High-Precision Bootstrapping for Approximate Homomorphic Encryption by Error Variance Minimization

Yongwoo Lee[1,2], Joon-Woo Lee[2], Young-Sik Kim[3(✉)], Yongjune Kim[4], Jong-Seon No[2], and HyungChul Kang[1]

[1] Samsung Advanced Institute of Technology, Suwon-si, Gyeonggi-do, Korea
{yw0803.lee,hc1803.kang}@samsung.com

[2] Department of Electrical and Computer Engineering, INMC, Seoul National University, Seoul, Korea
joonwoo3511@ccl.snu.ac.kr, jsno@snu.ac.kr

[3] Department of Information and Communication Engineering, Chosun University, Gwangju, Korea
iamyskim@chosun.ac.kr

[4] Department of Electrical Engineering and Computer Science, DGIST, Daegu, Korea
yjk@dgist.ac.kr

Abstract. The Cheon-Kim-Kim-Song (CKKS) scheme (Asiacrypt'17) is one of the most promising homomorphic encryption (HE) schemes as it enables privacy-preserving computing over real (or complex) numbers. It is known that bootstrapping is the most challenging part of the CKKS scheme. Further, homomorphic evaluation of modular reduction is the core of the CKKS bootstrapping. As modular reduction is not represented by the addition and multiplication of complex numbers, approximate polynomials for modular reduction should be used. The best-known techniques (Eurocrypt'21) use a polynomial approximation for trigonometric functions and their composition. However, all the previous methods are based on an indirect approximation, and thus it requires lots of multiplicative depth to achieve high accuracy. This paper proposes a direct polynomial approximation of modular reduction for CKKS bootstrapping, which is optimal in error variance and depth. Further, we propose an efficient algorithm, namely the lazy baby-step giant-step (BSGS) algorithm, to homomorphically evaluate the approximate polynomial, utilizing the lazy relinearization/rescaling technique. The lazy-BSGS reduces the computational complexity by half compared to the ordinary BSGS algorithm. The performance improvement for the CKKS scheme by the proposed algorithm is verified by implementation using HE libraries. The implementation results show that the proposed method has a multiplicative depth of 10 for modular reduction to achieve the state-of-the-art accuracy, while the previous methods have depths of 11 to 12. Moreover, we achieve higher accuracy within a small multiplicative depth, for example, 93-bit within multiplicative depth 11.

© International Association for Cryptologic Research 2022
O. Dunkelman and S. Dziembowski (Eds.): EUROCRYPT 2022, LNCS 13275, pp. 551–580, 2022.
https://doi.org/10.1007/978-3-031-06944-4_19

Keywords: Bootstrapping · Cheon-Kim-Kim-Song (CKKS) scheme · Fully homomorphic encryption (FHE) · Privacy-preserving machine learning (PPML) · Signal-to-noise ratio (SNR)

1 Introduction

Homomorphic encryption (HE) is a specific class of encryption schemes that enables computation over encrypted data. The Cheon-Kim-Kim-Song (CKKS) scheme [12] is one of the highlighted fully homomorphic encryption (FHE) schemes as it supports efficient computation on real (or complex) numbers, which are the usual data type for many applications such as deep learning. As the other HE schemes are designed for different domains, the CKKS scheme is known to be the most efficient for real numbers. For example, Brakerski-Fan-Vercauteren (BFV) [5,6,17] and Brakerski-Gentry-Vaikuntanathan (BGV) [4] schemes are designed for integer messages in $\mathbb{Z}_q$, and FHEW/TFHE [13–15] are designed for binary circuits.

Gentry's blueprint of bootstrapping provides the idea of homomorphic re-encryption of ciphertext. In CKKS bootstrapping, the modular reduction by an integer is performed homomorphically. However, the modular reduction function is not represented by the addition and multiplication of complex numbers. Hence, an approximate polynomial of trigonometric functions is used in prior arts [3,7,9,18,24], which have two limitations in practice: i) these are indirect approximations, which require larger multiplicative depths, and ii) the measure of approximation error is minimax-base (minimizing the upper bound of the approximation error). This paper shows that the minimax polynomial does not guarantee the minimax bootstrapping error. We propose that the error variance would be a better measure than the minimax error, especially for bootstrapping.

The CKKS scheme provides the trade-off between the efficiency and precision of messages as encrypted data of the CKKS scheme inherently has noise. Errors in encrypted data are propagated and added along with homomorphic operations. Hence, the error should be carefully measured when we design a circuit for efficiency and security in CKKS. Moreover, as attacks against CKKS have recently been proposed [11,26,27], reducing errors of the CKKS scheme becomes more crucial to mitigate the risk of the attacks.

1.1 Our Contributions

This paper contains contributions to the high-precision bootstrapping of the CKKS scheme. We propose i) a method to find the optimal approximate polynomial for the modular reduction in bootstrapping and ii) an efficient algorithm for homomorphic evaluation of polynomials.

First, we propose the optimal approximate polynomial for CKKS bootstrapping in the aspect of signal-to-noise ratio (SNR), which improves the precision of CKKS bootstrapping. As a result, we can reserve more levels after bootstrapping while achieving the best-known precision, where the level of a ciphertext is defined as the number of successive multiplications that can be performed to the ciphertext without bootstrapping. The proposed approximate polynomial has the following three features: i) an optimal measure of error for CKKS bootstrapping: we show that an approximate polynomial that achieves the least error variance is also optimal for CKKS bootstrapping in the aspect of SNR. ii) a direct approximation: the approximate polynomial of modular reduction is directly obtained from the whole polynomial space of degree n, i.e., P_n, using the error variance-minimizing method, and thus it has less multiplicative depths compared to the previous methods (In other words, less bootstrapping is required for the same circuit.) iii) reduction of error from noisy calculation: in the polynomial evaluation over CKKS, each polynomial basis has an error. Unlike previous bootstrapping methods, the proposed method minimizes the errors introduced by noisy basis as well as the approximation error.

Second, we propose a novel variant of the baby-step giant-step (BSGS) algorithm, called the lazy-BSGS algorithm, which reduces the number of relinearizations by half compared to ordinary BSGS algorithms. The proposed lazy-BSGS algorithm is more efficient for higher degree polynomial. The proposed approximate polynomial has a high degree, while the previous methods use a composition of small-degree polynomials. Thus, the lazy-BSGS algorithm makes the evaluation time of the proposed polynomial comparable to the previous methods.

Note for the First Contribution. Previous methods utilized the minimax approximate polynomial of modular reduction function for bootstrapping to reduce the bootstrapping error [18,24]. However, in CKKS bootstrapping, a linear transformation on slot values, called SlotToCoeff, is performed, and its resulting ciphertext is the sum of thousands of noisy values. Since many noisy values are added, the upper bound on the final error value is loose. Hence, we propose to minimize the error variance instead of the upper bound on the error.

Besides the approximation error, each polynomial basis also has an error in CKKS, and it is amplified when we multiply large coefficients of the approximate polynomial. The previous approximation method could not control these errors with the approximate polynomial coefficients. Thus, they used the trigonometric function and double angle formula instead, to make the approximation degree small [3,18,24]. This indirect approximation results in larger multiplicative depths. It is preferred to reserve more levels after bootstrapping as it can reduce the number of bootstrapping in the whole system; moreover, the number of remaining levels after bootstrapping is also important for an efficient circuit design of algorithms using CKKS, for example, in [23], the depth of activation layer is optimized for the levels after bootstrapping. The proposed method minimizes the basis error variance as well as the approximation error variance, so it has less multiplicative depths compared to the previous composition of trigonometric functions. To the best of our knowledge, this is the first method to find the optimal approximate polynomial that minimizes both the approximation error and the error in the basis at the same time.

We show that from the learning with error (LWE) assumption, the input of approximate polynomial follows a distribution similar to Irwin-Hall distribution, regardless of the security. The proposed method exploits this property to improve the approximation accuracy. Also, we derive an analytical solution for our error variance-minimizing approximate polynomial, while the previous minimax approximate polynomial was obtained by iterative algorithms [18,24].

Note for the Second Contribution. As rescaling and relinearization introduce additional errors, it is desirable to perform them as late as possible. In addition, the number of rescalings/relinearizations is also reduced by reordering the operations to delay the relinearization/rescaling. This technique, the so-called lazy rescaling/relinearization technique, has been applied to reduce the computational complexity in [1,8,22]. We propose a rigorous analysis on lazy rescaling and relinearization in the BSGS algorithm. Moreover, we propose the algorithm to find the optimal approximate polynomial, which fits the lazy-BSGS algorithm for odd functions.

1.2 Related Works

Bootstrapping of the CKKS Scheme. Since the CKKS bootstrapping was firstly proposed in [9], the Chebyshev interpolation has been applied to the homomorphic evaluation of modular reduction [7,18]. Then, a technique for direct approximation was proposed using the least squares method [25] and Lagrange interpolation [19]. However, the magnitudes of coefficients of those approximate polynomials are too large. The algorithm to find minimax approximate polynomial using improved multi-interval Remez algorithm and the use of arcsin to reduce approximation error of the modular reduction were presented in [24]. The bootstrapping for the non-sparse-key CKKS scheme was proposed, and the computation time for homomorphic linear transformations was significantly improved by using double hoisting in [3]. Julta and Manohar proposed to use sine series approximation [20], but as there exists a linear transformation from sine series $\{\sin(kx)\}$ to power of sine functions $\{\sin(x)^k\}$, this method is also based on trigonometric functions.

Attacks on the CKKS Scheme and High-Precision Bootstrapping. An attack to recover the secret key using the error pattern after decryption was recently proposed by Li and Micciancio [26], and thus it becomes more crucial to reduce the error in CKKS. One possible solution to this attack is to add a huge error, so-called the noise flooding technique [16] or perform rounding after decryption to make the plaintext error-free [26]. In order to use the noise flooding technique, the CKKS scheme requires much higher precision, and the bootstrapping error is the bottleneck of precision. Although a lot of research is required on how to exploit the bootstrapping error for cryptanalysis of CKKS, the high-precision bootstrapping is still an interesting topic [3,20,24,25].

1.3 Organization of This Paper

The remainder of the paper is organized as follows. In Sect. 2, we provide the necessary notations and SNR perspective on error. The CKKS scheme and its bootstrapping algorithm are summarized in Sect. 3. We provide a new method to find the optimal direct approximate polynomials for the CKKS scheme and also show its optimality in Sect. 4. Section 5 provides the novel lazy-BSGS algorithm for the efficient evaluation of approximate polynomial for CKKS bootstrapping. The implementation results and comparison for precision and timing performance for the CKKS bootstrapping are given in Sect. 6. Finally, we conclude the paper in Sect. 7.

