Bootstrapping for Approximate Homomorphic Encryption

Jung Hee Cheon[1], Kyoohyung Han[1], Andrey Kim[1],
Miran Kim[2], and Yongsoo Song[1,2]

[1] Seoul National University, Seoul, Republic of Korea
{jhcheon, satanigh, kimandrik, lucius05}@snu.ac.kr
[2] University of California, San Diego, United States
{mrkim, yongsoosong}@ucsd.edu

Abstract. This paper extends the leveled homomorphic encryption scheme for an approximate arithmetic of Cheon et al. (ASIACRYPT 2017) to a fully homomorphic encryption, i.e., we propose a new technique to refresh low-level ciphertexts based on Gentry's bootstrapping procedure.

The modular reduction operation is the main bottleneck in the homomorphic evaluation of the decryption circuit. We exploit a scaled sine function as an approximation of the modular reduction operation and present an efficient evaluation strategy. Our method requires only one homomorphic multiplication for each of iterations and so the total computation cost grows linearly with the depth of the decryption circuit.

We also show how to recrypt packed ciphertexts on the RLWE construction with an open-source implementation. For example, it takes 139.8 seconds to refresh a ciphertext that encrypts 128 numbers with 12 bits of precision, yielding an amortized rate of 1.1 seconds per slot.

Keywords. Homomorphic encryption, approximate arithmetic, bootstrapping

1 Introduction

Homomorphic encryption (HE) is a cryptographic scheme that allows us to evaluate an arbitrary arithmetic circuit on encrypted data without decryption. There have been a number of studies [22, 8, 9, 6, 7, 25, 30, 11, 5, 31, 19, 16] to improve the efficiency of HE cryptosystem after Gentry's blueprint [26]. This cryptographic primitive has a number of prospective real-world applications based on the secure outsourcing of computation in public clouds. For example, HE can be a solution to performing the computation of various algorithms on financial, medical, or genomic data without any information leakage [39, 38, 15, 41, 36].

Unfortunately, most of existing HE schemes support the exact arithmetic operations over some discrete spaces (e.g. finite field), so that they are not suitable for many real-world applications which require a floating point operation or real number arithmetic. To be specific, *bitwise encryption* schemes [24, 17] can evaluate a boolean gate with bootstrapping in much shorter time, but it is necessary to evaluate a deep circuit with a number of gates to perform a single arithmetic operation (e.g. addition or multiplication) between high-precision numbers. Moreover, a huge expansion rate of ciphertexts is another issue that stands in the way of the practical use of bitwise encryptions. On the other hand, *word encryption* schemes [7, 30, 6, 25] can encrypt multiple high-precision numbers in a single ciphertext but the rounding operation is difficult to be evaluated since it is not expressed as a small-degree polynomial. Therefore, they require either a plaintext space with an exponentially large bit size on the depth of a circuit, or an expensive computation such as rounding operation and extraction of the most significant bits.

Recently, Cheon et al. [14] proposed a HE scheme for an Arithmetic of Approximate Numbers (called `HEAAN` in what follows) based on the ring learning with errors (RLWE) problem. The main idea is to consider an encryption error as part of a computational error that occurs during approximate computations. For an encryption `ct` of a message m with

a secret key sk, the decryption algorithm $[\langle \mathsf{ct}, \mathsf{sk} \rangle]_q$ outputs an approximate value $m + e$ of the original message with a small error e. The main advantage of HEAAN comes from the *rescaling* procedure for managing the magnitude of plaintexts. It truncates a ciphertext into a smaller modulus, which leads to an approximate rounding of the encrypted plaintext. As a result, it achieved the first linear growth of the ciphertext modulus on the depth of the circuit being evaluated, against the exponential growth in previous word encryption schemes. In addition, the RLWE-based HEAAN scheme has its own packing strategy to encrypt multiple complex numbers in a single ciphertext and perform a parallel computation. However, HEAAN is a *leveled* HE scheme which can only evaluate a circuit of fixed depth. As homomorphic operations progresses, the ciphertext modulus decreases and finally becomes too small to carry out more computations.

In previous literature, Gentry's bootstrapping is the only known method to construct a fully homomorphic encryption (FHE) scheme which allows us to evaluate an arbitrary circuit. Technically, the bootstrapping method can be understood as a homomorphic evaluation of the decryption circuit to refresh a ciphertext for more computations. The HEAAN scheme does not support the modular arithmetic, however, its decryption circuit $[\langle \mathsf{ct}, \mathsf{sk} \rangle]_q$ requires the modular reduction operation, which makes its bootstrapping much harder. Therefore, the bootstrapping of HEAAN can be reduced to a problem that represents the modular reduction function $F(t) = [t]_q$ as a polynomial over the integers (or, complex numbers). One may use the polynomial interpolation of this function over the domain of $t = \langle \mathsf{ct}, \mathsf{sk} \rangle$, but it is a limiting factor for practical implementation due to a huge computational cost of an evaluation.

Our contributions. We present a methodology to refresh ciphertexts of HEAAN and make it bootstrappable for the evaluation of an arbitrary circuit. We take advantage of its intrinsic characteristic - *approximate computation* on encrypted data. Our bootstrapping procedure aims to evaluate the decryption formula approximately and obtain an encryption of the original message in a large ciphertext modulus. Hence, we find an approximation of the modular reduction function that can be evaluated efficiently using arithmetic operations of HEAAN. The approximation error should be small enough to maintain the precision of an input plaintext.

We first note that the modular reduction function $F(t) = [t]_q$ is the identity nearby zero and periodic with period q. If $t = \langle \mathsf{ct}, \mathsf{sk} \rangle$ is close to a multiple of the ciphertext modulus q (or equivalently, if the encrypted plaintext $m = [t]_q$ is small compared to q), then a trigonometric function can be a good approximation to the modular reduction. Namely, the decryption formula of HEAAN can be represented using the following scaled *sine* function as

$$[\langle \mathsf{ct}, \mathsf{sk} \rangle]_q = \frac{q}{2\pi} \cdot \sin\left(\frac{2\pi}{q} \cdot \langle \mathsf{ct}, \mathsf{sk} \rangle\right) + O(\epsilon^3 \cdot q),$$

when $|[\langle \mathsf{ct}, \mathsf{sk} \rangle]_q| \leq \epsilon \cdot q$. Hence we may use this analytic function instead of the modular reduction in the decryption formula.

Now our goal is to homomorphically evaluate the trigonometric function $\frac{q}{2\pi} \cdot \sin\left(\frac{2\pi}{q} \cdot t\right)$ with an input $t = \langle \mathsf{ct}, \mathsf{sk} \rangle$, which is bounded by Kq for some constant $K = O(\lambda)$ with λ the security parameter. We can consider the Taylor polynomial as an approximation to the trigonometric function, but its degree should be at least $O(Kq)$ to make an error term small enough on the interval $(-Kq, Kq)$. The evaluation of polynomial can be done in $O(\sqrt{Kq})$ homomorphic multiplications with Paterson-Stockmeyer method [40], but this complexity of recryption grows exponentially with the depth $L = O(\log q)$ of the decryption circuit - which is still quite substantial.

