META-BTS: Bootstrapping Precision Beyond the Limit

Youngjin Bae
youngjin.bae@cryptolab.co.kr
CryptoLab. Inc.

Jung Hee Cheon
jhcheon@snu.ac.kr
The Seoul National University/CryptoLab. Inc.

Wonhee Cho
wony0404@snu.ac.kr
The Seoul National University

Jaehyung Kim
jaehyungkim@cryptolab.co.kr
CryptoLab. Inc.

Taekyung Kim
taekyung.kim@cryptolab.co.kr
CryptoLab. Inc.

ABSTRACT

Bootstrapping, which enables the full homomorphic encryption scheme that can perform an infinite number of operations by restoring the modulus of the ciphertext with a small modulus, is an essential step in homomorphic encryption. However, bootstrapping is the most time and memory consuming of all homomorphic operations. As we increase the precision of bootstrapping, a large amount of computational resources is required. Specifically, for any of the previous bootstrap designs, the precision of bootstrapping is limited by rescaling precision.

In this paper, we propose a new bootstrapping algorithm of the Cheon-Kim-Kim-Song (CKKS) [11] scheme to use a known bootstrapping algorithm repeatedly, so called *Meta-BTS*. By repeating the original bootstrapping operation twice, one can obtain another bootstrapping with its precision essentially doubled; it can be generalized to be k-fold bootstrapping operations for some $k > 1$ while the ciphertext size is large enough. Our algorithm overcomes the precision limitation given by the rescale operation.

KEYWORDS

Fully Homomorphic Encryption, CKKS scheme, Approximate Bootstrapping, High Precision, Small parameters

1 INTRODUCTION

Homomorphic encryption (HE) refers to a class of encryption schemes that allows computing over encrypted data without decryption. Since Gentry et al. [17] introduced the first construction of FHE, there have been extensive studies [3–5, 8, 11–16, 18, 26] suggesting efficient designs and adding new functionalities of HE. Especially, the Cheon-Kim-Kim-Song (CKKS) [11] scheme is emerging as one of the notable HE schemes as it provides efficient privacy-preserving computation over real and complex numbers. To be specific, unlike other HE schemes that support exact computation over finite fields [3–5, 16] or of binary circuits [13–15], the CKKS scheme has complex vector space $\mathbb{C}^{N/2}$ as its default message space and naturally supports approximated SIMD(Single Instruction/Multiple Data) computation over real and complex numbers.

In order to support complicated applications, HE should be equipped with faster homomorphic operations and should be able to evaluate circuits of huge depth. However, as homomorphic operations progress, the ciphertext modulus decreases and finally becomes too small to sustain further operations in the CKKS scheme. To solve these problems, Gentry's blueprint of bootstrapping provides the idea of homomorphic re-encryption of a ciphertext. Re-encryption of a ciphertext means making the ciphertext modulus larger again while retaining the encrypted message of the ciphertext, and the resulting ciphertext is ready for further homomorphic operations.

In the CKKS scheme, the standard way for re-encryption, or bootstrapping (BTS) of ciphertexts is to evaluate an integral modular reduction homomorphically. Although the CKKS scheme provides the addition and the multiplication of ciphertexts, it is highly nontrivial to represent such a modular reduction with these basic algebraic operations. This means the bootstrapping requires a complicated combination of these basic operations, and it causes a major running time degradation and consumes a large amount of ciphertext modulus after bootstrapping. In practice, this is a severe bottleneck to use HE for various applications.

Also, in real use of HE, there are many cases require high precision bootstrapping. Evaluating a circuit of large multicative depth, including ML/DNN training, statistics, and sorting, require high precision bootstrapping since the ciphertext loses precision with each multiplication and bootstrapping procedure.

Recently kim *et al.* [21] achieves high precision with small N by using blind rotation which used for FHEW/TFHE bootstrapping. More precisely, when N is 2^{13}, their precision is 73.58 where the number of slot is 2^{10}. However, unlike other CKKS bootstrapping algorithms, it does not support SIMD(Single Instruction Multiple Data) computation during bootstrapping procedure. They split a ciphertext by the number of slots, bootstrap each piece, and combine them back into a ciphertext. Such split increases memory and time consumption in proportion to the number of slots, making it difficult for this method to deal with a large number of data.

Therefore, to preserve SIMD computation, various studies [2, 6, 7, 9, 19, 20, 22–24] have been conducted to represent modular reduction as a polynomial approximation to improve running time performance and precision of bootstrapping. Despite these efforts, improvements on the bootstrapping precision are limited with respect to the ciphertext dimension N, once we require a reasonable security level in practice, and it turns out that we cannot avoid from adopting larger parameters in order to obtain a bootstrapping with better precision.

Conference'22, July 2022, -
. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00

To address these problems, we propose a novel bootstrapping algorithm, called Meta-BTS, which can be used to obtain higher precision on given parameters.

1.1 Overview of Our Algorithm

CKKS bootstrapping can be understood as a homomorphic evaluation of the decryption circuit to re-encrypt a ciphertext. In bootstrapping, a bootstrapping error is added because the modular reduction is only approximated by evaluating a polynomial. Here we briefly present the main idea of Meta-BTS algorithm.

(1) Extract bootstrapping error by subtracting the original ciphertext from the naïvely bootstrapped ciphertext.
(2) Bootstrap the extracted error.
(3) Subtract the bootstrapped error from the bootstrapped ciphertext.

By repeating this procedure, the bootstrapping error can be partially removed from the bootstrapped ciphertext and the resulting ciphertext is in higher precision.

Techniques We describe our new algorithm to achieve a $2n$-bit precision bootstrapping $\mathsf{BTS}^{(2)}$ using an n-bit precision bootstrapping $\mathsf{BTS}^{(1)}$, as shown in Fig. 1. Before we define bootstrapping and its precision, we first define a ciphertext of HE. Let $R = \mathbb{Z}[X]/(X^N+1)$ be the cyclotomic ring with a power-of-two dimension N, and $\mathbf{m} \in \mathbb{C}^{N/2}$ be a message. We denote $R_Q = R/QR$ for a positive integer Q. When used in this context, we call Q a modulus. Let $\mathsf{ct}(\mathbf{m}, q) \in R_q^2$ denote a ciphertext whose modulus is q and the message is $\mathbf{m}$.

Now, we define bootstrapping precision as follows. By applying a bootstrapping to a ciphertext $\mathsf{ct}(2^r\mathbf{m}, q)$, we obtain the new ciphertext $\mathsf{ct}_{\mathsf{BTS}} = \mathsf{ct}(2^r\mathbf{m} + 2^{-n}\mathbf{e}, Q_{rem})$ where the infinite norm of the message $\|2^r\mathbf{m}\|_\infty$ is less than or equal to 2^r to apply the bootstrapping, Q_{rem} is the ciphertext modulus after bootstrapping, $\mathbf{e} \in \mathbb{C}^{N/2}$, and $\|\mathbf{e}\|_\infty$ is less than or equal to 1. The precision of the bootstrapping is defined to be $r + n$.

2-fold Bootstrapping, the simplest case of Meta-BTS, consists of two bootstrapping steps: bootstrapping the ciphertext and bootstrapping the error.

Bootstrapping of a ciphertext Given a ciphertext $\mathsf{ct}_I = \mathsf{ct}(2^n\mathbf{m}, 2^n \cdot q)$ where $\|\mathbf{m}\|_\infty$ is less than or equal to 1, we rescale the ciphertext by 2^n. Then, we get the new ciphertext $\mathsf{ct}_1 = \mathsf{ct}(\mathbf{m}, q)$. Now, we apply an n-bit precision $\mathsf{BTS}^{(1)}$ to ct_1 and obtain new ciphertext $\mathsf{ct}_2 = \mathsf{ct}(\mathbf{m} + 2^{-n}\mathbf{e}_1, Q_{rem})$ where $\|\mathbf{e}_1\|_\infty$ is less than or equal to 1. To extract the bootstrapping error, we multiply ct_2 by 2^n, and get new ciphertext $\mathsf{ct}_3 = \mathsf{ct}(2^n\mathbf{m} + \mathbf{e}_1, Q_{rem})$. Since Q_{rem} is multiple of q in the CKKS scheme, we can compute the ciphertext of the bootstrapping error, $\mathsf{ct}_{error} = [\mathsf{ct}_3]_q - [\mathsf{ct}_I]_q = \mathsf{ct}(\mathbf{e}_1, q)$. Now, we get the ciphertext of the bootstrapping error $\mathbf{e}_1$.

Bootstrapping the error Now, we apply an n-bit precision $\mathsf{BTS}^{(1)}$ to ct_{error} and obtain new ciphertext $\mathsf{ct}_4 = \mathsf{ct}(\mathbf{e}_1 + 2^{-n}\mathbf{e}_2, Q_{rem})$ where $\|\mathbf{e}_2\|_\infty$ is less than or equal to 1. Finally, we compute $\mathsf{ct}_o = ct_3 - \mathsf{ct}_4 = \mathsf{ct}(2^n\mathbf{m} - 2^{-n}\mathbf{e}_2, Q_{rem})$. Since $\|\mathbf{e}_2\|_\infty$ is less than or equal to 1, the ciphertext ct_o still has $2^n\mathbf{m}$ as a message with n-bit precision and Q_{rem} is the ciphertext modulus after bootstrapping. Therefore, we get a $2n$-bit precision $\mathsf{BTS}^{(2)}$ of using an n-bit precision $\mathsf{BTS}^{(1)}$.

In addition, Meta-BTS can easily be extended to iterate the basic BTS more than twice. The details on how to repeat multiple times of our algorithm are introduced in Section 3.

1.2 Our Contributions

We propose a Meta-BTS algorithm which enables us to obtain high precision bootstrapping by iterating lower precision bootstrapping. That is, given a FHE parameter, it is possible to construct a bootstrapping algorithm of which precision is even beyond the state-of-the-art limitation. More specifically, given an n-bit precision $\mathsf{BTS}^{(1)}$ with an input range of $[-1, 1]$, a kn-bit precision $\mathsf{BTS}^{(k)}$ can be obtained by repeating $\mathsf{BTS}^{(1)}$ k times for possible k. Until now, the parameters of HE are optimized for a specific precision, and if a higher precision operation is desired, a user needs to make new parameters for a higher precision. But it is hard to do unless a user is an expert of HE. In contrast, Meta-BTS enables higher precision operations without changing parameters.

We propose new HE parameters with a much higher precision than existing one according to the ciphertext polynomial dimension N using Meta-BTS method. Since the previous methods use a large number of modulus to increase precision of bootstrapping, precision is very limited by the ciphertext polynomial dimension N. The best known precision to be implemented was 15 [2], 45 [6], 100 [23] when N is of $2^{15}, 2^{16}, 2^{17}$, respectively.

