EvalRound Algorithm in CKKS Bootstrapping

Seonghak Kim[1][0000−0001−6366−5550], Minji Park[2], Jaehyung Kim[1][0000−0002−1624−6326], Taekyung Kim[1][0000−0001−8834−4796], and Chohong Min[2][0000−0002−6108−8742] ⋆

[1] Crypto Lab Inc., 1 Gwanak-ro, Gwanak-gu, Seoul, Korea, 08826
cryptolab@cryptolab.co.kr
https://cryptolab.co.kr
[2] Ewha Womans University, 52 Ewhayeodae-gil, Seodaemun-gu, Seoul, Korea, 03760
mathdept@ewha.ac.kr
http://math.ewha.ac.kr/

Abstract. Homomorphic encryption (HE) has opened an entirely new world up in the privacy-preserving use of sensitive data by conducting computations on encrypted data. Amongst many HE schemes targeting computation in various contexts, Cheon–Kim–Kim–Song (CKKS) scheme [8] is distinguished since it allows computations for encrypted real number data, which have greater impact in real-world applications.

CKKS scheme is a levelled homomorphic encryption scheme, consuming one level for each homomorphic multiplication. When the level runs out, a special computational circuit called bootstrapping is required in order to conduct further multiplications. The algorithm proposed by Cheon et al. [7] has been regarded as a standard way to do bootstrapping in the CKKS scheme, and it consists of the following four steps: ModRaise, CoeffToSlot, EvalMod and SlotToCoeff. However, the steps consume a number of levels themselves, and thus optimizing this extra consumption has been a major focus of the series of recent research.

Among the total levels consumed in the bootstrapping steps, about a half of them is spent in CoeffToSlot and SlotToCoeff steps to scale up the real number components of DFTmatrices and round them to the nearest integers. Each scale-up factor is very large so that it takes up one level to rescale it down. Scale-up factors can be taken smaller to save levels, but the error of rounding would be transmitted to EvalMod and eventually corrupt the accuracy of bootstrapping.

EvalMod aims to get rid of the superfluous qI term from a plaintext $\mathsf{pt} + qI$ resulting from ModRaise, where q is the bottom modulus and I is a polynomial with small integer coefficients. EvalRound is referred to as its opposite, obtaining qI. We introduce a novel bootstrapping algorithm consisting of ModRaise, CoeffToSlot, EvalRound and SlotToCoeff, which yields taking smaller scale-up factors without the damage of rounding errors.

Keywords: Homomorphic Encryption · CKKS Scheme · Bootstrapping

⋆ Corresponding author.

1 Introduction

Homomorphic encryption (HE) refers to a class of encryption schemes which enables computation over encrypted data. The Cheon–Kim–Kim–Song (CKKS) [8] scheme is recognized as one of the most efficient fully homomorphic encryption (FHE) scheme that supports computation on real/complex data. Unlike other FHE schemes that are designed for integer [2–4,14] or binary [10,11,13] messages, the CKKS scheme is designed for real/complex messages, as it supports efficient scaling down operation. Since real numbers are the usual data type for many applications including deep learning, there have been various studies and applications using the CKKS scheme.

In the CKKS scheme, each multiplication consumes certain amount of ciphertext modulus due to rescaling process. As computation including homomorphic multiplications progresses, the total ciphertext modulus in turn decreases, and eventually becomes too small to afford further multiplications. A homomorphic re-encryption of a ciphertext, so called bootstrapping, is required to recover the ciphertext modulus. In this way the levelled CKKS scheme becomes a fully homomorphic encryption (FHE) scheme [15].

The first step towards the conventional bootstrapping algorithm as presented in [7] is called ModRaise, which increases the ciphertext modulus from the bottom to the top modulus. Once the modulus has been raised, an integer polynomial multiple of the base modulus is added to the encrypted plaintext, and an appropriate modular reduction modulo the base modulus should be performed in order to recover the original data. Since it has no simple representation for the modular reduction by the basic algebraic manipulations (addition, multiplication or rotation), the modular reduction must be approximated by a polynomial evaluation with large degree, which is called EvalMod.

The step of CoeffToSlot and SlotToCoeff comes before and after the polynomial evaluation. After ModRaise, the multiples of the base modulus are added on the coefficients of the encrypted plaintext (the 'coefficient side'). However, the homomorphic polynomial evaluation should be performed to the slots of the encrypted message (the 'slot side'). Therefore, to map the value of coefficient side to slot side and vice versa, homomorphic evaluation of DFT/iDFTmatrix multiplication named CoeffToSlot and SlotToCoeff should be performed. As DFT/iDFTmatrices are complex matrices rather than integer ones, we should scale-up them and perform integer matrix multiplication. After multiplying such scaled-up matrices, we should scale-down to get the result which approximates the result of the complex matrix multiplication. Note that the scaling-down consumes modulus bits of the ciphertext.

These linear transformations (CoeffToSlot and SlotToCoeff), together with a polynomial approximation of the modular reduction function (EvalMod), consume a large ciphertext modulus and require a relatively high amount of running time. In particular, in most practical CKKS-FHE parameters, the remaining ciphertext modulus after bootstrapping is far less than the total ciphertext modulus available [1]. Hence, only a limited amount of homomorphic multiplication can be performed after bootstrapping, which degrades the overall performance of

the scheme, especially for deeper circuits. Furthermore, the amount of ciphertext modulus consumed in the bootstrapping process is a major factor keeping us from forming an FHE parameter with small ciphertext dimension under a fixed security level, for the total amount of ciphertext modulus needed to keep the security level is bounded once the ciphertext dimension has been fixed.

How much is the ciphertext modulus consumed during the homomorphic evaluation of the linear transformations in CoeffToSlot and SlotToCoeff steps? Since the evaluation of a linear transformation, or equivalently the product of a plaintext dense square matrix M and an encrypted vector $\mathbf{v}$ of dimension n, can be computed homomorphically by

$$M \cdot \mathbf{v} = \sum_{i=0}^{n-1} \mathrm{diag}_i(M) \odot \mathsf{rot}_i(\mathbf{v}) \qquad \text{(matrix-vector multiplication)} \qquad (1)$$

where $\mathrm{diag}_i(M)$ is the i-th diagonal of the matrix M, $\mathsf{rot}_i(\mathbf{v})$ is the rotation of $\mathbf{v}$ by index i and $\odot$ denotes the Hadamard multiplication, i.e. componentwise vector multiplication. This means that only one multiplicative depth is needed for each CoeffToSlot or SlotToCoeff step. However, this naïve method also requires n multiplications and n rotations, and it becomes quickly computationally infeasible as n grows exponentially in practical parameters.

As a remedy, in [17], the authors focused on the rich mathematical structure of the linear transformations in CoeffToSlot and SlotToCoeff, and proposed to decompose the linear transformations into the products of several sparse block diagonal matrices. In this way one can reduce the number of multiplications and rotations needed for the homomorphic evaluation of the linear transformations at the cost of using certain amount of multiplicative depths (cf. Subsection 2.4 and [17]). In practice, the depth consumption is equal to the number of decomposed matrices and usually taken to be 3 or 4, and it still requires a large amount of ciphertext modulus.

In this paper, we propose a new bootstrapping algorithm, replacing EvalMod by a new step called EvalRound, which addresses this modulus consuming problem on the evaluation of linear transformations and reduce the amount of ciphertext modulus consumed during bootstrapping.

1.1 Our contribution

In this work, we propose a novel bootstrap circuit, that yields a reduction of modulus consumed, compared to the conventional circuit [7]. The reduction of modulus amounts to lessening levels. Table 1 shows the reduction of modulus consumption on one of the practical parameters. Here N denotes the ciphertext dimension, $\log(QP)$ denotes the bit lengths of the largest RLWE modulus, and $\widetilde{\Delta}$ denotes the scaling factor of the CoeffToSlot matrix. The proposed method enables us to maintain the bootstrapping precision while using 32 bit smaller scaling factor in CoeffToSlot. The proposed method obtains better bootstrapping precision, while reducing the modulus consumption by 84 bits, which is equivalent

to preserving approximately two multiplicative depths. The details of this example is in Example 5.

Table 1: Comparison between conventional and proposed bootstrapping

	N	$\log(QP)$	$\widetilde{\Delta}$	Bootstrap Bit Precision	Modulus Consumption
Conventional	2^{17}	2900	2^{60}	-12.53	1160
Proposed			2^{29}	-14.80	1076 = 1160 - 84

We also constructed an improved parameter set based on the parameter set II proposed in [1], namely II'. New parameter set II' saves 1 depth while losing at most 1 bit of precision. Table 2 describes the comparison between set II and set II'. Here depth denotes the multiplicative level after bootstrapping, Δ denotes the size of the encoding scaling factor, and q_0 denotes the size of the base modulus. The detail of this example is in Example 6.

Table 2: Comparison between the set II in [1] and the proposed set II'.

Set	N	$\log(QP)$	depth	Δ	q_0	Bootstrap Precision	$\widetilde{\Delta}$
II	2^{16}	1547	5	2^{45}	2^{60}	-31.5	2^{58}
II'		1543	6			-30.5	2^{34}

It should be emphasized that only a negligible effort is needed to upgrade the conventional circuit to the proposed one. One may directly add one naive subtraction to the existing circuit to compute $\mathsf{EvalRound}(x) = x - \mathsf{EvalMod}(x)$, where $\mathsf{EvalMod}$ is the existing procedure of homomorphic modular reduction and $\mathsf{EvalRound}$ is the proposed homomorphic modular rounding.

1.2 Our proposal

Our main proposal is the use of $\mathsf{EvalRound}$ rather than $\mathsf{EvalMod}$, which is defined as $\mathsf{EvalRound} : x \mapsto x - \mathsf{EvalMod}(x)$. Below is the bird-eye view of the algorithm.

$$\mathsf{pt} \xrightarrow{\mathsf{ModRaise}} \mathsf{pt} + qI \xrightarrow{\mathsf{CoeffToSlot}^{\#}} (\mathsf{pt} + qI + e)^* \xrightarrow{\mathsf{EvalRound}} (qI)^* \xrightarrow{\mathsf{SlotToCoeff}} qI \xrightarrow{\mathsf{Subtract}} \mathsf{pt}.$$

In comparison, the conventional bootstrap algorithm, which is first presented on [7], would be viewed as below.

$$\mathsf{pt} \xrightarrow{\mathsf{ModRaise}} \mathsf{pt} + qI \xrightarrow{\mathsf{CoeffToSlot}} (\mathsf{pt} + qI)^* \xrightarrow{\mathsf{EvalMod}} (\mathsf{pt})^* \xrightarrow{\mathsf{SlotToCoeff}} \mathsf{pt}.$$

The conventional circuit consists of ModRaise, CoeffToSlot, EvalMod and SlotToCoeff. CoeffToSlot is a sequence of DFTmatrix multiplications. As the matrix elements are scaled up to integers with full scale factor Δ (e.g. 2^{60} or 2^{50}), each matrix multiplication consumes modulus bits. When d number of matrices are multiplied, CoeffToSlot consumes Δ^d modulus. The multiplication results mod-raised plaintext on slot-side, which is often denoted as $\mathsf{pt}+$ qI. EvalMod refers to the modular reduction $\mathsf{pt}+qI \equiv \mathsf{pt} (\mathrm{mod}\ q)$, so that $\mathsf{EvalMod}(\mathsf{pt}+qI) = \mathsf{pt}$. The following step, SlotToCoeff, computes the desired result of plaintext on coefficent-side.