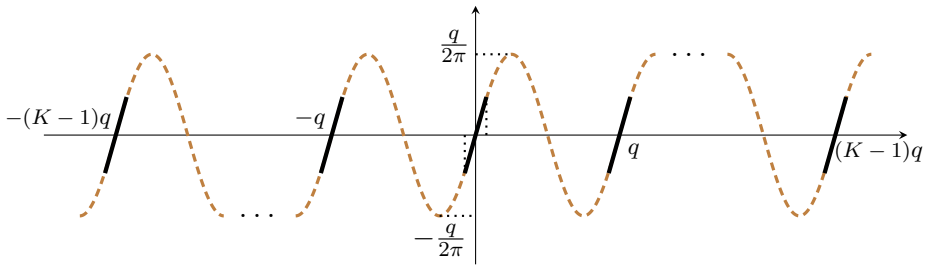

Fig. 1. Modular reduction and scaled sine functions

We suggest an evaluation strategy of the trigonometric function to reduce its computation cost by exploiting the following double-angle formulas:

$$\begin{cases} \cos(2\theta) = \cos^2\theta - \sin^2\theta, \\ \sin(2\theta) = 2\cos\theta \cdot \sin\theta, \end{cases}$$

which means that we can obtain some approximate values of $\cos(2\theta)$ and $\sin(2\theta)$ from approximate values of $\cos\theta$ and $\sin\theta$. In our bootstrapping process, we first compute the Taylor expansions of $\cos\left(\frac{2\pi}{q}\cdot\frac{t}{2^r}\right)$ and $\sin\left(\frac{2\pi}{q}\cdot\frac{t}{2^r}\right)$ of a small degree $d_0 = O(1)$ for some $r = O(\log(Kq))$. Then we use the doubling-angle formulas r times recursively to get an approximate value of $\sin\left(\frac{2\pi}{q}\cdot t\right)$. In the case of the RLWE-based construction, this evaluation can be even more simplified by encrypting the complex exponentiation $\exp(i\theta) = \cos\theta + i\cdot\sin\theta$ and adapting the identity $\exp(i\cdot 2\theta) = (\exp(i\cdot\theta))^2$.

Results. Our bootstrapping technique for `HEAAN` is a new cryptographic primitive for FHE mechanisms, which yields the first word encryption scheme for approximate arithmetic. For a ciphertext ct with a modulus q, our bootstrapping procedure generates a ciphertext ct' with a larger ciphertext modulus $Q \gg q$, satisfying the condition $[\langle \mathsf{ct}', \mathsf{sk}\rangle]_Q \approx [\langle \mathsf{ct}, \mathsf{sk}\rangle]_q$ while an error is kept small enough not to destroy the significant digits of a plaintext. The output ciphertext will have a sufficiently large modulus compared to a plaintext, thereby enabling further computation on the ciphertext. In addition, our approximation to a trigonometric function and efficient evaluation strategy reduce the complexity of the evaluation down to $O(L)$ homomorphic multiplications for the depth $L = O(\log q)$ of the decryption circuit.

We also give an open-source implementation [12] to demonstrate the performance of our bootstrapping method. It contains some optimization techniques including the linear transformation method of [33] for the recryption over the packed ciphertexts. When we want to preserve 12 bits of precision, our bootstrapping on a single-slot ciphertext takes about 26.6 seconds. We also optimize the linear transforms for sparsely packed ciphertexts and it takes about 139.8 seconds to recrypt a ciphertext that encrypts 128 complex numbers in plaintext slots, yielding an amortized rate of 1.1 seconds per slot.

Implications of our bootstrapping method. The main feature of approximate arithmetic is that every number contains an error of which could increase during computation. The precision of a number is reduced by approximately one bit after multiplication and finally we may not extract any meaningful information from the computation result if the depth of a circuit is larger than the bit precision of the input data. On the other hand, our bootstrapping procedure is to refresh ciphertexts and then perform further computation on encrypted data. This concept of an *unlimited* computation may seem a contradiction to the property of finite precision in the approximate arithmetic.

However, it turns out to be better for real-world applications that have a property of negative feedback or stability. For example, a cyber-physical system (CPS) is a compromised

mechanism of physical and computational components. A computational element commutes with the sensors and every signal contains a small error. One can guarantee the correctness of CPS only when it is stable because an error is reduced by negative feedback to the input. Another example is the gradient descent method, which is the most widely used algorithm to perform optimization. It has a number of applications in machine learning such as logistic regression and neural networks. It computes the gradient of a point and moves it closer to an optimal point, which reduces the effects of perturbations in the output.

As in the examples above, we do not have to worry about the precision of numbers when the overall system is stable. In fact, there are some proof-of-concept implementations about the secure control of CPS [35] and secure logistic regression using biomedical data [37]. We expect that our bootstrapping process can be applied to these real-world applications.

Related works. There have been several attempts to carry out an approximate arithmetic using HE. Downlin et al. [23] (see also [21, 4]) described a method to transform a real number into a polynomial with small and sparse coefficients to reduce the required size of a plaintext modulus. Costache et al. [20] suggested a similar encoding method with [14] to evaluate the discrete Fourier transformation efficiently, but a ciphertext could encrypt only one value. Chen et al. [10] uses a technique of [34] to encrypt a single high-precision number. However, they still have some problems: (1) the coefficients and the degree of encoded polynomial grow exponentially with the depth of a circuit and (2) there is no known result to achieve an FHE scheme because a polynomial should be re-encoded to be represented with a smaller degree and coefficients for more computations and the bootstrapping method of [33] is not enough for this functionality.

The original Gentry's bootstrapping technique was implemented by Gentry and Halevi [27], which took a half hour to recrypt a single bit ciphertext. Gentry et al. [28] represented the decryption circuit of RLWE-based HE with a lower depth circuit using a special modulus space. The Halevi-Shoup FHE implementation [33] reported a recryption time of approximately six minutes per slot. Meanwhile, Ducas and Micciancio [24] proposed the FHEW scheme that bootstraps a single-bit encryption in less than a second based on the framework of [2]. Chillotti et al. [17] obtained a speed up to less than 0.1 seconds. The following works [18, 3] improved the performance by using the evaluation of a look-up table before bootstrapping. However, the size of an input plaintext of bootstrapping is very limited since it is related to the ring dimension of an intermediate Ring GSW scheme. In addition, a huge expansion rate of ciphertexts is still an open problem in bitwise encryption schemes.

The previous FHE schemes evaluate the exact decryption circuit using the structure of a finite field or a polynomial ring in bootstrapping algorithm. The evaluation of an arbitrary polynomial of degree d requires $O(\sqrt{d})$ homomorphic multiplications, but Halevi and Shoup [33] used a polynomial with the lifting property to reduce the computational cost of bootstrapping. They used a recursive algorithm to extract some digits in an encrypted state, so the number of homomorphic multiplications for bootstrapping was reduced down to $O(\log^2 d)$. Contrary to the work of Halevi and Shoup, we find an approximate decryption circuit using a trigonometric function and suggest an even simpler recursive algorithm. As a result, our algorithm only requires $O(\log d)$ number of homomorphic multiplications, which results in an enhanced performance.

Road-map. Section 2 briefly introduces notations and some preliminaries about algebra. We also review the `HEAAN` scheme of Cheon et al. [14]. Section 3 explains our simplified decryption formula by using a trigonometric function. In Section 4, we recall the ciphertext packing method of `HEAAN` and describe a linear transformation on packed ciphertexts. In Section 5, we present our bootstrapping technique with a precise noise estimation. In Section 6, we implement the recryption procedure based on the proposed method and discuss the performance results.