Our Meta-BTS can increase the bootstrapping precision by using a fixed bootstrapping algorithm several times, so we only need to increase the precision in the multiplication process except the bootstrapping algorithm. Using Meta-BTS, we can obtain a higher precision compared to the previous methods for each corresponding N, as shown in Table 1. The upper bound of precision and the parameter sets for obtaining the high precision are described in Section 4.2, 5, respectively.

Table 1: Comparison of the bootstrapping precision of previous works and ours. *depth* denotes the number of possible multiplications after the bootstrapping algorithm.

Algorithm	N	*slot*	*iter*	*depth*	Bit prec.
BMT+21[2]	2^{15}	2^{14}	-	3	15.5
JM22 [6]	2^{16}	2^3	-	-	45
	2^{17}	2^3	-	-	100
LLK+22 [23]	2^{17}	2^3	-	-	100.11
	2^{17}	2^{12}	-	-	93.03
This work	2^{15}	2^{14}	3	1	48
	2^{16}	2^{15}	17	1	255
	2^{17}	2^{16}	14	1	420

Our Meta-BTS algorithm also improves asymptotic time complexity in the high precision bootstrapping algorithm. Our algorithm uses n/k-bit precision bootstrapping k times to obtain n-bit precision bootstrapping. The advantage of small precision bootstrapping is that small precision bootstrapping uses small ciphertext modulus and performs fewer operations. If the target precision n is large enough, the ciphertext modulus for an n-bit precision bootstrapping is approximated by $O(n^{5/4})$.

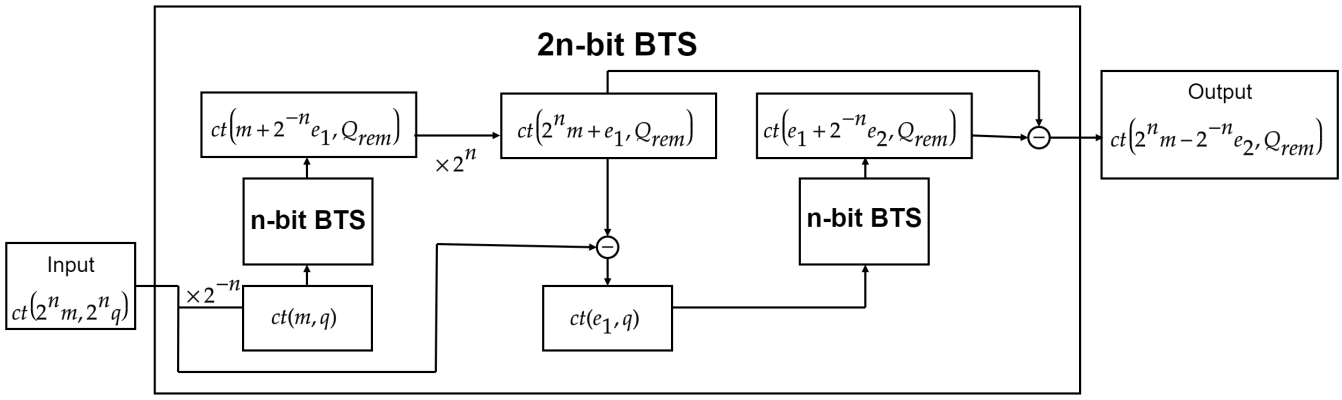

Figure 1: High-level Overview of the Meta-BTS algorithm

Let C_n be the the number of multiplications used by n-bit precision BTS, which is an incremental function for n. Since the time complexity of m-bit number multiplication is at least $O(m \log m)$, we assume of it as $O(m \log m)$, where our algorithm is least efficient. Using Meta-BTS, the time complexity of an n-bit precision bootstrapping decrease from $C_n \cdot O(n^{5/4} \log n)$ to $C_{n/k} \cdot O\left(\frac{n^{5/4}}{k^{1/4}} \cdot \log\left(\frac{n}{k}\right)\right)$ for a positive integer k.

Our algorithm provides higher precision bootstrapping under the same parameters, so smaller parameters are available in certain situations. If one wants to obtain 100-bit precision FHE, one uses the ciphertext polynomial dimension $N = 2^{17}$, but our new method achieves the same precision with $N = 2^{16}$. According to Lattigo preset I [2] which uses $N = 2^{16}$, total memory for storing bootstrapping keys is at least 10.25GB. If N grows to 2^{17}, total memory is more than four times larger. So Meta-BTS can be useful in environments such as hardware or IoT with limited memory.

Finally, our new algorithm uses bootstrapping algorithm as a black box, so it can be used for all bootstrapping algorithms of the CKKS scheme. We adapt the existing bootstrapping algorithms for CKKS schemes [7, 9, 19] to build a high precision bootstrapping algorithm. If one applies the state-of-the-art bootstrapping algorithm [2, 6, 23], one may get a better bootstrapping algorithm in terms of precision using Meta-BTS.

1.3 Related Works

Bootstrapping of the CKKS Scheme.

The core of the bootstrapping algorithm is the polynomial approximation of a modular reduction. The first proposed method is to approximate the modular reduction with a trigonometric function and approximate it with a polynomial function using Taylor series [9]. After that, the studies [7, 19] were conducted to change Taylor series to Chebyshev interpolation. Then, a technique for direct approximation of the modular reduction was proposed using the least squares method [24] and Lagrange interpolation [20]. Also, to reduce the error caused by trigonometric approximation, the use of the inverse sine approximation was presented in [22] and the sine series was presented in [6]. Separately, to reduce the computation time for homomorphic linear transformation, the use of double hoisting technique was presented in [2]. Most recently, a method of minimizing the error variance in homomorphic computation and bootstrapping algorithm was proposed in [23].

Difference of Our works.

Our method mitigates the overall bootstrapping precision which is the combination of error coming from the approximate modular reduction and error from the CKKS scheme itself (e.g. rescale error). Many papers in the past have focused on improving modular reduction errors, and the recent work [23] proposes a first technique that considered the noise added during the bootstrapping procedure by the different homomorphic operations. As our paper takes a completely different point of view from otherworks, we use a bootstrapping algorithm in a black-box manner. We defined boostrapping precision in Definition 2.2 as the sup-norm of the difference between the two decrypted messages of the "original ciphertext" and the "bootstrapped ciphertext" respectively. We propose a method to obtain higher precision bootstrapping by repeating a given bootstrapping.

2 BACKGROUND

2.1 Notation

We denote vectors in lower-case bold face, e.g. $\mathbf{a}$, and matrices in upper-case bold face, e.g. $\mathbf{A}$. We denote the inner product of two vector by $\langle \cdot, \cdot \rangle$ or simply $\cdot$. For a real number r, $\lceil r \rfloor$ denotes the rounding function. We denote by $[\cdot]_q$ the modular reduction by q. We denote the infinite norm by $\| \cdot \|_\infty$ and Hadamard multiplication by $\odot$. We use $x \leftarrow D$ to denote the sampling x according to distribution D. When a set S is used instead of a distribution, $x \leftarrow S$ means that x is sampled uniformly at random among elements of S. We set λ to be a fixed security parameter throughout the paper: all known valid attacks against the cryptographic scheme under scope should take $\Omega(2^\lambda)$ bit operations.

Let $R = \mathbb{Z}[X]/(X^N + 1)$ be the ring of integers of the $2N$-th cyclotomic field with a power-of-two dimension N and we write $R_q = R/qR$. A polynomial $a(X)$ can be denoted by a by omitting X.

2.2 The CKKS scheme

We first recap the FHE scheme CKKS [11] and its packing methods.

Packing method The CKKS scheme uses a complex vector as a message (i.e. $\mathbf{m} \in \mathbb{C}^{N/2}$) and provides homomorphic SIMD operations of ciphertexts (i.e. entry-wise operations) such as addition, subtraction, and Hadamard multiplication. To encrypt a vector of complex numbers $\mathbf{m} \in \mathbb{C}^{N/2}$, we use an isomorphism $\tau : \mathbb{R}[X]/(X^N + 1) \to \mathbb{C}^{N/2}$ called the canonical embedding and a positive real number Δ called scaling factor. A method of encoding/decoding a message is as follows:

- **Encoding:** $m(X) \leftarrow \mathsf{Ecd}(\mathbf{m}, \Delta)$. Given a message $\mathbf{m} \in \mathbb{C}^{N/2}$ and a scaling factor Δ, the encoding map returns $m(X) = \lfloor \Delta \cdot \tau^{-1}(\mathbf{m}) \rceil \in R$. When encoding, $\|\mathbf{m}\| \leq 1$.
- **Decoding:** $\mathbf{m} \leftarrow \mathsf{Dcd}(m(X), \Delta)$. Given a plaintext $m(X) \in R$ and a scaling factor Δ, the decoding map returns $\mathbf{m} = \tau(m'(X)) \in \mathbb{C}^{N/2}$, where $m'(X) = \Delta^{-1} \cdot m(X)$ is considered as an element of $\mathbb{Q}[X]/(X^N + 1)$.

The CKKS scheme Consider the CKKS scheme $\mathsf{Enc} : \mathcal{M} \times \mathcal{K} \to \mathcal{C}$ for the plaintext space $\mathcal{M} = R$, the key space $\mathcal{K}$ and the ciphertext space $\mathcal{C} = R_Q^2$. The ciphertext modulus Q is one of the positive integers $Q_0 < Q_1 < \cdots < Q_L$ depending on the level of the ciphertext, which decreases over the homomorphic computations from L to 0. When the level becomes 0, homomorphic multiplication can no longer be performed on the ciphertext. To enable extra homomorphic multiplications, a bootstrapping algorithm is performed to raise the modulus of the ciphertext. In the bootstrapping algorithm, the level is recovered only up to some $L_{\mathsf{BTS}} < L$ due to the levels consumed during the homomorphic operations inside of the bootstrapping algorithm. The distributions χ_{enc} and χ_{err} denote the discrete Gaussian distribution with some fixed standard deviation. The distribution χ_{key} outputs a polynomial with coefficients in $\{-1, 0, 1\}$.