2 Preliminaries

2.1 Basic Notation

Vectors are denoted in boldface, such as $\boldsymbol{v}$, and all vectors are column vectors. Matrices are denoted by boldfaced capital letters, i.e., $\mathbf{M}$. We denote the inner product of two vectors by $\langle\cdot,\cdot\rangle$ or simply $\cdot$. $\lfloor\cdot\rceil$, $\lfloor\cdot\rfloor$, and $\lceil\cdot\rceil$ denote the rounding, floor, and ceiling functions, respectively. $[m]_q$ is the modular reduction, i.e., the remainder of m dividing by q. $x \leftarrow \mathcal{D}$ denotes the sampling x according to a distribution $\mathcal{D}$. When a set is used instead of distribution, x is sampled uniformly at random among the set elements. Random variables are denoted by capital letters such as X. $E[X]$ and $Var[X]$ denote the mean and variance of random variable X, respectively. For a function f, $Var[f(X)]$ can be simply denoted by $Var[f]$. $\|\boldsymbol{a}\|_2$ and $\|\boldsymbol{a}\|_\infty$ denote the L-2 norm and the infinity norm, and when the input is a polynomial, those denote the norm of coefficient vector. We denote the supreme norm of a function $\|f\|_{\mathsf{sup}} := \sup_{t\in\mathcal{D}} |f(t)|$ for a given domain $\mathcal{D}$.

Let $\Phi_M(X)$ be the M-th cyclotomic polynomial of degree N, and when M is a power of two, $M = 2N$, and $\Phi_M(X) = X^N + 1$. Let $\mathcal{R} = \mathbb{Z}/\langle\Phi_M(X)\rangle$ be the ring of integers of a number field $\mathcal{S} = \mathbb{Q}/\langle\Phi_M(X)\rangle$, where $\mathbb{Q}$ is the set of rational numbers and we write $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. A polynomial $a(X) \in \mathcal{R}$ can be denoted by a by omitting X when it is obvious. Since the multiplicative depth of a circuit is crucial in CKKS, from here on, the multiplicative depth is referred to as depth.

2.2 The CKKS Scheme

The CKKS scheme and its residual number system (RNS) variants [10,18] provide operations on encrypted complex numbers, which are done by the canonical embedding and its inverse. Recall that the 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 of 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 and decoding are defined as follows.

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

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

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

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

For a real number σ, $\mathcal{DG}(\sigma^2)$ denotes the distribution in $\mathbb{Z}^N$, whose entries are sampled independently from the discrete Gaussian distribution of variance σ^2. $\mathcal{HWT}(h)$ is the set of signed binary vectors in $\{0, \pm 1\}^N$ with Hamming weight h. Suppose that we have ciphertexts of level l for $0 \leq l \leq L$.

The RNS-CKKS scheme performs all operations in RNS. The ciphertext modulus $Q_l = q \cdot \prod_{i=1}^{l} p_i$ is used, where p_i's are chosen as primes that satisfy $p_i = 1 \,(\mathrm{mod}\ 2N)$ to support efficient number theoretic transform (NTT). We note that $Q_0 = q$ is greater than p as the final message's coefficients should not be greater than the ciphertext modulus q. For a faster computation, we use the hybrid key switching technique in [18]. First, for predefined dnum, a small integer, we define partial products $\left\{\tilde{Q}_j\right\}_{0 \leq j < \mathsf{dnum}} = \left\{\prod_{i=j\alpha}^{(j+1)\alpha-1} p_i\right\}_{0 \leq j < \mathsf{dnum}}$, for a small integer $\alpha = \lceil (L+1)/\mathsf{dnum} \rceil$. For a ciphertext with level l and $\mathsf{dnum}' = \lceil (l+1)/\alpha \rceil$, we define [18]

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

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

Then, for any $(a, b) \in \mathcal{R}_{Q_l}^2$, we have
$$\langle \mathcal{WD}_l(a), \mathcal{PW}_l(b) \rangle = a \cdot b \,(\mathrm{mod}\ Q_l)\,.$$

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

- $\mathsf{KeyGen}(1^\lambda)$:
  - 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: $s \leftarrow \mathcal{HWT}(h)$.
  - The secret key is $\mathsf{sk} := (1, s)$.

- $\mathsf{KSGen}_{\mathsf{sk}}(s')$: For auxiliary modulus $P = \prod_{i=0}^{k} p'_i \approx \max_j \tilde{Q}_j$, sample $a'_k \leftarrow \mathcal{R}_{PQ_L}$ and $e'_k \leftarrow \mathcal{DG}(\sigma^2)$. Output the switching key
$$\mathsf{swk} := (\mathsf{swk}_0, \mathsf{swk}_1) = (\{b'_k\}_{k=0}^{\mathsf{dnum}'-1}, \{a'_k\}_{k=0}^{\mathsf{dnum}'-1}) \in \mathcal{R}_{PQ_L}^{2 \times \mathsf{dnum}'},$$
where $b'_k = -a'_k s + e'_k + P \cdot \mathcal{PW}(s')_k \,(\mathrm{mod}\ PQ_L)$.

  - Set the evaluation key as $\mathsf{evk} := \mathsf{KSGen}_{\mathsf{sk}}(s^2)$.
- $\mathsf{Enc}_{\mathsf{sk}}(m)$: Sample $a \leftarrow \mathcal{R}_{Q_L}$ and $e \leftarrow \mathcal{DG}(\sigma^2)$. The output ciphertext is

$$\mathsf{ct} = (-a \cdot s + e + m, a) \pmod{Q_L},$$

  where $\mathsf{sk} = (1, s)$. There is also a public-key encryption method [12], but omitted here.
- $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct})$: Output $\bar{m} = \langle \mathsf{ct}, \mathsf{sk} \rangle$.
- $\mathsf{Add}(\mathsf{ct}_1, \mathsf{ct}_2)$: For $\mathsf{ct}_1, \mathsf{ct}_2 \in \mathcal{R}^2_{Q_l}$, output $\mathsf{ct}_{\mathsf{add}} = \mathsf{ct}_1 + \mathsf{ct}_2 \pmod{Q_l}$.
- $\mathsf{Mult}(\mathsf{ct}_1, \mathsf{ct}_2)$: For $\mathsf{ct}_1 = (b_1, a_1)$ and $\mathsf{ct}_2 = (b_2, a_2) \in \mathcal{R}^2_{Q_l}$, return

$$\mathsf{ct}_{\mathsf{mult}} = (d_0, d_1, d_2) := (b_1 b_2, a_1 b_2 + a_2 b_1, a_1 a_2) \pmod{Q_l}.$$

- $\mathsf{RL}_{\mathsf{evk}}(d_0, d_1, d_2)$: For a three-tuple ciphertext (d_0, d_1, d_2) corresponding to secret key $(1, s, s^2)$, return $(d_0, d_1) + \mathsf{KS}_{\mathsf{evk}}((0, d_2))$.
- $\mathsf{cAdd}(\mathsf{ct}_1, \boldsymbol{a}; \Delta)$: For $\boldsymbol{a} \in \mathbb{C}^{N/2}$ and a scaling factor Δ, output $\mathsf{ct}_{\mathsf{cadd}} = \mathsf{ct} + (\mathsf{Ecd}(\boldsymbol{a}; \Delta), 0)$.
- $\mathsf{cMult}(\mathsf{ct}_1, \boldsymbol{a}; \Delta)$: For $\boldsymbol{a} \in \mathbb{C}^{N/2}$ and a scaling factor Δ, output $\mathsf{ct}_{\mathsf{cmult}} = \mathsf{Ecd}(\boldsymbol{a}; \Delta) \cdot \mathsf{ct}$.
- $\mathsf{RS}(\mathsf{ct})$: For $\mathsf{ct} \in \mathcal{R}^2_{Q_l}$, output $\mathsf{ct}_{\mathsf{RS}} = \lfloor p_l^{-1} \cdot \mathsf{ct} \rceil \pmod{q_{l-1}}$.
- $\mathsf{KS}_{\mathsf{swk}}(\mathsf{ct})$: For $\mathsf{ct} = (b, a) \in \mathcal{R}^2_{Q_l}$ and $\mathsf{swk} := (\mathsf{swk}_0, \mathsf{swk}_1)$, output

$$\mathsf{ct}_{\mathsf{KS}} = \left(b + \left\lfloor \frac{\langle \mathcal{WD}_l(a), \mathsf{swk}_0 \rangle}{P} \right\rceil, \left\lfloor \frac{\langle \mathcal{WD}_l(a), \mathsf{swk}_1 \rangle}{P} \right\rceil \right) \pmod{Q_l}.$$

The key-switching techniques are used to provide various operations such as complex conjugate and rotation. To remove the error introduced by approximate scaling factors, one can use different scaling factors for each level as given in [21], or we can use the scale-invariant method proposed in [3] for polynomial evaluation. We note that (`FullRNS-`)`HEAAN` and `SEAL` are ($\mathsf{dnum} = 1$) and ($\mathsf{dnum} = L + 1$) cases, respectively, and `Lattigo` supports for arbitrary dnum.

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

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

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 follows:

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

where S and N denote the signal (message) and noise (error), respectively. As shown in the definition, the higher SNR corresponds to the better quality.

A simple way to increase SNR is to increase the signal power, but it would be limited due to regulatory or physical constraints. The CKKS scheme has the same problem; a larger scaling factor should be multiplied to the message to increase the message power, but if one uses a larger scaling factor, the ciphertext level decreases, or larger parameters should be used for security. Hence, to increase SNR, it is beneficial to reduce the noise power in the CKKS scheme rather than increasing the signal power.

Error estimation of the CKKS scheme so far has been focused on the high-probability upper bound of the error after several operations [9,12] and also minimax for approximation [24]. However, the bound becomes quite loose as the homomorphic operation continues, and its statistical significance may diminish. Thus, we maximize SNR in this paper, which is equivalent to minimizing error variance when the scaling factor is fixed.

3 Bootstrapping of the CKKS Scheme

3.1 Outline of the CKKS Bootstrapping

There are extensive studies for bootstrapping of the CKKS scheme [3,7,9,18–20,24,25]. The CKKS bootstrapping consists of the following four steps: ModRaise, CoeffToSlot, EvalMod, and SlotToCoeff.

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

Homomorphic Evaluation of Encoding (CoeffToSlot). Homomorphic operations are performed in plaintext slots, but we need component-wise operations on coefficients. Thus, to deal with $t(X)$, we should put polynomial coefficients in plaintext slots. In CoeffToSlot step, $\mathsf{Emb}^{-1} \circ \pi^{-1}$ is performed homomorphically using matrix multiplication [9], or FFT-like hybrid method [7]. Then, we have two ciphertexts encrypting $\boldsymbol{z}'_0 = (t_0, \ldots, t_{\frac{N}{2}-1})$ and $\boldsymbol{z}'_1 = (t_{\frac{N}{2}}, \ldots, t_{N-1})$ (when the number of slots is small, we can put $\boldsymbol{z}'_0$ and $\boldsymbol{z}'_1$ in a ciphertext, see [9]), where t_j denotes the j-th coefficient of $t(X)$. The matrix multiplication is composed of three steps [9]: i) rotate ciphertexts, ii) multiply diagonal components of matrix to the rotated ciphertexts, and iii) sum up the ciphertexts.

Evaluation of the Approximate Modular Reduction (EvalMod). An approximate evaluation of the modular reduction function is performed in this step. As additions and multiplications cannot represent the modular reduction function, an approximate polynomial for $[\cdot]_q$ is used. For approximation, it is desirable to control the message size to ensure $m_i \leq \epsilon \cdot q$ for a small ϵ [9].