2 Preliminaries

The binary logarithm will be simply denoted by $\log(\cdot)$. We denote vectors in bold, e.g. $\mathbf{a}$, and every vector in this paper is a column vector. For a $n_1 \times m$ matrix A_1 and a $n_2 \times m$ matrix A_2, $(A_1; A_2)$ denotes the $(n_1 + n_2) \times m$ matrix obtained by concatenating matrices A_1 and A_2 in a vertical direction.

We denote by $\langle \cdot, \cdot \rangle$ the usual dot product of two vectors. For a real number r, $\lfloor r \rceil$ denotes the nearest integer to r, rounding upwards in case of a tie. For an integer q, we identify $\mathbb{Z} \cap (-q/2, q/2]$ as a representative of $\mathbb{Z}_q$ and use $[z]_q$ to denote the reduction of the integer z modulo q into that interval. We use $x \leftarrow D$ to denote the sampling x according to distribution D. The uniform distribution over a finite set S is denoted by $U(S)$. We let λ denote the security parameter throughout the paper: all known valid attacks against the cryptographic scheme under scope should take $\Omega(2^\lambda)$ bit operations.

2.1 Cyclotomic Ring

For a positive integer M, let $\Phi_M(X)$ be the M-th cyclotomic polynomial of degree $N = \phi(M)$. Let $\mathcal{R} = \mathbb{Z}[X]/(\Phi_M(X))$ be the ring of integers of a number field $\mathbb{Q}[X]/(\Phi_M(X))$. We write $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ for the residue ring of $\mathcal{R}$ modulo an integer q. An arbitrary element of the set $\mathcal{P} = \mathbb{R}[X]/(\Phi_M(X))$ will be represented as a polynomial $a(X) = \sum_{j=0}^{N-1} a_j X^j$ of degree strictly less than N and identified with its coefficients vector $\boldsymbol{a} = (a_0, \ldots, a_{N-1}) \in \mathbb{R}^N$. We define $\|a\|_\infty$ and $\|a\|_1$ by the relevant norms on the coefficients vector $\boldsymbol{a}$.

Write $\mathbb{Z}_M^* = \{x \in \mathbb{Z}_M : \gcd(x, M) = 1\}$ for the multiplicative group of units in $\mathbb{Z}_M$. Recall that the canonical embedding of $a(X) \in \mathbb{Q}[X]/(\Phi_M(X))$ into $\mathbb{C}^N$ is the vector of evaluations of $a(X)$ at the M-th primitive roots of unity. We use its natural extension σ to $\mathcal{P}$, defined by $\sigma(a) = (a(\zeta^j))_{j \in \mathbb{Z}_M^*}$ for $\zeta = \exp(2\pi i/M)$. Its ℓ_∞-norm is called the *canonical embedding norm*, denoted by $\|a\|_\infty^{\mathsf{can}} = \|\sigma(a)\|_\infty$.

2.2 Homomorphic Encryption for Arithmetic of Approximate Numbers

HE is one of the prospective cryptographic primitives for secure outsourcing computation without information leakage. However, an inefficiency of real number computation is one of the main obstacles to apply HE schemes in real-world applications. Recently Cheon et al. [14] proposed a method to construct the HE scheme for approximate arithmetic, called `HEAAN`. Their scheme supports an efficient rounding operation of encrypted plaintext as well as basic arithmetic operations. This subsection gives a concrete description of the RLWE-based `HEAAN` scheme.

For a real $\sigma > 0$, $\mathcal{DG}(\sigma^2)$ denotes a distribution over $\mathbb{Z}^N$ which samples its components independently from the discrete Gaussian distribution of variance σ^2. For an positive integer h, $\mathcal{HWT}(h)$ denotes a uniform distribution over the set of signed binary vectors in $\{\pm 1\}^N$ whose Hamming weight is exactly h. For a real $0 \le \rho \le 1$, the distribution $\mathcal{ZO}(\rho)$ draws each entry in the vector from $\{0, \pm 1\}$, with probability $\rho/2$ for each of -1 and $+1$, and probability being zero $1 - \rho$.

- $\underline{\mathsf{KeyGen}(1^\lambda)}$.
  - For a base p and an integer L, let $q_\ell = p^\ell$ for $\ell = 1, \ldots, L$. Given the security parameter λ, choose a power-of-two M, an integer h, an integer P, and a real number $\sigma > 0$ for an RLWE problem that achieves λ-bit of security level.
  - Sample $s \leftarrow \mathcal{HWT}(h)$, $a \leftarrow U(\mathcal{R}_{q_L})$ and $e \leftarrow \mathcal{DG}(\sigma^2)$. Set the secret key as $\mathsf{sk} \leftarrow (1, s)$ and the public key as $\mathsf{pk} \leftarrow (b, a) \in \mathcal{R}_{q_L}^2$ where $b \leftarrow -as + e \pmod{q_L}$.

- $\underline{\mathsf{KSGen}_{\mathsf{sk}}(s')}$. For $s' \in \mathcal{R}$, sample $a' \leftarrow U(\mathcal{R}_{P \cdot q_L})$ and $e' \leftarrow \mathcal{DG}(\sigma^2)$. Output the switching key as $\mathsf{swk} \leftarrow (b', a') \in \mathcal{R}^2_{P \cdot q_L}$ where $b' \leftarrow -a's + e' + Ps' \pmod{P \cdot q_L}$.
  - Set the evaluation key as $\mathsf{evk} \leftarrow \mathsf{KSGen}_{\mathsf{sk}}(s^2)$.
- $\underline{\mathsf{Enc}_{\mathsf{pk}}(m)}$. For $m \in \mathcal{R}$, sample $v \leftarrow \mathcal{ZO}(0.5)$ and $e_0, e_1 \leftarrow \mathcal{DG}(\sigma^2)$. Output $v \cdot \mathsf{pk} + (m + e_0, e_1) \pmod{q_L}$.
- $\underline{\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct})}$. For $\mathsf{ct} = (c_0, c_1) \in \mathcal{R}^2_{q_\ell}$, output $m = c_0 + c_1 \cdot s \pmod{q_\ell}$.
- $\underline{\mathsf{Add}(\mathsf{ct}_1, \mathsf{ct}_2)}$. For $\mathsf{ct}_1, \mathsf{ct}_2 \in \mathcal{R}^2_{q_\ell}$, output $\mathsf{ct}_{\mathsf{add}} \leftarrow \mathsf{ct}_1 + \mathsf{ct}_2 \pmod{q_\ell}$.
- $\underline{\mathsf{Mult}_{\mathsf{evk}}(\mathsf{ct}_1, \mathsf{ct}_2)}$. For $\mathsf{ct}_1 = (b_1, a_1), \mathsf{ct}_2 = (b_2, a_2) \in \mathcal{R}^2_{q_\ell}$, let $(d_0, d_1, d_2) = (b_1 b_2,\ a_1 b_2 + a_2 b_1, a_1 a_2) \pmod{q_\ell}$. Output $\mathsf{ct}_{\mathsf{mult}} \leftarrow (d_0, d_1) + \lfloor P^{-1} \cdot d_2 \cdot \mathsf{evk} \rceil \pmod{q_\ell}$.
- $\underline{\mathsf{RS}_{\ell \to \ell'}(\mathsf{ct})}$. For a ciphertext $\mathsf{ct} \in \mathcal{R}^2_{q_\ell}$ at level ℓ, output $\mathsf{ct}' \leftarrow \lfloor p^{\ell' - \ell} \cdot \mathsf{ct} \rceil \pmod{q_{\ell'}}$. We will omit the subscript $(\ell \to \ell')$ when $\ell' = \ell - 1$.