The proposed circuit performs $\mathsf{CoeffToSlot}^{\#}$ instead of CoeffToSlot, which utilizes small scale factor $\widetilde{\Delta} \ll \Delta$, which reduces modulus consumption to $\widetilde{\Delta}^d$. In exchange for the modulus reduction, a non-negligible error e is appended to $pt + qI$.

The following step of the proposed algorithm is EvalRound, which refers to the rounding operation, so that $\mathsf{EvalRound}(\mathsf{pt}+qI) = qI$ under assumption $\|\mathsf{pt}\| \ll q$. As EvalRound is piecewise constant, $\mathsf{EvalRound}(\mathsf{pt} + qI + e) = \mathsf{EvalRound}(\mathsf{pt} + qI) = qI$ when $\|\mathsf{pt} + e\|$ remains to be much smaller than q. In other words, the error from the small scale factor is annihilated by EvalRound, allowing the use of the small scale factor while keeping the accuracy. This is the reason why we use EvalRound instead of EvalMod, which propagates error as $\mathsf{EvalMod}(\mathsf{pt}+qI+e) = \mathsf{pt} + e$ and corrupts the overall accuracy.

The output ciphertext of EvalRound then goes into SlotToCoeff, resulting a ciphertext encrypting qI in the coefficient side. So the extra Subtract step is needed for subtracting it from the ciphertext originally resulting from the ModRaise step, encrypting $\mathsf{pt} + qI$, to get the final ciphertext encrypting pt.

To sum up, our proposed circuit consists of ModRaise, CoeffToSlot, EvalRound, SlotToCoeff and Subtract. Since the subtraction is ignorable compared to the overall cost, our proposal is equivalent to the conventional circuit in computational cost, while taking the reduced scale factor $\widetilde{\Delta}$ on $\mathsf{CoeffToSlot}^{\#}$ that saves modulus and levels.

1.3 Related works

Bootstrapping of the CKKS scheme was first introduced in [7]. The notions of evaluating DFT/iDFTmatrix homomorphically and evaluating modular reduction via polynomial approximation of trigonometric function were proposed here. In order to improve the time complexity of the linear transformations, FFT-like decomposition was adopted in [5] and [17]. Since then, numerous studies [5,17–19,21,23] have been conducted to improve the approximation of a modular reduction function. Recently, the use of sine series to reduce the error caused by approximating a trigonometric function was presented in [20] and the method of directly approximating a modular reduction function while minimizing error variance was presented in [22]. Independently, using double hoisting technique to reduce the computation time of homomorphic linear transformation was proposed in [1].

2 Preliminaries

2.1 Encoding and decoding

In this subsection and what follows, we review the basic CKKS scheme([8]); it will provide us a bird's-eye view of the entire scheme, and it also serves us to fix notations used in our discussions hereafter. For a power-of-two N, denote by $R = \mathbb{Z}[x]/(x^N + 1)$, the ring of integers of the $2N$-th cyclotomic field, which is a fundamental ring for the CKKS scheme and the RLWE problem the CKKS scheme is based on. For a positive q, let $R_q = R/qR = \mathbb{Z}_q[x]/(x^N + 1)$. Here N is determined at the parameter selection step of the CKKS scheme. A CKKS ciphertext can encrypt a complex vector of a power-of-two length which is maximally $N/2$. This vector is called a *(complex) message*, and its encryption is called a ciphertext. Here for the ease of description, we assume every message has an exact length of $N/2$.

Let ζ be a primitive $2N$-th root of unity contained in $\mathbb{C}$, e.g., $\zeta = \exp(\pi\sqrt{-1}/N)$, where $\sqrt{-1}$ is a complex imaginary unit. For integers i, write $\zeta_i := \zeta^{5^i}$. The map

$$\mathsf{DFT}_N : \mathbb{R}[x]/(x^N + 1) \to \mathbb{C}^{N/2}, \quad m(x) \mapsto (m(\zeta_0), m(\zeta_1), \cdots, m(\zeta_{N/2-1})) \quad (2)$$

is known to be an isomorphism by [5], with inverse iDFT_N. When the dimension N is understood, we also omit the subscript N so we write $\mathsf{DFT} = \mathsf{DFT}_N$ and $\mathsf{iDFT} = \mathsf{iDFT}_N$. With these algebraic maps, we can encode a complex message $\mathbf{z} \in \mathbb{C}^{N/2}$ to a *plaintext* $\mathsf{pt} \in R$ and in reverse decode from pt to $\mathbf{z}$.

- $\mathsf{Encode}(\mathbf{z}; \Delta)$. For an $N/2$-dimensional vector $\mathbf{z}$ of complex numbers and a scale factor Δ, the encoding process first transforms $\mathbf{z}$ to a polynomial in $\mathbb{R}[x]/(x^N + 1)$ and quantize it into an element of R. It returns

$$\mathsf{pt} = \mathsf{Encode}(\mathbf{z}; \Delta) = \lfloor \Delta \cdot \mathsf{iDFT}(\mathbf{z}) \rceil, \quad (3)$$

  where $\lfloor \cdot \rceil$ is the coefficient-wise rounding to the nearest integers.
- $\mathsf{Decode}(\mathsf{pt}; \Delta)$. For a plaintext pt and its scale factor Δ, the decoding process returns

$$\mathbf{z} = \mathsf{Decode}(\mathsf{pt}; \Delta) = \mathsf{DFT}(\mathsf{pt}/\Delta). \quad (4)$$

  Here the polynomial pt/Δ is computed in $\mathbb{R}[x]/(x^N + 1)$.

2.2 Basic operations of the CKKS scheme

Let χ_{key} be the distribution that outputs polynomials in R with coefficients in $\{-1, 0, 1\}$ with a fixed Hamming weight (the number of nonzero coefficients). By χ_{err} and χ_{enc} denote discrete Gaussian distribution with mean 0 and with some fixed standard deviation.

- **SetUp**. $\mathsf{Params} \leftarrow \mathsf{SetUp}(1^\lambda)$. Take a security level λ as an input and return the public parameters Params such as the ciphertext dimension N and the chain of moduli $Q_0 < Q_1 < \cdots < Q_L$ with maximal level L.

- **Key Generation**. $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{Params})$. Take Params and output a pair of a secret key $\mathsf{sk} = (1, s) \in R \times R$ and an encryption key $\mathsf{pk} = (\mathsf{pk}_0, \mathsf{pk}_1) \in R_{Q_L} \times R_{Q_L}$. More precisely,
  - Sample $s \leftarrow \chi_{\text{key}}$, and set $\mathsf{sk} = (1, s) \in R \times R$. For convenience, denote by h the Hamming weight of the polynomial s. This is fixed once Params has been set.
  - Sample $\mathsf{pk}_1 \leftarrow R_{Q_L}$ and $e \leftarrow \chi_{\text{err}}$. Output $\mathsf{pk} = (\mathsf{pk}_0 := [-\mathsf{pk}_1 \cdot s + e]_{Q_L}, \mathsf{pk}_1)$.
- **Switching Key Generation**. $\mathsf{swk}_{\mathsf{sk}' \to \mathsf{sk}} \leftarrow \mathsf{KSGen}_{\mathsf{sk}}(\mathsf{sk}')$. Given two secret keys $\mathsf{sk} = (1, s)$ and $\mathsf{sk}' = (1, s')$, sample $a \leftarrow R_{PQ_L}$ with auxiliary modulus P and $e \leftarrow \chi_{\text{err}}$ and output $\mathsf{swk}_{\mathsf{sk}' \to \mathsf{sk}} := (\mathsf{swk}_0, \mathsf{swk}_1)$ with $\mathsf{swk}_1 = a$ and $\mathsf{swk}_0 = -a \cdot s + e + P \cdot s' \pmod{PQ_L}$.
  - Set the relinearization key as $\mathsf{rlk} := \mathsf{KSGen}_{\mathsf{sk}}(s^2)$.
  - Set the rotation keys for j-step rotation as $\mathsf{rk}_j := \mathsf{KSGen}_{\mathsf{sk}}(s(x^{5^j}))$ for $1 \le j < N/2$.
- **Encryption**. $\mathsf{ct} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mathsf{pt})$. Given a plaintext pt given by a polynomial $m(x) \in R$, sample $v \leftarrow \chi_{\text{enc}}$ and $e_0, e_1 \leftarrow \chi_{\text{err}}$, output the ciphertext $\mathsf{ct} = v \cdot \mathsf{pk} + (m(x) + e_0, e_1) \pmod{Q_L}$.
- **Decryption**. $\mathsf{pt} \leftarrow \mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct})$. Given a ciphertext $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$, output the plaintext $\mathsf{pt} = [\langle \mathsf{ct}, \mathsf{sk} \rangle]_{Q_0} = [\mathsf{ct}_0 + \mathsf{ct}_1 \cdot s]_{Q_0}$, where Q_0 is the modulus for level zero.
- **Addition and Subtraction**. ct_{add}, $\mathsf{ct}_{\text{sub}} \leftarrow \mathsf{Add}(\mathsf{ct}, \mathsf{ct}')$, $\mathsf{Sub}(\mathsf{ct}, \mathsf{ct}')$, respectively. Given two ciphertexts ct and ct' in $R^2_{Q_\ell}$, output the ciphertext $\mathsf{ct}_{\text{add}} = [\mathsf{ct}+\mathsf{ct}']_{Q_\ell}$ and $\mathsf{ct}_{\text{sub}} = [\mathsf{ct}-\mathsf{ct}']_{Q_\ell}$. The resulting ciphertext ct_{add} and ct_{sub} are encrypting message vectors $\mathbf{z} + \mathbf{z}'$ and $\mathbf{z} - \mathbf{z}'$, respectively, where $\mathbf{z}$ (resp. $\mathbf{z}'$) is the message for ct (resp. ct').
- **Multiplication**. $\mathsf{ct}_{\text{mult}} \leftarrow \mathsf{Mult}(\mathsf{ct}, \mathsf{ct}')$. Given two ciphertexts $\mathsf{ct} = (c_0, c_1)$ and $\mathsf{ct}' = (c'_0, c'_1)$ in $R^2_{Q_\ell}$, output the ciphertext $\mathsf{ct}_{\text{mult}} := (c_0c'_0, c_1c'_0 + c_0c'_1, c_1c'_1)$. This seemingly unconventional ciphertext can be decrypted by taking the inner product with $(1, s, s^2)$, where $\mathsf{sk} = (1, s)$ is the secret key. One can get rid of the additional component $c_1c'_1$ which is multiplied by the component s^2 of the secret key by applying the key switching algorithm with the relinearization key rlk.
- **Rotation**. $\mathsf{ct}_{\text{rot},\, j} \leftarrow \mathsf{Rot}_j(\mathsf{ct})$. Given a ciphertext $\mathsf{ct} = (c_0, c_1) \in R^2_{Q_\ell}$, output $\mathsf{ct}_{\text{rot},\, j} := (c_0(x^{5^j}), c_1(x^{5^j}))$. The resulting ciphertext $\mathsf{ct}_{\text{rot},\, j}$ can be decrypted with the secret key $(1, s(x^{5^j}))$ where $(1, s)$ is the secret key for the original ciphertext ct. This discrepancy can also be resolved by key switching algorithm and one can transform $\mathsf{ct}_{\text{rot},\, j}$ to another ciphertext with secret key $(1, s)$.
- **Key Switching**. $\mathsf{ct} \leftarrow \mathsf{KS}_{\mathsf{swk}}(\mathsf{ct}')$. Given a ciphertext ct' which decrypts to a message with a secret key sk' and given a switching key $\mathsf{swk} = \mathsf{swk}_{\mathsf{sk}' \to \mathsf{sk}}$ for another secret key sk, output another ciphertext ct which decrypts to the same message as ct' but with secret key sk. This process is used for eliminating the s^2 term after the multiplication process and also for transforming the result of the rotation process.