Basic Operations of the CKKS scheme

- **Setup:** $\mathsf{params} \leftarrow \mathsf{FHE.Setup}(1^\lambda)$. Take the security parameter as an input and return the public parameters such as the degree N and the chain of modulus $Q_0 < \cdots < Q_L$.
- **Key Generation:** $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{params})$. Output a pair of secret and public key.
  - Sample $s \leftarrow \chi_{key}$ and set the secret key as $\mathsf{sk} = (1, s)$. We denote by $h = \mathrm{wt}(s)$ the number of nonzero numbers of s.
  - Sample $a \leftarrow R_{Q_L}$ and $e \leftarrow \chi_{err}$. Set the public key as $\mathsf{pk} = (b, a) \in R_{Q_L}^2$ where $b = [-a \cdot s + e]_{Q_L}$.
- **Switching Key Generation:** $\mathsf{swk} \leftarrow \mathsf{KSGen}_{\mathsf{sk}}(s')$. For auxiliary modulus $P = \prod_{i=0}^{k} p_i$, sample $a'_k \leftarrow R_{PQ_L}$ and $e'_k \leftarrow \chi_{err}$, output the switching key $\mathsf{swk} := (\mathsf{swk}_0, \mathsf{swk}_1) = (b'_k, a'_k) \in R_{PQ_L}^2$ where $b'_k = -a'_k s + e'_k + P \cdot s' \bmod PQ_L$.
  - Set the evaluation key as $\mathsf{evk} := \mathsf{KSGen}_{\mathsf{sk}}(s^2)$.
  - Set the rotation key as $\mathsf{rk}_j := \mathsf{KSGen}_{\mathsf{sk}}(s(X^{5^j}))$ for $1 \leq j \leq N/2$.
  - Set the bootstrapping key as $\mathsf{btk} := \{rk_j\}_{j \in J}$ where J is a subset of $[1, 2^{N/2}]$.
- **Encryption:** $\mathsf{ct} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m(X))$. Given a plaintext $m(X) \in R$, sample $v \leftarrow \chi_{enc}$ and $e_0, e_1 \leftarrow \chi_{err}$, output the ciphertext $\mathsf{ct} = [v \cdot \mathsf{pk} + (m(X) + e_0, e_1)]_{Q_L}$. We call the message $\mathbf{m}$ for $\mathsf{Dcd}(m(X), \Delta)$.
- **Decryption:** $m(X) \leftarrow \mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct})$. Given a ciphertext $\mathsf{ct} \in R_{Q_\ell}^2$, output $m(X) = [\langle \mathsf{ct}, \mathsf{sk} \rangle]_{Q_0}$ where Q_0 is 0-level modulus.
- **Rescale:** $\mathsf{ct}_{\mathsf{RS}} \leftarrow \mathsf{RS}(Q, \mathsf{ct})$. For a given ciphertext $\mathsf{ct} \in R_{Q_\ell}^2$, output $\mathsf{ct}_{\mathsf{RS}} = \lfloor Q^{-1}\mathsf{ct} \rceil \bmod Q_\ell/Q$ for some integer Q which is a factor of Q_ℓ. Rescale process plays two roles: reducing the size of the error and maintaining the scaling factor of each slot.
- **Addition/Subtraction:** $\mathsf{ct}_{add}/\mathsf{ct}_{sub} \leftarrow \mathsf{Add}/\mathsf{Sub}(\mathsf{ct}, \mathsf{ct}')$. Given two ciphertext $\mathsf{ct}, \mathsf{ct}' \in R_{Q_\ell}^2$, output the ciphertext $\mathsf{ct}_{add}/\mathsf{ct}_{sub} = [\mathsf{ct} \pm \mathsf{ct}']_{Q_\ell}$ with the corresponding message vector $\mathbf{m} \pm \mathbf{m}'$.
- **Multiplication:** $\mathsf{ct}_{mult} \leftarrow \mathsf{Mult}_{\mathsf{evk}}(\mathsf{ct}, \mathsf{ct}')$. Given two ciphertexts $\mathsf{ct}, \mathsf{ct}' \in R_{Q_\ell}^2$, output a level-downed ciphertext $\mathsf{ct}_{mult} \in R_{Q_{\ell-1}}^2$ with the corresponding message vector $\mathbf{m} \odot \mathbf{m}'$.
- **Homomorphic evaluation:**

  $$\hat{\mathsf{ct}} \leftarrow \mathsf{FHE.Eval}(C, (\mathsf{ct}_1, \cdots, \mathsf{ct}_l), \{\mathsf{swk}_{s'}\}).$$

  Given a circuit C, a tuple of ciphertexts $(\mathsf{ct}_1, \cdots, \mathsf{ct}_l)$ and the switching keys $\{\mathsf{swk}_{s'}\}$ for computations, outputs a evaluated ciphertext $\hat{\mathsf{ct}}$.

For the rest of the paper, we may denote the operations between ciphertexts or ciphertext and plain vector by common symbols, such as $\mathsf{Add}(\mathsf{ct}_1, \mathsf{ct}_2) = \mathsf{ct}_1 + \mathsf{ct}_2$ or $\mathsf{Mult}(\mathsf{ct}, \mathsf{ct}') = \mathsf{ct} \cdot \mathsf{ct}'$ for simplicity. Also, we denote $\mathsf{ct}(\mathbf{m}, q)$ by a ciphertext with the message $\mathbf{m} \in \mathbb{C}^{N/2}$ and the modulus q.

2.3 Bootstrapping of the CKKS scheme

There have been extensive studies for the bootstrapping of the CKKS scheme. Although they vary in details, all of CKKS bootstrapping consists of four steps: StC, ModRaise, CtS, EvalMod. StC and CtS are specific linear transformations, which require rotation operations on encrypted vectors.

Slot to Coefficient (StC). In CKKS scheme, the canonical embedding is used to encode a complex vector to a polynomial which is an element of R. Homomorphic operations are performed slot-wise on $\mathbb{C}^{N/2}$, but changing the modulus is only possible for coefficients of ciphertext polynomials. Therefore, for modulus raising, the homomorphic linear transformation is performed to move the value in the slot to the coefficient.

Modulus raising (ModRaise). ModRaise increase the ciphertext modulus to a larger modulus to recover the level of the ciphertext. Let $\mathsf{ct}(\mathbf{m}, Q_\ell) = (b = -as + m(X) + e, a) \bmod Q_\ell$ be a ciphertext where $m(X) = \mathsf{Ecd}(\mathbf{m}, \Delta)$. By only changing its modulus, the ciphertext $\mathsf{ct}(\mathbf{m}, Q_\ell)$ can be considered as $\mathsf{ct}(\mathbf{m}', Q_L)$ for $Q_\ell < Q_L$ where $\mathbf{m}' = \mathsf{Dcd}(m(X) + Q_\ell I(X), \Delta)$ for some $I(X) \in \mathbb{Z}[X]/(X^N + 1)$.

Coefficient to Slot (CtS). To compute modular reduction, a homomorphic linear transformation is performed to move the value in the coefficient back to the slot. Then each slot of the ciphertext has $m_i + Q_\ell I_i$ as its encrypted message where $m_i + Q_\ell I_i$ is the i-th coefficient of $m(X) + Q_\ell I(X)$.

Approximate Evaluation of the Modular Reduction (EvalMod). An approximate evaluation of the modular reduction is performed in this step. The modular reduction cannot be accurately expressed

only by addition and multiplication, but it can be approximated by evaluating a polynomial. Thus, the modular reduction is approximated through a polynomial homomorphic operation, where a bootstrapping error occurs due to approximation, which is greater than the rescale error that occurs during homomorphic operations. Various studies are being conducted to increase the precision of this polynomial approximation [6, 7, 9, 19, 22, 23].

Definition 2.1. (BTS) *A bootstrapping algorithm is an algorithm that allows to re-encrypt a ciphertext and makes homomorphic operations sustainable in homomorphic encryption. A bootstrapping algorithm* BTS *is a PPT algorithm with the following properties:*

- $\mathsf{BTS}(\mathsf{ct}(\mathbf{m}, q), \mathsf{btk}) \rightarrow \mathsf{ct}_{\mathsf{BTS}} = \mathsf{ct}(\mathbf{m}', Q_{rem})$*: Given input keys for bootstrapping* btk *and a ciphertext* $\mathsf{ct}(\mathbf{m}, q)$ *whose message* $\mathbf{m}$ *satisfies* $\|\mathbf{m}\|_\infty \leq 2^r$ *for some* r*, the bootstrapping algorithm returns a ciphertext* $\mathsf{ct}_{\mathsf{BTS}}(\mathbf{m}', Q_{rem})$ *where* $Q_{rem} > q$ *is a ciphertext modulus after the bootstrapping algorithm.*

The upper bound of $\|\mathbf{m}\|_\infty$, 2^r*, is called the* ***input bound*** *of the* BTS*. When the input bound is* 1, BTS *is said to be* ***standard****.*

Definition 2.2. ($\mathsf{PERF}_{\mathsf{BTS}}$) *Given a bootstrapping algorithm* BTS*, Performance* $\mathsf{PERF}_{\mathsf{BTS}}$ *is the measure*

$$\mathsf{PERF}_{\mathsf{BTS}} = (\textit{Precision}, \textit{RemainModulus}, \textit{Time})$$

of BTS *with the following properties:*

- $\mathsf{PERF}_{\mathsf{BTS}}$*.Precision* $\rightarrow r + n$*: Returns the average number of bits preserved by* BTS*. n is the largest positive integer such that* $\|\mathbf{m} - \mathbf{m}'\|_\infty \leq 2^{-n}$ *holds without negligible probability where* $\mathbf{m}'$ *is the decrypted message of the bootstrapped ciphertext* $\mathsf{ct}_{\mathsf{BTS}} = \mathsf{BTS}(\mathsf{ct}(\mathbf{m}, q))$*.* 2^r *is the input bound of the* BTS*. That is,* $\mathbf{m}$ *is assumed to satisfy* $\|\mathbf{m}\|_\infty \leq 2^r$*.*
- $\mathsf{PERF}_{\mathsf{BTS}}$*.RemainModulus* $\rightarrow Q_{rem}/q$*: Returns the modulus after bootstrapping except the minimal modulus* q *for* BTS*. This means that the size of modulus that we can perform homomorphic operations excluding bootstrapping.*
- $\mathsf{PERF}_{\mathsf{BTS}}$*.Time* $\rightarrow t$*: Returns the running time of* BTS*.*

As bootstrapping consists of homomorphic operations, an error is added to the message of bootstrapped ciphertext. By Definition 2.1, 2.2, after bootstrapping, bootstrapped ciphertext contains a bootstrapped message $\mathbf{m}' = \mathbf{m} + 2^{-n}\mathbf{e}$ where $\mathbf{m}$ is the message of input ciphertext, $\|\mathbf{m}\|_\infty \leq 2^r$, $\mathbf{e} \in \mathbb{C}^{N/2}$, and $\|\mathbf{e}\|_\infty \leq 1$.

3 THE META-BTS ALGORITHM

In this section, we present the Meta-BTS, a new method of combining bootstraps (in other words, iterating bootstraps) to get a higher precision bootstrap. We describe the algorithms of 2-fold bootstrapping and combining different bootstrapping. Since Meta-BTS use bootstrapping algorithm as a black box, it can be used even if a new bootstrapping algorithm is developed in the future.

