is a small value when i is an odd number. That is, since even degree terms have a large mean of squares, the error is large when multiplication of even terms is used when calculating the Chebyshev basis.

Therefore, when n is even, T_n should be calculated by $T_n(x) = 2T_{2^k-1}(x) \cdot T_{n+1-2^k} - T_{2^{k+1}-n-2}$ rather than $T_n(x) = 2T_{2^k}(x) \cdot T_{n-2^k} - T_{2^{k+1}-n}$. The power-of-two degree terms T_{2^k} can only be found by the product of $T_{2^{k-1}}$, to consume the least level. Thus, the power-of-two degree terms have a large error, and it is also shown in implementation in Subsection 6.1. Now, it is clear that it is

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

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

better to avoid using the power-of-two degree terms as possible as we can when evaluating polynomials.

As a simple example, it is assumed in Table 1 that the input messages follow the Gaussian distribution with zero means. It is common to normalize or standardize the input value in deep learning [26–29], which is the most attractive application of HE. Thus we are interested in inputs that follow Gaussian distribution. Besides, x should be in $[-1, 1]$ to use Chebyshev polynomials, and thus the input is concentrated in the center (zero). Table 1 shows that the smaller the order of the messages and more centered, the larger the even terms and the smaller the odd terms.

There are several reasons why this property is essential. In practice, errors in lower degree terms are essential in homomorphic polynomial evaluation because the evaluation algorithms such as the baby-step giant-step algorithm mostly utilize the lower degree terms. Most importantly, as the higher degree terms are obtained from lower degree terms, and the error is accumulated, minimizing error in lower degree terms is crucial. Finally, in the case of bootstrapping, the input distribution is much more concentrated in the center compared to the case of examples in Table 1 and thus, the proposed reordering method is quite efficient in bootstrapping.

In Subsection 6.1, the implementation shows that the error in encrypted polynomial basis can be reduced by reordering homomorphic operations. For example, the error variance for the proposed method becomes smaller by 1/1973 compared to the case without minimization for T_{74}. We can also conclude that the modified baby-step giant-step algorithm in [6] is advantageous in terms of error because the baby-step giant-step algorithm proposed in [4] utilizes power-of-two degree terms.

Lazy Rescaling We generalize the technique to treat rescaling as a part of multiplication, which is proposed in [8] to do rescaling as lazy as possible. In the method suggested by Kim et al. [7], the scaling factor of the ciphertext is a value around Δ^2, and scaling is performed right before a multiplication, not after a multiplication. It was shown that error could be reduced and especially, the encryption error can be removed by this method [7]. We generalize this so that the ciphertext can have any scaling factor such as $\approx \Delta^3$ or Δ^4, and the rescaling is done as lazy as possible.

As shown in Lemma 2, error in encrypted data is divided by the scaling factor, and rounding error is added when the ciphertext is rescaled. The critical point is that the rescaling has a distributive property, and thus, rescaling can be reordered to reduce errors. In other words, since the rounding errors occur through rescaling in general, it should be done as lazy as possible. For example, as suggested in [7], ciphertexts can be rescaled right before a multiplication. This method prevents further amplification of rescaling errors by addition as well as reduces the number of required rescalings. Moreover, this method reduces the number of rescaling, and thus the more additions out of the total operation, the better the effect.

To reduce the error, rescaling can also be reordered with constant multiplications as well as additions. For example, when one calculating encryption of $a\boldsymbol{x}^8$ by using $\boldsymbol{c}$, which is the encryption of $\boldsymbol{x}$ with scaling factor $\approx \Delta^2$, previously, the calculation was as follows [7]

$$\begin{aligned} \boldsymbol{c}_2 &\leftarrow \mathsf{Mult}(\mathsf{RS}(\boldsymbol{c}), \mathsf{RS}(\boldsymbol{c})) \\ \boldsymbol{c}_4 &\leftarrow \mathsf{Mult}(\mathsf{RS}(\boldsymbol{c}_2), \mathsf{RS}(\boldsymbol{c}_2)) \\ \boldsymbol{c}_8 &\leftarrow \mathsf{Mult}(\mathsf{RS}(\boldsymbol{c}_4), \mathsf{RS}(\boldsymbol{c}_4)) \\ \boldsymbol{c}_{output} &\leftarrow \mathsf{cMult}(\mathsf{RS}(\boldsymbol{c}_8), a; \Delta), \end{aligned} \tag{3}$$

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

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

When $a\boldsymbol{x}^8$ is obtained using (3), the error introduced by rescaling is amplified by a unlike (4). However, when a is an integer, it is not necessary to consume level; in other words, a is not scaled by Δ, and thus (3) may be advantageous in terms of depth. In that case, unless a is a prime number, depth and error can be reduced simultaneously by multiplying the factors of a in advance.