– **Rescaling**. $\mathsf{ct}_{\mathrm{rs}} \leftarrow \mathsf{RS}(\ell, \mathsf{ct})$. For a given ciphertext $\mathsf{ct} \in R^2_{Q_\ell}$, output $\mathsf{ct}_{\mathrm{rs}} = \lfloor q_\ell^{-1}\mathsf{ct} \rceil \pmod{Q_{\ell-1}}$, where $Q_\ell = q_\ell \cdot Q_{\ell-1}$. Rescaling process is used for reducing the error by throwing off the LSB of the ciphertext, and at the same time it makes the ciphertext to keep its scaling factor Δ in the similar scale as computation progresses.

2.3 Full CKKS homomorphic encryption scheme and bootstrapping

For a homomorphic encryption scheme to be used in practice, it must support the computation of complicated circuits. It is only possible if it can conduct homomorphic operations with sufficient efficiency, and if it can compute arbitrarily deep circuits. A major problem for homomorphic encryption schemes is that certain amount of noise must have been accumulated when homomorphic operations proceed. As a result, after certain number of homomorphic operations has been done, HE ciphertexts are "deteriorated" so that it is impossible to conduct further homomorphic multiplications on the ciphertexts. In practice, this phenomenon of deterioration reveals itself as the modulus of the ciphertexts being decreased. For example, in the CKKS scheme, any multiplication requires a rescaling operation to keep the noise under control, and each rescaling consumes certain amount of ciphertext modulus.

The notion of *fully homomorphic encryption*, or in short FHE, indicates homomorphic encryption schemes for which the deterioration problem is resolved, and thus FHE allows their users to do multiplication on their ciphertexts indefinitely. Craig Gentry, in [15], proposed an algorithm for a FHE scheme. His method to renew the deteriorated ciphertexts is called *recryption*, and it is conducted by evaluating the decryption circuits homomorphically. He called HE schemes equipped with such a recryption algorithm *bootstrappable*, and we use the terminology *bootstrapping* to indicate an algorithm to transform such deteriorated ciphertexts to refreshed ciphertexts so one can continue to apply homomorphic operations to them. After [15], an enormous amount of contributions was made to improve Gentry's original idea and to apply bootstrapping to existing HE schemes: [2–4, 6, 8–14, 16].

Bootstrapping in the CKKS scheme As decryption circuit of the CKKS scheme involves modular reduction, the bootstrapping in the CKKS scheme is reduced to the problem of performing modular reduction: see [7]. A typical way to achieve bootstrapping in the CKKS homomorphic encryption scheme consists of two steps: we first raise the modulus of the ciphertext to the maximal one, and take a modular reduction modulo the modulus the ciphertext begins with. Suppose that a CKKS ciphertext ct is given with modulus q, encrypting a plaintext $\mathsf{pt} \in R_q$. Mathematically, raising the modulus of ct is equivalent to just treating the same ciphertext ct having a bigger modulus Q. In the perspective of its encrypted plaintext, however, ct with modulus Q now decrypts to $\mathsf{pt} + qI$, where $I \in R$ is a polynomial with sufficiently small size of coefficients. In order to retain the original plaintext, one needs to do modular reduction modulo q

at the plaintext side of ct, and this needs to be done in "homomorphic" way, because the decryption is needed to see the plaintext ct has encrypted, and it is not accessible unless the secret key is known.

One problem is the inconsistency at which such two operations take places: the "raising modulus" or ModRaise operation takes place at the plaintext side (or *coefficient side*) so it transforms ct from encrypting pt to a ciphertext encrypting $\mathsf{pt} + qI$, and the "evaluating modular reduction" or EvalMod step takes places at the message side (or *slot side*), because it consists of homomorphic operations like addition/subtraction, multiplication and rotation. So one needs to move coefficients of the plaintext to slot side and *vice versa*, and these can be achieved using two additional steps CoeffToSlot and SlotToCoeff.

Let ct be a ciphertext encrypting a plaintext pt that is so deteriorated that one cannot proceed further multiplication on ct, i.e. ct has ground level, of level 0 with ground modulus q_0. The ciphertext ct is an input of the bootstrapping process.

- ModRaise. Raise the ciphertext modulus from q_0 to the maximal modulus Q_L, which is determined at SetUp step. We can see that the resulting ciphertext now encrypts $\mathsf{pt} + q_0 I$ where $I \in R$ is a polynomial with small integer coefficients.
- CoeffToSlot. Apply iDFThomomorphically to transfer the additional $q_0 I$ part of the plaintext to the slot side. Being linear map, one can do this with cost of couples of homomorphic multiplications and rotations.
- EvalMod. In this step, conduct the modular reduction $[\cdot]_{q_0}$ at the encrypted message of the ciphertext. Since only algebraic operations (addition/subtraction, multiplication and rotation) are provided in the CKKS scheme, one can do this by approximating the modular reduction function with some polynomial function. After the first feasible algorithmic breakthrough [7], this has been a major topic in homomorphic encryption to improve the quality of such approximation: see [1, 5, 17, 18, 21–23].
- SlotToCoeff. Final step of bootstrapping is to restore message by transferring the message of EvalMod'd ciphertext to its original space, the coefficient side. Naturally this is the inverse process of CoeffToSlot, and thus this can be done with applying DFThomomorphically.

2.4 On decomposition of DFT/iDFT matrices

In CoeffToSlot and SlotToCoeff steps of the bootstrapping algorithm, one computes linear transformations homomorphically on the input ciphertexts, and the linear transformations are represented by the iDFTand DFTmatrices, respectively. Although mathematically they are just matrix-vector multiplications, their homomorphic computations involve homomorphic rotations of various indices and become infeasible while the ciphertext dimension grows exponentially.

In [17], the authors made some clever use of the rich structures of DFTand iDFTto decompose DFTand iDFTinto the products of several sparse block diagonal matrices; it turns out that the homomorphic evaluation of these sequences of

matrix-vector multiplications reduces their homomorphic constant multiplication complexity from $O(n)$ to $O(r \log_r n)$ and their homomorphic rotation complexity from $O(\sqrt{n})$ to $O(\sqrt{r} \log_r n)$ (see §5 in [17]), where n is the number of slots encrypted in the ciphertext and r is the radix of the decomposition. Of course, it does not come for free; the multiplicative depth taken in the CoeffToSlot and SlotToCoeff steps gets larger from 1 to $O(\log_r n)$.

In a nutshell, our new bootstrapping algorithm actually reduces this additional multiplicative depth by degrading CoeffToSlot, in such a way that the precision of the final output of the bootstrapping is on par with the original method (Table 1).

3 Error Analysis of CoeffToSlot

CoeffToSlot is basically a procedure to take a matrix multiplication on the message. To do so in integer arithmetic, the real numbers of the matrix is multiplied by a large number Δ, so called scaling factor, and rounded to integers. This section is devoted to a thorough analysis on the error of the rounding.

There are three types of errors in CoeffToSlot, the rounding error, the key switching error, and the rescaling error. One of the very common technique in CoeffToSlot is lazy rescaling, which delays all the rescalings and rescale once only at the end of CoeffToSlot. This technique enables us to remove the effect of the key switching error. In this sense, we only have to consider the remaining two types of errors. Hence, in this section, we analyze the CoeffToSlot error in the plaintext side, instead of in the ciphertext side.

3.1 Rounding error in matrix multiplication

In this section, we estimate the rounding error which occurs when we homomorphically compute matrix multiplication. Encoding a message m into a plaintext pt involves rounding. To focus on the rounding error, we can think of a plaintext without rounding pt^{raw}.

$$\mathsf{pt} = \lfloor \mathsf{iDFT}(z) \cdot \Delta \rceil, \quad \mathsf{pt}^{raw} = \mathsf{iDFT}(z) \cdot \Delta,$$

Let the rounding error $e = \mathsf{pt}^{raw} - \mathsf{pt}$. Note that each entry of the error $[e]_i$ belongs to $[-\frac{1}{2}, \frac{1}{2})$ and distributed uniformly through the range.

When homomorphically computing a matrix multiplication $z \mapsto Az$, we compute

$$Az = V_1 \odot z_1 + V_2 \odot z_2 + \cdots + V_k \odot z_k$$

where V_i are the diagonals of the matrix A ($k < N$ if A is sparse), and z_i are the rotated copies of z, to match V_i. This computation corresponds to the plaintext computation

$$\mathsf{pt}_{Az} = \mathsf{pt}_{V_1} * \mathsf{pt}_{z_1} + \cdots + \mathsf{pt}_{V_k} * \mathsf{pt}_{z_k}$$

where pt_{V_i} are encoded with same scale factor Δ_A, so that the scale factor of pt_{Az} becomes $\Delta_A \Delta_z$.

Here each V_i is being rounded during the encoding of pt_{V_i}. Using the notations above, we can split the rounding error of matrix multiplication into parts as below. To focus on only the rounding error that occurs when multiplying A, we ignore the rounding error of z which has been occurred before the multiplication. The rounding error e_{Az} can be described as

$$e_{Az} = \mathsf{pt}_{Az}^{raw} - \mathsf{pt}_{Az} = \sum_{i=1}^{k} \mathsf{pt}_{V_i}^{raw} * \mathsf{pt}_{z_i} - \sum_{i=1}^{k} \mathsf{pt}_{V_i} * \mathsf{pt}_{z_i}$$
$$= \sum_{i=1}^{k} (pt_{V_i}^{raw} - \mathsf{pt}_{V_i}) * \mathsf{pt}_{z_i} = \sum_{i=1}^{k} e_{V_i} * \mathsf{pt}_{z_i}.$$

To take a deeper look into each entry of the sum, we introduce Lemma 1 and Lemma 2 as stated below.