3.1 Difficulties in improving the performance of bootstrapping

In the CKKS bootstrapping, the modular reduction over encrypted data is required without decryption. Since the CKKS scheme provides only the multiplication and addition of ciphertexts, the modular reduction for the bootstrapping algorithms should be approximated by polynomials to compute over encrypted data. In this process, the bootstrapping error is caused by the homomorpic operations of the polynomial approximation as well as several rescalings with errors. In order to obtain high precision, one reduces the bootstrapping error by increasing a scaling factor Δ and/or increasing the degree of polynomial approximation. Since the high precision bootstrapping algorithm consumes a lot of bits of the ciphertext modulus, it reduces the ciphertext modulus significantly after the bootstrapping and results in the large HE parameters.

3.2 Algorithm description

In this subsection, we propose a new algorithm of increasing the bootstrapping precision by bootstrapping the errors caused by the bootstrapping algorithm. Furthermore, we propose a method of repeating the bootstrapping more than twice by generalizing our method. In case of using the RNS variant of CKKS[CITE], every rescaling by 2^n below means that we rescale the ciphertext by a modulus which is close to 2^n.

3.2.1 2-fold Bootstrapping. We propose an algorithm for constructing a $2n$-bit precision bootstrapping $\mathsf{BTS}^{(2)}$ using an n-bit precision bootstrapping $\mathsf{BTS}^{(1)}$.

We start with a standard bootstrapping algorithm $\mathsf{BTS}^{(1)}$ with $\mathsf{PERF}_{\mathsf{BTS}^{(1)}} = (n, Q_{rem}, t)$. In other words, $\mathsf{BTS}^{(1)}$ converts a ciphertext $\mathsf{ct}(\mathbf{m}, q)$ into $\mathsf{ct}(\mathbf{m} + 2^{-n}\mathbf{e}, Q_{rem})$ where $\|\mathbf{m}\|_\infty \leq 1$ and $\|\mathbf{e}\|_\infty \leq 1$.

Algorithm 1: Construct $\mathsf{BTS}^{(2)}$ using $\mathsf{BTS}^{(1)}$

Input : $\mathsf{ct}_I = \mathsf{ct}(2^n\mathbf{m}, 2^n \cdot q)$
Output: $\mathsf{ct}_O = \mathsf{ct}(2^n\mathbf{m} - 2^{-n}\mathbf{e}_2, Q_{rem})$

1 $\mathsf{ct}_1 \leftarrow RS(2^n, \mathsf{ct}_I)$ ▷ $\mathsf{ct}_1 = \mathsf{ct}(\mathbf{m} + \mathbf{e}_{rs}, q)$
2 $\mathsf{ct}_2 \leftarrow \mathsf{BTS}^{(1)}(\mathsf{ct}_1)$ ▷ $\mathsf{ct}_2 = \mathsf{ct}(\mathbf{m} + \mathbf{e}_{rs} + 2^{-n}\mathbf{e}_1, Q_{rem})$
3 $\mathsf{ct}_3 \leftarrow 2^n \cdot \mathsf{ct}_2$ ▷ $\mathsf{ct}_3 = \mathsf{ct}(2^n\mathbf{m} + 2^n\mathbf{e}_{rs} + \mathbf{e}_1, Q_{rem})$
4 $\mathsf{ct}_4 \leftarrow [\mathsf{ct}_3]_{2^n \cdot q}$ ▷ $\mathsf{ct}_4 = \mathsf{ct}(2^n\mathbf{m} + 2^n\mathbf{e}_{rs} + \mathbf{e}_1, 2^n \cdot q)$
5 $\mathsf{ct}_5 \leftarrow \mathsf{ct}_4 - \mathsf{ct}_I$ ▷ $\mathsf{ct}_5 = \mathsf{ct}(2^n\mathbf{e}_{rs} + \mathbf{e}_1, 2^n \cdot q)$
6 $\mathsf{ct}_6 \leftarrow \mathsf{BTS}^{(1)}(\mathsf{ct}_5)$ ▷ $\mathsf{ct}_6 = \mathsf{ct}(2^n\mathbf{e}_{rs} + \mathbf{e}_1 + 2^{-n}\mathbf{e}_2, Q_{rem})$
7 $\mathsf{ct}_O \leftarrow \mathsf{ct}_3 - \mathsf{ct}_6$
8 **return** $\mathsf{ct}_O = \mathsf{ct}(2^n\mathbf{m} - 2^{-n}\mathbf{e}_2, Q_{rem})$

The $\mathsf{BTS}^{(2)}$ is as follows. Given an input ciphertext $\mathsf{ct}_I = \mathsf{ct}(2^n\mathbf{m}, 2^n \cdot q)$ where $\|\mathbf{m}\|_\infty \leq 1$, rescale ct_I to 2^n to reduce the size of message for applying $\mathsf{BTS}^{(1)}$. After rescale process, the ciphertext is $\mathsf{ct}_1 = \mathsf{ct}(\mathbf{m} + \mathbf{e}_{rs}, q)$. Here, we assume that the rescale error $\mathbf{e}_{rs}$ is smaller than the error of the basic bootstrapping $\mathsf{BTS}^{(1)}$, so it satisfies $\|\mathbf{e}_{rs}\|_\infty \leq 2^{-n}$. By applying $\mathsf{BTS}^{(1)}$ to ct_1, we get $\mathsf{ct}_2 = \mathsf{ct}(\mathbf{m} + \mathbf{e}_{rs} + 2^{-n}\mathbf{e}_1, Q_{rem})$ where $\|\mathbf{e}_1\|_\infty \leq 1$. To extract the error, multiplying ct_2 by 2^n, we get $\mathsf{ct}_3 = 2^n \cdot \mathsf{ct}_2 = \mathsf{ct}(2^n\mathbf{m} + 2^n\mathbf{e}_{rs} + \mathbf{e}_1, Q_{rem})$.

To calculate the two ciphertexts $\mathsf{ct}_I, \mathsf{ct}_3$, we make modulus of ct_3 down. So, we obtain the ciphertext $\mathsf{ct}_4 = [\mathsf{ct}_3]_{2^n \cdot q} = \mathsf{ct}(2^n\mathbf{m} + 2^n\mathbf{e}_{rs} + \mathbf{e}_1, 2^n \cdot q)$. And then, computing $\mathsf{ct}_4 - \mathsf{ct}_I$, we get $\mathsf{ct}_5 = \mathsf{ct}_4 - \mathsf{ct}_I = \mathsf{ct}(2^n\mathbf{e}_{rs} + \mathbf{e}_1, 2^n \cdot q)$.

Now, we get the ciphertext ct_5 whose message is the rescale and bootstrapping error $2^n\mathbf{e}_{rs} + e_1$. Since $\|2^n\mathbf{e}_{rs} + e_1\|_\infty \leq 1$, we can apply $\mathsf{BTS}^{(1)}$ to ct_5. By applying $\mathsf{BTS}^{(1)}$ to ct_5, we get $\mathsf{ct}_6 = \mathsf{ct}(2^n\mathbf{e}_{rs} + \mathbf{e}_1 + 2^{-n}\mathbf{e}_2, Q_{rem})$ where $\|\mathbf{e}_2\|_\infty \leq 1$.. Finally, computing $\mathsf{ct}_3 - \mathsf{ct}_6$, we get $\mathsf{ct}_O = \mathsf{ct}(2^n\mathbf{m} - 2^{-n}\mathbf{e}_2, Q_{rem})$.

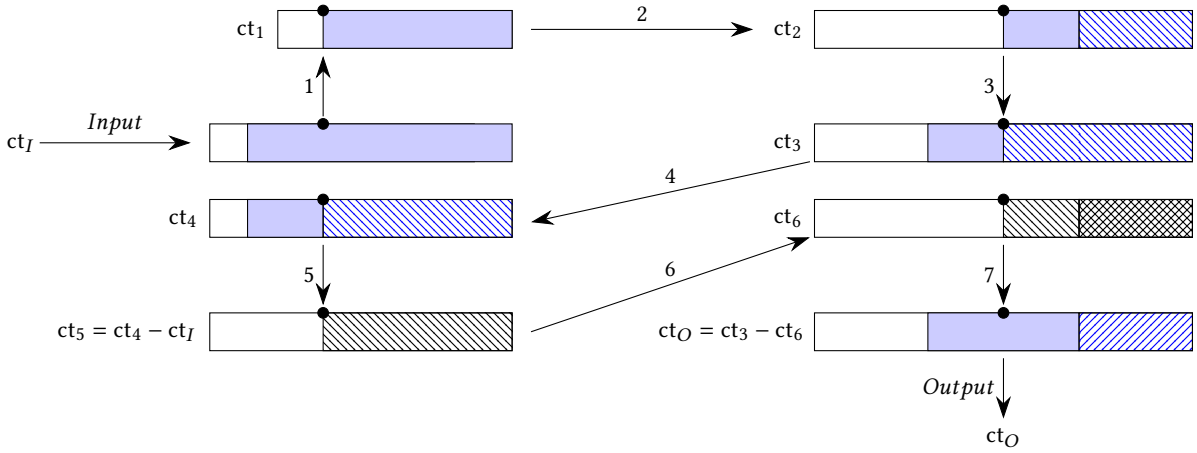

Figure 2: Changes in Decrypted Messages along the Algorithm 1. Each rectangle represents the modulus of the ciphertext. The wider the rectangle, the greater the modulus. The blue colored region is the bits occupied by the decrypted message, the white colored part is non-occupied(i.e. zero) region. Since capacity in the message space $\mathbb{C}^{N/2}$ is not uniform as in the plaintext space R_Q , the horizontal length of a rectangle cannot be expressed with an exact number of bits. However, it is clear that the overall capacity increases through the bootstrapping process, so it is reasonable to draw a wider rectangle for a ciphertext on larger modulus space. The black dot indicates the decimal point. Since the infinity norm of the message to be bootstrapped must be no greater than 1, the blue colored part must be located to the right of the decimal point. Parts occupied by the error are indicated by hatching.