Equation (4) can be improved further by replacing the first line in (4) as

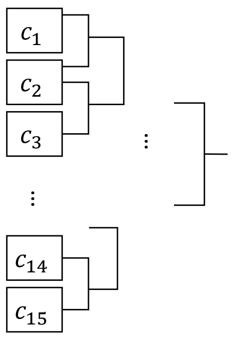
$$\boldsymbol{c}'_a \leftarrow \mathsf{RS}\left(\mathsf{cMult}(\boldsymbol{c}, a^{1/8}; \Delta)\right).$$

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

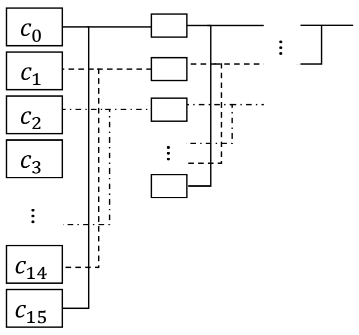
$$\left(\boldsymbol{c}, l, \Delta^2, \boldsymbol{\mu}_{\mathsf{m}}, \boldsymbol{v}_{\mathsf{m}}, \boldsymbol{v}_{\mathsf{e}}\right).$$

Then, from Lemma 2, the full ciphertext of $\mathsf{RS}(\boldsymbol{c})$ is expressed as

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



(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 $\prod_{i=0}^{15} \boldsymbol{c}_i$.

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

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

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

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

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

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

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

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

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

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

for $i = 0, \ldots, 15$, where $\Delta = 2^{30}$ and $N = 2^{14}$. We compare two ways to obtain the multiplication $\prod_{i=0}^{15} \boldsymbol{c}_i$ in Fig. 1. The results and computation time are the same. However, the variances of errors are $2^{44.4}$ for the operation in 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) Resizing should be done as lazy as possible.
iii) The error can be reduced by pairing the large and small values and multiplying the largest and smallest values first when successively multiplying ciphertexts.

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

5 Optimal Approximate Polynomial and Bootstrapping of the CKKS Scheme

Usually, HE schemes support addition and multiplication, and thus only polynomials can be evaluated. However, non-polynomial functions such as ReLU, min/max function, multiplicative inverse, and modulus reduction are frequently required in their applications [36]. Hence, approximate polynomials are used to replace those non-polynomial functions in real-world applications [24]. This subsection proposes a new method to find the optimal approximate polynomial for the CKKS scheme using the generalized least mean square method.

5.1 Polynomial Basis Error and Polynomial Evaluation in the CKKS Scheme

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

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

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

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

High-degree approximate polynomials have large coefficients in general. The outputs of the approximate polynomial for modulus reduction, which is essential for the bootstrapping of the CKKS scheme, are in $[-\epsilon, \epsilon]$, where $\epsilon = \frac{|m|}{q}$. There have been series of studies in approximate polynomials in the CKKS scheme [1]–[6], but the error amplified by coefficients were not considered in the previous studies. The magnitude of coefficients of approximate polynomial should be controlled when we find the approximate polynomial of any non-polynomial functions, as well as the modulus reduction, which deteriorates message precision by the successive homomorphic operations.

5.2 Variance Minimizing Polynomial Approximation

In the encrypted data of the CKKS scheme, errors include errors added for security, rounding errors, approximation errors, and errors added during homomorphic operations. Therefore, from the central limit theorem, the basis errors can be considered Gaussian random variables with zero means. The approximate polynomial can be optimized by minimizing the variance of the approximation error, rather than using minimax approximate polynomial [6].

As shown in Subsection 5.1, basis error is amplified by coefficients of the approximate polynomial. Thus, the magnitude of its coefficients should not be large values and using the generalized least squares method, the optimal coefficients vector $\boldsymbol{c}^*$ of the approximate polynomial is obtained as

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} \left(Var[e_{\mathsf{aprx}}] + \sum w_i \boldsymbol{c}_i^2 \right) \tag{5}$$

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 (5) as the *error variance minimizing approximate polynomial*, and there exists an analytic solution. It is noted that when w_i's are all zero, the approximate polynomial minimizes the variance of approximation error only.

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

5.3 Optimal Approximate Polynomial for Bootstrapping and Magnitude Its Coefficients

The key part of the bootstrapping of the CKKS scheme is the homomorphic evaluation of the modulus reduction. In [2], the modulus reduction is approximated by the sine function, and the approximate polynomial for the sine function is homomorphically evaluated using a Taylor series expansion and the double-angle formula. Moreover, with optimized nodes for the Chebyshev interpolation, the polynomial approximation is significantly improved [4]. The least-square method to find better approximate polynomial without trigonometric functions is proposed in [5] and the method to find minimax approximate polynomial is also proposed in [6].