Homomorphic Evaluation of Decoding (SlotToCoeff). SlotToCoeff is the inverse operation of CoeffToSlot. Since the matrix elements do not have to be precise as much in CoeffToSlot, we can use a smaller scaling factor here [3]. In SlotToCoeff, the ciphertext is multiplied by the CRT matrix $\mathbf{U}$, whose elements have magnitudes of one. Thus, the N errors in slots are multiplied by a constant of size one and then added.

3.2 Polynomial Approximation of Modular Reduction

Previous works approximated the modular reduction function as $\frac{q}{2\pi}\sin\left(\frac{2\pi t}{q}\right)$ [7, 9, 18]. Approximate polynomial of sine function is found by using Taylor expansion of exponent function and $e^{it} = \cos(t) + i \cdot \sin(t)$ in [9]. The Chebyshev approximation of sine function improved the approximation in [7]. The modified Chebyshev approximation in cosine function and the double-angle formula reduced the error and evaluation time in [18]. However, in these approaches, the sine function is used, and thus there is still the fundamental approximation error, that is,

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

Direct-approximation methods were proposed in [19, 25], but their coefficients are large and amplify errors of polynomial basis. A composition with inverse sine function that offers a trade-off between the precision and the remaining level was proposed to remove the fundamental approximation error between the sine function and the modular reduction [24]. However, the evaluation of inverse sine function has a considerable multiplicative depth.

Those prior researches tried to find the minimax approximate polynomial p_n, which minimizes $\|f - p_n\|_{\mathsf{sup}}$, where f is the function to approximate, such as sine function [7, 9, 18, 24]. Lee *et al.* proposed the multi-interval Remez algorithm [24], which is an iterative method to find minimax approximate polynomial of an arbitrary piece-wise continuous function.

3.3 Baby-Step Giant-Step Algorithms

There are several baby-step giant-step algorithms for a different purpose in the context of HE. In this paper, BSGS only refers to the polynomial evaluation algorithm proposed in [18] and its variants. The BSGS algorithm is presented in Algorithm 1 composed of SetUp, BabyStep, and GiantStep. SetUp calculates all the polynomial bases required to evaluate the given polynomial. The GiantStep divides the input polynomial by a polynomial of degree $2^i k$ and calls GiantStep recursively for its quotient and remainder, where $i \leq \lfloor \log(\mathsf{deg}/k) \rfloor$ for an integer k, and deg is the degree of the polynomial. When the given polynomial has a degree less than k, it calls BabyStep, and it evaluates the given polynomial of a small degree, namely a baby polynomial.

Originally, Han and Ki proposed to use a power-of-two k [18], and Lee *et al.* generalized k to an arbitrary even number and proposed to omit even-degree terms for odd polynomial,[1] which reduces the number of ciphertext-ciphertext

[1] This technique appears in their first version in Cryptology ePrint Archive.

Algorithm 1. BSGS Algorithm [18,24]

```
Instance: A ciphertext $\mathsf{ct}$ of $t$, a polynomial $p(X) = \sum_i c_i \cdot T_i(X)$.
Output: A ciphertext encrypting $p(t)$.
Let $l$ be the smallest integer satisfying $2^l k > n$ for an even number $k$.
procedure SetUp($\mathsf{ct}, l, k$)
    $\mathsf{ct}_i \leftarrow$ encryption of $T_i(t)$
    $\mathsf{ct}_{2^i k} \leftarrow$ encryption of $T_{2^i k}(t)$        ▷ for $0 \le i < l$.
end procedure
procedure BabyStep($p(X), \{\mathsf{ct}_i\}, k$)
    return $\sum_j c_j \cdot \mathsf{ct}_j$        ▷ baby polynomials.
end procedure
procedure GiantStep($p(X), \{\mathsf{ct}_i\}, l, k$)
    if $deg(p) < k$ then
        return BabyStep($p(X), \{\mathsf{ct}_i\}, k$)
    end if
    Find $q(X), r(X)$ s.t. $p(X) = q(X) \cdot T_{2^i k}(X) + r(X)$
    $\mathsf{ct}_q \leftarrow$GiantStep($q(X), \{\mathsf{ct}_i\}, l, k$)
    $\mathsf{ct}_r \leftarrow$GiantStep($r(X), \{\mathsf{ct}_i\}, l, k$)
    return $\mathsf{ct}_q \cdot \mathsf{ct}_{2^i k} + \mathsf{ct}_r$
end procedure
```

multiplications [24]. The number of ciphertext-ciphertext multiplications is given as

$$k - 2 + l + 2^l$$

in general, and

$$\lfloor \log (k-1) \rfloor + k/2 - 2 + l + 2^l$$

for odd polynomials, where $\mathsf{deg} < k \cdot 2^l$ is satisfied. Also, Bossuat *et al.* improved to do more recursion for high-degree terms [3] to optimize the multiplicative depth. In the BSGS algorithm of Bossuat *et al.*, we can evaluate a polynomial of degree up to $2^d - 1$ within multiplicative depth d by applying $O(\log k)$ additional multiplications.

4 Optimal Approximate Polynomial of Modular Reduction for Bootstrapping of the CKKS Scheme

This section proposes a new method to find the optimal approximate polynomial of the modular reduction function for the CKKS bootstrapping, considering the noisy computation nature of the CKKS Scheme. The optimality of the proposed approximate polynomial is proved, and statistics of input for an approximate polynomial are also analyzed to improve the approximation.

4.1 Error Variance-Minimizing Polynomial Approximation

We use the variance of error as the objective function for the proposed polynomial approximation and show that it is also optimal for CKKS bootstrapping.

As described later in the following subsections, the error in the noisy polynomial basis, namely the basis error, might be amplified by coefficients of the approximate polynomial. Thus, the magnitude of its coefficients should be small, and using the generalized least square method, the optimal coefficient vector $\boldsymbol{c}^*$ of the approximate polynomial is obtained as

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} \left(Var[e_{\mathsf{aprx}}] + \sum w_i \boldsymbol{c}_i^2 \right), \tag{1}$$

where e_{aprx} is the approximation error, and the constant values w_i are determined by the basis error given by CKKS parameters such as key Hamming weight, number of slots, and scaling factor.

We call the proposed approximate polynomial obtained by (1) as the *error variance-minimizing approximate polynomial*, and we derived an analytic solution. We note that the optimized solution attempts to minimize the variance of the approximation error as well as the variance of amplified basis error. The error variance-minimizing approximate polynomial is described in detail by taking bootstrapping as a specific example in the following subsection. It is worth noting that the approximation can be applied arbitrary function.

4.2 Optimality of the Proposed Direct Approximate Polynomial

In this subsection, we show that the proposed error variance-minimizing approximate polynomial is optimal for CKKS bootstrapping in the following aspects. First, we show that an approximate polynomial that minimizes the error variance after EvalMod also minimizes the bootstrapping error variance, and thus it is optimal in terms of SNR. Next, we show that the direct approximation to the modular reduction allows a more accurate approximation than previous indirect approximations using trigonometric functions [3,7,9,18,24] for fixed multiplicative depth.

Error-Optimality of the Proposed Approximate Polynomial in CKKS Bootstrapping. Here, we show that error variance-minimizing approximate polynomial guarantees the minimal error after bootstrapping in the aspect of SNR, while the minimax approach in [3,7,9,18,24] does not guarantee the minimax error after bootstrapping. In EvalMod, the operations between different slots do not happen, and thus we can assume that the error in each slot is independent. The SlotToCoeff is the homomorphic operation of decoding, and the decoding of $m(X)$ is given as $(m(\zeta_0), m(\zeta_1), \ldots, m(\zeta_{N/2-1}))$. Hence, the error in the j-th slot after SlotToCoeff is given as $e_{\mathsf{boot},j}(\zeta_j) = \sum_{i=0}^{N-1} e_{\mathsf{mod},i} \cdot \zeta_j^i$ which is the sum of thousands of independent random variables, where $e_{\mathsf{mod},i}$ denotes the error in the i-th slot after EvalMod and $|\zeta_j| = 1$.

The minimax approximate polynomial minimizes $\|e_{\mathsf{aprx}}(t)\|_{\mathsf{sup}}$ [7,24]. In this case, we have $e_{\mathsf{mod},i} = e_{\mathsf{aprx}}(t_i) + e_{\mathsf{noise},i}$, where t_i is the i-th slot value after CoeffToSlot and $e_{\mathsf{noise},i}$ is the random error by the noisy polynomial basis of CKKS. Hence, the minimax approximation minimizes $\max\left(\left|e_{\mathsf{aprx}}(t_i) \cdot \zeta_j^i\right|\right) =$

$\|e_{\mathsf{aprx}}(t_i)\|_{\mathsf{sup}}$, not $\max(|e_{\mathsf{boot},j}|)$. In other words, we observe that the final bootstrapping error is $e_{\mathsf{boot},j}$, and we have

$$\begin{aligned}\max(|e_{\mathsf{boot},j}|) = \max\left(\left|\sum_{i=0}^{N-1} e_{\mathsf{mod},i} \cdot \zeta_j^i\right|\right) &= \max\left(\left|\sum_{i=0}^{N-1} (e_{\mathsf{aprx}}(t_i) + e_{\mathsf{noise},i}) \cdot \zeta_j^i\right|\right) \\ &\leq \max\left(\left|\sum_{i=0}^{N-1} e_{\mathsf{aprx}}(t_i) \cdot \zeta_j^i\right|\right) + \max\left(\left|\sum_{i=0}^{N-1} e_{\mathsf{noise},i} \cdot \zeta_j^i\right|\right), \end{aligned} \tag{2}$$

where

$$\max\left(\left|\sum_{i=0}^{N-1} e_{\mathsf{aprx}}(t_i) \cdot \zeta_j^i\right|\right) \leq \left\|\zeta_j^0 \cdot e_{\mathsf{aprx}}\right\|_{\mathsf{sup}} + \cdots + \left\|\zeta_j^{N-1} \cdot e_{\mathsf{aprx}}\right\|_{\mathsf{sup}}.$$

Hence, the minimax approximate polynomial does not guarantee the minimum infinity norm of bootstrapping error but provides an upper bound only for the approximation error term. Besides, it is challenging to optimize polynomial coefficients for noisy basis in the existing minimax approximation.

In contrast, the proposed error variance-minimizing approximate polynomial minimizes $Var[e_{\mathsf{mod},j}]$. Thus, it also minimizes the final bootstrapping error $Var[e_{\mathsf{boot},j}]$, as

$$Var[e_{\mathsf{boot},j}] = Var[e_{\mathsf{mod},0} \cdot \zeta_j^0] + \cdots + Var[e_{\mathsf{mod},N-1} \cdot \zeta_j^{(N-1)}].$$

The above equation implies that minimizing the variance of the approximate error is optimal to reduce the bootstrapping error of the CKKS scheme in the aspect of SNR. Due to the characteristics of SlotToCoeff, we have the tight value of the variance of bootstrapping error, while the minimax provides an upper bound of infinity norm. In other words, we can optimize our objective function by the proposed error variance-minimizing approximate polynomial, whereas the minimax approach optimizes an upper bound (the right-hand side of (2)) instead of the bootstrapping error (the left-hand side of (2)).