The native plaintext space of HEAAN can be understood as the set of polynomials $m(X)$ in $\mathbb{Z}[X]/(\Phi_M(X))$ such that $\|m\|^{\mathsf{can}}_\infty < q/2$. For convenience, we allow an arbitrary element of $\mathcal{P} = \mathbb{R}[X]/(\Phi_M(X))$ as a plaintext polynomial, so that a ciphertext $\mathsf{ct} = (c_0, c_1) \in \mathcal{R}^2_{q_\ell}$ at level ℓ will be called an encryption of $m(X) \in \mathcal{P}$ with an error bound B if it satisfies $\langle \mathsf{ct}, \mathsf{sk} \rangle = m + e \pmod{q_\ell}$ for some polynomial $e(X) \in \mathcal{P}$ satisfying $\|e\|^{\mathsf{can}}_\infty \le B$. The set $\mathcal{P} = \mathbb{R}[X]/(\Phi_M(X))$ can be identified with the complex coordinate space $\mathbb{C}^{N/2}$ using a ring isomorphism. This decoding map allows us to encrypt at most $(N/2)$ numbers in a single ciphertext and carry out parallel operations in a Single Instruction Multiple Data (SIMD) manner. A simple description of the packing method will be described in Section 4.1.

We will make the use of the following lemmas from [14] for noise estimation. We adapt some notations from [14], defining the constants B_{ks} and B_{rs}.

Lemma 1 ([14, Lem. 1]). *Let* $\mathsf{ct} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)$ *be an encryption of* $m \in \mathcal{R}$. *Then* $\langle \mathsf{ct}, \mathsf{sk} \rangle = m + e \pmod{q_L}$ *for some* $e \in \mathcal{R}$ *satisfying* $\|e\|^{\mathsf{can}}_\infty \le B_{\mathsf{clean}}$ *for* $B_{\mathsf{clean}} = 8\sqrt{2}\sigma N + 6\sigma\sqrt{N} + 16\sigma\sqrt{hN}$.

Lemma 2 ([14, Lem. 2]). *Let* $\mathsf{ct}' \leftarrow \mathsf{RS}_{\ell \to \ell'}(\mathsf{ct})$ *for a ciphertext* $\mathsf{ct} \in \mathcal{R}^2_{q_\ell}$. *Then* $\langle \mathsf{ct}', \mathsf{sk} \rangle = \frac{q_{\ell'}}{q_\ell} \langle \mathsf{ct}, \mathsf{sk} \rangle + e \pmod{q_{\ell'}}$ *for some* $e \in \mathcal{P}$ *satisfying* $\|e\|^{\mathsf{can}}_\infty \le B_{\mathsf{rs}}$ *for* $B_{\mathsf{rs}} = \sqrt{N/3} \cdot (3 + 8\sqrt{h})$.

Lemma 3 ([14, Lem. 3]). *Let* $\mathsf{ct}_{\mathsf{mult}} \leftarrow \mathsf{Mult}_{\mathsf{evk}}(\mathsf{ct}_1, \mathsf{ct}_2)$ *for two ciphertexts* $\mathsf{ct}_1, \mathsf{ct}_2 \in \mathcal{R}^2_{q_\ell}$. *Then* $\langle \mathsf{ct}_{\mathsf{mult}}, \mathsf{sk} \rangle = \langle \mathsf{ct}_1, \mathsf{sk} \rangle \cdot \langle \mathsf{ct}_2, \mathsf{sk} \rangle + e_{\mathsf{mult}} \pmod{q_\ell}$ *for some* $e \in \mathcal{R}$ *satisfying* $\|e_{\mathsf{mult}}\|^{\mathsf{can}}_\infty \le B_{\mathsf{mult}}(\ell)$ *for* $B_{\mathsf{ks}} = 8\sigma N/\sqrt{3}$ *and* $B_{\mathsf{mult}}(\ell) = P^{-1} \cdot q_\ell \cdot B_{\mathsf{ks}} + B_{\mathsf{rs}}$.

A rescaling (rounding) error is the smallest error type of homomorphic operations. The least digits of a plaintext is destroyed by some error after multiplication or rescaling, so its significand should be placed in higher digits not to lose the precision of the resulting plaintext.

3 Decryption Formula over the Integers

The goal of bootstrapping is to refresh a ciphertext and keep computing on encrypted data. Recall that HEAAN supports arithmetic operations on a characteristic zero plaintext space such as $\mathbb{C}$ However, its decryption formula consists of two steps: the inner product $t = \langle \mathsf{ct}, \mathsf{sk} \rangle$ over the integers and the modular reduction $m = [t]_q$. We therefore have to express this decryption formula efficiently using homomorphic operations provided in the HEAAN scheme.

The main difficulty comes from the fact that the reduction modular q function $F(t) = [t]_q$ is not represented as a small-degree polynomial. A naive approach such as the polynomial interpolation causes a huge degree, resulting in a large parameter size and an expensive computational cost for bootstrapping process. Instead, we reduce the required circuit depth and the evaluation complexity by exploiting a polynomial approximation of the decryption formula and taking advantage of approximate arithmetic.

3.1 Approximation of the Modular Reduction Function

Let ct be a ciphertext relative to a secret key sk and a modulus q. Since sk is sampled from a small distribution, the size of its decryption structure $t = \langle \mathsf{ct}, \mathsf{sk} \rangle$ is bounded by Kq for some fixed constant K. So we can say that the decryption formula of `HEAAN` is defined on the set $\mathbb{Z} \cap (-Kq, Kq)$ and it maps an arbitrary integer $t \in \mathbb{Z} \cap (-Kq, Kq)$ to the reduction modular q.

It is infeasible to find a good approximation of the modular reduction function since it is not continuous. We first assume that a message m of an input ciphertext is still much smaller than a ciphertext modulus q, so that $t = \langle \mathsf{ct}, \mathsf{sk} \rangle$ can be expressed as $qI + m$ for some I and m such that $|I| < K$ and $|m| \ll q$. This assumption is reasonable because one can start the bootstrapping procedure on a ciphertext before its modulus becomes too small. Then the modular reduction $F(t) = [t]_q$ on a restricted domain becomes a *piecewise linear* function (see Fig. 1). We point out that this function is the identity near zero and periodic, so it looks like a part of the scaled sine

$$S(t) = \frac{q}{2\pi} \sin\left(\frac{2\pi t}{q}\right).$$

Note that it gives a good approximation to the piecewise linear function when an input value $t = qI + m$ is close to a multiple of q. Specifically, an error between $F(t)$ and $S(t)$ is bounded by

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

which is equivalently $O(1)$ when $m = O(q^{2/3})$.

3.2 Homomorphic Evaluation of the Complex Exponential Function

As discussed before, the scaled sine function $S(t)$ is a good approximation of the reduction modulo q. However, this function cannot be evaluated directly using HE since it is not a polynomial function. The goal of this subsection is to explain how to approximately and efficiently evaluate this trigonometric function based on `HEAAN`.