In [6], the modulus reduction function, $t \pmod{q}$ is considered as

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

and the approximate polynomials for $\arcsin(\cdot)$ and $\sin(\cdot)$ are evaluated sequentially. This paper focuses on the direct-approximation of modulus reduction rather than trigonometric function approximation to minimize the bootstrapping error and depth. However, as the proposed error variance minimizing method can be applied to any function, the composite method and double-angle formula for faster evaluation in [4, 6] can also be applied to the proposed method.

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, $|m_i|/q \le \epsilon$, where m_i denotes a coefficient of message polynomial $m(X)$.

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

The approximation error in t is given as

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

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

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

where the mean of e_{aprx} is zero by Lemma 8. This gives us the following equation

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

It is noted that the integral can be directly calculated or replaced by the sum of discretized values as in [5].

Lemma 8. *Let $p(t)$ be an approximate polynomial that minimizes $Var[f(t) - p(t)]$ for a function f. Then, $E[f(t) - p(t)] = 0$ is satisfied.*

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

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

which is a contradiction.

□

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

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} \left(Var[e_{\mathsf{aprx}}] + \sum_{i=0}^{n} w_i c_i^2 \right), \tag{6}$$

and the solution satisfies

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

where $\boldsymbol{c} = (c_0, c_1, \ldots, c_n)$ 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 $\phi_i(t)$ may be different. For example, when the ciphertext of x^4 is obtained by multiplying the ciphertext of x^2, while the ciphertext of x^2 contains a rounding error $e_{\mathsf{rnd},2}$. Then, the ciphertext of x^4 has error $2e_{\mathsf{rnd},2}x^2 + e_{\mathsf{rnd},2}^2 + e_{\mathsf{rnd},4}$. In general, we can say that a high-degree term of polynomial basis causes a larger error. Hence, a precise adjustment of the magnitude of polynomial coefficients can also be made using multiple weight constants, w_i's.

Theorem 1. *There exists a polynomial-time algorithm that finds $\boldsymbol{c} = (c_0, \ldots, c_n)$ satisfying*

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

Proof. From Lemma 8, we can assume that $E[e_{\mathsf{aprx}}] = 0$. Then, we have the following equation

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

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

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

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

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

$$\mathbf{T} \cdot \boldsymbol{c} = \boldsymbol{y}, \tag{7}$$

where

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

and

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

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

Theorem 1 states that the approximate polynomial for $p(t)$ is found efficiently. In other words, the computation time of solving this system of linear equations is the same as that of finding an interpolation polynomial for given points, which is faster than the modified Remez algorithm [6].

5.4 Reducing Complexity and Error Using Odd Function

When the approximate polynomial is an odd function, one can save time to find and homomorphically evaluate the approximate polynomial. Using the fact that $f_{\mathsf{mod}}(t)$ is an odd function and the minimax approximate polynomial of an odd function is also an odd function, the approximate polynomial for $f_{\mathsf{mod}}(t)$ with only odd degree terms was derived [5, 6]. Moreover, the number of operations to evaluate the approximate polynomial can also be reduced by omitting even-order terms. Besides, the amplified basis error is also reduced as there are only half of the terms to be added when the approximate polynomial is an odd function. Theorem 2, it is shown that when the target function of polynomial approximation such as $f_{\mathsf{mod}}(t)$ is odd and the probability density function is even, the error variance minimizing approximate polynomial is also an odd function from the following theorem. In the following section, odd approximate polynomials are obtained and implemented based on the following theorem.

Theorem 2. *When* $\Pr_T(t)$ *is an even function and* $f(t)$ *is an odd function, the error variance minimizing approximate polynomial for* $f(t)$ *is an odd function.*

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

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

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

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

□

5.5 Generalization of Weight Constants and Numerical Method

Earlier it is noted that the weight constant vector $\boldsymbol{w}$ provides the trade-off between the magnitude of coefficients and variance of polynomial approximation

Algorithm 1 Generalized Odd Baby-Step Giant-Step Algorithm [6]

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

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

error. 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 baby-step giant-step algorithm. The numerical method to select the weight constant vector $\boldsymbol{w}$ is also proposed.