Depth Optimality of Direct Approximation. As shown in (1), the proposed method approximates the objective function directly, while the prior works approximate trigonometric functions [3,7,9,18,24]. Let $P_{\mathsf{deg}} \subset \mathbb{C}[X]$ be the set of all polynomials whose degree is less than or equal to deg. When we perform a direct approximation, the algorithm finds an approximate polynomial among all elements of P_{deg}, and its multiplicative depth is $\lceil \log(\mathsf{deg}) \rceil$.

When we use the approximation of trigonometric function, the search space of the approximation algorithm is much more limited. For example, as in [3,24], suppose that we use the double angle formula twice and approximate polynomial for cosine and arcsine of degree deg_1 and deg_2, respectively. Then the search space is

$$\left\{f_2 \circ g \circ f_1 | f_1 \in P_{\mathsf{deg}_1}, f_2 \in P_{\mathsf{deg}_2}, \text{ and } g(x) = (x^2-1)^2 - 1\right\}.$$

We can see that the search space is much smaller than $P_{4\mathsf{deg}_1\mathsf{deg}_2}$, and its multiplicative depth is $\lceil \log(\mathsf{deg}_1 + 1) \rceil + 2 + \lceil \log(\mathsf{deg}_2 + 1) \rceil \geq \lceil \log(4\mathsf{deg}_1\mathsf{deg}_2 + 1) \rceil$.

Hence, the direct approximation in $P_{4\mathsf{deg}_1\mathsf{deg}_2}$ has more chance to find a better approximation as well as it has less multiplicative depth.

4.3 Noisy Polynomial Basis and Polynomial Evaluation in the CKKS Scheme

Let $\{\phi_0(x), \phi_1(x), \ldots, \phi_n(x)\}$ denote a polynomial basis of degree n such that every $\phi_k(t)$ is odd for an odd k. When a polynomial $p(x) = \sum c_i\phi_i(x)$ is evaluated homomorphically, it is expected that the result is $p(x) + e$ for a small error e. In the CKKS scheme, there exists an error in encrypted data, and thus, each $\phi_i(x)$ contains independent $e_{\mathsf{basis},i}$, namely the basis error. Thus, the output is

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

In general, $\sum c_i e_{\mathsf{basis},i}$ is small as $e_{\mathsf{basis},i}$ are small. However, when $|c_i|$ are much greater than $p(x)$, $\sum c_i e_{\mathsf{basis},i}$ dominates $p(x)$.

The basis errors, $e_{\mathsf{basis},i}$ are introduced by rescaling, key switching, and encryption errors, which are independent of the message. Each $\phi_i(x)$ is usually obtained from smaller-degree polynomials, and thus there may be some correlation between $e_{\mathsf{basis},i}$'s. If we assume that each $e_{\mathsf{basis},i}$ is independent, then the variance of $\sum c_i \cdot e_{\mathsf{basis},i}$ becomes $\sum c_i^2 \cdot Var(e_{\mathsf{basis},i})$ and w_i in (1) corresponds to $Var(e_{\mathsf{basis},i})$. The experiments in Sect. 6 support that our approximation with this independence assumption obtains accurate approximations for bootstrapping in practice. In other words, we do not need exact distributions of $e_{\mathsf{basis},i}$ in practice.

In conclusion, the magnitude of c_i's should be controlled when we find an approximate polynomial. A high-degree approximate polynomial for modular reduction and piece-wise cosine function has large coefficients magnitude in previous works [19,24]. There have been series of studies in approximate polynomials in the CKKS scheme [3,7,12,18,19,24,25], but the errors amplified by coefficients were not considered in the previous studies.

4.4 Optimal Approximate Polynomial for Bootstrapping and the Magnitude of Its Coefficients

The most depth-consuming and noisy part of bootstrapping is EvalMod. In this subsection, we show how to find the optimal approximate polynomial for EvalMod in the aspect of SNR. By scaling the modular reduction function $[\cdot]_q$ by $\frac{1}{q}$, we define

$$f_{\mathsf{mod}} : \bigcup_{i=-K+1}^{K-1} I_i \to [-\epsilon, \epsilon]\,, \text{ that is, } f_{\mathsf{mod}}(t) = t - i \text{ if } t \in I_i,$$

where $I_i = [i - \epsilon, i + \epsilon]$ for an integer $-K < i < K$. 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 $m(X)$. Let T be the random variable of input t of $f_{\mathsf{mod}}(t)$. Then, $T = R + I$, where R is the random variable of the rational part r, and I is the random variable of the integer part i. We note 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 the probability mass functions or probability density functions of T, I, and R, respectively.

The approximation error for t is given as

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

where a polynomial $p(t) = \sum c_i \phi_i(t)$ approximates $f_{\mathsf{mod}}(t)$. We can set $p(t)$ as an odd function because $f_{\mathsf{mod}}(t)$ is odd. Then the variance of e_{aprx} is given as

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

where the mean of e_{aprx} is zero by assuming that $\Pr_T(t)$ is even. It is noted that the integral can be directly calculated or approximated by the sum of discretized values as in [25].

The basis error $\sum c_i^2 \cdot Var(e_{\mathsf{basis},i})$ is also added as discussed in Subsect. 4.3. We generalize $Var(e_{\mathsf{basis},i})$ by w_i. Then, we find $\boldsymbol{c}^*$ such that

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

and its solution satisfies

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

where $\boldsymbol{c} = (c_1, c_3, \ldots, c_n)$ and $\boldsymbol{w} = (w_1, w_3, \ldots, w_n)$ are coefficient and weight constant vectors, respectively. We note that the objective function is convex.

It is noted that $Var(e_{\mathsf{basis},i})$ may differ by i, and thus, a precise adjustment of the magnitude of polynomial coefficients can be made by multiple weight constants, w_i's. The following theorem states that we can find the approximate polynomial for $p(t)$ efficiently; the computation time of solving this system of linear equations is the same as that of finding an interpolation polynomial for given points. It will be faster than the improved multi-interval Remez algorithm [24], as the Remez algorithm requires an interpolation per each iteration.

Theorem 1. *There exists a polynomial-time algorithm that finds the odd polynomial $p(t) = \sum c_i \phi_i(t)$ satisfying*

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

when $\Pr_T(t)$ is an even function.

Proof. By substituting $p(t) = \sum c_i \phi_i(t)$ from $Var[e_{\mathsf{aprx}}] = E[e_{\mathsf{aprx}}^2] = E[f_{\mathsf{mod}}(t)^2] - 2E[f_{\mathsf{mod}}(t) \cdot p(t)] + E[p(t)^2]$, we have

$$\frac{\partial}{\partial c_j} Var[e_{\mathsf{aprx}}] = -2E[f_{\mathsf{mod}}(t)\phi_j(t)] + 2\sum_i c_i \cdot E[\phi_i(t)\phi_j(t)].$$

Therefore, one can find $\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} \left(Var[e_{\mathsf{aprx}}] + \sum w_i c_i^2\right)$ by solving the following system of linear equations:

$$(\mathbf{T} + \boldsymbol{w}\mathbf{I}) \cdot \boldsymbol{c} = \boldsymbol{y}, \tag{4}$$

where $\boldsymbol{w}$ is a diagonal matrix where $\boldsymbol{w}_{ii} = w_i$,

$$\mathbf{T} = \begin{bmatrix} E[\phi_1 \cdot \phi_1] & E[\phi_1 \cdot \phi_3] & \dots & E[\phi_1 \cdot \phi_n] \\ E[\phi_3 \cdot \phi_1] & E[\phi_3 \cdot \phi_3] & \dots & \vdots \\ \vdots & & \ddots & \vdots \\ E[\phi_n \cdot \phi_1] & E[\phi_n \cdot \phi_3] & \dots & E[\phi_n \cdot \phi_n] \end{bmatrix}, \text{ and } \boldsymbol{y} = \begin{bmatrix} E[f_{\mathsf{mod}} \cdot \phi_1] \\ E[f_{\mathsf{mod}} \cdot \phi_3] \\ \vdots \\ E[f_{\mathsf{mod}} \cdot \phi_n] \end{bmatrix}.$$

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

4.5 Statistical Characteristics of Modular Reduction

The input distribution of the proposed approximate polynomial, represented by $\Pr_I$ and $\Pr_R$, is required to find $\mathbf{T}$ and $\boldsymbol{y}$. Unfortunately, in HE, it is not always possible to utilize the message distribution as it might be related to security. However, we observe and analyze that the major part of the input distribution of approximate polynomial is unrelated to the security.

After ModRaise, the plaintext in the ciphertext $\mathsf{ct} = (b, a)$ is given as

$$t(X) = q \cdot I(X) + m(X) = \langle \mathsf{ct}, \mathsf{sk} \rangle \left(\bmod X^N + 1\right),$$

where sk has Hamming weight h and each coefficient of a ciphertext (b, a) is an element of $\mathbb{Z}_q$. The RLWE assumption states that a ciphertext is uniformly distributed over $\mathcal{R}_q^2$, and thus each coefficient of b and a is distributed uniformly at random. In other words, coefficients of $b + a \cdot s$ follow the well-known Irwin-Hall distribution. Especially, it is a sum of $h + 1$ independent and identically distributed uniform random variables.

We note that one can exploit the distribution of I without security concerns. This is because the probability distribution $\Pr_I$ is given by the RLWE assumption (that b and a are uniformly distributed), regardless of the message distribution. Also, the implementation results in Sect. 6 show that we can achieve high approximation accuracy of the proposed approximate polynomial using $\Pr_I$ even if we set to the worst-case of $\Pr_R$.

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

i	$\Pr_I(i)$		i	$\Pr_I(i)$		i	$\Pr_I(i)$	
	Experiment	Theory		Experiment	Theory		Experiment	Theory
0	$9.94{\cdot}10^{-2}$	$9.91{\cdot}10^{-2}$	±8	$1.36{\cdot}10^{-2}$	$1.37{\cdot}10^{-2}$	±16	$3.34{\cdot}10^{-5}$	$3.48{\cdot}10^{-5}$
±1	$9.64{\cdot}10^{-2}$	$9.61{\cdot}10^{-2}$	±9	$8.02{\cdot}10^{-3}$	$8.10{\cdot}10^{-3}$	±17	$1.16{\cdot}10^{-5}$	$1.23{\cdot}10^{-5}$
±2	$8.78{\cdot}10^{-2}$	$8.76{\cdot}10^{-2}$	±10	$4.44{\cdot}10^{-3}$	$4.50{\cdot}10^{-3}$	±18	$3.84{\cdot}10^{-6}$	$4.09{\cdot}10^{-6}$
±3	$7.52{\cdot}10^{-2}$	$7.51{\cdot}10^{-2}$	±11	$2.30{\cdot}10^{-3}$	$2.34{\cdot}10^{-3}$	±19	$1.20{\cdot}10^{-6}$	$1.27{\cdot}10^{-6}$
±4	$6.05{\cdot}10^{-2}$	$6.05{\cdot}10^{-2}$	±12	$1.12{\cdot}10^{-3}$	$1.15{\cdot}10^{-3}$	±20	$3.40{\cdot}10^{-7}$	$3.71{\cdot}10^{-7}$
±5	$4.58{\cdot}10^{-2}$	$4.58{\cdot}10^{-2}$	±13	$5.15{\cdot}10^{-4}$	$5.26{\cdot}10^{-4}$	±21	$9.41{\cdot}10^{-8}$	$1.01{\cdot}10^{-7}$
±6	$3.25{\cdot}10^{-2}$	$3.26{\cdot}10^{-2}$	±14	$2.20{\cdot}10^{-4}$	$2.27{\cdot}10^{-4}$	±22	–	$2.58{\cdot}10^{-8}$
±7	$2.17{\cdot}10^{-2}$	$2.18{\cdot}10^{-2}$	±15	$8.84{\cdot}10^{-5}$	$9.15{\cdot}10^{-5}$	±23	–	$6.15{\cdot}10^{-8}$

We can numerically obtain the distribution of I or analytically derive its distribution. Table 1 is the probability mass function of I, obtained numerically using `SEAL` and analytically derived by using Irwin-Hall distribution. It is shown that the experimental results and our probability analysis using the Irwin-Hall distribution agree. In previous researches, a heuristic assumption is used, and a high-probability upper bound $K = O(\sqrt{h})$ for $\|I\|_\infty$ is used for polynomial approximation [3,9,18,24], but they could not utilize the distribution of I.