We may consider the Taylor polynomial $\frac{q}{2\pi} \sum_{j=0}^{d-1} \frac{(-1)^j}{(2j+1)!} \left(\frac{2\pi t}{q}\right)^{2j+1}$ of $S(t)$. The size of error converges to zero very rapidly as the degree grows, i.e., an error between $S(t)$ and its Taylor polynomial of degree $2d$ is bounded by $\frac{q}{2\pi} \cdot \frac{1}{(2d+1)!} (2\pi K)^{2d+1}$ when $|t| < Kq$, and it becomes small enough when the degree of the Taylor polynomial is $O(Kq)$. However, despite its high precision, this naive method has an ineffective problem in practice. The complexity grows exponentially with the depth of a circuit, e.g. $O(\sqrt{d})$ using the Paterson-Stockmeyer algorithm [40] for an evaluation of a degree-d polynomial.

Instead, we can reduce the computational cost by exploiting the following double-angle formulas: $\cos(2\theta) = \cos^2\theta - \sin^2\theta$ and $\sin(2\theta) = 2\cos\theta \cdot \sin\theta$. From approximate values of trigonometric functions in a small domain, we extend to find good approximations of the sign function on a wider (doubled) range. In particular, the RLWE-based `HEAAN` scheme can encrypt the complex numbers, so that the evaluation algorithm can be more simplified using the complex exponential function. Specifically, we use the identities

$$\begin{cases} \exp(i\theta) = \cos\theta + i \cdot \sin\theta, \\ \exp(2i\theta) = (\exp(i\theta))^2, \end{cases}$$

and the error growth from squaring can be bounded by about one bit since $(\exp(i\theta) \pm \epsilon)^2 \approx \exp(2i\theta) \pm 2\epsilon$.

We take the Taylor polynomial of a small degree $d_0 \geq 1$ as a high-precision approximation of the complex exponential function within a small range. Then we perform the squaring operation repeatedly to get an approximation of the complex exponential function over the desirable domain. Note that we multiply a scale factor of Δ to prevent the precision loss and divide the intermediate ciphertexts by a constant Δ using the rescaling procedure of `HEAAN`.

The use of the complex exponential function has another advantage in error analysis. When we consider the RLWE-based `HEAAN` scheme, small *complex* errors are added to plaintext slots during encryption, evaluation, rescaling and slot permutation. Therefore, we have only one constraint such that a decryption formula should be tolerant of small complex errors. Another advantage of our method comes from the fact that the complex exponential function is analytic with a bounded derivative over the whole complex plane, and therefore an error does not blow up by the decryption formula. The whole procedure is explicitly described as follows.

A value $t \in (-Kq, Kq)$ is given as an input of the decryption formula.

1. Consider the complex exponential function of $\exp\left(\frac{2\pi i t}{2^r \cdot q}\right)$ and compute its (scaled) Taylor expansion as
$$P_0(t) = \Delta \cdot \sum_{k=0}^{d_0} \frac{1}{k!} \left(\frac{2\pi i t}{2^r \cdot q}\right)^k$$
of degree $d_0 \geq 1$.
2. For $j = 0, 1, \ldots, r-1$, repeat the squaring $P_{j+1}(t) \leftarrow \Delta^{-1} \cdot (P_j(t))^2$.
3. Return $P_r(t)$.

The degree d_0 of the initial Taylor polynomial, the scaling factor of Δ, and the number r of the iterations (squaring) are determined by the following noise analysis. Since the size of the initial input $(2\pi t)/(2^r \cdot q)$ of the complex exponential function has a small upper bound $(2\pi K/2^r)$, even the Taylor polynomial of a small degree d_0 can be a good approximation to the complex exponential function $\exp\left(\frac{2\pi i t}{2^r \cdot q}\right)$. From the above observation, the output $P_r(t)$ is a polynomial of degree $d_r = d_0 \cdot 2^r$ and it is an approximation of $E(t) := \Delta \cdot \exp\left(\frac{2\pi i t}{q}\right)$ on a wide interval $t \in (-Kq, Kq)$. After the evaluation of the complex exponential function, we can extract the imaginary (sine) part by conjugation operation (i.e., $2\sin\theta = \exp(i\theta) - \exp(-i\theta)$), which will be described in the next section.

For the estimation of noise, we start from an initial error between $P_0(t)$ and $\Delta \cdot \exp\left(\frac{2\pi i t}{2^r \cdot q}\right)$, which is bounded by $\frac{\Delta}{(d_0+1)!}\left|\frac{2\pi K}{2^r}\right|^{d_0+1}$ from the Taylor remainder theorem. As described above, the error bound is almost doubled after each squaring. Therefore, we get a bound from an approximation as follows:

$$\begin{aligned} |P_r(t) - E(t)| &\leq \frac{\Delta \cdot 2^r}{(d_0+1)!} \left(\frac{2\pi K}{2^r}\right)^{d_0+1} \\ &\leq \frac{\Delta \cdot 2^r}{\sqrt{2\pi(d_0+1)}} \left(\frac{e\pi K}{2^{r-1}(d_0+1)}\right)^{d_0+1} \end{aligned}$$

from Stirling's formula. Asymptotically the choice of parameters $d_0 = O(1)$ and $r = O(\log(Kq))$ gives us a sufficiently small error bound. Note that the complexity of the algorithm is $r = O(\log(Kq))$ homomorphic multiplications and it grows linearly with the depth of the decryption circuit.

4 Linear Transformation on Packed Ciphertexts

In this section, we explain how to homomorphically evaluate the linear transformations over the vector of plaintext slots. We first present a simple description of the packing method of `HEAAN`. We then explain how to compute the rotation and the complex conjugation over the plaintext slots using the key-switching technique. These functionalities can be applied to the evaluation of a linear transformation over plaintext slots.

4.1 Packing Method

The packing technique of HE schemes allows us to encrypt multiple messages in a single ciphertext and enables a parallel computation in a SIMD manner. Cheon et al. [14] proposed a method to identify a cyclotomic polynomial with real coefficients to a vector of complex numbers. We clarify this encoding method and give a simpler description using the structure of a cyclotomic ring with a power-of-two dimension.

Recall that for a power-of-two integer $M > 4$, we have $N = M/2$ and $\Phi_M(X) = X^N + 1$. The integer 5 has the order of $(N/2)$ modulo M and spans $\mathbb{Z}_M^*$ with the integer "-1". Hence $\{\zeta_j, \overline{\zeta_j} : 0 \le j < N/2\}$ forms the set of the primitive M-th roots of unity for $\zeta_j := \zeta^{5^j}$ and $0 \le j < N/2$. We use the notation $\tau : \mathcal{P} = \mathbb{R}[X]/(X^N + 1) \to \mathbb{C}^{N/2}$ to denote a variant of the complex canonical embedding map defined by $\tau : m(X) \mapsto \boldsymbol{z} = (z_j)_{0 \le j < N/2}$ such that $z_j = m(\zeta_j)$. Note that τ is an isometric ring homomorphism between metric spaces $(\mathcal{P}, \|\cdot\|_\infty^{\mathsf{can}})$ and $(\mathbb{C}^{N/2}, \|\cdot\|_\infty)$. We use this isomorphism τ as the decoding function for packing of $(N/2)$ complex numbers in a single polynomial. By identifying a polynomial $m(X) = \sum_{i=0}^{N-1} m_i X^i \in \mathcal{P}$ with the vector of its coefficients $\boldsymbol{m} = (m_0, \ldots, m_{N-1})$, the decoding algorithm τ can be understood as a linear transformation from $\mathbb{R}^N$ to $\mathbb{C}^{N/2}$. Its matrix representation is given by

$$U = \begin{bmatrix} 1 & \zeta_0 & \zeta_0^2 & \cdots & \zeta_0^{N-1} \\ 1 & \zeta_1 & \zeta_1^2 & \cdots & \zeta_1^{N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \zeta_{\frac{N}{2}-1} & \zeta_{\frac{N}{2}-1}^2 & \cdots & \zeta_{\frac{N}{2}-1}^{N-1} \end{bmatrix}$$

which is the $(N/2) \times N$ Vandermonde matrix generated by $\{\zeta_j : 0 \le j < N/2\}$.