The basis error can be found using the method proposed in Subsection 4.4 or numerically. Let $v_{\mathsf{b},i}$ be the variance of basis error in a slot of ciphertext which is an encryption of $T_i(x)$. Then, the basis errors are multiplied by c_i and the amplified error is given as $\sum_{i=0}^{n} c_i^2 v_{\mathsf{b},i}$. Considering the approximation error, it should be noted that a large scaling factor $\Delta_{\mathsf{bs}} = O(q)$ is multiplied to the result of EvalMod [2, 5]. For brevity of description, we let $\Delta_{\mathsf{bs}} = q$; it is proper, because there are coefficients of $m(X)$, m_i's in the slots after CoeffToSlot, and by letting its scaling factor q, the slot values become m_i/q with scaling value q. Hence, the approximation error is given as $q^2 \cdot Var[e_{\mathsf{aprx}}]$. Therefore, it is optimal when w_i has the value $v_{\mathsf{b},i}/q^2$.

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

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

$$p(t) = (p_3(t)T_k(t) + p_2(t))T_{2k}(t) + p_1(t)T_k(t) + p_0(t). \tag{8}$$

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

Let E_p be a function of $\boldsymbol{d}$, which is the variance of basis error amplified by coefficients $\boldsymbol{d} = (d_{0,1}, d_{0,3}, \ldots, d_{2^l-1,k-1})$. 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 simplifies E_p. Let $\hat{T}_i$ be the product of all $T_{2^j k}$'s multiplied by p_i, for example, $\hat{T}_0 = 1$

and $\hat{T}_3 = T_k T_{2k}$ in (8). Considering the error multiplied by $d_{i,j}$, $e_j \cdot \hat{T}_i$ is the dominant term as T_i has zero mean and very small variance. Thus, it can be said that $E_p = \sum_i \sum_j d_{i,j}^2 E[\hat{T}_i^2] v_{\mathsf{b},j}$, which is a quadratic function of $\boldsymbol{d}$. In other words, we have $E_p = \boldsymbol{d}^{\mathsf{T}}\mathbf{H}\boldsymbol{d}$, where $\mathbf{H}$ is a diagonal matrix that

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

Thus, (6) is generalized as

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

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

$$\begin{aligned} \boldsymbol{c} &= \mathbf{L}\boldsymbol{d} \\ &= \begin{bmatrix} \mathbf{A}_{2^{l-1}k} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{A}_{2^{l-2}k} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{2^{l-2}k} \end{bmatrix} \cdots \begin{bmatrix} \mathbf{A}_k & & \\ & \ddots & \\ & & \mathbf{A}_k \end{bmatrix} \cdot \boldsymbol{d}, \end{aligned} \tag{9}$$

where

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

$\mathbf{I}_{k/2}$ is the $k/2 \times k/2$ identity matrix, and $\mathbf{J}_{k/2}$ is the $k/2 \times k/2$ exchange matrix. Equation (9) is derived from $T_m(x)T_n(x) = \frac{1}{2}(T_{m+n}(x) + T_{|m-n|}(x))$. Then, we have $\nabla_{\boldsymbol{c}} E_p = 2\mathbf{L}^{-1^{\mathsf{T}}}\mathbf{H}\mathbf{L}^{-1}\boldsymbol{c}$. Hence, the optimal coefficient $\boldsymbol{c}^*$ satisfies

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

Instead of finding E_p, a simple numerical method can also be used. Actually, the value of $\hat{T}_i$ is close to one, and moreover, the numerical method shows good error performance in the implementation in Subsection 6.2. We can let $w_i = w$ for all i and find the value using the numerical method for brevity. When w increases, the coefficients $\boldsymbol{c}$ decreases, and $Var[e_{\mathsf{aprx}}]$ increases, and thus its sum is a convex function of w. Thus, the optimal polynomial is found by using numerical methods by finding the optimal w. $\boldsymbol{c}$ is uniquely determined by w, and using $\boldsymbol{c}$, the coefficients for the baby-step giant-step algorithm or the Paterson-Stockmeyer algorithm can be calculated. The magnitude of the basis errors that are amplified by coefficients is similar to the rounding error whose variance is $\frac{N(h+1)}{12}$. After multiplying $\|\mathbf{L}^{-1}\boldsymbol{c}\|_2$ with the variance, it is added to $q^2 \cdot Var[e_{\mathsf{aprx}}]$. In other words, we adjust w to minimize

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

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

Table 2. Probability mass function of I.

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

The proposed method is efficient when an accurate approximation is required. In [8], a bootstrapping for the CKKS scheme with a non-sparse key was proposed; in other words, the secret key has Hamming weight $h \approx N/2$. In that case, K is a considerable value, so that a high-degree approximate polynomial is required. Therefore, if the method proposed in this paper is applied to the non-sparse key case, its impact on bootstrapping error reduction will be significant.