For $\Pr_R$, we can set the worst-case scenario; message $m(X)$ is uniformly distributed over $\|m\|_\infty < \epsilon \cdot q$, as it results in the most significant entropy of the message. The experimental results in Sect. 6 show that even though the worst-case scenario is used and the distribution of $m(X)$ is different from the actual one, the error value in the proposed method is comparable to the prior arts [3,24] while consuming less depth. Also, in the experiment of [3], a uniformly distributed message is used to simulate the bootstrapping error and utilized the fact that $m(X)$ is highly probable to be in the center to use a small-degree arcsine Taylor expansion. We note that we can also heuristically assume a specific distribution in our bootstrapping when we specify $\Pr_R$ for (1) and improve the precision.

5 Lazy Baby-Step Giant-Step Algorithm

This section proposes error and complexity optimization when evaluating the error variance-minimizing approximate polynomial in bootstrapping. There are two optimizations: First, we show that the error variance-minimizing approximate polynomial is odd, and thus, we can ignore the even-degree terms. Second, we propose a novel evaluation algorithm, namely the lazy-BSGS algorithm, to reduce the computational complexity of EvalMod.

5.1 Reducing Error and Complexity Using Odd Function Property

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

Error Variance-Minimizing Polynomial for an Odd Function. This subsection shows that the variance-minimizing polynomial is an odd function, where $\Pr_T(t)$ is even. Using an odd polynomial, we can reduce the approximation error and the computation time to find the proposed approximate polynomial. First of all, we only need to integrate over the positive domain when obtaining each element of (4). Second, the number of operations to evaluate the approximate polynomial can also be reduced by omitting even-degree terms when using the lazy-BSGS algorithm Algorithm 2 in the following subsection. Finally, the basis error is also reduced as only half of the terms are added.

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

Theorem 2. *If $\Pr_T(t)$ 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: Equation (3) is a quadratic polynomial for the coefficients $\boldsymbol{c}$, and thus there exists one and only solution.
Oddness: Let P_m be the subspace of the polynomials of degree at most m and $f_m(t)$ denote the unique element in P_m that is closest to $f(t)$ in terms of 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

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

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 $f_m(t) = -f_m(-t)$.

□

5.2 Lazy Baby-Step Giant-Step Algorithm

In this subsection, we propose a new algorithm that efficiently evaluates arbitrary polynomials over the CKKS scheme, namely the lazy-BSGS algorithm in Algorithm 2, and we extend it to the odd polynomials. We apply the lazy relinearization and rescaling technique [1,2,8,22] to the BSGS algorithm to improve its time complexity and error performance. For example, when we evaluate a polynomial of degree 711 by using the ordinary BSGS algorithm in [18], 58 non-scalar multiplications are required; however, when we use the odd-BSGS

Algorithm 2. Lazy-BSGS Algorithm

Instance: A ciphertext ct of t, a polynomial $p(X)$ of degree deg.
Output: A ciphertext encrypting $p(t)$.

```
Let $l$ be the smallest integer satisfying $2^l k > n$ for an even number $k$.
procedure SetUpLazy($\mathsf{ct}, l, k$)
    for $i = 2; i < k; i \leftarrow 2i$ do
        $\mathsf{ct}_i \leftarrow 2 \cdot \mathsf{ct}_{i/2}\mathsf{ct}_{i/2} - 1$
        $\mathsf{ct}_i \leftarrow \mathsf{RL}(\mathsf{ct}_i)$
    end for
    for $i = 3; i < k; i \leftarrow i + 1$ do
        $i_0, i_1 \leftarrow 2^{\lfloor \log i \rfloor}, i - 2^{\lfloor \log i \rfloor}$
        $\mathsf{ct}_{i_0} \leftarrow \mathsf{RL}(\mathsf{ct}_{i_0})$
        $\mathsf{ct}_i \leftarrow 2 \cdot \mathsf{ct}_{i_0}\mathsf{ct}_{i_1} - \mathsf{ct}_{i_0 - i_1}$
    end for
    if $k/2$ is even then                    ▷ To reduce the error, see Fig. 2
        $\mathsf{ct}_k \leftarrow 2 \cdot \mathsf{ct}_{k/2+1}\mathsf{ct}_{k/2-1} - \mathsf{ct}_2$
    else
        $\mathsf{ct}_k \leftarrow 2 \cdot \mathsf{ct}_{k/2}\mathsf{ct}_{k/2} - 1$
    end if
    $\mathsf{ct}_k \leftarrow \mathsf{RL}(\mathsf{ct}_k)$
    for $i = 2k; i < \mathsf{deg}; i \leftarrow 2i$ do
        $i_0, i_1 \leftarrow 2^{\lfloor \log i \rfloor}, i - 2^{\lfloor \log i \rfloor}$
        $\mathsf{ct}_i \leftarrow 2 \cdot \mathsf{ct}_{i/2}\mathsf{ct}_{i/2} - 1$
        $\mathsf{ct}_i \leftarrow \mathsf{RL}(\mathsf{ct}_i)$
    end for
    $\{\mathsf{ct}_i\} \leftarrow$ encryptions of $T_i(t)$
    $\{\mathsf{ct}_{2^i k}\} \leftarrow$ encryptions of $T_{2^i k}(t)$
end procedure
procedure GiantStepLazy($p(X), \{\mathsf{ct}_i\}, l, k$)
    if $deg(p) < k$ then
        return BabyStep($p(X), \{\mathsf{ct}_i\}, k$)
    end if
    Find $q(X), r(X)$ s.t. $p(X) = q(X) \cdot T_{2^i k}(X) + r(X)$
    $\mathsf{ct}_q \leftarrow$GiantStep($q(X), \{\mathsf{ct}_i\}, l, k$)
    $\mathsf{ct}_r \leftarrow$GiantStep($r(X), \{\mathsf{ct}_i\}, l, k$)
    $\mathsf{ct}_q \leftarrow \mathsf{RL}(\mathsf{ct}_q)$
    return $\mathsf{ct}_q \cdot \mathsf{ct}_{2^i k} + \mathsf{ct}_r$
end procedure
```

algorithm [24], 46 non-scalar multiplications are required. Moreover, the lazy relinearization method reduces the number of relinearizations to 33, which is the same number of relinearizations for a polynomial of degree 220 using the ordinary BSGS algorithm.

The relinearization and rescaling introduce additional errors in the CKKS scheme, and the error propagates along with homomorphic operations. Hence, we should delay the relinearization and rescaling to reduce the error of the resulting ciphertext. Moreover, those operations, especially relinearization, require many

NTTs, and thus it requires lots of computation. For some circuits, we can reduce the numbers of relinearizations and rescalings by delaying them. We observe that we can perform plaintext addition, ciphertext addition, and scalar multiplication to a ciphertext before relinearization.

A ciphertext is a three-tuple $(d_0, d_1, d_2) \in \mathcal{R}^3_{q_L}$ such that $\langle (d_0, d_1, d_2), (1, s, s^2) \rangle = m + e$. A plaintext $u \in \mathcal{R}$ can be multiplied homomorphically by calculating $(u \cdot d_0, u \cdot d_1, u \cdot d_2)$, but we note that the error is amplified by the magnitude of u. When we add a ciphertext (b, a) to (d_0, d_1, d_2), we get $(d_0 + b, d_1 + a, d_2)$. However, as the scaling factor of ciphertext is changed along with homomorphic operations, we should make sure that the scaling factors of the two ciphertexts are identical when we add two ciphertexts. If not, we can multiply a constant, Δ_1/Δ_2, to a ciphertext which has a smaller scaling factor and then add, where Δ_1 is the larger scaling factor, and Δ_2 is the smaller scaling factor. Alternatively, we can use the scaling factor management technique proposed in [21].

We propose the lazy-BSGS algorithm, which reduces the numbers of rescalings and relinearizations, and we analyze its computational complexity. Here, we rigorously analyze the number of relinearizations as its complexity is much higher than other operations, and we note that the number of rescalings is also similar. As we use the Chebyshev polynomial of the first kind as the polynomial basis, we explain the lazy-BSGS algorithm with Chebyshev polynomial. For the sake of brevity, we denote ciphertext-ciphertext multiplication by $\cdot$, and the ciphertext of $T_j(t_0)$ is denoted by ct_j, where T_j is Chebyshev polynomial of the first kind with degree j.

SetUp finds all the Chebyshev polynomials of degree less than or equal to k, and $T_{2^i k}$ for $i < l$, for given parameter k and l. We use $T_a = 2 \cdot T_{2^i} \cdot T_{a-2^i} - T_{2^{i+1}-a}$ to find ct_a, where $i = \lfloor \log(a) \rfloor$. We note that one can alternatively use multiplication of odd degree polynomials to reduce the basis error, which is presented in Subsect. 5.4.

First, we find ct_{2^i} for $i < k$, and these are used to find other Chebyshev bases with degrees less than k. Thus, we rescale and relinearize them, which requires $\lfloor \log(k-1) \rfloor$ rescalings and relinearizations. When calculating $\mathsf{ct}_a = 2 \cdot \mathsf{ct}_{2^i} \cdot \mathsf{ct}_{a-2^i} - \mathsf{ct}_{2^{i+1}-a}$, if ct_{a-2^i} is a three-tuple ciphertext, we relinearize it (and rescale it if needed.) We note that the lazy rescaling makes it possible to accurately subtract $\mathsf{ct}_{2^{i+1}-a}$ from $2 \cdot \mathsf{ct}_{2^i} \cdot \mathsf{ct}_{a-2^i}$ without level consumption as follows. We do not rescale $\mathsf{ct}_{2^i} \cdot \mathsf{ct}_{a-2^i}$ here, and thus the scaling factor of $2 \cdot \mathsf{ct}_{2^i} \cdot \mathsf{ct}_{a-2^i}$ is maintained as $\approx q^2$. Obviously, the level of $\mathsf{ct}_{2^{i+1}-a}$ is larger than that of $\mathsf{ct}_{2^i} \cdot \mathsf{ct}_{a-2^i}$. When their scaling factors are different, we multiply $(\Delta_{2^i} \cdot \Delta_{a-2^i}) \cdot p_l / \Delta_{2^{i+1}-a}$ to $\mathsf{ct}_{2^{i+1}-a}$ and rescale if $\Delta_{2^{i+1}-a} \approx q^2$, or multiply $(\Delta_{2^i} \cdot \Delta_{a-2^i}) / \Delta_{2^{i+1}-a}$ if $\Delta_{2^{i+1}-a} \approx q$, where Δ_j denotes the scaling factor of ct_j, and p_l is the last prime of modulus chain for ct_{a-2^i}. Now, the scaling factors of $\mathsf{ct}_{2^{i+1}-a}$ and $2\mathsf{ct}_{2^i} \cdot \mathsf{ct}_{a-2^i}$ are the same, and thus we can subtract them without additional error from the difference of scale.