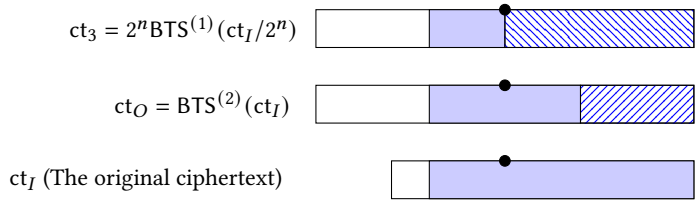

Figure 3: Comparison of $\mathrm{ct}_3 = 2^n \mathsf{BTS}^{(1)}(\mathrm{ct}_I/2^n)$ and $\mathrm{ct}_O = \mathsf{BTS}^{(2)}(\mathrm{ct}_I)$. $\mathsf{BTS}^{(2)}$ is twice more precise than $\mathsf{BTS}^{(1)}$

Theorem 3.1 (2-fold Bootstrapping Algorithm). *Given a standard bootstrapping algorithm* $\mathsf{BTS}^{(1)}$ *with performance*

$$\mathsf{PERF}_{\mathsf{BTS}^{(1)}} = (n, Q_{rem}/q, t),$$

there is a bootstrapping algorithm $\mathsf{BTS}^{(2)}$ *whose performance is*

$$\mathsf{PERF}_{\mathsf{BTS}^{(2)}} = (2n, Q_{rem}/(2^n \cdot q), 2t),$$

if

$$Q_{rem}/(2^n \cdot q) \geq 1.$$

Proof. Given a ciphertext $\mathrm{ct}(2^n\mathbf{m}, 2^n \cdot q)$ where $\mathbf{m} \in \mathbb{C}^{N/2}$ and $\|\mathbf{m}\|_\infty \leq 1$. According to Algorithm 1, the output ciphertext of $\mathsf{BTS}^{(2)}$ is $\mathrm{ct}(2^n\mathbf{m} - 2^{-n}\mathbf{e}_2, Q_{rem})$. Since $2^{-n}\mathbf{e}_2$ is the error of BTS_1, $\mathbf{e}_2 \in \mathbb{C}^{N/2}$ and $\|\mathbf{e}_2\|_\infty \leq 1$. Therefore, $\mathsf{BTS}^{(2)}$ preserves $2n$-bit precision of the message $\mathbf{m}$. Also, the minimal modulus for BTS_2 grows from q to $2^n \cdot q$ and the running time of BTS_2 is almost $2t$ because the running time of BTS is overwhelmingly large compared to other times. So, the performance of $\mathsf{BTS}^{(2)}$ is $(2n, Q_{rem}/(2^n \cdot q), 2t)$. □

3.2.2 k-fold bootstrapping. We propose an algorithm for constructing a kn-bit precision $\mathsf{BTS}^{(k)}$ using an n-bit precision bootstrapping $\mathsf{BTS}^{(1)}$ for a positive integer k.

Let $\mathsf{BTS}^{(1)}$ be a standard bootstrapping algorithm whose performance is $(n, Q_{rem}/q, t)$. Assume that $\mathsf{BTS}^{(k)}$ holds the following conditions for a positive integer k.

- Input: $\mathrm{ct}(2^{(k-1)n} \cdot \mathbf{m}, 2^{(k-1)n} \cdot q)$.
- Output: $\mathrm{ct}(2^{(k-1)n} \cdot \mathbf{m} + 2^{-n}\mathbf{e}_k, Q_{rem})$ where $\|\mathbf{m}\|_\infty, \|\mathbf{e}_k\|_\infty \leq 1$.
- $\mathsf{PERF}_{\mathsf{BTS}^{(k)}} = (kn, Q_{rem}/(2^{(k-1)n} \cdot q), kt)$.

To prove above argument, we use mathematical induction for a positive integer k. When k is 1, 2, we already propose the Algorithm 1 and $\mathsf{BTS}^{(1)}$, $\mathsf{BTS}^{(2)}$ satisfy all conditions by Definition 2.2 and Theorem 3.1. If we can construct $\mathsf{BTS}^{(k+1)}$ using $\mathsf{BTS}^{(k)}$ and $\mathsf{BTS}^{(1)}$ for a positive integer k and $\mathsf{BTS}^{(k+1)}$ satisfies the all conditions, we can construct $\mathsf{BTS}^{(k)}$ which satisfies the all conditions using only $\mathsf{BTS}^{(1)}$ for a positive integer k.

The $\mathsf{BTS}^{(k+1)}$ is as follows. Given an input ciphertext $\mathrm{ct}_I = \mathrm{ct}(2^{kn}\mathbf{m}, 2^{kn} \cdot q)$ where $\|\mathbf{m}\|_\infty \leq 1$, rescale ct_I to 2^n to reduce the size of message for applying $\mathsf{BTS}^{(k)}$. After rescale process, the

Algorithm 2: Construct $\mathsf{BTS}^{(k+1)}$ using $\mathsf{BTS}^{(k)}$ and $\mathsf{BTS}^{(1)}$

Input : $\mathrm{ct}_I = \mathrm{ct}(2^{kn}\mathbf{m}, 2^{kn} \cdot q)$
Output: $\mathrm{ct}_O = \mathrm{ct}(2^{kn}\mathbf{m} - 2^{-n}\mathbf{e}_{k+1}, Q_{rem})$

$\mathrm{ct}_1 \leftarrow RS(2^n, \mathrm{ct}_I)$ ▷ $\mathrm{ct}_1 = \mathrm{ct}(2^{(k-1)n}\mathbf{m} + \mathbf{e}_{rs}, q)$
$\mathrm{ct}_2 \leftarrow \mathsf{BTS}^{(k)}(\mathrm{ct}_1)$ ▷ $\mathrm{ct}_2 = \mathrm{ct}((2^{(k-1)n}\mathbf{m} + \mathbf{e}_{rs} + 2^{-n}\mathbf{e}_k, Q_{rem})$
$\mathrm{ct}_3 \leftarrow 2^n \cdot \mathrm{ct}_2$ ▷ $\mathrm{ct}_3 = \mathrm{ct}(2^{kn}\mathbf{m} + 2^n\mathbf{e}_{rs} + \mathbf{e}_k, Q_{rem})$
$\mathrm{ct}_4 \leftarrow [\mathrm{ct}_3]_{2^n q}$ ▷ $\mathrm{ct}_4 = \mathrm{ct}(2^{kn}\mathbf{m} + 2^n\mathbf{e}_{rs} + \mathbf{e}_k, 2^n \cdot q)$
$\mathrm{ct}_5 \leftarrow \mathrm{ct}_4 - \mathrm{ct}_I$ ▷ $\mathrm{ct}_5 = \mathrm{ct}(2^n\mathbf{e}_{rs} + \mathbf{e}_k, 2^n \cdot q)$
$\mathrm{ct}_6 \leftarrow \mathsf{BTS}^{(1)}(\mathrm{ct}_5)$ ▷ $\mathrm{ct}_6 = \mathrm{ct}(2^n\mathbf{e}_{rs} + \mathbf{e}_k + 2^{-n}\mathbf{e}_{k+1}, Q_{rem})$
$\mathrm{ct}_O \leftarrow \mathrm{ct}_3 - \mathrm{ct}_6$
return $\mathrm{ct}_O = \mathrm{ct}(2^{kn}\mathbf{m} - 2^{-n}\mathbf{e}_{k+1}, Q_{rem})$

ciphertext is $\mathrm{ct}_1 = \mathrm{ct}(2^{(k-1)n}\mathbf{m} + \mathbf{e}_{rs}, q)$. Here, we assume that the rescale error $\mathbf{e}_{rs}$ is smaller than the error of the basic bootstrapping $\mathsf{BTS}^{(1)}$, so it satisfies $\|\mathbf{e}_{rs}\|_\infty \le 2^{-n}$. By applying $\mathsf{BTS}^{(k)}$ to ct_1, we get $\mathrm{ct}_2 = \mathrm{ct}((2^{(k-1)n}\mathbf{m} + \mathbf{e}_{rs} + 2^{-n}\mathbf{e}_k, Q_{rem})$ by the condition of $\mathsf{BTS}^{(k)}$ where $\|\mathbf{e}_k\|_\infty \le 1$. To extract the error, multiplying ct_2 by 2^n, we get $\mathrm{ct}_3 = 2^n \cdot \mathrm{ct}_2 = \mathrm{ct}(2^{kn}\mathbf{m} + 2^n\mathbf{e}_{rs} + \mathbf{e}_k, Q_{rem})$.

To calculate the two ciphertexts $\mathrm{ct}_I, \mathrm{ct}_3$, we make modulus of ct_3 down. So, we obtain the ciphertext $\mathrm{ct}_4 = [\mathrm{ct}_3]_{2^n \cdot q} = \mathrm{ct}(2^{kn}\mathbf{m} + 2^n\mathbf{e}_{rs} + \mathbf{e}_k, 2^n \cdot q)$. And then, computing $\mathrm{ct}_4 - \mathrm{ct}_I$, we get $\mathrm{ct}_5 = \mathrm{ct}_4 - \mathrm{ct}_I = \mathrm{ct}(2^n\mathbf{e}_{rs} + \mathbf{e}_k, 2^n \cdot q)$.

Now, we get the ciphertext ct_5 whose message is the rescale and bootstrapping error $2^n\mathbf{e}_{rs} + e_k$. Since $\|2^n\mathbf{e}_{rs} + e_k\|_\infty \le 1$, we can apply $\mathsf{BTS}^{(1)}$ to ct_5. By applying $\mathsf{BTS}^{(1)}$ to ct_5, we get $\mathrm{ct}_6 = \mathrm{ct}(2^n\mathbf{e}_{rs} + \mathbf{e}_k + 2^{-n}\mathbf{e}_{k+1}, Q_{rem})$ where $\|\mathbf{e}_{k+1}\|_\infty \le 1$. Finally, computing $\mathrm{ct}_3 - \mathrm{ct}_6$, we get $\mathrm{ct}_O = \mathrm{ct}(2^{kn}\mathbf{m} - 2^{-n}\mathbf{e}_{k+1}, Q_{rem})$.

Theorem 3.2 (k-fold Bootstrapping Algorithm). *Given a standard bootstrapping algorithm* $\mathsf{BTS}^{(1)}$ *with*

$$\mathsf{PERF}_{\mathsf{BTS}^{(1)}} = (n, Q_{rem}/q, t),$$

one can construct a new bootstrapping algorithm $\mathsf{BTS}^{(k)}$ *whose performance is*

$$\mathsf{PERF}_{\mathsf{BTS}^{(k)}} = (kn, Q_{rem}/(2^{(k-1)n} \cdot q), kt)$$

by repeating $\mathsf{BTS}^{(1)}$ k *times where* k *is a positive integer with*

$$Q_{rem}/(2^{(k-1)n} \cdot q) \ge 1.$$

Proof. We only need to show that $\mathsf{BTS}^{(k+1)}$ constructed by Algorithm 2 satisfies all conditions. The input and output of $\mathsf{BTS}^{(k+1)}$ are $\mathrm{ct}_I = \mathrm{ct}(2^{kn}\mathbf{m}, 2^{kn} \cdot q)$, $\mathrm{ct}_O = \mathrm{ct}(2^{kn}\mathbf{m} - 2^{-n}\mathbf{e}_{k+1}, Q_{rem})$, respectively, and satisfy the condition of $\mathsf{BTS}^{(k+1)}$. Also, $\mathsf{BTS}^{(k+1)}$ preserves $(k+1)n$-bit precision of the message $\mathbf{m}$ and the running time of $\mathsf{BTS}^{(k+1)}$ is $(k+1)t$. Therefore, $\mathsf{PERF}^{(k+1)}_{\mathsf{BTS}} = ((k+1)n, Q_{rem}/(2^{kn} \cdot q), (k+1)t)$. Thus, $\mathsf{BTS}^{(k+1)}$ holds all conditions. By mathematical induction, we can construct a kn-bit precision $\mathsf{BTS}^{(k)}$ which satisfies the all conditions with just an n-bit precision bootstrapping $\mathsf{BTS}^{(1)}$ for a positive integer k. □

Since the *RemainModulus* of $\mathsf{BTS}^{(k)}$ decreases as k increases, Meta-BTS algorithm also cannot grow k infinitely. Therefore, there is a upper bound of precision that can be obtained for a fixed bootstrapping algorithm $\mathsf{BTS}^{(1)}$.