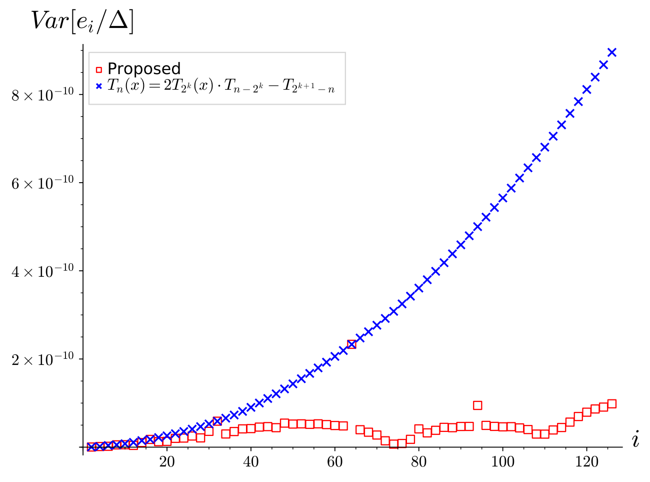
6 Implementation of the Proposed Method and Performance Comparison

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

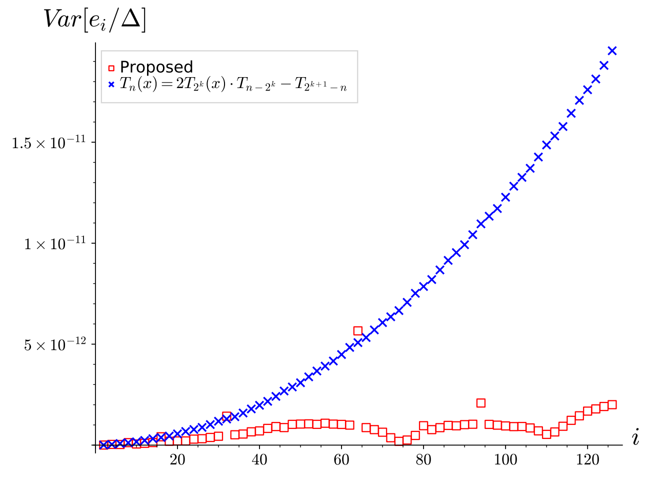
6.1 Error Variance Minimization

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

As the domain of Chebyshev polynomial is $[-1, 1]$, $\{T_i(t/K)\}_{0 \le i \le n}$ is used as the polynomial basis, and it can be seen that the distribution of the input t/K is concentrated at zero. As shown in Subsection 4.4, multiplication between two even degree terms should be avoided when we calculate the even degree terms. Fig. 2 shows the variance of error in $T_i(t)$ for even i's. It can be seen that error in encrypted data can be greatly reduced by only changing the order of calculating $T_i(t)$. In particular, for $T_{74}(t)$, the variance of error for the proposed method becomes smaller by 1/1973 compared to that without minimization.



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



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

Fig. 2. Variance of error in $T_i(t)$ for even i using HEAAN and SEAL with various parameters.

6.2 Weight Constant and Minimum Error Variance

In Subsection 5.5, we discussed analytic solution and numerical method for optimal approximate polynomial. In this subsection, the above methods are implemented and verified, together with the theoretical values of approximation error and the amplified basis error. Besides, we confirm that although the numerical method finds a polynomial that is very close to the value obtained through Subsection (5.5), it has a slightly larger error than this.

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

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

The variance of amplified basis errors by coefficients are given as

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

Finally, in the SlotToCoeff step, the plaintext vectors are multiplied with the first half of the encoding matrix $\mathbf{U}$ and their diagonal vectors have the magnitude of one. Hence, the variance of the approximation error is multiplied by the number of slots n. In summary, the total errors by bootstrapping are given as

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

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

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

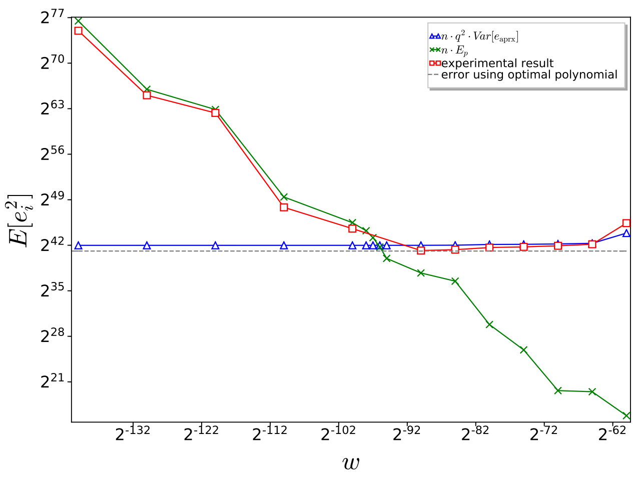
$$n \cdot q^2 \cdot Var[e_{\mathsf{aprx}}].$$