To evaluate $\mathsf{ct}_{2^i} \cdot \mathsf{ct}_{a-2^i}$, we need to relinearize ct_{a-2^i} if it is not relinearized yet. Hence, we need relinearized ct_j's for $j < 2^{\lfloor \log k-1 \rfloor -1}$ to find ct_i for all

$i < 2^{\lfloor \log k-1 \rfloor}$. Moreover, if $k \geq 2^{\lfloor \log k-1 \rfloor} + 2^{\lfloor \log k-1 \rfloor -1}$, we need $k - 2^{\lfloor \log k-1 \rfloor} + 2^{\lfloor \log k-1 \rfloor -1}$ more relinearizations. Each $\mathsf{ct}_{2^i k}$ should be relinearized as it is used for multiplication in GiantStep, which requires l relinearizations. In conclusion, we do

$$\lfloor \log(k-1) \rfloor + (2^{\lfloor \log k-1 \rfloor -1} - 1) + l$$

relinearizations in SetUp. If $k \geq 2^{\lfloor \log k-1 \rfloor} + 2^{\lfloor \log k-1 \rfloor -1}$, $\left(k - 3 \cdot 2^{\lfloor \log k-1 \rfloor -1}\right)$ additional relinearizations are required.

BabyStep performs only plaintext multiplication and addition. Hence, it does not require relinearization in our lazy-BSGS algorithm, but the scale for baby-step polynomial coefficients should be adequately scaled to make the added ciphertexts have identical scaling factors, but this process does not involve additional computation at all. Note that the resulting ciphertext of BabyStep is not relinearized, i.e., it has size 3.

In GiantStep, the ct_q is relinearized before multiplied to $\mathsf{ct}_{2^i k}$. Hence, the number of relinearizations is $2^{l-1} + 2^{l-2} + \cdots + 1 = 2^l - 1$, and the final result is not relinearized. Thus, we perform relinearization once more right before SlotToCoeff.

Finally, the number of relinearizations in lazy-BSGS is

$$\lfloor \log(k-1) \rfloor + (2^{\lfloor \log k-1 \rfloor -1} - 1) + l + 2^l$$

if $k < 2^{\lfloor \log k-1 \rfloor} + 2^{\lfloor \log k-1 \rfloor -1}$ and otherwise

$$\lfloor \log(k-1) \rfloor + \left(2^{\lfloor \log k-1 \rfloor -1} - 1\right) + l + 2^l + \left(k - 3 \cdot 2^{\lfloor \log k-1 \rfloor -1}\right).$$

Lazy-BSGS for Odd Polynomial. We can naturally extend the lazy-BSGS for the odd polynomials. Here, SetUp finds all the odd-degree Chebyshev polynomials of degrees less than k. To find an odd-degree Chebyshev polynomial, we need an even-degree Chebyshev polynomial because the multiplication of odd-degree Chebyshev polynomials is not an odd-degree polynomial. Hence, we use T_{2^i} to find ct_a, where $i = \lfloor \log(a) \rfloor$, and thus we rescale and relinearize them, which requires $\lfloor \log(k-1) \rfloor$ rescaling and relinearization. Thus, the number of relinearizations in lazy-BSGS for odd polynomial is

$$\lfloor \log(k-1) \rfloor + (2^{\lfloor \log k-1 \rfloor -1}/2 - 1) + l + 2^l$$

if $k < 2^{\lfloor \log k-1 \rfloor} + 2^{\lfloor \log k-1 \rfloor -1}$ and otherwise

$$\lfloor \log(k-1) \rfloor + \left(2^{\lfloor \log k-1 \rfloor -1}/2 - 1\right) + l + 2^l + \left(k - 3 \cdot 2^{\lfloor \log k-1 \rfloor -1}\right)/2.$$

Using the error variance-minimizing approximate polynomial in bootstrapping requires evaluating a polynomial with a higher degree than the previous composition methods. However, the lazy-BSGS algorithm reduces the time complexity by half, compared to ordinary BSGS mentioned in Sect. 2. As a result, the lazy-BSGS algorithm makes the time complexity of evaluating our polynomial comparable to the previous algorithm. Figure 1 compares our lazy-BSGS algorithm, odd-BSGS algorithm [24], and the original BSGS algorithm [18].

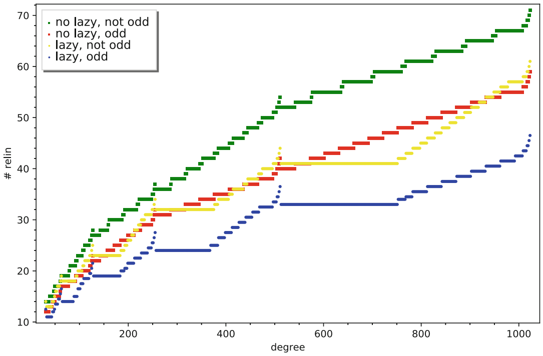


Fig. 1. Number of relinearizations for the variants of BSGS algorithms.

The lazy-BSGS algorithm is given in Algorithm 2 in detail. We note that the methods in [3] should be applied for optimal depth and scale-invariant evaluation, but we omit it for the sake of brevity. However, we note that Fig. 1 considers the depth optimization in [3], and thus, the number of relinearizations is high when the degree is close to a power of two. The BSGS coefficients are pre-computed for optimal parameters k and l to minimize the complexity.

5.3 Error Variance-Minimizing Approximate Polynomial for BSGS Algorithm

In this subsection, we propose a method to find the variance-minimizing approximate polynomial for the odd-BSGS algorithm. We generalize the amplified basis error and find the variance-minimizing coefficients for the odd-BSGS algorithm. The numerical method to select the weight constantly is also proposed.

BSGS Algorithm Coefficients and Minimizing the Approximation Error Variance. In the lazy-BSGS algorithm, we divide the given polynomial by $T_{2^i k}$ and evaluate its quotient and remainder. Hence, each polynomial basis is multiplied by a divided coefficient, not c_i. We define $\boldsymbol{d}$ by the vector of coefficients multiplied to the basis in BabyStep, in other words, we have 2^l polynomials in BabyStep such that $p_i^{\mathsf{baby}}(t) = \sum_{j \in \{1,3,\ldots,k-1\}} d_{i,j} T_j(t)$ for $i = 0, 1, \ldots, 2^l - 1$, and $\boldsymbol{d} = (d_{0,1}, d_{0,3}, \ldots, d_{2^l-1,\mathsf{deg}-k\cdot 2^{l-1}})$.

We should reduce the magnitude of $\boldsymbol{d}$, to reduce the basis error. Let $p(t) = \sum c_i T_i(t)$, and then, $\boldsymbol{c}$ and $\boldsymbol{d}$ have the following linearity:

$$\boldsymbol{c} = \mathbf{L} \cdot \boldsymbol{d} = \begin{bmatrix} \mathbf{A}_{2^{l-1}k} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{A}_{2^{l-2}k} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{2^{l-2}k} \end{bmatrix} \cdots \begin{bmatrix} \mathbf{A}_k & & \\ & \ddots & \\ & & \mathbf{A}_k \end{bmatrix} \cdot \boldsymbol{d}, \qquad (5)$$

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. Hence, the linear equation to find the error variance-minimizing approximate polynomial (4) is modified for the BSGS algorithm as

$$(\mathbf{L}^{\mathsf{T}}\mathbf{T}\mathbf{L} + \boldsymbol{w}\mathbf{I}) \cdot \boldsymbol{d} = \boldsymbol{y}. \tag{6}$$

Generalization of Weight Constant. Let E_p be a function of $\boldsymbol{d}$, which is the variance of basis error amplified by the BSGS algorithm. We simplify E_p by a heuristic assumption that T_i's are independent and the encryptions of $T_k(t), \ldots, T_{2^{l-1}k}(t)$ have small error. Let $\hat{T}_i$ be the product of all $T_{2^j k}$'s multiplied to p_i in the giant step, for example, $\hat{T}_0 = 1$ and $\hat{T}_3 = T_k T_{2k}$. Considering the error multiplied by $d_{i,j}$, $e_j \cdot \hat{T}_i$ is the dominant term as T_i has zero mean for odd integer i as it is an odd polynomial. Thus, we can say that

$$E_p \approx \sum_i \sum_j d_{i,j}^2 E[\hat{T}_i^2] Var[e_{\mathsf{basis},j}],$$

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 $\mathbf{H}_{ki+j,ki+j} = E[\hat{T}_i^2] Var[e_{\mathsf{basis},j}]$. Thus, (3) is generalized as

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

Equation (5) gives us that the optimal coefficient $\boldsymbol{d}^*$ satisfies

$$\left(\mathbf{L}^{\mathsf{T}}\mathbf{T}\mathbf{L} + \mathbf{H}\right) \boldsymbol{d}^* = \mathbf{L}^{\mathsf{T}}\boldsymbol{y}. \tag{7}$$

Numerical Method of Finding Optimal Approximate Polynomial. Instead of finding E_p, a simple numerical method can also be used. In practice, the numerical method shows good error performance in the implementation in Subsect. 6.1. We can let $w_i = w$ for all i and find w numerically. When w increases, the magnitude of coefficients decreases, and $Var[e_{\mathsf{aprx}}]$ increases, and thus its sum is a convex function of w. The magnitude of the basis errors that are amplified by coefficients $\boldsymbol{d}$ has the order of the rescaling error whose variance is $\frac{2n(h+1)}{12 \cdot q^2}$, where n is the number of slots. In other words, we adjust w to minimize

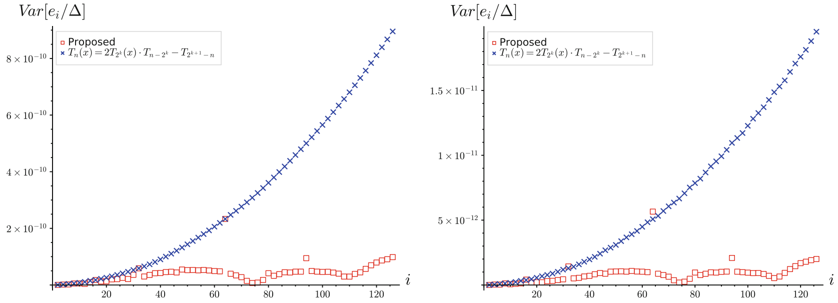
$$Var[e_{\mathsf{aprx}}] + w \cdot \|\boldsymbol{d}\|_2^2, \tag{8}$$

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

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

Lemma 1 (Rescaling error [9]). *The error variance of rescaling error is $\frac{2n(h+1)}{12}$, where h is key Hamming weight and n is the number of slots.*