4 APPLICATION

In this section, we propose a new method of obtaining high precision FHE using our Meta-BTS algorithm. Also, we analyze the upper bound of precision and the time complexity in that case.

4.1 Standardization of Meta-BTS

Before the construction of FHE, we revisit the encoding process of the message.

Revisit encoding and scaling factor Given a scaling factor Δ and a message $\mathbf{m} \in \mathbb{C}^{N/2}$, the encoding of $\mathbf{m}$ is $m(X) = \lfloor \Delta \cdot \tau^{-1}(\mathbf{m}) \rceil \in R$ where τ is the canonical embedding $\tau : \mathbb{R}[X]/(X^N + 1) \to \mathbb{C}^{N/2}$. Since τ is a linear map, $\tau(m(X))$ is approximately equal to $\Delta \cdot m$. In this sense, $\tau(m(X))$ could be regarded as a slot-side value of the ciphertext that encrypts $m(X)$. Let $\tau(m(X))$ be a *scaled message* of the ciphertext.

For example, when we multiply two ciphertexts without rescaling, the output ciphertext has a scaled message of $\Delta^2 \cdot \mathbf{m}_1 \odot \mathbf{m}_2$. We rescale the ciphertext by Δ to maintain scaling factor, and the scaled message becomes $\Delta \cdot \mathbf{m}_1 \odot \mathbf{m}_2$.

Given a $(r+n)$-bit precision bootstrapping algorithm BTS with input bound 2^r and scaling factor Δ, we can understand BTS from the perspective of scaled message. BTS outputs $\mathrm{ct}(\mathbf{m}, q)$ to $\mathrm{ct}(\mathbf{m} + 2^{-n}\mathbf{e}, Q_{rem})$ where $\|\mathbf{m}\|_\infty \le 2^r, \|\mathbf{e}\|_\infty \le 1$. In the perspective of scaled message, BTS outputs $\Delta \cdot \mathbf{m}$ to $\Delta \cdot (\mathbf{m} + 2^{-n}\mathbf{e})$.

Lemma 4.1 (Standardization). *Let* BTS *be an* $r+n$*-bit precision bootstrapping algorithm with input bound* 2^r *and a scaling factor* Δ*. If one uses a scaling factor as* $\Delta' = \Delta \cdot 2^r$ *for encoding, one can construct a fully homomorphic encryption having* BTS *as its standard bootstrapping algorithm, retaining* $(r+n)$*-bit precision.*

Proof. Let Q_{rem}/q be the remaining modulus of BTS. Let $\mathbf{m}$ be a message such that $\|\mathbf{m}\|_\infty \le 1$. Given an input ciphertext $\mathrm{ct}(\mathbf{m}, q)$ with a scaling factor Δ', it can be considered as $\mathrm{ct}(2^r\mathbf{m}, q)$ with a scaling factor Δ. After performing BTS in the initial sense, we get the output ciphertext $\mathrm{ct}(2^r\mathbf{m} + 2^{-n}\mathbf{e}, Q_{rem})$ with a scaling factor Δ, where $\|\mathbf{e}\|_\infty \le 1$. Again, the output can be considered as $\mathrm{ct}(\mathbf{m} + 2^{-r-n}\mathbf{e}, Q_{rem})$ with a scaling factor Δ'. This gives a complete construction of a fully homomorphic encryption having BTS as its standard bootstrapping algorithm. □

Lemma 4.1 gives a standardized version of Theorem 3.2. We get a following Corollary 4.2.

Corollary 4.2 (Theorem 3.2, standardized). *Let* $\mathsf{BTS}^{(1)}$ *be a standard bootstrapping algorithm with scaling factor* Δ_1 *and performance* $(n, Q_{rem}/q, t)$*. If one uses a scaling factor as* $\Delta_k = \Delta_1 \cdot 2^{(k-1)n}$ *to encode a message for possible* k*, one can construct a fully homomorphic encryption having* $\mathsf{BTS}^{(k)}$ *as its standard bootstrapping algorithm, with* kn*-bit precision.*

4.2 Getting unbounded operations of high precision with fixed bootstrapping

We analyze the upper bound of the precision which supports unbounded operations using our Meta-BTS algorithm. However, the CKKS bootstrapping can be implemented in various methods [2, 6, 7, 9, 19, 20, 22–24], and each method has different performance of bootstrapping. Even the best bootstrapping algorithm can vary

depending on the desired precision. Therefore, we assume that the bootstrapping algorithm BTS is given, and compute the upper bound of the number of BTS iterations and the precision at this time.

Theorem 4.3 (Upper bound of the precision). *Let* $\mathsf{BTS}^{(1)}$ *be a standard bootstrapping algorithm with performance* $(n, Q_{rem}/q, t)$. *Let* N *be the ciphertext polynomial dimension, and* h *be the number of nonzero coefficients in the secret key polynomial. Then the* k*-fold bootstrapping* $\mathsf{BTS}^{(k)}$ *can be constructed for*

$$k \leq \frac{\log(Q_{rem}/(q\sqrt{Nh})) + n}{2n}.$$

Let $n_{\max}$ *be the maximum precision of* $\mathsf{BTS}^{(k)}$ *in order for ciphertexts to be multiplied indefinitely. Then* $n_{\max}$ *is given by*

$$n_{\max} = \lfloor \frac{\log(Q_{rem}/q\sqrt{Nh}) + n}{2n} \rfloor \cdot n.$$

Proof. Suppose that the *multiplicative depth* which is the number of possible multiplication after $\mathsf{BTS}^{(k)}$ is 1 to compute the upper bound of the iteration number k and the precision $n_{\max}$. According to Theorem 3.2, we can construct a new bootstrapping algorithm $\mathsf{BTS}^{(k)}$ and its performance is $(kn, Q_{rem}/(2^{(k-1)n} \cdot q))$ with a positive integer k. Then, we set the scaling factor Δ as $Q_{rem}/(2^{(k-1)n} \cdot q)$ and the precision after a ciphertext multiplication is approximated by $(\log(Q_{rem}/(2^{(k-1)n} \cdot q)) - \log\sqrt{Nh})$-bit.

Until the precision after the bootstrapping is increased to the precision after a ciphertext multiplication, the ciphertext can continue homomorphic operations while maintaining the precision. So, for unbounded homomorphic operations, we obtain the following inequality.

$$\begin{aligned} kn &\leq \log(Q_{rem}/(2^{(k-1)n} \cdot q)) - \log\sqrt{Nh}, \\ 2kn &\leq \log(Q_{rem}/(q\sqrt{Nh})) + n, \\ k &\leq \frac{\log(Q_{rem}/(q\sqrt{Nh})) + n}{2n}. \end{aligned}$$

Therefore,

$$k \leq \frac{\log(Q_{rem}/(q\sqrt{Nh})) + n}{2n}$$

and

$$n_{\max} = \lfloor \frac{\log(Q_{rem}/q\sqrt{Nh}) + n}{2n} \rfloor \cdot n.$$ □

Based on this analysis, we introduce experiments to obtain a high precision using Meta-BTS according to N in section 5.

4.3 Accelerating bootstrapping with fixed precision

In this subsection, we analyze the bootstrapping time complexity when making an n-bit precision bootstrapping algorithm by repeating n/k-bit precision bootstrapping algorithm k times. The time complexity of m-bit integer multiplication is at least $O(m \log m)$.

Before we analyze our Meta-BTS algorithm, we analyze the time complexity of an n-bit precision bootstrapping algorithm. A parameter set of HE such as the ciphertext polynomial dimension N and the number of nonzero numbers in the secret key h is given. Also, for a fair comparison, we assume that modulus of a ciphertext after a bootstrapping is Q_{rem}.

The CKKS bootstrapping consists of four step: StC, ModRaise, CtS, EvalMod. Since ModRaise does not use the level of a ciphertext and the number of levels to perform StC, CtS are not affected by the precision, we only define the level for EvalMod, Rest steps of BTS. Let d_{EvalMod}, d_{rest} be the number of levels to perform EvalMod, rest steps. Then, the size of a ciphertext modulus used by an n-bit bootstrapping is $(d_{\mathsf{EvalMod}} + d_{rest}) \cdot O(n + \log\sqrt{Nh})$-bit where $\sqrt{Nh}$ is added because of a rescale error. Therefore, the total size of a ciphertext modulus is $(d_{\mathsf{EvalMod}} + d_{rest}) \cdot O(n + \log\sqrt{Nh}) + \log(Q_{rem})$-bit.

Also, we define C_n as the the number of multiplications used by an n-bit precision bootstrapping with fixed N, h. Since higher precision needs a higher degree of polynomial approximation, C_n is an incremental function for n.

Lemma 4.4. *The time complexity of an* n*-bit precision bootstrapping algorithm is at least* $C_n \cdot O(Q_n \log Q_n)$ *where the* Q_n *is* $(d_{\mathsf{EvalMod}} + d_{rest}) \cdot O(n + \log\sqrt{Nh}) + \log(Q_{rem})$.

Proof. The size of a ciphertext modulus is $(d_{\mathsf{EvalMod}} + d_{rest}) \cdot O(n + \log\sqrt{Nh}) + \log(Q_{rem})$. Let Q_n be $(d_{\mathsf{EvalMod}} + d_{rest}) \cdot O(n + \log\sqrt{Nh}) + \log(Q_{rem})$. In the bootstrapping, since multiplication of Q_n-bit integer is performed C_n times, the time complexity is at least $C_n \cdot O(Q_n \log Q_n)$. □

Furthermore, d_{EvalMod} increase with precision because higher precision needs a higher degree of polynomial approximation. We consider only the cosine function among approximations of the trigonometric functions in EvalMod. The relationship between degree and bit precision (i.e. $-\log_2(\|p(x) - \cos(m\pi x)\|_\infty)$) is investigated when several cosine functions are approximated by minimax approximation $p(x)$ using the Remez algorithm.