In order to compute the encoding function, which is the inverse of τ, we first note that the relation $\overline{\boldsymbol{z}} = \overline{U} \cdot \boldsymbol{m}$ is obtained from $\boldsymbol{z} = U \cdot \boldsymbol{m}$ by taking the conjugation. If we write $\mathsf{CRT} = (U; \overline{U})$ as the CRT matrix generated by the set $\{\zeta_j, \overline{\zeta_j} : 0 \le j < N/2\}$ of M-th primitive roots of unity, we have the identities $(\boldsymbol{z}; \overline{\boldsymbol{z}}) = \mathsf{CRT} \cdot \boldsymbol{m}$ and $\mathsf{CRT}^{-1} = \frac{1}{N}\overline{\mathsf{CRT}}^T$. This implies that the inverse of τ can be computed by

$$\boldsymbol{m} = \frac{1}{N}(\overline{U}^T \cdot \boldsymbol{z} + U^T \cdot \overline{\boldsymbol{z}}).$$

Throughout this paper, we will identify two spaces $\mathcal{P}$ and $\mathbb{C}^{N/2}$ via the map τ, and hence a ciphertext will be called an encryption of $\boldsymbol{z} \in \mathbb{C}^{N/2}$ if it encrypts the corresponding polynomial $m(X) = \tau^{-1}(\boldsymbol{z})$.

4.2 Rotation and Conjugation

The purpose of the key-switching operation is to convert a ciphertext under a secret s' into a ciphertext of the same message with respect to another secret key $\mathsf{sk} = (1, s)$. The switching key swk can be generated by the procedure of $\mathsf{KSGen}_{\mathsf{sk}}(s')$. Given a ciphertext $\mathsf{ct} = (c_0, c_1)$ at level ℓ, the procedure $\mathsf{KS}_{\mathsf{swk}}(\mathsf{ct})$ proceeds as follows.

- $\underline{\mathsf{KS}_{\mathsf{swk}}(\mathsf{ct})}$. Output the ciphertext $\mathsf{ct}' \leftarrow (c_0, 0) + \lfloor P^{-1} \cdot c_1 \cdot \mathsf{swk} \rceil \pmod{q_\ell}$.

The following lemma shows the correctness of key-switching procedure and estimates a noise bound. It has a similar noise bound $P^{-1} \cdot q \cdot B_{\mathsf{ks}} + B_{\mathsf{rs}} \approx B_{\mathsf{rs}}$ with the rescaling process.

Lemma 4 (Key-switching). *Let $\mathsf{ct} = (c_0, c_1) \in \mathcal{R}_q^2$ be a ciphertext with respect to a secret key $\mathsf{sk}' = (1, s')$ and let $\mathsf{swk} \leftarrow \mathsf{KSGen}_{\mathsf{sk}}(s')$. Then $\mathsf{ct}' \leftarrow \mathsf{KS}_{\mathsf{swk}}(\mathsf{ct})$ satisfies $\langle \mathsf{ct}', \mathsf{sk} \rangle = \langle \mathsf{ct}, \mathsf{sk}' \rangle + e_{\mathsf{ks}} \pmod{q}$ for some $e_{\mathsf{ks}} \in \mathcal{R}$ with $\|e_{\mathsf{ks}}\|_\infty^{\mathsf{can}} \le P^{-1} \cdot q \cdot B_{\mathsf{ks}} + B_{\mathsf{rs}}$.*

Proof. Let $e' = \langle \mathsf{swk}, \mathsf{sk} \rangle - P \cdot s' \pmod{P \cdot q_L}$ be the inserted error of the switching key swk. It was shown in [14, Lem. 3] that the ciphertext ct' contains an additional error $e'' = c_1 \cdot e'$ in the modulus $P \cdot q$ from the key-switching operation. The key-switching error is the sum of $P^{-1} \cdot e''$ and a rounding error e_{rs} of $P^{-1} \cdot c_1 \cdot \mathsf{swk}$, so its size is bounded by

$$\|e_{\mathsf{ks}}\|_\infty^{\mathsf{can}} \le P^{-1} \cdot \|e''\|_\infty^{\mathsf{can}} + \|e_{\mathsf{rs}}\|_\infty^{\mathsf{can}} \le P^{-1} \cdot q \cdot B_{\mathsf{ks}} + B_{\mathsf{rs}},$$

as desired. □

For an integer k co-prime with M, let $\kappa_k : m(X) \mapsto m(X^k) \pmod{\Phi_M(X)}$ be a mapping defined on the set $\mathcal{P}$. As noted in [29], this transformation can be used to provide more functionalities on plaintext slots. In some more details, given a ciphertext ct of a message m with $\mathsf{sk} = (1, s)$, we denote $\kappa_k(\mathsf{ct})$ the ciphertext which is obtained by applying κ_k to each entry of ct. Then $\kappa_k(\mathsf{ct})$ is a valid encryption of $\kappa_k(m)$ with the secret key $\kappa_k(s)$. The key-switching technique can be applied to the ciphertext $\kappa_k(\mathsf{ct})$ to get an encryption of the same message $\kappa_k(m)$ with respect to the original secret key sk.

Rotation. Suppose that ct is an encryption of a message $m(X)$ with the corresponding plaintext vector $\boldsymbol{z} = (z_j)_{0 \le j < \frac{N}{2}} \in \mathbb{C}^{N/2}$. For any $0 \le i, j < N/2$, there is a mapping κ_k which sends an element in the slot of index i to an element in the slot of index j. Let us define $k = 5^{i-j} \pmod{M}$ and $\tilde{m} = \kappa_k(m)$. Denoting $\tilde{\boldsymbol{z}} = (\tilde{z}_j)_{0 \le j < \frac{N}{2}}$ as the corresponding plaintext vector of $\tilde{m}$, we have

$$\tilde{z}_j = \tilde{m}(\zeta_j) = m(\zeta_j^{5^{i-j}}) = m(\zeta_i) = z_i,$$

so the j-th slot of $\tau(\tilde{m})$ and the i-th entry of $\tau(m)$ have the same value. In general, we may get a ciphertext $\kappa_{5^r}(\mathsf{ct})$ encrypting $\rho(\boldsymbol{z}; r) := (z_r, \ldots, z_{\frac{N}{2}-1}, z_0, \ldots, z_{r-1})$, which is the vector obtained from $\boldsymbol{z}$ by rotation. Below, we describe the rotation procedure including the key-switching operation.

- Generate the rotation key $\mathsf{rk}_r \leftarrow \mathsf{KSGen}_{\mathsf{sk}}(\kappa_{5^r}(s))$.
- $\underline{\mathsf{Rot}(\mathsf{ct}; r)}$. Output the ciphertext $\mathsf{KS}_{\mathsf{rk}_r}(\kappa_{5^r}(\mathsf{ct}))$.