Lemma 1 (convolution of pt and its error). *Let $X_0, \cdots, X_{N-1}$ be independent and identically distributed (i.i.d.) random variables following the uniform distribution $U\left(-\frac{1}{2}, \frac{1}{2}\right)$ on range $\left(-\frac{1}{2}, \frac{1}{2}\right)$ and $X \in \mathbb{R}[x] / \left(x^N + 1\right)$ be a random polynomial with its i-th coefficient being X_i, for all i. Suppose $\mathsf{pt} \in \mathbb{R}[x]/(x^N + 1)$ is given. Then $Y = \mathsf{pt} * X \in \mathbb{R}[x]/x^N + 1$ satisfies*

$$\mathbb{E}\left(\|Y\|^2\right) = \frac{N}{12} \|\mathsf{pt}\|^2.$$

Proof. The negative-wrapped convolution is defined as

$$Y_i = \sum_{j=0}^{i} \mathsf{pt}_j X_{i-j} - \sum_{j=i+1}^{N-1} \mathsf{pt}_j \mathsf{X}_{i-j+N}$$

or shortly

$$Y_i = \sum_{j=0}^{N-1} \mathsf{sgn}_{i-j} \mathsf{pt}_j \mathsf{X}_{[i-j]_N}.$$

where $\mathsf{sgn}_x = \begin{cases} 1, & \text{if } x \geq 0 \\ -1, & \text{otherwise} \end{cases}$

Using the additivity of expectation, we have the following.

$$\mathbb{E}\left(\|Y\|^2\right) = \mathbb{E}\left(\sum_{i=0}^{N-1} Y_i^2\right) = \mathbb{E}\left(\sum_{i=0}^{N-1} \left(\sum_{j=0}^{N-1} \mathsf{sgn}_{i-j} \cdot \mathsf{pt}_j \mathsf{X}_{[i-j]_N}\right)^2\right)$$
$$= \mathbb{E}\left(\sum_{i=0}^{N-1} \left(\sum_{j=0}^{N-1} \mathsf{pt}_j^2 \mathsf{X}_{[i-j]_N}^2 + \sum_{j=0}^{N-1} \sum_{k \neq j} \mathsf{sgn}_{i-j} \cdot \mathsf{sgn}_{i-k} \cdot \mathsf{pt}_j \mathsf{pt}_k X_{[i-j]_N} X_{[i-k]_N}\right)\right)$$
$$= \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \mathsf{pt}_j^2 \cdot \mathbb{E}\left(X_{[i-j]_N}^2\right) + \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \sum_{k \neq j} \mathsf{sgn}_{i-j} \cdot \mathsf{sgn}_{i-k} \cdot \mathsf{pt}_j \mathsf{pt}_k \cdot \mathbb{E}\left(X_{[i-j]_N}\right) \mathbb{E}\left(X_{[i-k]_N}\right)$$

Since X_j, X_k are independent and pt_j are fixed,

$$= \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \mathsf{pt}_j^2 \mathbb{E}(X_{[i-j]_N}^2) = \sum_{i=0}^{N-1} ||\mathsf{pt}||_2^2 \cdot \frac{1}{12} = \frac{N}{12} \cdot ||\mathsf{pt}||_2^2$$

The above result with single plaintext can be generalized to the following with multiple plaintexts.

Lemma 2 (Sum of convolutions of $\mathsf{pt}_1, \cdots, \mathsf{pt}_k$ and error). *Suppose $\mathsf{pt}_1, \cdots, \mathsf{pt}_k \in \mathbb{R}[x]/\left(x^N+1\right)$ are given with $\|\mathsf{pt}_1\| = \cdots = \|\mathsf{pt}_k\|$. Let X_{ij} be i.i.d. following $U\left(-\frac{1}{2}, \frac{1}{2}\right)$, $X_i = [X_{i,0}, \cdots, X_{i,N-1}] \in \mathbb{R}[x]/\left(x^N+1\right)$ and $Y_i = X_i * \mathsf{pt}_i$ for $i = 1, \cdots, k$ and $j = 0, \cdots, N-1$. Then we have*

$$\mathbb{E}\left(\|Y_1 + \cdots + Y_k\|^2\right) = \frac{kN}{12} \|\mathsf{pt}_1\|^2 .$$

Proof. Note that

$$\begin{aligned}
\|Y_1 + \cdots + Y_k\|^2 &= \sum_{j=0}^{N-1} \left(\sum_{i=1}^{k} Y_{ij} \right)^2 \\
&= \sum_{j=0}^{N-1} \left(\sum_{i=1}^{k} Y_{ij}^2 + \sum_{i=1}^{k} \sum_{l \neq i} Y_{ij} Y_{lj} \right) \\
&= \sum_{i=1}^{k} \|Y_i\|^2 + \sum_{j=0}^{N-1} \sum_{i=1}^{k} \sum_{l \neq i} Y_{ij} Y_{lj}.
\end{aligned}$$

For the single entry of $Y_{ij}Y_{lj}$, the following holds.

$$\begin{aligned}
\mathbb{E}\left(Y_{ij}Y_{lj}\right) &= \mathbb{E}\left(\left(\sum_{m=0}^{N-1} \mathsf{sgn}_{j-m} \cdot \mathsf{pt}_{im} \mathsf{X}_{i[j-m]_N} \right) \left(\sum_{o=0}^{N-1} \mathsf{sgn}_{j-o} \cdot \mathsf{pt}_{lo} \mathsf{X}_{l[j-o]_N} \right) \right) \\
&= \mathbb{E}\left(\sum_{m=0}^{N-1} \sum_{o=0}^{N-1} \mathsf{sgn}_{j-m} \mathsf{sgn}_{j-o} \cdot \mathsf{pt}_{im} \mathsf{pt}_{lo} \mathsf{X}_{i[j-m]_N} \mathsf{X}_{l[j-o]_N} \right) \\
&= \sum_{m=0}^{N-1} \sum_{o=0}^{N-1} \mathsf{sgn}_{j-m} \mathsf{sgn}_{j-o} \cdot \mathsf{pt}_{im} \mathsf{pt}_{lo} \mathbb{E}\left(\mathsf{X}_{i[j-m]_N} \mathsf{X}_{l[j-o]_N} \right) = 0.
\end{aligned}$$

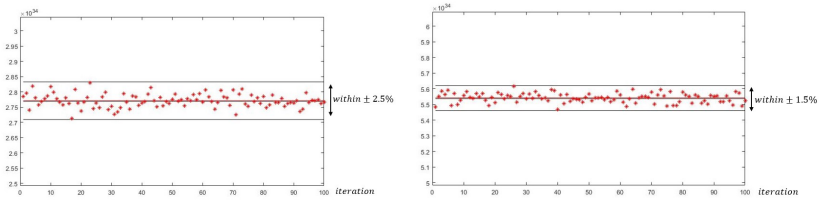
Therefore,

$$\begin{aligned}
\mathbb{E}\left(\|Y_1 + \cdots + Y_k\|^2\right) &= \mathbb{E}\left(\sum_{i=1}^{k} \|Y_i\|^2 + \sum_{j=0}^{N-1} \sum_{i=1}^{k} \sum_{l \neq i} Y_{ij} Y_{lj} \right) \\
&= \sum_{i=1}^{k} \mathbb{E}\left(\|Y_i\|^2\right) + \sum_{j=0}^{N-1} \sum_{i=1}^{k} \sum_{l \neq i} \mathbb{E}\left(Y_{ij} Y_{lj}\right) \\
&= \sum_{i=1}^{k} \mathbb{E}\left(\|Y_i\|^2\right) = \sum_{i=1}^{k} \frac{N}{12} \|\mathsf{pt}_i\|^2 = \frac{kN}{12} \|\mathsf{pt}_1\|^2 .
\end{aligned}$$

Example 1. We verify Lemma 2 by experiment in cases of $N = 2^{15}$ and 2^{16}. k and $[\mathsf{pt}_i]_{i=1,\cdots,k}$ is set to 16 and the ith rotated plaintexts of the plaintext pt, which is the encoding of $z \in \mathbb{C}^{N/2}$ given by $z_i = \frac{\cos(i)}{\sqrt{2}} + \frac{\sin(i)}{\sqrt{-2}}$. The empirical mean of 100 trials and the expectation are quite similar and their relative differences are less than 10^{-3}, as stated in the following table. Figure 1 depicts the trials and show that the deviations of the empirical observation to the estimation of Lemma 2 are less than 2.5% and 1.5%, when $N = 2^{15}$ and 2^{16}, respectively.

N	$\mathbb{E}\left(\lVert Y_1 + \cdots + Y_k \rVert^2\right)$	$\mathbb{E}^{\text{empirical}}\left(\lVert Y_1 + \cdots + Y_k \rVert^2\right)$	$\frac{\mathbb{E}\left(\lVert Y_1+\cdots+Y_k \rVert^2\right) - \mathbb{E}^{\text{empirical}}\left(\lVert Y_1+\cdots+Y_k \rVert^2\right)}{\mathbb{E}\left(\lVert Y_1+\cdots+Y_k^2 \rVert\right)}$
2^{15}	2.7692×10^{34}	2.7718×10^{34}	9.2777×10^{-4}
2^{16}	5.5384×10^{34}	5.5430×10^{34}	8.1469×10^{-4}

Remark 1.



(a) Case of $N = 2^{15}$ (b) Case of $N = 2^{16}$

Fig. 1: 100 trials of $\lVert Y_1 + \cdots + Y_k \rVert^2$ in Example 1 that are fairly close to the expectation $\frac{kN}{12} \lVert \mathsf{pt}_1 \rVert^2$ stated in Lemma 2. The deviation of the trials to the expectation decreases as N increases.

In this section, to estimate the magnitude of error, we eagerly utilize this approximation on $\lVert Y_1 + \cdots + Y_k \rVert$ as below.

$$\lVert Y_1 + \cdots + Y_k \rVert = \sqrt{\lVert Y_1 + \cdots + Y_k \rVert^2} \approx \sqrt{\mathbb{E}\left(\lVert Y_1 + \cdots + Y_k \rVert^2\right)} = \sqrt{\frac{kn}{12}} \lVert \mathsf{pt} \rVert .$$

Recall that the following equality on the matrix multiplication error holds :

$$e_{Az} = \sum_{i=1}^{k} e_{V_i} * \mathsf{pt}_{z_i}.$$

Applying Lemma 2 on e_{Az}, we prove the following Theorem 1.

Theorem 1. *Let $A \in \mathbb{C}^{\frac{N}{2} \times \frac{N}{2}}$ be a matrix with diagonals $V_1, \cdots, V_k$, so that $Az = V_1 \odot z_1 + V_2 \odot z_2 + \cdots + V_k \odot z_k$ where each z_i is a rotation of $z \in \mathbb{C}^{N/2}$.*

Assume that A and z are encoded into plaintexts by scale factor Δ_A and Δ_z respectively. Then

$$\left\| Az - \widetilde{Az} \right\| \approx \sqrt{\frac{kN}{12}} \frac{1}{\Delta_A} \|z\|$$

Proof. Let pt_z be the plaintext encoding z and $\mathsf{pt}_{V_1}, \cdots, \mathsf{pt}_{V_k}$ be the plaintext encoding $V_1, \cdots, V_k$, respectively.