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

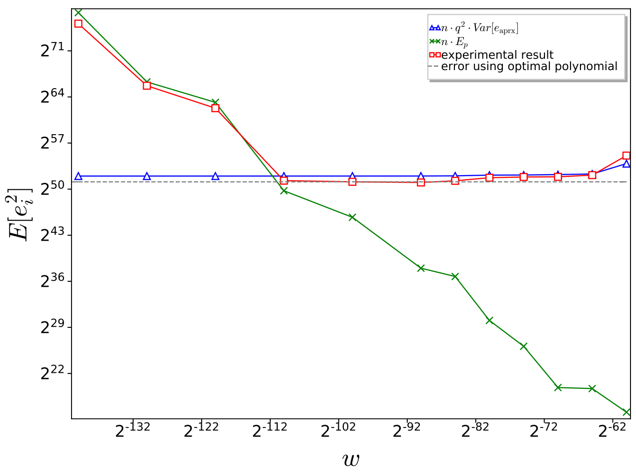
$$n \cdot E_p$$

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

In Fig. 3, the sum of blue lines with triangular legend and green lines with x mark legend meets the red lines with the square legend. Thus, it shows that the theoretical derivation and experimental results are agreed upon. It can also be



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



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

Fig. 3. Theoretical energy of approximation error, amplified basis error, and energy of experimental results, implemented in HEAAN. A polynomial of degree 81 is used. The gray dot line is the variance of bootstrapping error that is achieved by using the polynomial with coefficients that $\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} \left(Var[q \cdot e_{\mathsf{aprx}}] + E_p \right)$, which is the lower bound of bootstrapping error.

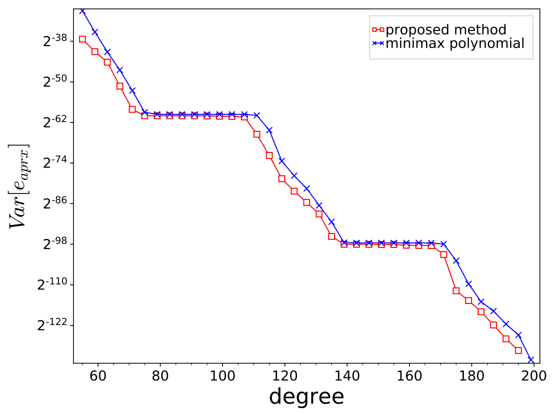


Fig. 4. Comparison of the minimum achievable variance of approximation error of the proposed method and that of the minimax polynomial, when $\epsilon = 2^{-10}$ and no controlling of coefficients of both approximate polynomials.

seen that it is possible to obtain an approximate polynomial with a small error even by using the numerical method, but the error is slightly larger than that of an accurately calculated approximate polynomial. It is noted that the variance of the rescaling error is $\frac{(h+2)2n}{12}$ and the optimal w is close to $\frac{(h+2)2n}{12q^2} \approx 2^{6.4}$ because each element of Chebyshev polynomials has error mainly introduced by rescaling.

6.3 Comparison of the Proposed Method with the Previous Methods

Minimized Error Variance by the Proposed Method and Error Variance of Minimax Polynomial The best-known approximation method for the CKKS bootstrapping so far is the modified Remez algorithm [6]. The modified Remez algorithm is an iterative method that finds the minimax approximate polynomial for piece-wise continuous functions such as $f_{\mathsf{mod}}(t)$. Using the modified Remez algorithm, the minimax approximate polynomial for $f_{\mathsf{mod}}(t)$ can be found. The minimax approach is reasonable when the input distribution is unknown. However, in the CKKS bootstrapping, the input distribution is partially known; the probability mass function of I follows a distribution similar to the Irwin–Hall distribution. We use the worst-case assumption that r is uniformly distributed when we derive the variance minimizing approximate polynomial. However, in the experiment of finding the variance of approximate error for the given approximate polynomials for both methods, we let r follow the Gaussian distribution, not the uniform distribution because the message polynomial is assumed to be the resultant value of compound operations and summations. In other words, by assuming the distribution of the message polynomial differently for finding the variance minimizing approximate polynomial and actually calculating the variance, the experiment is conducted in an unfavorable environment to the proposed method. It is noteworthy that a lower error variance than minimax polynomial is achieved when using the proposed method, despite the worst-case assumption. It is shown that the distribution of I has a dominant effect on the error.