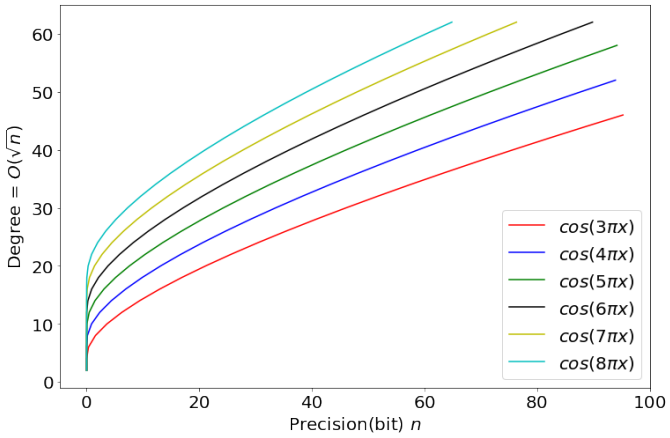

Figure 4: Precision and Degree of the Minimax Approximation of $cos(m\pi x)$ by Remez algorithm.

As shown in Fig.4, the degree of approximate polynomial as proportional to the square root of bit precision. Therefore, we assume that the minimum polynomial degree to achieve n-bit precision

is approximately $O(\sqrt{n})$. Using the BSGS algorithm [2], a polynomial approximation can be computed only by square root of the polynomial degree multiplications. Thus, the number of ciphertext multiplication d_{EvalMod} is roughly $O(n^{1/4})$. To sum up, in the n-bit precision bootstrapping, the total size of the ciphertext modulus is $O(n^{1/4}(n+\log\sqrt{Nh}))+d_{rest}\cdot O(n+\log\sqrt{Nh})\log(Q_{rem})$.

Lemma 4.5. **(Heuristic).** *If n is large enough, the time complexity of an n-bit precision bootstrapping algorithm is at least $C_n \cdot O(n^{5/4}\log n)$.*

Proof. Since the number of ciphertext multiplication d_{EvalMod} is roughly $O(n^{1/4})$, the ciphertext modulus is approximated by $O(n^{1/4}(n+\log\sqrt{Nh}))+d_{rest}\cdot O(n+\log\sqrt{Nh})\log(Q_{rem})$. If n is large enough to ignore other constant values, the ciphertext modulus can be simplified to $O(n^{5/4})$. Therefore, the time complexity of an n-bit precision bootstrapping algorithm is at least $C_n \cdot O(n^{5/4}\log n)$. □

Using our Meta-BTS algorithm, an n-bit precision bootstrapping can be constructed by repeating a n/k-bit precision bootstrapping k times for some k. If n is large enough, the time complexity of an n-bit precision bootstrapping is superlinear for n due to the time complexity of multiplication. By Theorem 3.2, our Meta-BTS algorithm linearly increases the bootstrapping time with the number of iterations k. Therefore, Our Meta-BTS algorithm can improve the time complexity of high precision bootstrapping.

Theorem 4.6. **(Heuristic).** *Given a parameter set of FHE. Assume that n is large enough to ignore other parameters and the time complexity of n-bit integer multiplication is $O(n\log n)$.*

When making an n-bit precision bootstrapping by repeating n/k-bit precision bootstrapping k times using Meta-BTS, the time complexity of the n-bit precision bootstrapping is $C_{n/k}\cdot O\left(\frac{n^{5/4}}{k^{1/4}}\log\left(\frac{n}{k}\right)\right)$.

Proof. For the iteration number k, which n/k is high enough, the time complexity of n/k-bit precision bootstrapping algorithm is $C_{n/k}\cdot O\left(\left(\frac{n}{k}\right)^{5/4}\log\left(\frac{n}{k}\right)\right)$ by Lemma 4.5.

When making an n-bit precision bootstrapping algorithm by repeating n/k-bit precision bootstrapping algorithm k times, the time complexity of Meta-BTS is k times the time complexity of n/k-bit precision bootstrapping algorithm by Theorem 3.2.

Therefore, using Meta-BTS, the time complexity of the n-bit precision bootstrapping decrease from $C_n\cdot O(n^{5/4}\log n)$ to $C_{n/k}\cdot O\left(\frac{n^{5/4}}{k^{1/4}}\log\left(\frac{n}{k}\right)\right)$. □

5 CONCRETE PARAMETERS AND EXPERIMENTS

We provide a proof-of-concept implementation to show the performance of Meta-BTS. Our source code is developed in C++ with our **HEaaN** library. We summarize our optimization techniques, recommended parameter sets and some experimental results in this section. All experiments are conducted single-threaded on an Intel Xeon Gold 6242 at 2.8 GHz with 502GB of RAM running Linux. Our parameter sets satisfy at least 128-bit security according to Lattice-estimator [1] and known best attacks [10, 25].

Using Meta-BTS, one can repeat the existing bootstrapping algorithm to increase the bootstrapping precision without having to increase the size of primes for bootstrapping algorithm. Therefore, since we only need to increase the prime used in a ciphertext multiplication, we can obtain a higher precision than the existing algorithms on the same parameter.

5.1 A basic example

In this subsection, we provide a basic example of Meta-BTS. We use a parameter set which maintains an 8-bit precision as described in Table 2. In this parameter, the bootstrapping algorithm uses maximum 41-bit primes and the ciphertext multiplication uses 28-bit primes.

Table 2: The base parameter set. $\log QP$ **and** $\log q_i$ **denote the bit lengths of the largest RLWE modulus and individual RNS primes, respectively.** h **denotes the hamming weight of secret key.** $depth$ **denotes the number of possible multiplications after the bootstrapping algorithm.**

$\log N$	$\log QP$	h	$\log q_i$	$depth$	bit prec.
15	762	192	28-41	6	9

We apply Meta-BTS to this parameter and observe that the precision of a ciphertext increases with the number of iterations. All the parameters satisfy 128-bit security.

Table 3: Experimental result of Meta-BTS. $\log q_{\mathsf{Mult}}$ **denotes the bit lengths of individual primes which uses a ciphertext multiplication.** $depth$ **denotes the number of possible multiplications after the bootstrapping algorithm.**

$iter$	$\log q_{\mathsf{Mult}}$	$depth$	bit prec.	boot time
1	28	6	9	7.86s
2	36	4	17	14.2s
3	44	3	25	19.2s
4	52	2	34	23.8s

Theoretically, the bootstrapping time for the kth iteration should be roughly k times the initial bootstrapping time. However, it is actually smaller than the expected value, since we should reduce the number of multiplication primes in order to satisfy the same security level.

Remark 1. *Even when parameter modification is not allowed, Meta-BTS could still be applied to achieve higher precision. We can group several multiplication primes together and consider them as a rescaling unit, enabling a high precision multiplication.*

5.2 Achieving the maximum precision

In Table 1, we described the maximum precision using Meta-BTS. In this section, we propose the parameter sets obtaining the maximum precision.

Table 4: Overview of the base parameter for $N = 2^{17}$. $\log QP$, $\log(q_i)$, and $\log(p_j)$ denote the bit lengths of the largest RLWE modulus, individual RNS primes, and temporary primes for Modulus switching, respectively. Δ_{Ecd} denotes the encoding scaling factor, L denotes the maximum ciphertext level, and h denotes the hamming weight of a secret key. Base, StC, Mult, Sine, and CtS denote sloToCoeff, multiplication, evalMod, and coeffToSlot primes, respectively. The left and right operands of the dot product denotes the number and the bit lengths of primes, respectively.

$\log QP$	Δ_{Ecd}	L	h	λ	boot time
2783	2^{54}	35	128	> 128	51.4s

$log(q_i)$					$log(p_j)$
Base	StC	Mult	Sine	CtS	
61	$3 \cdot 54$	$16 \cdot 54$	$13 \cdot 61$	$3 \cdot 57$	$12 \cdot 61$

5.2.1 Implementation for $N = 2^{17}$. We construct a parameter for $N = 2^{17}$, and achieve the maximum precision with actual implementation. First, we introduce the base parameter in Table 4.

We iterated Meta-BTS until we achieve the maximum precision described in theorem 4.3. Table 5 describes the implementation result including parameter constructions, ordered by number of iterations.

Table 5: Experimental result of Meta-BTS, achieving the maximum possible precision. Div denotes the primes reserved for dividing by 2^n.

$iter$	$\log QP$	Δ_{Ecd}	$log(q_i)$		$log(p_j)$	boot time
			Div	Mult[1]		
1	2783	2^{54}	-	$16 \cdot 54$	$12 \cdot 61$	51.4s
2	2897	2^{84}	$1 \cdot 30$	$10 \cdot (2 \cdot 42)$	$14 \cdot 60$	128s
3	2777	2^{114}	$2 \cdot 30$	$7 \cdot (2 \cdot 57)$	$12 \cdot 61$	156s
4	2790	2^{144}	$3 \cdot 30$	$5 \cdot (3 \cdot 48)$	$13 \cdot 61$	230s
5	2735	2^{174}	$4 \cdot 30$	$4 \cdot (3 \cdot 58)$	$12 \cdot 61$	259s
6	2742	2^{204}	$5 \cdot 30$	$3 \cdot (4 \cdot 51)$	$13 \cdot 61$	344s
7	2868	2^{234}	$6 \cdot 30$	$3 \cdot (4 \cdot 59)$	$13 \cdot 61$	405s
8	2720	2^{264}	$7 \cdot 30$	$2 \cdot (5 \cdot 53)$	$13 \cdot 61$	465s
9	2810	2^{294}	$8 \cdot 30$	$2 \cdot (5 \cdot 59)$	$13 \cdot 61$	514s
14	2881	2^{444}	$13 \cdot 30$	$1 \cdot (9 \cdot 50)$	$14 \cdot 61$	903s

Let k be the number of iterations. According to theorem 4.3,

$$k \leq \lfloor \frac{\log(Q_{rem}/q) - \log(\sqrt{Nh}) + n}{2n} \rfloor = \lfloor \frac{16 \cdot 54 - 12 + 30}{60} \rfloor = 14,$$

and the maximum possible precision is $nk = 420$ bits. The last parameter shown in Table 5 achieves the maximum precision. The actual bootstrapping time for this parameter is $903s$, which is greater than the expected $51.4s \cdot 14 = 719.6s$. Due to the implementation issues, the maximum ciphertext level L increased, so the bootstrapping time increased more. When $iter$ is 3 or 5, the bootstrapping time, $156s$, $259s$, is almost same as expected, $51.4s \cdot 3 = 154.2s$, $51.4s \cdot 5 = 257s$.

5.2.2 Construction for $N = 2^{15}, 2^{16}$. We construct parameters for $N = 2^{15}, 2^{16}$, that achieve the maximum precision. The Table 6, 7 describes the base parameters for the construction.

Table 6: Overview of the base parameter for $N = 2^{15}$.

$log(QP)$	Δ_{Ecd}	L	h	λ	boot time
759	2^{36}	14	192	>128	6.43s

$log(q_i)$					$log(p_j)$
Base	StC	Mult	Sine	CtS	
49	$2 \cdot 33$	$3 \cdot 36$	$8 \cdot 49$	$2 \cdot 47$	50

Table 7: Overview of the base parameter for $N = 2^{16}$.