Recall that the rounding error of matrix multiplication $e_{Az} = \mathsf{pt}_{Az}^{raw} - \mathsf{pt}_{Az} = \sum_{i=1}^{k} e_{V_i} * \mathsf{pt}_{z_i}$. Since we can assume that each entry of e_{V_i} follows $U(-\frac{1}{2}, \frac{1}{2})$ and $\left\|\mathsf{pt}_{z_1}\right\| = \cdots = \left\|\mathsf{pt}_{z_k}\right\|$, we can apply Lemma 2 on $\sum_{i=1}^{k} e_{V_i} * \mathsf{pt}_{z_i}$.

$$\left\| \sum_{i=1}^{k} e_{V_i} * \mathsf{pt}_{z_i} \right\| \approx \sqrt{\frac{kN}{12}} \|\mathsf{pt}_z\|$$

Note that the following holds for any $z \in \mathbb{C}^{N/2}$ and $\mathsf{pt} = \mathsf{Encode}(\mathbf{z}; \Delta_z)$.

$$\|z\| \approx \left\| \frac{\mathsf{DFT}\,(\mathsf{pt})}{\Delta_z} \right\| = \frac{1}{\Delta_z} \|\mathsf{DFT}\,(\mathsf{pt})\| = \frac{1}{\Delta_z} \sqrt{\frac{N}{2}} \|\mathsf{pt}\|$$

Therefore,

$$\left\| Az - \widetilde{Az} \right\| = \frac{1}{\Delta_{Az}} \sqrt{\frac{N}{2}} \left\| \mathsf{pt}_{Az - \widetilde{Az}} \right\| = \frac{1}{\Delta_A \Delta_z} \sqrt{\frac{N}{2}} \|\mathsf{pt}_{Az}^{raw} - \mathsf{pt}_{Az}\|$$

$$\approx \frac{1}{\Delta_A \Delta_z} \sqrt{\frac{N}{2}} \sqrt{\frac{kN}{12}} \|\mathsf{pt}_z\| = \frac{1}{\Delta_A} \sqrt{\frac{kN}{12}} \left(\frac{1}{\Delta_z} \sqrt{\frac{N}{2}} \|\mathsf{pt}_z\| \right) \approx \frac{1}{\Delta_A} \sqrt{\frac{kN}{12}} \|z\|$$

For convenience, we denote $\frac{1}{\Delta_A} \sqrt{\frac{kN}{12}}$ as p_A so that the following holds :

$$\left\| Az - \widetilde{Az} \right\| \approx p_A \|z\| .$$

Theorem 1, which is the error analysis of single matrix multiplication, can be applied to the case of homomorphically multiplying the two matrices A and B successively, i.e. $z \mapsto Az \mapsto BAz$. The first type of rounding error will occur during the multiplication of z by A. Let $\widetilde{Az}$ be the actual result of such computation, which contains the rounding error. The second type of rounding error occurs during the multiplication of $\widetilde{Az}$ by B. Let $\widetilde{B\widetilde{Az}}$ be the result of such computation, which contains the rounding error with respect to the matrix multiplication by B. The total error generalized to a series of matrix multiplications in a straightforward manner as follows.

Theorem 2 (Rounding error in serial matrix multiplication). *Let $A_1, \cdots, A_d \in \mathbb{C}^{N/2 \times N/2}$ and let $\mathbf{z} \in \mathbb{C}^{N/2}$. Let p_i be the multiplier in Theorem 1, i.e. $p_i = \frac{1}{\Delta_{A_i}} \sqrt{\frac{k_i \cdot N}{12}}$. Then we have*

$$\left\| A_d \cdots A_1 \mathbf{z} - \widetilde{(A_d \cdots \widetilde{A_1 \mathbf{z}})} \right\| \lesssim \left(\sum_{i=1}^{d} p_i \|A_i\|^{-1} \right) \cdot \prod_{i=1}^{d} \|A_i\| \cdot \|\mathbf{z}\|$$

Proof. We use an induction on d. The base case with $d = 1$ is just Theorem 1. Now suppose the theorem holds up to $d-1$. Write $B := A_d \cdots A_2$. Then we have

$$\left\| A_d \cdots A_1 \mathbf{z} - \widetilde{\left(A_d \cdots \widetilde{A_1 \mathbf{z}}\right)} \right\| \leq \left\| BA_1\mathbf{z} - B\widetilde{A_1\mathbf{z}} \right\| + \left\| B\widetilde{A_1\mathbf{z}} - \widetilde{\left(A_d \cdots \widetilde{A_1\mathbf{z}}\right)} \right\|,$$

by the triangular inequality. Then

$$\left\| BA_1\mathbf{z} - B\widetilde{A_1\mathbf{z}} \right\| \lesssim \|B\| \cdot p_1 \cdot \|\mathbf{z}\|$$

by Theorem 1 and

$$\left\| B\widetilde{A_1\mathbf{z}} - \widetilde{\left(A_d \cdots \widetilde{A_1\mathbf{z}}\right)} \right\| \lesssim \left(\sum_{i=2}^{d} p_i \|A_i\|^{-1} \right) \prod_{i=2}^{d} \|A_i\| \cdot \left\| \widetilde{A_1\mathbf{z}} \right\|$$

by the induction hypothesis. Since $p_1 \ll 1$, we get $\left\| \widetilde{A_1\mathbf{z}} \right\| \leq \|A_1\mathbf{z}\| + \left\| A_1\mathbf{z} - \widetilde{A_1\mathbf{z}} \right\| \leq \|A_1\| \|\mathbf{z}\| + p_1 \|\mathbf{z}\| \approx \|A_1\| \|\mathbf{z}\|$, and hence

$$\left\| A_d \cdots A_1 \mathbf{z} - \widetilde{\left(A_d \cdots \widetilde{A_1\mathbf{z}}\right)} \right\| \lesssim \left(\sum_{i=1}^{d} p_i \|A_i\|^{-1} \right) \cdot \prod_{i=1}^{d} \|A_i\| \cdot \|\mathbf{z}\|,$$

as expected.

Example 2. We empirically verify Theorem 2 in cases of $N = 2^{15}, 2^{16}$. Again, $z \in \mathbb{C}^{N/2}$ is given by $z_i = \frac{\cos(i)}{\sqrt{2}} + \frac{\sin(i)}{\sqrt{-2}}$. To demonstrate the case of CoeffToSlot, we use A the decomposed iDFTmatrices. [17] introduces the decomposition, so that the iDFTmatrix $\frac{1}{N}\overline{U_0^{NR}}^T$ is decomposed into $A_1 \cdot A_2 \cdots A_{\log N - 1}$ and a permutation matrix, where each of A_is have at most 3 diagonals and has norm of $\|A_i\| = \frac{1}{\sqrt{2}}$. z and A are encoded with the scale factor of $\Delta_z = \Delta_A = 2^{50}$ during the homomorphic computation. The following table shows that the observed error of the matrix multiplication is close to and less than the estimate of Theorem 2.

N	$\|A_d \cdots A_1\mathbf{z} - (A_d \cdots (A_1\mathbf{z})^{\sim})^{\sim}\|$	$\left(\sum_{i=1}^{d} p_i \|A_i\|^{-1}\right) \cdot \prod_{i=1}^{d} \|A_i\| \cdot \|\mathbf{z}\|$
2^{15}	7.5697×10^{-10}	9.2196×10^{-9}
2^{16}	2.1414×10^{-9}	2.7939×10^{-8}

3.2 CoeffToSlot

As explained in preliminaries, the linear transform of DFTmatrix maps $\frac{\mathsf{pt}}{\Delta}$ to z, so it holds that $z = [U_0 : U_0\sqrt{-1}](\mathsf{pt}/\Delta)$, where $\sqrt{-1}$ is a complex imaginary unit and U_0 is the DFTmatrix of size $N/2 \times N/2$. Refer to Section 5.1 of [7] for the details. CoeffToSlot is the inversion of the encoding. It calculates the two parts of the plaintext from the message by taking a matrix multiplication and conjugated sums as follows.

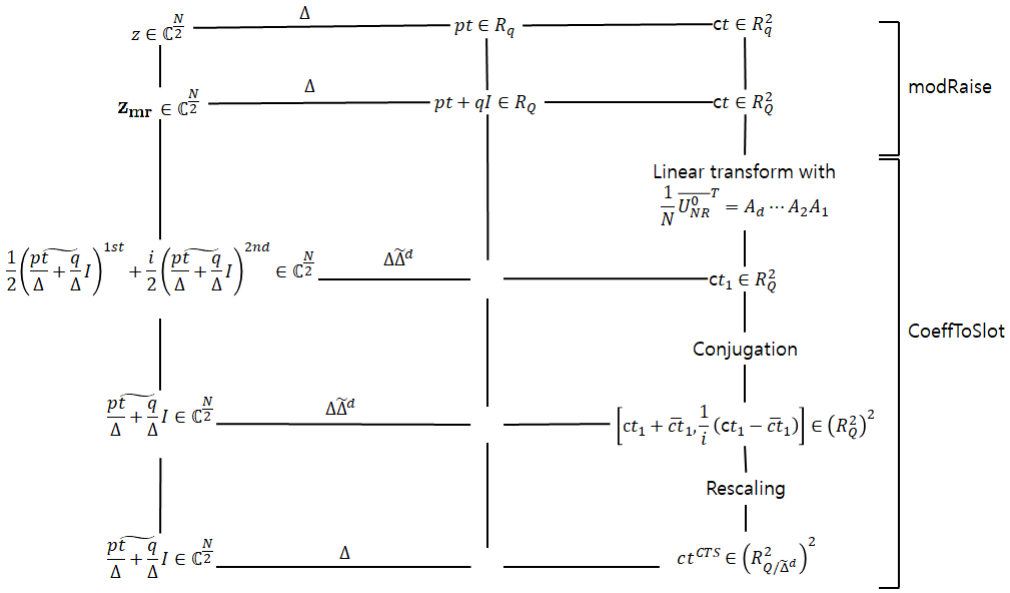


Fig. 2: The diagram illustrates the details of CoeffToSlot procedure. $\widetilde{\Delta}$ is the scale factor to round the real numbers of each matrix $A_i (i = 1, \cdots, d)$ into integers. Because of the rounding errors, the homomorphically calculated message $\widetilde{\frac{\mathsf{pt}}{\Delta}} + \frac{q}{\Delta}I$ does not equal $\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta}I$, but approximates it. The error of the approximation is estimated in theorem 3 as $O\left(\frac{1}{\widetilde{\Delta}} N^{1+\frac{1}{2d}} \frac{q}{\Delta}\right)$.

$$z_1 = \frac{1}{N}\overline{U}_0^T z = \frac{1}{2}\left(\frac{\mathsf{pt}^{1st}}{\Delta} + \frac{\mathsf{pt}^{2nd}}{\Delta}\sqrt{-1}\right)$$

$$z_{CTS}^{1st} = z_1 + \overline{z_1} = \frac{\mathsf{pt}^{1st}}{\Delta}$$

$$z_{CTS}^{2nd} = \frac{1}{\sqrt{-1}}(z_1 - \overline{z_1}) = \frac{\mathsf{pt}^{2nd}}{\Delta}$$

In the process of bootstrapping, CoeffToSlot is applied on the ModRaise'd plaintext, which is known to have form of pt+ qI. The result of CoeffToSlot becomes two plaintexts encoding $z_{cts}^{1st} = \left(\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta}I\right)^{1st}$ and $z_{cts}^{2nd} = \left(\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta}I\right)^{2nd}$.

On the computation of CoeffToSlot, $\frac{1}{N}\overline{U_0}^T$ is a full matrix of size $N/2 \times N/2$ which is a huge burden to compute naively. In [17], the authors utilized its FFT decomposition

$$\frac{1}{N}\overline{U}_0^{NR^T} = \frac{1}{N}V_{logN-1}\cdots V_2 V_1,$$

where each V_i is the matrix of butterfly action having matrix norm $\|V_i\| = \sqrt{2}$ and has up to three diagonal vectors. Each matrix multiplication requires a spending of modulus to scale and round its real numbers into integers. It is customary to group logN-1number of matrix multiplications into fewer number, let us say d(e.g 3 or 4).

$$\frac{1}{N}\overline{U}_0^{NR^T} = A_d \cdots A_2 A_1$$

Let $\widetilde{\Delta}$ be the scale factor to round the real numbers of each matrix A_i into integers. Figure 3 illustrates CoeffToSlot that consists of d number of matrix multiplications and two conjugated sums. The following theorem estimates the size of error in CoeffToSlot.