Conjugation. We see that $\kappa_{-1}(\mathsf{ct})$ is a valid encryption of the plaintext vector $\overline{\boldsymbol{z}} = (\overline{z_j})_{0 \le j < N/2}$ with the secret key $\kappa_{-1}(s)$. It follows from that fact that

$$\overline{z_j} = \overline{m(\zeta_j)} = m(\overline{\zeta_j}) = m(\zeta_j^{-1}).$$

Then the homomorphic evaluation of the conjugation operation over plaintext slots consists of two procedures:

- Generate the conjugation key $\mathsf{ck} \leftarrow \mathsf{KSGen}_{\mathsf{sk}}(\kappa_{-1}(s))$.
- $\underline{\mathsf{Conj}(\mathsf{ct})}$. Output the ciphertext $\mathsf{KS}_{\mathsf{ck}}(\kappa_{-1}(\mathsf{ct}))$.

4.3 Linear Transformations

In general, an arbitrary linear transformation over the vector of plaintext slots in $\mathbb{C}^{N/2}$ can be represented as $\boldsymbol{z} \mapsto A \cdot \boldsymbol{z} + B \cdot \overline{\boldsymbol{z}}$ for some complex matrices $A, B \in \mathbb{C}^{N/2 \times N/2}$. As discussed in [32], it can be efficiently done by handling the matrix in a diagonal order and making use of SIMD computation. Specifically, let $\boldsymbol{u}_j = (A_{0,j}, A_{1,j+1}, \ldots, A_{\frac{N}{2}-j-1,\frac{N}{2}-1}, A_{\frac{N}{2}-j,0}, \ldots, A_{\frac{N}{2}-1,j-1}) \in \mathbb{C}^{N/2}$ denote the shifted diagonal vector of A for $0 \leq j < N/2$. Then we have

$$A \cdot \boldsymbol{z} = \sum_{0 \leq j < N/2} (\boldsymbol{u}_j \odot \rho(\boldsymbol{z}; j)) \tag{1}$$

where $\odot$ denotes the Hadamard (component-wise) multiplication between vectors. Therefore, if the matrix A is given in plaintext and ct is given as an encryption of the vector $\boldsymbol{z}$, the matrix-vector multiplication $A \cdot \boldsymbol{z}$ is expressed as combination of rotations and scalar multiplications. The vector rotation $\rho(\boldsymbol{z}; j)$ can be homomorphically computed by $\mathsf{Rot}(\mathsf{ct}; j)$ and the Hadamard (component wise) scalar multiplication is done by multiplying the polynomial $\tau^{-1}(\boldsymbol{u}_j)$. See Algorithm 1 for an explicit description of the homomorphic matrix multiplication. Similarly, the second term $B \cdot \overline{\boldsymbol{z}}$ can be obtained by multiplying the matrix B after applying the slot-wise conjugation on $\boldsymbol{z}$.

Algorithm 1 Homomorphic evaluation of matrix multiplication

```
1: procedure MatMult(ct ∈ R_q^2, A ∈ C^{N/2×N/2})
2:     ct' ← ⌊τ^{-1}(u_0)⌉ · ct (mod q)
3:     for j = 1 to N/2 − 1 do
4:         ct_j ← ⌊τ^{-1}(u_j)⌉ · Rot(ct; j) (mod q)
5:         ct' ← Add(ct', ct_j) (mod q)
6:     end for
7:     return ct'
8: end procedure
```

Algorithm 1 requires $(N/2 - 1)$ rotations and N multiplications with scalar polynomials but the complexity can be reduced down using the idea of Baby-Step Giant-Step algorithm. Let $N_1 = O(\sqrt{N})$ be a divisor of $N/2$ and denote $N_2 = N/2N_1$. It follows from [33] that Equation (1) can be expressed as

$$\begin{aligned} A \cdot \boldsymbol{z} &= \sum_{0 \leq j < N_2} \sum_{0 \leq i < N_1} (\boldsymbol{u}_{N_1 \cdot j + i} \odot \rho(\boldsymbol{z}; N_1 \cdot j + i)) \\ &= \sum_{0 \leq j < N_2} \rho\left(\sum_{0 \leq i < N_1} \rho(\boldsymbol{u}_{N_1 \cdot j + i}; -N_1 \cdot j) \odot \rho(\boldsymbol{z}; i); N_1 \cdot j \right). \end{aligned}$$

For the homomorphic evaluation of the arithmetic circuit, we first compute the ciphertexts of $\rho(\boldsymbol{z}; k)$ for $i = 1, \ldots, N_1 - 1$. For each index j, we perform N_1 scalar multiplications and aggregate the resulting ciphertexts. In total, the matrix multiplication can be homomorphically evaluated with $(N_1 - 1) + (N_2 - 1) = O(\sqrt{N})$ rotations and $N_1 \cdot N_2 = O(N)$ scalar multiplications.

We provide a trade-off between the precision of a plaintext and the size of a ciphertext modulus. The output plaintext contains errors from the rounding operation of scalar polynomial and the homomorphic rotation. We can reduce the relative size of the rounding error by multiplying a scaling factor of $\Delta \geq 1$ to the scalar polynomials, and the relative size of the

rotation error can be controlled by placing the significand of an input ciphertext in higher digits. Therefore, the modulus of an output ciphertext is reduced after the evaluation of a linear transformation if we use a scaling factor to get a better precision of plaintexts.

4.4 Sparsely Packed Ciphertext

The packing method described in Section 4.1 allows us to encrypt $(N/2)$ complex numbers in a single ciphertext. However, it is sufficient to deal with a small number of slots in some applications of HE. In this section, we explain how to encode a sparse plaintext vector and describe the relation with the ordinary packing method. This idea will be applied to our bootstrapping method and provide a trade-off between the latency time and the amortized time of bootstrapping procedure.

Let $n \geq 2$ be a divisor of N and let $Y = X^{N/n}$. The native plaintext space of `HEAAN` is the set of small polynomials in $\mathbb{Z}[X]/(X^N + 1)$, and it has a subring $\mathbb{Z}[Y]/(Y^n + 1)$. Note that $\mathbb{Z}[Y]/(Y^n + 1)$ can be identified with the complex coordinate space $\mathbb{C}^{n/2}$ by adapting the idea of ordinary packing method in Section 4.1. Specifically, a polynomial $m(Y)$ is mapped to the vector $\boldsymbol{w} = (w_j)_{0 \leq j < n/2}$ where $w_j = m(\xi_j)$, $\xi = \exp(-2\pi i/n) = \zeta^{N/n}$, and $\xi_j = \xi^{5^j}$ for $0 \leq j < n/2$. If we consider a plaintext polynomial $m(Y) \in \mathbb{Z}[Y]/(Y^n + 1)$ as a polynomial $\tilde{m}(X) = m(X^{N/n})$ in X, then the image of $\tilde{m}$ through the ordinary decoding function τ is obtained by $\tau(\tilde{m}) = (z_j)_{0 \leq j < N/2}$ where

$$z_j = \tilde{m}(\zeta_j^{N/n}) = m(\xi^{5^j}) = w_j \pmod{n/2}$$

for $0 \leq j < N/2$. Hence $\boldsymbol{z} = (z_j)_{0 \leq j < N/2}$ can be understood as the vector obtained from $\boldsymbol{w}$ by concatenating itself (N/n) times.