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



(a) Error variance in HEAAN $\epsilon = 2^{-10}$ and $q = 2^{40}$

(b) Error Variance in SEAL $\epsilon = 2^{-5}$ and $q = 2^{45}$

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

5.4 Basis Error Variance Minimization for Even-Degree Terms

In this subsection, we show that the even-degree Chebyshev polynomials in CKKS have huge errors and propose a method to find a small-error Chebyshev polynomial. In the BSGS algorithm, we use even-degree Chebyshev polynomials, namely, $T_{2^i k}(t)$. For depth and simplicity, we usually obtain $T_a(t)$ by using

$$T_a(t) = 2 \cdot T_{2^i}(t) \cdot T_{a-2^i}(t) - T_{2^{i+1}-a}(t),$$

where $i = \lfloor \log(a) \rfloor$. Let ct_i be the ciphertext of message $T_i(t)$ with scaling factor $\varDelta$, and it contains error $e_{\mathsf{basis},i}$. Then, the error in ct_{i+j} obtained by $\mathsf{ct}_{i+j} = 2\mathsf{ct}_i \cdot \mathsf{ct}_j - \mathsf{ct}_{|i-j|}$ is given as

$$(2T_i(t)e_{\mathsf{basis},j} + 2T_j(t)e_{\mathsf{basis},i})\varDelta + 2e_{\mathsf{basis},i}e_{\mathsf{basis},j} - e_{\mathsf{basis},|i-j|}. \tag{9}$$

As $\varDelta \gg e_{\mathsf{basis},i}, e_{\mathsf{basis},j}$, the dominant term of error variance in (9) is

$$\begin{aligned} &Var[2T_i(t)e_{\mathsf{basis},j} + 2T_j(t)e_{\mathsf{basis},i}] \\ &\qquad \approx 4E[T_i(t)^2]Var[e_{\mathsf{basis},j}] + 4E[T_j(t)^2]Var[e_{\mathsf{basis},i}]. \end{aligned} \tag{10}$$

As a simple example, it is shown that $E[T_i(t)^2]$ is close to one when i is an even number for low-degree polynomials, where t is a value after CoeffToSlot. Meanwhile, $E[T_i(t)]$ is zero and $Var[T_i(t)]$ is a small value when i is odd. Thus, following to (10), the error remains large when it is multiplied by an even-degree Chebyshev polynomial in the calculation of the next Chebyshev polynomial. Therefore, when a is even, ct_a should be calculated by $\mathsf{ct}_a = 2\mathsf{ct}_{2^i-1} \cdot \mathsf{ct}_{a+1-2^i} - \mathsf{ct}_{2^{i+1}-2-a}$ rather than $\mathsf{ct}_a = 2\mathsf{ct}_{2^i} \cdot \mathsf{ct}_{a-2^i} - \mathsf{ct}_{2^{i+1}-a}$. Also, it is noted that, for the above reasons, the power-of-two polynomials should have a large basis error.

Figure 2 shows the experimental results of the variance of error in encryption of $T_i(t)$ for even i's, where t is the output value of CoeffToSlot.

Table 2. The second moment of $T_i(t)$ when t is value after SlotToCoeff and $N = 2^{15}$

	i	0	1	2	3	4	5	6	7	8	9
$E[T_i(t)^2]$	$h = 192$	1.00	0.035	0.905	0.187	0.718	0.361	0.579	0.457	0.520	0.491
	$h = \sqrt{N}$	1.00	0.013	0.950	0.105	0.828	0.239	0.696	0.358	0.596	0.426

Square mark and x mark legends are the results with and without operation reordering, respectively. In other words, square marks are results from $\mathsf{ct}_a = 2\mathsf{ct}_{2^i-1} \cdot \mathsf{ct}_{a-2^i+1} - \mathsf{ct}_{2^{i+1}-2-a}$ for even a. The experimental result in Fig. 2 supports our argument that multiplying even-degree Chebyshev polynomials amplifies the error. We can see that the basis error is significantly improved by reordering operations. For example, the variance of error in ct_{74} is reduced to 1/1973 compared to that of without reordering (Fig. 2).

6 Performance Analysis and Comparison

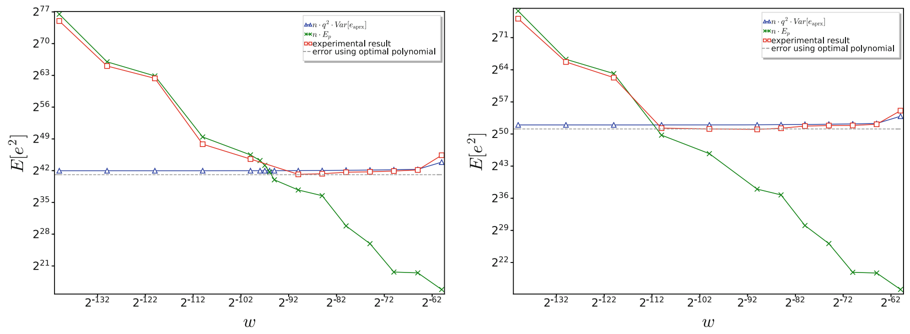
In this section, several implementation results and comparisons for the previous bootstrapping algorithms are presented. The bootstrapping using the proposed approximate polynomial is implemented on the well-known HE library `Lattigo`, as `Lattigo` is the only open-source library that supports bootstrapping of RNS-CKKS at the time of writing. We also provide a proof-of-concept implementation of bootstrapping with high precision such as 93 bits, based on the `HEAAN` library.

6.1 Error Analysis

Weight Parameter and Approximation Error. In Subsect. 5.3, we discussed analytic and numerical solutions for error variance-minimizing approximate polynomial. In this subsection, these methods are implemented and verified. We confirm that the numerical method in Subsect. 5.3 finds a polynomial that is very close but has a slightly larger error than that of the optimal one, and $w \approx \frac{(h+1)2n}{q^2 12}$, where n is the number of slots.

The experimental results are shown in Fig. 3 with parameters $N = 2^{16}, h = 64$, and the slot size $n = 2^3$. The blue lines with triangular legend show the error by polynomial approximation as $2n \cdot q^2 \cdot Var[e_{\mathsf{aprx}}]$. The green lines with x mark legend show the amplified basis errors as $2n \cdot q^2 \cdot E_p$, and the red lines with square legend are for the mean square of bootstrapping errors without scale obtained by experiments using the proposed approximate polynomial in (8). The gray dot line is the variance of bootstrapping error without scale, achieved by the analytic solution of the error variance-minimizing approximate polynomial (7) of the same degree, which is the lower bound of bootstrapping error variance. The reason for multiplying the above result by $2n$ is because of SlotToCoeff as discussed in Subsect. 4.2. For the worst-case assumption, we assume that m is distributed uniformly at random.

In Fig. 3, the sum of blue lines with triangular legend and green lines with x mark legend meets the red lines with the square legend. In other words, it shows



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

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

Fig. 3. The theoretical variance of the approximation error, amplified basis error, and experimental results implemented in HEAAN. Polynomials of degree 81 are used. (Color figure online)

that the theoretical derivation and experimental results are agreed upon. It can also be seen that it is possible to obtain an approximate polynomial with a small error with the proposed numerical method, but the error is slightly larger than that of the analytical solution. It is noted that the optimal w is close to the variance of the rescaling error $\frac{(h+1)2n}{q^2 12}$.

Polynomial Degree and Minimum Error. This subsection presents the experimental result of the approximate error variance of the proposed error variance-minimizing approximate polynomial for the given degree and constant w. In the above paragraphs, we show that when $w \approx \frac{(h+1)2n}{q^2 12}$, the variance of approximation error achieves the optimality. Unlike the previous methods that find the approximate polynomial without considering the CKKS parameters, the proposed approximation algorithm finds an approximate polynomial that is optimal for the given parameter of the CKKS scheme, such as the number of slots, key Hamming weight, and scaling factor.

In Fig. 4, we represent the variance of approximation error with $w = \frac{(h+1)2n}{q^2 12}$, where $\|m/q\|_\infty < 2^{-5}$. $w = 2^{-104}$ corresponds to $q \approx 2^{60}$ and slot size $n = 2^{14}$. $w = 2^{-200}$ corresponds to $q \approx 2^{109}$ for the same slot size. In this figure, we can see that the proposed method approaches the maximal accuracy of polynomial approximation for $q \approx 2^{60}$ within depth 10. Moreover, we can see that the proposed error variance-minimizing approximate polynomial achieves approximate error variance 2^{-209} within depth only 11.

6.2 Comparison of Bootstrapping and High-Precision Bootstrapping

Experimental Result of Bootstrapping Error. The proposed method is implemented using Lattigo, and it is compared with the most accurate bootstrapping techniques in the literature [3,24] in Table 3. In this table, the proposed

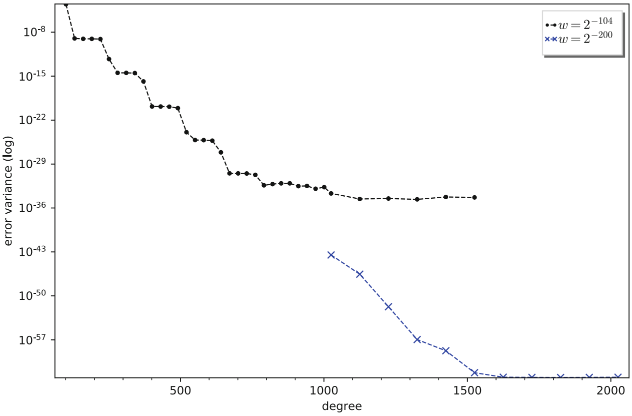


Fig. 4. Error variance of the proposed polynomial, for $w = 2^{-104}$ and 2^{-200}.

Table 3. Comparison of the variance of bootstrapping error of the proposed error variance-minimizing polynomial and prior arts. Columns "cos" and "$\sin^{-1}$" are for the degree of the approximate polynomial of each, and "double" is for the number of double angle formulas of cosine applied. The proposed method uses direct approximation, so the degree of the approximate polynomial is indicated by f_{mod}.

Algorithm	h	N	n	$\log QP_L$	λ	$\log q$	$\log p$	$\log \lVert r \rVert_\infty$	EvalMod					$Var[e_{\mathsf{boot}}]$	Bit prec.	Runtime(s)
									cos	Double	$\sin^{-1}$	Depth	#relin			
[24]	192	2^{16}	2^{14}	1553	≈ 128	60	50	-10	68	2	5	12	24	$2^{-64.5}$	32.6	451.5
[3]	192	2^{16}	2^{14}	1547	≈ 128	60	45	-5	62	2	7	11	24	$2^{-62.6}$	31.6	22.8
				1547	≈ 128	60	45	-5	62	2	3	10	22	$2^{-44.4}$	22.4	25.3
Proposed	192	2^{16}	2^{14}	1487	> 128	60	45	-5	f_{mod}: 711			**10**	33	$2^{-62.1}$	31.4	28.3
Proposed (high prec^2.)	192	2^{17}	2^{12}	–	–	115	102	-5	f_{mod}: 1625			**11**	46	$2^{-185.4}$	93.03	–
	192	2^{17}	2^{3}	–	–	115	106	-5						$2^{-199.0}$	100.11	–

error variance-minimizing polynomial directly approximates f_{mod}, and the previous methods approximate the cosine function and use the double-angle formula. For a high precision achieved in [3,24], approximate polynomials of $\frac{1}{2\pi}\arcsin(t)$ by multi-interval Remez algorithm and Taylor expansion are evaluated, respectively, and the evaluation of those algorithms consumes three more levels. For a fair comparison, we fix the message precision as $\approx$ 31-bits and compare the depth of modular reduction. The timing result is measured using Intel Xeon Silver 4210 CPU @ 2.20 GHz, single core. The scale-invariant evaluation [3] is also applied for a precise evaluation. The same parameter set as [3] is used for the proposed method, and thus the same levels are consumed for CoeffToSlot and SlotToCoeff.