Theorem 3 (Error of CoeffToSlot). *Let* $\widetilde{\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta}I}$ *be the approximation of* $\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta}I$ *calculated by the CoeffToSlot in figure* (2)*. Then the error* $e = \left(\widetilde{\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta}I}\right) - \left(\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta}I\right)$ *satisfies*

$$\|e\| \lesssim \frac{C_1}{\widetilde{\Delta}} N^{1+\frac{1}{2d}} \frac{q}{\Delta},^{3}$$

where $C_1 = \frac{d\sqrt{(h+1)3^{\left\lceil \frac{\log N-1}{d} \right\rceil}}}{12} \cdot 2^{\frac{1}{2d}}$.

[3] Provided that $\widetilde{\Delta}$ is sufficiently small, we can assume that the rescale error is negligible. Since we are focusing on the case when $\widetilde{\Delta}$ is as small as possible, such assumption is valid.

Proof. The random integer coefficient polynomial I in $\mathsf{pt}+qI$ is known to follow the Irwin-Hall distribution, and each coefficient of $\mathsf{pt}+qI$ follows a normal distribution $N\left(0, \frac{h+1}{12}q^2\right)$ very accurately when the hamming weight h is large enough (e.g. 64 or 128). Then we have $\|\mathsf{pt}+qI\| \simeq \sqrt{\frac{(h+1)N}{12}}q$ and

$$\|z_{mr}\| = \sqrt{\frac{N}{2}}\frac{1}{\Delta}\|\mathsf{pt}+qI\| \simeq \sqrt{\frac{h+1}{24}}N\frac{q}{\Delta},$$

where z_{mr} is the message being encoded into $\mathsf{pt}+qI$ after the step of ModRaise.

The iDFTmatrix $\frac{1}{N}\overline{U}_0^{NR^T}$ splits into $V_{\log N-1}\cdots V_2V_1$, where each V_i has matrix norm of $\sqrt{2}$ and consists of upto 3 diagonal vectors. Merging the matrices into d number of matrices and scaling by $\frac{1}{N}$, we can assume that

$$\|A_d\| = \cdots = \|A_1\| = \left(\frac{\sqrt{2}^{\log N-1}}{N}\right)^{\frac{1}{d}} = 2^{-\frac{1}{2d}}N^{-\frac{1}{2d}},$$

and each A_i consists of up to k diagonal vectors, where

$$k = 3^{\lceil\frac{logN-1}{d}\rceil}.$$

Utilizing the error analysis in Theorem 2 for the matrix multiplication $A_d\cdots A_2A_1$ on z_{mr}, we obtain the estimation on $z_1 = A_d\cdots A_2A_1z = \frac{1}{2}\left(\left(\frac{\mathsf{pt}}{\Delta}+\frac{q}{\Delta}I\right)^{1st}+\left(\frac{\mathsf{pt}}{\Delta}+\frac{q}{\Delta}I\right)^{2nd}\sqrt{-1}\right)$.

$$\begin{aligned}\|\widetilde{z_1}-z_1\| &\lesssim \|z_{mr}\|\sqrt{\frac{kN}{12}}\frac{1}{\widetilde{\Delta}}\|A_1\|\cdots\|A_d\|\left(\frac{1}{\|A_1\|}+\cdots+\frac{1}{\|A_d\|}\right)\\ &\simeq \sqrt{\frac{h+1}{24}}N\frac{q}{\Delta}\sqrt{\frac{kN}{12}}\frac{1}{\widetilde{\Delta}}\left(2^{-\frac{1}{2d}}N^{-\frac{1}{2d}}\right)^{d-1}d\\ &= \frac{d\sqrt{(h+1)\,3^{\lceil\frac{logN-1}{d}\rceil}}}{24}2^{\frac{1}{2d}}\frac{1}{\widetilde{\Delta}}N^{1+\frac{1}{2d}}\frac{q}{\Delta}\end{aligned}$$

$$
\begin{aligned}
\|e\| &= \left\| \left(\frac{\mathsf{pt}}{\Delta} + \widetilde{\frac{q}{\Delta} I} \right) - \left(\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta} I \right) \right\| \\
&= \sqrt{\left\| \left(\frac{\mathsf{pt}}{\Delta} + \widetilde{\frac{q}{\Delta} I} \right)^{1st} - \left(\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta} I \right)^{1st} \right\|^2 + \left\| \left(\frac{\mathsf{pt}}{\Delta} + \widetilde{\frac{q}{\Delta} I} \right)^{2nd} - \left(\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta} I \right)^{2nd} \right\|^2} \\
&= \left\| \left(\left(\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta} I \right)^{1st} + \widetilde{\left(\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta} I \right)^{2nd} \sqrt{-1}} \right) - \left(\left(\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta} I \right)^{1st} + \left(\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta} I \right)^{2nd} \sqrt{-1} \right) \right\| \\
&= \| 2 \left(\widetilde{z_1} - z_1 \right) \| = 2 \left\| \widetilde{z_1} - z_1 \right\| \\
&\lesssim \frac{d \sqrt{(h+1)\, 3^{\left\lceil \frac{logN-1}{d} \right\rceil}}}{12} 2^{\frac{1}{2d}} \frac{1}{\widetilde{\Delta}} N^{1+\frac{1}{2d}} \frac{q}{\Delta} = C_1 \frac{1}{\widetilde{\Delta}} N^{1+\frac{1}{2d}} \frac{q}{\Delta}.
\end{aligned}
$$

Example 3. We provide a proof-of-concept implementation of Theorem 3, at https://github.com/CryptoLabInc/EvalRound. We developed our own source code in C++, which implements the binary version of CKKS bootstrapping. Table 3 describes the parameters used in this example. N denotes the ciphertext dimension, $\log(QP)$ denotes the bit lengths of the largest RLWE modulus, h denotes the hamming weight, λ denotes the security bits, Δ denotes the encoding scale factor, q denotes the base modulus, and d denotes the decomposition number for CoeffToSlot matrix.

Table 3: Parameters for Example 3

Parameter	N	$\log(QP)$	h	λ	Δ	q	d
P1	2^9	2900	128	-	2^{50}	2^{60}	4
P2	2^{13}	2900	128	-	2^{50}	2^{60}	4
P3	2^{17}	2900	128	128	2^{50}	2^{60}	4

C_1 could be computed directly from the statement of Theorem 3, and so are the estimate of $\|z_{mr}\|$ and the bound of $\|e\|$. We checked that $\|z_{mr}\|$ is close to $\|z_{mr}\|^{est}$ and $\|e\|$ is bounded to $\|e\|^{bound}$, as shown in Table 4.

Table 4: Implementation result for Example 3

Parameter	C_1	$\|z_{mr}\|$	$\|z_{mr}\|^{est}$	$\|e\|$	$\|e\|^{bound}$
P1	12.3858	1.2543×10^6	1.1723×10^6	1.93821×10^{-9}	1.25792×10^{-8}
P2	28.6846	1.9509×10^7	1.87575×10^7	1.50875×10^{-7}	9.59557×10^{-7}
P3	37.1574	3.1176×10^8	3.0012×10^8	1.6208×10^{-6}	1.9321×10^{-5}

4 EvalRound instead of EvalMod

Let $[\cdot]_{\frac{q}{\Delta}} : \mathbb{R} \to \mathbb{R}$ be the modular reduction by integer multiple of $\frac{q}{\Delta}$. $\mathsf{EvalMod}$ subsequent to $\mathsf{CoeffToSlot}$ is a homomorphic evaluation of $[\cdot]_{\frac{q}{\Delta}}$, which removes the ambiguity $\frac{q}{\Delta}I$ from $\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta}I$. Let $\mathsf{Mod}_{\frac{q}{\Delta}} : \mathbb{R}^N \to \mathbb{R}^N$ be an element-wise evaluation of $[\cdot]_{\frac{q}{\Delta}}$. We first take a look at $\mathsf{Mod}_{\frac{q}{\Delta}}$, focusing on its role during $\mathsf{EvalMod}$.

We pointed out in the previous section that the homomorphically calculated message $\widetilde{\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta}I}$ does not equal $\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta}I$ mainly because of the rounding error. Let e be the error of the approximation, then we get

$$\widetilde{\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta}I} = \frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta}I + e \text{ and}$$
$$\mathsf{Mod}_{\frac{q}{\Delta}}\left(\widetilde{\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta}I}\right) = \frac{\mathsf{pt}}{\Delta} + e.$$

Note that the rounding error is added to the output of $\mathsf{EvalMod}$ and deteriorates the overall accuracy of bootstrapping. Thus the scale factor $\widetilde{\Delta}$ should be taken as large as Δ to keep the rounding error small, in spite of its consumption of d number of levels. Let $\mathsf{Round}_{\frac{q}{\Delta}} : \mathbb{R}^N \to \mathbb{R}^N$ denote the counterpart of $\mathsf{Mod}_{\frac{q}{\Delta}}$, so that

$$\mathsf{Round}_{\frac{q}{\Delta}}(x) := x - \mathsf{Mod}_{\frac{q}{\Delta}}(x).$$

A standard assumption is $\|z\|_\infty \leq 1$, from which $\left\|\frac{\mathsf{pt}}{\Delta}\right\|_\infty \leq 1$ is derived, and $\frac{q}{\Delta}$ is much larger than $\frac{\mathsf{pt}}{\Delta}$ (e.g. 2^{10}). When the magnitude of e is also negligible compared to $\frac{q}{\Delta}$, the sum $\frac{\mathsf{pt}}{\Delta} + e$ does not change the qI component in $\frac{\mathsf{pt}}{\Delta} + e + \frac{q}{\Delta}I$. In other words,

$$\mathsf{Round}_{\frac{q}{\Delta}}\left(\widetilde{\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta}I}\right) = \frac{q}{\Delta}I = \mathsf{Round}_{\frac{q}{\Delta}}\left(\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta}I\right) \text{ but}$$
$$\mathsf{Mod}_{\frac{q}{\Delta}}\left(\widetilde{\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta}I}\right) = \frac{\mathsf{pt}}{\Delta} + e \neq \frac{\mathsf{pt}}{\Delta} = \mathsf{Mod}_{\frac{q}{\Delta}}\left(\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta}I\right).$$

Let $\mathsf{EvalRound}(\mathsf{ct}) := \mathsf{ct} - \mathsf{EvalMod}(\mathsf{ct})$. The equality of $\mathsf{Round}_{\frac{q}{\Delta}}$ allows us to ignore the rounding error, and the scale factor $\widetilde{\Delta}$ can be taken much smaller than the canonical choice Δ while maintaining the same accuracy. Figure 3 shows the details of our proposed bootstrapping utilizing $\mathsf{EvalRound}$. Our main aim is to take smaller scale factor $\widetilde{\Delta}$ while maintaining the same accuracy as the conventional bootstrapping using $\mathsf{EvalMod}$, and reduce the consumption of modulus bit from Δ^{l+2d} to $\Delta^{l+d}\widetilde{\Delta}^d$, where d is the number of matrix multiplications in $\mathsf{DFT}$and $\mathsf{iDFT}$and l is the number of levels consumed in $\mathsf{EvalMod}$/$\mathsf{EvalRound}$.