$log(QP)$	Δ_{Ecd}	L	h	λ	boot time
1549	2^{36}	29	192	128	16.3s

$log(q_i)$					$log(p_j)$
Base	StC	Mult	Sine	CtS	
49	$3 \cdot 36$	$14 \cdot 36$	$9 \cdot 49$	$3 \cdot 49$	$6 \cdot 50$

Table 8, 9 describes the maximum precision parameters constructed from the base parameters achieving 48, 255 bit precision, respectively.

Table 8: Overview of the high precision parameter for $N = 2^{15}$.

$log(QP)$	Δ_{Ecd}	L	h	λ
755	$2^{68} = 2^{32} \cdot 2^{36}$	15	192	>128

$log(q_i)$					$log(p_j)$
Base	StC	Div + Mult	Sine	CtS	
49	$2 \cdot 33$	$32 + 1 \cdot (2 \cdot 34)$	$8 \cdot 49$	$2 \cdot 47$	54

Table 9: Overview of the high precision parameter for $N = 2^{16}$.

$log(QP)$	Δ_{Ecd}	L	h	λ
1570	$2^{276} = 2^{240} \cdot 2^{36}$	24	192	>128

$log(q_i)$					$log(p_j)$
Base	StC	Div + Mult	Sine	CtS	
49	$3 \cdot 36$	$4 \cdot 60 + 1 \cdot (5 \cdot 56)$	$9 \cdot 49$	$3 \cdot 49$	$5 \cdot 61$

[1]The product inside the bracket means that the group of primes act as a rescaling unit.

5.3 An improved parameter set based on a set in [2]

Table 10 describes the overview of the parameter sets I, II and III suggested in [2]. Here N denotes the dimension of the polynomial space, Δ_{Ecd} denotes the scaling factor of the input ciphertext, $log(Q_{L-k}/q_0)$ denotes the remaining coefficient modulus excluding the base prime after bootstrapping, and ϵ denotes the precision of the bootstrapping. As you can see, Set III leaves twice as much remaining coefficient modulus as Set II, while providing more than half a precision compared to that of Set II and maintaining faster bootstrapping time.

Table 10: Overview of Set I, II and III in [2]

Set	N	Δ_{Ecd}	boot time[2]	$log(Q_{L-k}/q_0)$	$log(\epsilon^{-1})$
I	2^{16}	2^{40}	23.0s	360	25.7(*mean)
II	2^{16}	2^{45}	23.4s	225	31.5(*mean)
III	2^{16}	2^{30}	18.1s	450	19.1(*mean)

We can construct a new parameter set based on Set III, by taking Meta-BTS as its default bootstrapping method. The new set is more precise in terms of bootstrapping precision and faster in terms of amortized multiplication time, compared with Set II. Our new parameter set III' is given as follows. Notations in Table 11 follow from Table 5 in [2].

Table 11: Suggested parameter Set III'

III'				
$\log(QP)$	N	Δ	h	L
1545	2^{16}	2^{43}	192	24
$\log(q_i)$				$\log(p_j)$
Base + Div + Mult	StC	Sine	CtS	
$55 + (13 + 43) + 9 \cdot 43$	$30 + 60$	$8 \cdot 55$	$4 \cdot 53$	$5 \cdot 61$

Let us compare the performance of Set III' with other parameter sets given in Table 10. Homomorphic operations except bootstrapping under Set III' should be as fast as Set I since they have the same parameter structure. If we take Meta-BTS using Theorem 3.1 as the default bootstrapping strategy for Set III', then we see that the running time for bootstrapping would be about 46 seconds which is twice of the bootstrapping running time of Set I. Since Set III' has multiplicative depth of 10, which means we can do twice as much multiplications with Set III' as with Set II, we see that the amortized multiplication of Set III' is approximately the same as that of Set II.

6 CONCLUSION

In this paper, we presented the Meta-BTS algorithm for obtaining the high precision bootstrapping algorithm in the CKKS scheme. We provided the experimental results of our algorithms by implementing existing bootstrapping algorithm. The main contribution is to propose much higher precision after bootstrapping algorithm with fixed N compared to prior works [2, 6, 7, 9, 19, 20, 22–24]. Our technique supported high-precision operations by repeating the bootstrapping algorithm even on a fixed parameter set. Finally, we proposed the parameter sets that can obtain higher precision according to N than existing one, respectively. It is an open problem that applies the technique of obtaining high precision by repeating the bootstrapping algorithm in the BGV or BFV scheme with similar properties to the CKKS scheme.

[2]Data from [2].

REFERENCES

[1] Martin R. Albrecht, Rachel Player, and Sam Scott. 2015. On the concrete hardness of Learning with Errors. *Journal of Mathematical Cryptology* (2015). https://github.org/malb/lattice-estimator..

[2] Jean-Philippe Bossuat, Christian Mouchet, Juan Troncoso-Pastoriza, and Jean-Pierre Hubaux. 2021. Efficient Bootstrapping for Approximate Homomorphic Encryption with Non-sparse Keys. In *Advances in Cryptology – EUROCRYPT 2021*, Anne Canteaut and François-Xavier Standaert (Eds.). Springer International Publishing, Cham, 587–617.

[3] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2012. (Leveled) Fully Homomorphic Encryption without Bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference* (Cambridge, Massachusetts) *(ITCS '12)*. Association for Computing Machinery, New York, NY, USA, 309–325. https://doi.org/10.1145/2090236.2090262

[4] Zvika Brakerski and Vinod Vaikuntanathan. 2011. Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. In *Advances in Cryptology – CRYPTO 2011*, Phillip Rogaway (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 505–524.

[5] Zvika Brakerski and Vinod Vaikuntanathan. 2014. Efficient Fully Homomorphic Encryption from (Standard) LWE. *SIAM J. Comput.* 43, 2 (2014), 831–871. https://doi.org/10.1137/120868669 arXiv:https://doi.org/10.1137/120868669

[6] Nathan Manohar Charanjit S. Jutla. 2022. Sine Series Approximation of the Mod Function for Bootstrapping of Approximate HE. In *Advances in Cryptology – EUROCRYPT 2022*.

[7] Hao Chen, Ilaria Chillotti, and Yongsoo Song. 2019. Improved Bootstrapping for Approximate Homomorphic Encryption. In *Advances in Cryptology – EUROCRYPT 2019*, Yuval Ishai and Vincent Rijmen (Eds.). Springer International Publishing, Cham, 34–54.

[8] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. 2013. Batch Fully Homomorphic Encryption over the Integers. In *Advances in Cryptology – EUROCRYPT 2013*, Thomas Johansson and Phong Q. Nguyen (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 315–335.

[9] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. 2018. Bootstrapping for Approximate Homomorphic Encryption. In *Advances in Cryptology – EUROCRYPT 2018*, Jesper Buus Nielsen and Vincent Rijmen (Eds.). Springer International Publishing, Cham, 360–384.

[10] Jung Hee Cheon, Minki Hhan, Seungwan Hong, and Yongha Son. 2019. A Hybrid of Dual and Meet-in-the-Middle Attack on Sparse and Ternary Secret LWE. *IEEE Access* 7 (2019), 89497–89506.

[11] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 409–437.

[12] Jung Hee Cheon and Damien Stehlé. 2015. Fully Homomorphic Encryption over the Integers Revisited. In *Advances in Cryptology – EUROCRYPT 2015*, Elisabeth Oswald and Marc Fischlin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 513–536.

[13] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2017. Faster Packed Homomorphic Operations and Efficient Circuit Bootstrapping for TFHE. In *Advances in Cryptology – ASIACRYPT 2017*, Tsuyoshi Takagi and Thomas Peyrin (Eds.). Springer International Publishing, Cham, 377–408.

[14] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2019. TFHE: Fast Fully Homomorphic Encryption Over the Torus. *Journal of Cryptology* (2019). https://doi.org/10.1007/s00145-019-09319-x

[15] Léo Ducas and Daniele Micciancio. 2015. FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second. In *Advances in Cryptology – EUROCRYPT 2015*, Elisabeth Oswald and Marc Fischlin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 617–640.

[16] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat Practical Fully Homomorphic Encryption. Cryptology ePrint Archive, Report 2012/144. https://ia.cr/2012/144.

[17] Craig Gentry, Amit Sahai, and Brent Waters. 2013. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Annual Cryptology Conference*. Springer, 75–92.

[18] Craig Gentry, Amit Sahai, and Brent Waters. 2013. Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. In *Advances in Cryptology – CRYPTO 2013*, Ran Canetti and Juan A. Garay (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 75–92.

[19] Kyoohyung Han and Dohyeong Ki. 2020. Better Bootstrapping for Approximate Homomorphic Encryption. In *Topics in Cryptology – CT-RSA 2020*, Stanislaw Jarecki (Ed.). Springer International Publishing, Cham, 364–390.

[20] Charanjit S. Jutla and Nathan Manohar. 2020. Modular Lagrange Interpolation of the Mod Function for Bootstrapping of Approximate HE. Cryptology ePrint Archive, Report 2020/1355. https://ia.cr/2020/1355.

[21] Andrey Kim, Maxim Deryabin, Jieun Eom, Rakyong Choi, Yongwoo Lee, Whan Ghang, and Donghoon Yoo. 2021. General Bootstrapping Approach for RLWE-based Homomorphic Encryption. Cryptology ePrint Archive, Paper 2021/691. https://eprint.iacr.org/2021/691 https://eprint.iacr.org/2021/691.

[22] Joon-Woo Lee, Eunsang Lee, Yongwoo Lee, Young-Sik Kim, and Jong-Seon No. 2021. High-Precision Bootstrapping of RNS-CKKS Homomorphic Encryption Using Optimal Minimax Polynomial Approximation and Inverse Sine Function. In *Advances in Cryptology – EUROCRYPT 2021*, Anne Canteaut and François-Xavier Standaert (Eds.). Springer International Publishing, Cham, 618–647.

[23] Joon-Woo Lee, Yongwoo Lee, Young-Sik Kim, Youngjune Kim, Jong-Seon No, and HyungChul Kang. 2022. High-Precision Bootstrapping for Approximate Homomorphic Encryption by Error Variance Minimization. In *Advances in Cryptology – EUROCRYPT 2022*.

[24] Yongwoo Lee, Joon-Woo Lee, Young-Sik Kim, and Jong-Seon No. 2020. Near-Optimal Polynomial for Modulus Reduction Using L2-Norm for Approximate Homomorphic Encryption. *IEEE Access* PP (08 2020), 1–1. https://doi.org/10.1109/ACCESS.2020.3014369

[25] Yongha Son and Jung Hee Cheon. 2019. Revisiting the Hybrid Attack on Sparse Secret LWE and Application to HE Parameters *(WAHC'19)*. Association for Computing Machinery, 11––20.

[26] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. 2010. Fully Homomorphic Encryption over the Integers. In *Advances in Cryptology – EUROCRYPT 2010*, Henri Gilbert (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 24–43.