It is shown in Fig. 4 that the variance of approximation error is smaller when the error variance minimizing polynomial is used, as expected. This means that the proposed method reduces the approximation error during bootstrapping. As the magnitude of the coefficients of the approximate polynomial cannot be controlled in the modified Remez algorithm, the approximate polynomials for both methods are compared without controlling the magnitude of coefficients in Fig. 4. It is noted here that the variance of approximation errors shown in Fig. 4 is not practical in the CKKS scheme due to the basis error and the enormous coefficients of the approximate polynomials. However, it is possible to reduce the magnitude of the coefficients of the approximate polynomial in the proposed method with slightly increased error variance. In contrast, the previous methods cannot control the magnitude of its coefficients, and thus the use of the double-angle formula is essential, which results in a large error variance and more depth.

Experimental Result of Bootstrapping Error There are experimental results, but they are being visualized.

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

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

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

Then the giant-step such as

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

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

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

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

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

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

7 Conclusion

In this paper, we introduced two novel methods to improve the precision of the CKKS scheme. First, SNR, a widely-used measure of performance when dealing with noisy media such as communication systems, was adopted for the error variance minimization of the CKKS scheme. To maximize the SNR of encrypted data, we proposed a method to minimize the variance of errors; To do this, we replaced the high-probability upper bound that has been in the tagged information so far with the variance of errors. As a result, we could tightly manage the error, and 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 minimizing the error variance. Especially, the newly proposed operation reordering and approximate polynomial were applied to the bootstrapping of the CKKS scheme, and it is shown that the error performance of the bootstrapping of the CKKS scheme was greatly improved compared to that of the previous methods. To our best knowledge, this is the first bootstrapping algorithm that contemplates various parameters, slot size, the error characteristics of the CKKS scheme, and the polynomial evaluation algorithm. From its implementation on HEAAN and SEAL, it was shown that the proposed bootstrapping algorithm achieves less variance of error in encrypted data while consuming less level, compared to the previous works. Even though the number of non-scalar multiplication is slightly increased compared to the composite method, the lower depth is more important for future parallel implementation.

From the 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 proposed three 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 the deeper operations. Since many studies to adjust the order of operations for general purposes have been done in the field of compilers, applying the results of these researches will lead to significant improvement in many applications using the CKKS scheme. We leave application-specific reordering of homomorphic operations with compiler techniques as further work.

References

1. J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Proc. Intl. Conf. on the Theory and Appl. of Cryptol. and Inf. Secur. (ASIACRYPT),* Springer, 2017, pp. 409–437.
2. J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "Bootstrapping for approximate homomorphic encryption," in *Proc. Annu. Intl. Conf. on the Theory and Appl. of Cryptograph. Techn. (EUROCRYPT),* Springer, 2018, pp. 360–384.
3. H. Chen, I. Chillotti, and Y. Song, "Improved bootstrapping for approximate homomorphic encryption," in *Proc. Annu. Intl. Conf. on the Theory and Appl. of Cryptograph. Techn. (EUROCRYPT),* Springer, 2019, pp. 34–54.
4. K. Han and D. Ki, "Better bootstrapping for approximate homomorphic encryption," in *Proc. Cryptographers' Track at the RSA Conf.,* Springer,2020, pp. 364–390.
5. Y. Lee, J. W. Lee, Y. S. Kim, and J. S. No, "Near-optimal polynomial for modulus reduction using L2-norm for approximate homomorphic encryption," *IEEE Access,* vol. 8, pp. 144 321-144 330, 2020.
6. J. W. Lee, E. Lee, Y. Lee, Y. S. Kim, and J. S. No, "High-precision bootstrapping of RNS-CKKS homomorphic encryption using optimal minimax polynomial approximation and inverse sine function," *IACR Cryptol. ePrint Arch.,* vol. 2020.552, 2020.
7. A. Kim, A. Papadimitriou, and Y. Polyakov, "Approximate homomorphic encryption with reduced approximation error," *IACR Cryptol. ePrint Arch.,* vol. 2020.1118, 2020.
8. J. P. Bossuat, C. Mouchet, J. Troncoso-Pastoriza, and J. P. Hubaux,"Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys," *IACR Cryptol. ePrint Arch.,* vol. 2020.1203, 2020.
9. J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "A full RNS variant of approximate homomorphic encryption," in *Proc. Intl. Conf. on Sel. Areas in Cryptogr.,* Springer, 2018, pp. 347–368.
10. M. Blatt, A. Gusev, Y. Polyakov, K. Rohloff, and V. Vaikuntanathan, "Optimized homomorphic encryption solution for secure genome-wide association studies," *BMC Medical Genomics,* 13(7):1–13, 2020.
11. K. Han, M. Hhan, and J. H. Cheon, "Improved homomorphic discrete Fourier transforms and FHE bootstrapping," *IEEE Access,* vol. 7, pp. 57 361–57 370, 2019.
12. M. Kim, Y. Song, S. Wang, Y. Xia, and X. Jiang, "Secure logistic regression based on homomorphic encryption: Design and evaluation," *JMIR Med. Inform.,* vol. 6, no. 2, p. e19, 2018.
13. C. Gentry, S. Halevi, and N. P. Smart, "Homomorphic evaluation of the AES circuit," in *Proc. Annu. Cryptol. Conf. (CRYPTO),* Springer, 2012, pp. 850–867.
14. I. Chillotti, N. Gama, M. Georgieva, and M. Izabach'ene, "TFHE: fast fully homomorphic encryption over the torus," *J. of Cryptol.,* vol. 33, no. 1, pp. 34–91, 2020.