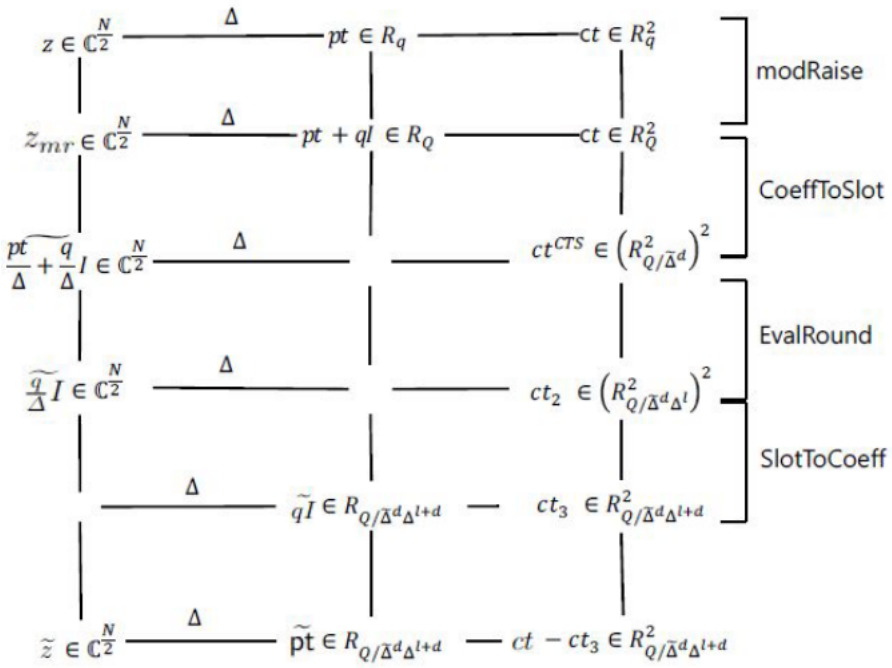


Fig. 3: The proposed bootstrapping utilizing EvalRound : d is the number of matrix multiplications in DFTand iDFT. l is the number of levels consumed in EvalRound, which equals that of EvalMod. The overall modulus bit spent is $\widetilde{\Delta}^d \Delta^{l+d}$. Our main aim is to reduce $\widetilde{\Delta}$, while maintaining the accuracy.

From the identity $\mathsf{Round}_{\frac{q}{\Delta}}(x) = x - \mathsf{Mod}_{\frac{q}{\Delta}}(x)$, $\mathsf{EvalRound}$ can be readily implemented just using one of the successful implementations [5, 18, 21–23] of $\mathsf{EvalMod}$. Referencing one of them, let $\mathsf{EvalMod} : \mathsf{ct} \mapsto \mathsf{ct}_{em}$ be a homomorphic approximation of $\mathsf{Mod}_{\frac{q}{\Delta}}$. Consider the map of end-to-end evaluation of $\mathsf{EvalMod}$, $\mathsf{EvalMod}_z : z \mapsto z_{em}$, or $\mathsf{EvalMod}_z = \psi^{-1} \circ \mathsf{EvalMod} \circ \psi$ where $\psi : z \mapsto \mathsf{ct} = \mathsf{Encrypt} \circ \mathsf{Encode}$. Then $\mathsf{EvalMod}_z$ approximates $\mathsf{Mod}_{\frac{q}{\Delta}}$ with an error bound B^{EvalMod} (e.g. 2^{-20} or 2^{-30}) for each $x \in \mathbb{C}^N$ close to lattice points $\left\{0, \pm\frac{q}{\Delta}, \pm\frac{2q}{\Delta}, \cdots\right\}$ within distance ϵ (e.g. 1), so that

$$\mathsf{dist}\left(x, \left\{0, \pm\frac{q}{\Delta}, \pm\frac{2q}{\Delta}, \cdots\right\}\right) \leq \epsilon \\ \implies \left\|\mathsf{Mod}_{\frac{q}{\Delta}}(x) - \mathsf{EvalMod}_z(x)\right\|_\infty < B^{\mathsf{EvalMod}}, \quad (5)$$

where dist denotes the maximum distance among its elements.

For $x = \widetilde{\frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta} I} = \frac{\mathsf{pt}}{\Delta} + e + \frac{q}{\Delta} I$, its distance to the lattice points is $\frac{\mathsf{pt}}{\Delta} + e$, and we have

$$\left\|\frac{\mathsf{pt}}{\Delta} + e\right\|_\infty \leq \epsilon \implies \left\|\frac{q}{\Delta} I - \mathsf{EvalRound}_z(x)\right\|_\infty < B^{\mathsf{EvalMod}}. \quad (6)$$

From a standard assumption $\|z\|_\infty \leq 1$ and theorem 3, we have the following L^2 estimates.

$$\left\|\frac{\mathsf{pt}}{\Delta}\right\| \leq 1 \\ \|e\| \leq \frac{C_1 N^{1+\frac{1}{2d}} q}{\widetilde{\Delta}\Delta} \quad (7)$$

For any x, it holds that $\frac{1}{\sqrt{N}}\|x\| \leq \|x\|_\infty \leq \|x\|$. The equivalent condition for $\|x\|_\infty = \|x\|$ is that x is a discrete delta function, which is extremely concentrated at one point. When x is not that extreme, we observed in practice that there exists a constant C_2 of moderate size (< 10) that satisfies

$$\left\|\frac{\mathsf{pt}}{\Delta}\right\|_\infty \leq \frac{C_2}{\sqrt{N}}\left\|\frac{\mathsf{pt}}{\Delta}\right\| \text{ and} \\ \|e\|_\infty \leq \frac{C_2}{\sqrt{N}}\|e\|. \quad (8)$$

Example 4. Let the parameters be determined as in Example 3. We checked that the approximations in inequality 8 are valid, as shown in Table 5. $\|\frac{\mathsf{pt}}{\Delta}\|_\infty^{bound}$ and $\|e\|_\infty^{bound}$ denote the estimates of the actual values under the choice of $C_2 = 5$.

Now we analyze the proposed bootstrapping in figure 3 and show the proper range of $\tilde{\Delta}$ that enables the bootstrapping to maintain the same accuracy as $\mathsf{EvalMod}$.

Table 5: Checking the inequality 8

Parameter	$\|\frac{\mathsf{pt}}{\Delta}\|_\infty$	$\|\frac{\mathsf{pt}}{\Delta}\|_\infty^{bound}$	$\|e\|_\infty$	$\|e\|_\infty^{bound}$
P1	8.5345×10^{-2}	1.2273×10^{-1}	4.46107×10^{-10}	2.7796×10^{-9}
P2	2.4474×10^{-2}	3.1755×10^{-2}	3.5204×10^{-9}	2.7234×10^{-8}
P3	7.0179×10^{-3}	7.9874×10^{-3}	2.8059×10^{-8}	2.6684×10^{-7}

Theorem 4 (Proper range of $\widetilde{\Delta}$). *In the bootstrapping proposed in figure 3, we have*

$$\left\| \frac{\mathsf{pt}}{\Delta} - \frac{\widetilde{\mathsf{pt}}}{\Delta} \right\|_\infty < B^{\mathsf{EvalMod}}$$

if $\widetilde{\Delta} \geq \widetilde{\Delta}_{min} = \frac{1}{\frac{\epsilon\sqrt{N}}{C_2}-1} \frac{C_1 N^{1+\frac{1}{2d}} q}{\Delta}$. *Here* C_1 *and* C_2 *are the constants in theorem 3 and the inequality* (8)*, respectively.*

Proof. If $\widetilde{\Delta} \geq \widetilde{\Delta}_{min}$, the inequalities (8) and (7) lead to

$$\begin{aligned}
\left\| \frac{\mathsf{pt}}{\Delta} + e \right\|_\infty &\leq \frac{C_2}{\sqrt{N}} \left(\left\| \frac{\mathsf{pt}}{\Delta} \right\| + \|e\| \right) \\
&\leq \frac{C_2}{\sqrt{N}} \left(1 + \frac{C_1 N^{1+\frac{1}{2d}} q}{\widetilde{\Delta}\Delta} \right) \\
&\leq \frac{C_2}{\sqrt{N}} \left(1 + \left(\frac{\epsilon\sqrt{N}}{C_2} - 1 \right) \right) \\
&= \epsilon.
\end{aligned}$$

From (6), we have

$$\left\| \frac{q}{\Delta} I - \widetilde{\frac{q}{\Delta} I} \right\|_\infty < B^{\mathsf{EvalMod}}.$$

Since $\frac{\widetilde{\mathsf{pt}}}{\Delta} = \frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta} I - \widetilde{\frac{q}{\Delta} I}$[4], we have the desired result,

$$\left\| \frac{\mathsf{pt}}{\Delta} - \frac{\widetilde{\mathsf{pt}}}{\Delta} \right\|_\infty = \left\| \frac{q}{\Delta} I - \widetilde{\frac{q}{\Delta} I} \right\|_\infty < B^{\mathsf{EvalMod}}.$$

Example 5. We validate our main argument, Theorem 4 on the two parameter sets in Example 3. A standard EvalMod is employed. The minimax polynomial approximation of degree 31 is sought for $\frac{q}{2\pi\Delta} \sin\left(\frac{2\pi\Delta x}{q}\right)$ on $\left[-3\frac{q}{\Delta}, 3\frac{q}{\Delta}\right]$. Repeatedly applying half-angle identity, the domain is extended to $\left[-24\frac{q}{\Delta}, 24\frac{q}{\Delta}\right]$. EvalMod is then the composition of the arcsine polynomial of degree three and the polynomial approximation on the extended domain. The following Table 6 reports the constants in Theorem 4.

Table 6: Constants in Theorem 4

Parameter	C_2	ϵ	B^{EvalMod}	$\widetilde{\Delta}_{min}$
P1	5	1	1.7441×10^{-8}	2^{22}
P2	5	1	1.8069×10^{-8}	2^{25}
P3	5	1	2.01234×10^{-8}	2^{29}

Table 7: Results of the conventional and proposed bootstrappings

	Parameter	$\widetilde{\Delta} = \widetilde{\Delta}_{min}$		$\widetilde{\Delta} = \widetilde{\Delta}_{can}$	
		Conventional	Proposed	Conventional	Proposed
$\left\|\frac{\mathsf{pt}}{\Delta} - \frac{\widetilde{\mathsf{pt}}}{\widetilde{\Delta}}\right\|_\infty$	P1	1.0778×10^{-1}	4.3459×10^{-10}	4.7695×10^{-10}	4.4524×10^{-10}
	P2	1.1448×10^{-1}	1.0692×10^{-9}	8.4596×10^{-10}	1.0636×10^{-10}
	P3	5.9085×10^{-2}	1.0979×10^{-8}	1.3277×10^{-8}	5.2136×10^{-8}

A conventional bootstrapping takes $\widetilde{\Delta} = \widetilde{\Delta}_{can} = 2^{60}$ and utilizes $\mathsf{EvalMod}$. Our proposed bootstrapping allows for taking any $\widetilde{\Delta} \geq \widetilde{\Delta}_{min}$.

As stated in Theorem 4, the proposed bootstrapping satisfies $\left\|\frac{\mathsf{pt}}{\Delta} - \frac{\widetilde{\mathsf{pt}}}{\widetilde{\Delta}}\right\|_\infty < B^{\mathsf{EvalMod}}$ in all the cases, while the conventional one does not in the case of $\widetilde{\Delta} = \widetilde{\Delta}_{min}$. Let $\tilde{z}$ be the final output of the proposed bootstrapping in Figure 3. $\|z - \tilde{z}\|_\infty$ represents the precision of bootstrapping. The following Table 8 re-interprets the result in Table 7 in terms of bootstrapping precision.