In the experiment, we sample each slot value $a + bi \in \mathbb{C}$, where a and b are uniformly distributed over $[-1, 1]$, and thus, from the central limit theorem, the coefficient of the encoded plaintext follows a Gaussian distribution. On the other hand, the proposed error variance-minimizing approximate polynomial is obtained under assumption that the coefficients of plaintext are distributed uniformly at random, that is, $\Pr_R(r) = \frac{1}{2\epsilon}$ for all $r \in [-\epsilon, \epsilon]$ as a worst-case assumption discussed in Sect. 4. We note that the difference of message distribution for approximation and actual experiment is a harsh environment for the proposed error variance-minimizing approximate polynomial.

The first three rows of Table 3 show that the proposed method requires less depth compared to the prior arts. This is due to the indirect approximation using trigonometric functions of previous methods. Compared to the previous method with the same depth of 10, our method has 9-bit higher precision. In another aspect, the proposed approximate polynomial achieves the same precision as the previous methods by only the depth of 10. The proposed bootstrapping consumes one to two fewer levels in EvalMod, thus we used smaller parameters in the experiment which improves security. We can utilize the additional level depending on the application, for example, one can exploit it for efficient circuit design to reduce the total number of bootstrapping of the whole system (e.g., inference of privacy-preserving deep learning [23],) or we might speed up CoeffToSlot or SlotToCoeff using this remaining level. However, in terms of bootstrapping runtime for $\approx$ 31-bit precision, our method is slower than previous methods due to the evaluation of high-degree polynomial. Our algorithm is more advantageous for higher precision as it is efficiently scalable, which is discussed in the next subsection.

Comparison of Numerical and Analytical Error. Experiments in Fig. 3 show that the error variance-minimizing approximate polynomial has $Var[e_{\mathsf{aprx}}] + \sum w_i \boldsymbol{d}_i^2 = 2^{-103.33}$ when $w = 2^{-104}$. We can easily find the expected bootstrapping error variance with this value. The error variance is multiplied by $2n$ in SlotToCoeff; thus, the error variance after SlotToCoeff should be $2^{-88.33}$. The scaling factor in bootstrapping is $\approx q$, and thus, the error without scaling is $2^{-88.33} \cdot q^2 \approx 2^{31.67}$. The scaling factor of a message is $\approx 2^{45}$, and thus the expected bootstrapping error variance is $2^{31.67}/p^2 \approx 2^{-58.33}$. Compared with the experimental result in Table 3, $2^{-61.12}$, we can see that the numerical result roughly meets the analysis. The difference seems to be due to various methods to reduce the error introduced in Sect. 5.

Scalability and High-Precision Bootstrapping. The last two rows in Table 3 represent the proof-of-concept implementation of high-precision CKKS bootstrapping[2]. In the table, we can see that the proposed method achieves high precision such as 93-bit with 2^{12} slots. It is worth noting that our method uses the same depth as previous methods [3,24], but achieves much higher accuracy. We use the variance-minimizing approximate polynomial of degree 1625 with parameter $w = 2^{-200}$ which minimizes (8) (the minimum value is $\approx 2^{-209}$), as

[2] It is implemented using a multi-precision CKKS library HEAAN which supports rescaling by an arbitrary-length integer. As this proof-of-concept implementation is only interested in high precision, we omitted runtime and parameter QP_L (as a side note, $(h, N, \log QP_L) = (192, 2^{17}, 3069)$ achieves 128-bit security [3].) We note that the implementation is slow due to the non-RNS nature of HEAAN and has less level due to the use of $\mathsf{dnum} = 1$.

shown in Fig. 4. We can also say that bottleneck of bootstrapping error is the scaling factor, not the approximation.

The previous bootstrapping methods so far cannot achieve such high accuracy, and even if so, it will have a huge multiplicative depth to perform a high-degree polynomial approximation. Julta and Manohar proposed sine series approximation and 100-bit accuracy CKKS bootstrapping using their approximation which consumes modulus of 2^{2157} (see Table 1 in [20]). In contrast, due to the direct approximation of the proposed method, it has much less depth compared to indirect approximations, and thus we can achieve the same accuracy (shown in the last row or Table 3) with modulus about 2^{1495}, which corresponds to 6 more levels after bootstrapping. Also, the proposed lazy-BSGS algorithm reduces the number of relinearizations; the previous BSGS algorithm for odd polynomial [24] requires 66 relinearizations to evaluate polynomial of degree 1625.

This high accuracy is essential in the presence of the Li-Micciancio attack [26]. The "noise flooding" method is currently the only known way to make CKKS provably secure against Li-Micciancio attack, but it was impractical with bootstrapping as it makes CKKS noisy by losing about 30-40 bits of accuracy [26]. Although a lot of research is required on how to exploit the bootstrapping error for cryptanalysis, at least, we can directly apply the noise flooding technique [16] with the high-precision bootstrapping.

7 Conclusion

In this paper, we have two contributions for accurate and fast bootstrapping of CKKS scheme, that is, we proposed i) a method to find the optimal approximate polynomial of modular reduction for bootstrapping and its analytical solution, and ii) a more efficient algorithm to homomorphically evaluate a high-degree polynomial. The proposed error variance-minimizing approximate polynomial guarantees the minimum error after bootstrapping in the aspect of SNR; in contrast, the previous minimax approximation does not guarantee the minimum infinity norm of the bootstrapping error. Moreover, we proposed an efficient algorithm, the lazy-BSGS algorithm, to evaluate the approximate polynomial. The lazy-BSGS algorithm reduces the number of relinearizations by half compared to the ordinary BSGS algorithm, and the error is also reduced. We also proposed the algorithm to find the error variance-minimizing approximate polynomial designed for the lazy-BSGS algorithm.

The proposed algorithm reduces the level consumption of the most depth-consuming part of bootstrapping, approximate modular reduction. Thus we can reserve more levels after bootstrapping or we can use the level to speed up bootstrapping. The number of the levels after bootstrapping is significant for efficient circuit design of algorithms using CKKS [23], as well as it reduces the number of bootstrappings.

The bootstrapping performance improvement by the proposed algorithm was verified by an implementation. The implementation showed that we could reduce the multiplicative depth of modular reduction in CKKS bootstrapping while achieving the best-known accuracy. Also, we discussed that the proposed method achieves the CKKS bootstrapping with very high accuracy, so we can directly apply the noise flooding technique to the CKKS scheme for IND-CPA$^{\mathrm{D}}$ security.

Acknowledgements. We would like to thank anonymous reviewers of Crypto 2021 and Eurocrypt 2022 for their suggestions and comments which improves the paper.

References

1. Blatt, M., Gusev, A., Polyakov, Y., Rohloff, K., Vaikuntanathan, V.: Optimized homomorphic encryption solution for secure genome-wide association studies. BMC Med. Genomics **13**(7), 1–13 (2020)
2. Boemer, F., Costache, A., Cammarota, R., Wierzynski, C.: nGraph-HE2: a high-throughput framework for neural network inference on encrypted data. In: The ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography, pp. 45–56 (2019)
3. Bossuat, J.-P., Mouchet, C., Troncoso-Pastoriza, J., Hubaux, J.-P.: Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12696, pp. 587–617. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77870-5_21
4. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. ACM Trans. Comput. Theory (TOCT) **6**(3), 1–36 (2014)
5. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_29
6. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. SIAM J. Comput. **43**(2), 831–871 (2014)
7. Chen, H., Chillotti, I., Song, Y.: Improved bootstrapping for approximate homomorphic encryption. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11477, pp. 34–54. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17656-3_2
8. Chen, H., Kim, M., Razenshteyn, I., Rotaru, D., Song, Y., Wagh, S.: Maliciously secure matrix multiplication with applications to private deep learning. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12493, pp. 31–59. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64840-4_2
9. Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: Bootstrapping for approximate homomorphic encryption. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 360–384. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_14
10. Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: A full RNS variant of approximate homomorphic encryption. In: Cid, C., Jacobson Jr., M. (eds.) SAC 2018. LNSC, vol. 11349, pp. 347–368 Springer, Cham (2018). https://doi.org/10.1007/978-3-030-10970-7_16
11. Cheon, J.H., Hhan, M., Hong, S., Son, Y.: A hybrid of dual and meet-in-the-middle attack on sparse and ternary secret LWE. IEEE Access **7**(89), 497–506 (2019)

12. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 409–437. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_15
13. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 377–408. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_14
14. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: fast fully homomorphic encryption over the torus. J. Cryptol. **33**(1), 34–91 (2020)
15. Ducas, L., Micciancio, D.: FHEW: bootstrapping homomorphic encryption in less than a second. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 617–640. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_24
16. Ducas, L., Stehlé, D.: Sanitization of FHE ciphertexts. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 294–310. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_12
17. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive 2012/144 (2012)
18. Han, K., Ki, D.: Better bootstrapping for approximate homomorphic encryption. In: Cryptographers' Track at the RSA Conference, pp. 364–390 (2020)
19. Jutla, C.S., Manohar, N.: Modular Lagrange interpolation of the mod function for bootstrapping for approximate HE. Cryptology ePrint Archive 2020/1355 (2020)
20. Jutla, C.S., Manohar, N.: Sine series approximation of the mod function for bootstrapping of approximate HE. Cryptology ePrint Archive 2021/572 (2021)
21. Kim, A., Papadimitriou, A., Polyakov, Y.: Approximate homomorphic encryption with reduced approximation error. In: Galbraith, S.D. (ed.) CT-RSA 2022. LNCS, vol. 13161, pp. 120–144. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-95312-6_6
22. Kim, M., Song, Y., Li, B., Micciancio, D.: Semi-parallel logistic regression for GWAS on encrypted data. BMC Med. Genomics **13**(7), 1–13 (2020)
23. Lee, J.W., et al.: Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. IEEE Access (2021). https://ieeexplore.ieee.org/abstract/document/9734024
24. Lee, J.-W., Lee, E., Lee, Y., Kim, Y.-S., No, J.-S.: High-precision bootstrapping of RNS-CKKS homomorphic encryption using optimal minimax polynomial approximation and inverse sine function. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12696, pp. 618–647. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77870-5_22
25. Lee, Y., Lee, J.W., Kim, Y.S., No, J.S.: Near-optimal polynomial for modular reduction using L2-norm for approximate homomorphic encryption. IEEE Access **8**(144), 321–330 (2020)
26. Li, B., Micciancio, D.: On the security of homomorphic encryption on approximate numbers. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12696, pp. 648–677. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77870-5_23
27. Son, Y., Cheon, J.H.: Revisiting the hybrid attack on sparse and ternary secret LWE. Cryptology ePrint Archive 2019/1019 (2019)