15. I. Chillotti, N. Gama, M. Georgieva, and M. Izabachene, "Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE," in *Proc. Intl. Conf. on the Theory and Application of Cryptol. and Inf. Secur. (ASIACRYPT),* Springer, 2017, pp. 377–408.
16. Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-LWE and security for key dependent messages," in *Proc. Annu. Cryptol. Conf. (CRYPTO),* Springer, 2011, pp. 505–524.
17. Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," *SIAM J. on Comput.,* vol. 43, no. 2, pp. 831–871, 2014.
18. Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *ACM Trans. on Computation Theory (TOCT),* vol. 6, no. 3, pp. 1–36, 2014.
19. M. Albrecht, M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, S. Halevi, J. Hoffstein, K. Laine, K. Lauter, S. Lokam, D. Micciancio, D. Moody, T. Morrison, A. Sahai, and V. Vaikuntanathan, "Homomorphic encryption security standard," HomomorphicEncryption.org, Toronto, Canada, Tech. Rep., November 2018.
20. J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *IACR Cryptology ePrint Arch.,* vol. 2012.144, 2012.
21. L. Ducas and D. Micciancio, "FHEW: bootstrapping homomorphic encryption in less than a second," in *Proc. Annu. Intl. Conf. on the Theory and Appl. of Cryptograph. Techn. (EUROCRYPT),* Springer, 2015, pp. 617–640.
22. J. H. Cheon, D. Kim, Y. Kim, and Y. Song, "Ensemble method for privacy-preserving logistic regression based on homomorphic encryption," *IEEE Access,* vol. 6, pp. 46 938–46 948, 2018.
23. R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proc. Intl. Conf. on Machine Learning,* 2016, pp. 201–210.
24. E. Chou, J. Beal, D. Levy, S. Yeung, A. Haque, and L. Fei-Fei, "Faster cryptonets: Leveraging sparsity for real-world encrypted inference," *arXiv preprint arXiv*:1811.09953, 2018.
25. H. Chabanne, A. de Wargny, J. Milgram, C. Morel, and E. Prouff, "Privacy-preserving classification on deep neural network," *IACR Cryptology ePrint Arch.,* vol. 2017.35, 2017.
26. I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning," MIT press Cambridge, pp. 483–484, 2016.
27. S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.,* 2015, pp. 448–456.
28. T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.,* 2016, pp. 901–909.
29. J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450,* 2016.
30. E. Hesamifard, H. Takabi, and M. Ghasemi, "Cryptodl: Deep neural networks over encrypted data," *arXiv preprint arXiv*:1711.05189, 2017.
31. P. Ramachandran, B. Zoph, and Q. V. Le, "Swish: A self-gated activation function," arXiv preprint arXiv: 1710.05941, 2017.
32. D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," in *Proc. Annu. ACM Symp. on Theory of Comput.,* 1987, pp. 1-6.
33. J. C. Mason and D. C. Handscomb, "Chebyshev interpolation," in *Chebyshev Polynomials,* Boca Raton, FL, USA: CRC Press, 2002, pp. 154-172.

34. M. Fasi, “Optimality of the Paterson–Stockmeyer method for evaluating matrix polynomials and rational matrix functions,” *Linear Algebra and its Appl.,* vol. 574, pp. 182–200, 2019.
35. M. S. Paterson and L. J. Stockmeyer, “On the number of nonscalar multiplications necessary to evaluate polynomials,” *SIAM J. on Comput.,* vol. 2, no. 1, pp. 60–66, 1973.
36. J. H. Cheon, D. Kim, D Kim, H. H. Lee, and K. Lee, “Numerical method for comparison on homomorphically encrypted numbers,” in *Proc. Intl. Conf. on the Theory and Application of Cryptol. and Inf. Secur. (ASIACRYPT),* Springer, 2019, pp. 415–445.
37. C. Lattner and V. Adve, “LLVM: A compilation framework for lifelong program analysis & transformation,” in *International Symposiumon Code Generation and Optimization,* 2004, pp. 75–86.