Table 8: Precision of bootstrapping obtained from conventional and proposed methods

	Parameter	$\widetilde{\Delta} = \widetilde{\Delta}_{min}$		$\widetilde{\Delta} = \widetilde{\Delta}_{can}$	
		Conventional	Proposed	Conventional	Proposed
$\|z - \tilde{z}\|_\infty$	P1	1.11	-26.43	-26.40	-26.41
	P2	3.02	-22.14	-22.20	-22.29
	P3	4.27	-18.23	-18.14	-18.25

Finally, we compute the amount of preserved modulus in Parameter II, as in Table 1. In $\mathsf{CoeffToSlot}$, we use $\widetilde{\Delta} = 2^{29}$ instead of 2^{60}, preserving $(60-29)\cdot 4 = 124$ bits of modulus. Meanwhile, since the input of $\mathsf{SlotToCoeff}$ becomes larger from the size of pt to the size of $\mathsf{pt} + qI$, we should increase the $\mathsf{SlotToCoeff}$ scaling factor by $\frac{q}{\Delta} = 2^{10}$, losing a total of $10 \cdot 4 = 40$ bits of modulus. In sum, we save $124 - 40 = 84$ bits of ciphertext modulus. Since $\Delta = 2^{50}$, it is equivalent to preserving approximately two multiplicative depths.

In the state of the art implementations, we multiply a constant $c > 1$ before bootstrapping and divide by c after bootstrapping. This technique increases the

[4] Here we assumed that $\mathsf{SlotToCoeff}$ error is negligible.

bootstrapping precision by $\log_2(c)$ bits. In particular, we use as large c and possible. Let $c-$bootstrapping be a bootstrapping circuit with such constant c.

Corollary 1. *Let $\mathsf{BTS}_c = (\times c^{-1}) \circ \mathsf{BTS} \circ (\times c)$ be a c-bootstrapping circuit with $\mathsf{EvalMod}$ input bound ϵ and $\mathsf{EvalMod}$ error bound B^{EvalMod}, so that $\left\| \frac{c \cdot \mathsf{pt}}{\Delta} \right\|_\infty \leq \epsilon$. This bootstrapping satisfies*

$$\left\| \frac{\mathsf{pt}}{\Delta} - \frac{\widetilde{\mathsf{pt}}}{\Delta} \right\|_\infty < \frac{1}{c} B^{\mathsf{EvalMod}}.$$

where $\widetilde{\mathsf{pt}}$ is defined to be a final output of BTS_c. Let $\mathsf{BTS}^{\#}$ be an $\mathsf{EvalRound}$ version of BTS. We define $\mathsf{BTS}^{\#}_{c/2} := (\times 2/c) \circ \mathsf{BTS}^{\#} \circ (\times c/2)$. For $\mathsf{BTS}^{\#}_{c/2}$, we have

$$\left\| \frac{\mathsf{pt}}{\Delta} - \frac{\widetilde{\mathsf{pt}}}{\Delta} \right\|_\infty < \frac{2}{c} B^{\mathsf{EvalMod}}.$$

if $\widetilde{\Delta} \geq \widetilde{\Delta}_{min} = \frac{2C_2}{\epsilon\sqrt{N}} \cdot \frac{C_1 N^{1+\frac{1}{2d}} q}{\Delta}$. Here C_1 and C_2 are the constants in theorem 3 and the inequality (8), respectively.

Proof. If $\widetilde{\Delta} \geq \widetilde{\Delta}_{min}$, the inequalities (8) and (7) lead to

$$\begin{aligned} \left\| \frac{c}{2} \cdot \frac{\mathsf{pt}}{\Delta} + e \right\|_\infty &\leq \frac{\epsilon}{2} + \frac{C_2}{\sqrt{N}} \|e\| \\ &\leq \frac{\epsilon}{2} + \frac{C_2}{\sqrt{N}} \cdot \frac{C_1 N^{1+\frac{1}{2d}} q}{\widetilde{\Delta}\Delta} \\ &\leq \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon \\ &= \epsilon. \end{aligned}$$

From (6), we have

$$\left\| \frac{q}{\Delta} I - \frac{\widetilde{q}}{\Delta} I \right\|_\infty < B^{\mathsf{EvalMod}}.$$

Since $\frac{c}{2} \cdot \frac{\widetilde{\mathsf{pt}}}{\Delta} = \frac{c}{2} \cdot \frac{\mathsf{pt}}{\Delta} + \frac{q}{\Delta} I - \frac{\widetilde{q}}{\Delta} I$, we have the desired result,

$$\left\| \frac{\mathsf{pt}}{\Delta} - \frac{\widetilde{\mathsf{pt}}}{\Delta} \right\|_\infty = \frac{2}{c} \left\| \frac{q}{\Delta} I - \frac{\widetilde{q}}{\Delta} I \right\|_\infty < \frac{2}{c} B^{\mathsf{EvalMod}}.$$

Example 6 (Parameter Construction based on a set in [1]). Table 9 describes the overview of the parameter set II proposed in [1]. Here N denotes the ciphertext dimension, $\log(QP)$ denotes the bit length of the largest RLWE modulus, h denotes the hamming weight, depth denotes the number of available multiplication after bootstrapping, Δ denotes the scaling factor for encoding, and q_0 denotes the size of the base modulus.

Using Corollary 1, $\widetilde{\Delta}_{min}$ is computed as 2^{34}. Based on Set II, by applying $\mathsf{EvalRound}$ technique, we can construct a parameter with $\mathsf{depth} = 6$, while losing

Table 9: Overview of Set II in [1].

Set	N	$\log(QP)$	h	depth	Δ	q_0	Bootstrap Precision
II	2^{16}	1547	192	5	2^{45}	2^{60}	-31.5

at most 1 bit of precision. Table 10 describes the new parameter, namely Set II'. Here L denotes the maximum ciphertext level. $\log(q_i)$ and $\log(p_j)$ denote the bit lengths of individual RNS primes and temporary primes for Modulus switching, respectively. Base, Mult, StC, EvalRound, CtS denote the base prime, the multiplication primes, the SlotToCoeff primes, the EvalRound primes, and the CoeffToSlot primes, respectively. The left operand of the dot product denotes the number of primes, and the right operand denotes the bit lengths of primes.

Table 10: Proposed parameter Set II'

II'				
h	N	Δ	$\log(QP)$	L
192	2^{16}	2^{45}	1543	23

$\log(q_i)$				$\log(p_j)$
Base + Mult	StC	EvalRound	CtS	
$60 + 6 \cdot 45$	$3 \cdot 57$	$11 \cdot 60$	$3 \cdot 46$	$4 \cdot 61$

5 Conclusion

In this article, we proposed a method called EvalRound, which is a modification of the EvalMod step in the CKKS bootstrapping. One can reduce the amount of ciphertext modulus consumed in the bootstrapping by modifying the original algorithm with EvalRound. The modulus spent in CoeffToSlot are for the scale factors, each of which has been chosen large enough to take up one level to rescale it down. Smaller scale factors lead to non-negligible rounding errors that are transmitted to EvalMod and eventually corrupt the overall accuracy of the bootstrapping. We introduce a scrutinized analysis that estimates the size of the rounding error with respect to the new scale factors.

Although the rounding error at this step is stuck to the input of the next step EvalMod, it does not pass through EvalRound if it is of a similar size to $\epsilon = \Delta/q$. Thus, when using EvalRound, we can use smaller scale factors for CoeffToSlot compared to the conventional method. In particular, we observed that we can preserve almost half of the modulus consumption in CoeffToSlot. The utilization of the error analysis of CoeffToSlot, reduced scale factors, and EvalRound yielded a saving of approximately two multiplicative levels, in one of practical settings.

Acknowledgements

The research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (Grant No. 2019R1A6A1A11051177 and 2021R1A2C1095703) and Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) [NO.2022-0-01047, Development of statistical analysis algorithm and module using homomorphic encryption based on real number operation].

References

1. 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.) Advances in Cryptology – EUROCRYPT 2021. pp. 587–617. Springer International Publishing, Cham (2021)
2. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. p. 309–325. ITCS '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2090236.2090262, https://doi.org/10.1145/2090236.2090262
3. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-lwe and security for key dependent messages. In: Rogaway, P. (ed.) Advances in Cryptology – CRYPTO 2011. pp. 505–524. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
4. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) lwe. SIAM Journal on Computing **43**(2), 831–871 (2014). https://doi.org/10.1137/120868669, https://doi.org/10.1137/120868669
5. Chen, H., Chillotti, I., Song, Y.: Improved bootstrapping for approximate homomorphic encryption. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2019. pp. 34–54. Springer International Publishing, Cham (2019)
6. Cheon, J.H., Coron, J.S., Kim, J., Lee, M.S., Lepoint, T., Tibouchi, M., Yun, A.: Batch fully homomorphic encryption over the integers. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology – EUROCRYPT 2013. pp. 315–335. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
7. Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: Bootstrapping for approximate homomorphic encryption. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018. pp. 360–384. Springer International Publishing, Cham (2018)
8. Cheon, J.H., Kim, A., Kim, M., Song, Y.: 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. pp. 409–437. Springer (2017)
9. Cheon, J.H., Stehlé, D.: Fully homomophic encryption over the integers revisited. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015. pp. 513–536. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
10. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster packed homomorphic operations and efficient circuit bootstrapping for tfhe. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017. pp. 377–408. Springer International Publishing, Cham (2017)

11. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Tfhe: Fast fully homomorphic encryption over the torus. Journal of Cryptology (2019). https://doi.org/10.1007/s00145-019-09319-x
12. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. pp. 24–43. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
13. Ducas, L., Micciancio, D.: Fhew: Bootstrapping homomorphic encryption in less than a second. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015. pp. 617–640. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
14. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012), https://ia.cr/2012/144
15. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing. p. 169–178. STOC '09, Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1536414.1536440, https://doi.org/10.1145/1536414.1536440
16. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013. pp. 75–92. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
17. Han, K., Hhan, M., Cheon, J.H.: Improved homomorphic discrete fourier transforms and fhe bootstrapping. IEEE Access **7**, 57361–57370 (2019). https://doi.org/10.1109/ACCESS.2019.2913850
18. Han, K., Ki, D.: Better bootstrapping for approximate homomorphic encryption. In: Jarecki, S. (ed.) Topics in Cryptology – CT-RSA 2020. pp. 364–390. Springer International Publishing, Cham (2020)
19. Jutla, C.S., Manohar, N.: Modular lagrange interpolation of the mod function for bootstrapping of approximate he. Cryptology ePrint Archive, Report 2020/1355 (2020), https://ia.cr/2020/1355
20. Jutla, C.S., Manohar, N.: Sine series approximation of the mod function for bootstrapping of approximate he. In: Advances in Cryptology – EUROCRYPT 2022 (2022)
21. 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.) Advances in Cryptology – EUROCRYPT 2021. pp. 618–647. Springer International Publishing, Cham (2021)
22. Lee, J.W., Lee, Y., Kim, Y.S., Kim, Y., No, J.S., Kang, H.: High-precision bootstrapping for approximate homomorphic encryption by error variance minimization. In: Advances in Cryptology – EUROCRYPT 2022 (2022)
23. Lee, Y., Lee, J.W., Kim, Y.S., No, J.S.: Near-optimal polynomial for modulus reduction using l2-norm for approximate homomorphic encryption. IEEE Access **PP**, 1–1 (08 2020). https://doi.org/10.1109/ACCESS.2020.3014369