An encryption of a plaintext polynomial $m(Y) \in \mathbb{Z}[Y]/(Y^n + 1)$ with respect to a secret key $\mathsf{sk} = (1, s)$ will be pairs as $\mathsf{ct} \in \mathcal{R}_q^2$ satisfying $\langle \mathsf{ct}, \mathsf{sk} \rangle = m(Y) + e(X) \pmod{q}$ for some small polynomial $e(X) \in \mathcal{R}$. We may employ key-switching technique to get the functionalities of rotation, conjugation, and linear transformation on sparsely packed slots. The main advantage of this method is that it reduces the complexity of an arbitrary linear transformation: the total complexity on $(n/2)$-sparsely packed ciphertexts is bounded by $O(\sqrt{n})$ rotations and $O(n)$ scalar multiplications.

5 Bootstrapping for `HEAAN`

5.1 Overview of the Recryption Procedure

This section gives a high level structure of the bootstrapping process for the `HEAAN` scheme. We employ the ciphertext packing method and combine it with our efficient evaluation strategy to achieve a better performance in terms of memory and computation cost. Below we describe all the parts of the recryption procedure in more details. We denote the following five steps by ModRaise, CoeffToSlot, EvalExp, ImgExt and SlotToCoeff, respectively. See Fig. 2 for an illustration.

Modulus raising. Let ct be an input ciphertext of the bootstrapping procedure with a ciphertext modulus q satisfying $m(X) = [\langle \mathsf{ct}, \mathsf{sk} \rangle]_q$. We start with the point that its inner product $t(X) = \langle \mathsf{ct}, \mathsf{sk} \rangle \pmod{X^N + 1}$ is of the form $t = qI + m$ for some small $I(X) \in \mathcal{R}$ with a bound $\|I\|_\infty < K$. Thus ct itself can be viewed as an encryption of $t(X) = t_0 + t_1 X + \cdots + t_{N-1} X^{N-1}$ in a large modulus $Q_0 \gg q$ due to $[\langle \mathsf{ct}, \mathsf{sk} \rangle]_{Q_0} = t(X)$. Our bootstrapping procedure aims to homomorphically and approximately evaluate the reduction mod q $F(t) = [t]_q$ using arithmetic operations over the integers, and hence we can generate an encryption of the original message $m = [t]_q$ with a ciphertext modulus larger than q.

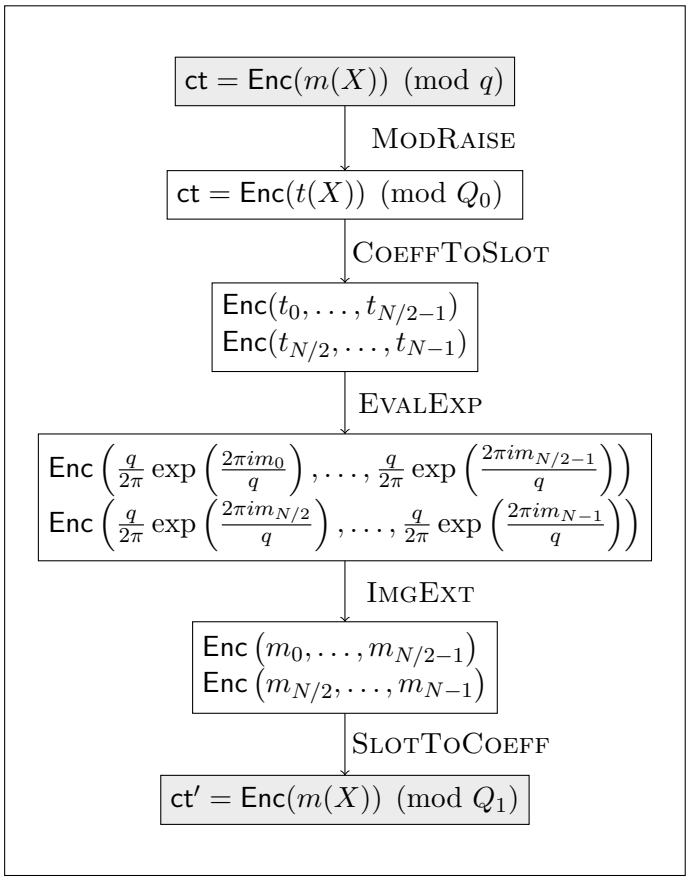

Fig. 2. Pipeline of our bootstrapping process

Putting polynomial coefficients in plaintext slots. Given the input ciphertext $\mathsf{ct} \in \mathcal{R}^2_{Q_0}$ with a decryption structure $t(X) = \langle \mathsf{ct}, \mathsf{sk} \rangle$, this step aims to put the coefficients $t_0, \ldots, t_{N-1}$ in plaintext slots in order to evaluate the modular reduction function $F(t)$ coefficient-wisely. Let $\boldsymbol{z}' = \tau(t) \in \mathbb{C}^{N/2}$ be the corresponding vector of plaintext slots of the ciphertext ct. Since each ciphertext can store at most $N/2$ plaintext values, we will generate two ciphertexts encrypting the vectors $\boldsymbol{z}'_0 = (t_0, \ldots, t_{\frac{N}{2}-1})$ and $\boldsymbol{z}'_1 = (t_{\frac{N}{2}}, \ldots, t_{N-1})$, respectively.

As mentioned in Section 4.1, recall the linear relation between the coefficient vector of a polynomial and its corresponding vector of plaintext slots. If we divide the matrix U into two square matrices

$$U_0 = \begin{bmatrix} 1 & \zeta_0 & \ldots & \zeta_0^{\frac{N}{2}-1} \\ 1 & \zeta_1 & \ldots & \zeta_1^{\frac{N}{2}-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \zeta_{\frac{N}{2}-1} & \ldots & \zeta_{\frac{N}{2}-1}^{\frac{N}{2}-1} \end{bmatrix} \quad \text{and} \quad U_1 = \begin{bmatrix} \zeta_0^{\frac{N}{2}} & \zeta_0^{\frac{N}{2}+1} & \ldots & \zeta_0^{N-1} \\ \zeta_1^{\frac{N}{2}} & \zeta_1^{\frac{N}{2}+1} & \ldots & \zeta_1^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ \zeta_{\frac{N}{2}-1}^{\frac{N}{2}} & \zeta_{\frac{N}{2}-1}^{\frac{N}{2}+1} & \ldots & \zeta_{\frac{N}{2}-1}^{N-1} \end{bmatrix},$$

then we get an identity $\boldsymbol{z}'_k = \frac{1}{N}(\overline{U_k}^T \cdot \boldsymbol{z}' + U_k^T \cdot \overline{\boldsymbol{z}'})$ for $k = 0, 1$. Therefore, we can generate encryptions of $\boldsymbol{z}'_0$ and $\boldsymbol{z}'_1$ using the linear transformations on the plaintext vector $\boldsymbol{z}'$. One can apply our general method in Section 4.3 to this step.

Evaluation of the complex exponential function. This step takes the results of the previous step and homomorphically computes the reduction mod q function $F(t) = [t]_q$ homomorphically. We use the trigonometric function $S(t) := \frac{q}{2\pi}\sin\left(\frac{2\pi t}{q}\right)$ as an approximation

of $F(t)$ and adapt the optimized evaluation strategy for the complex exponential function $E(t) := \frac{q}{2\pi}\exp\left(\frac{2\pi i t}{q}\right)$.

Since the plaintext slots of the input ciphertexts contain the coefficients $t_j = qI_j + m_j$ for $0 \le j < N$, the output ciphertexts will encrypt $\frac{q}{2\pi}\exp\left(\frac{2\pi i t_j}{q}\right) = \frac{q}{2\pi}\exp\left(\frac{2\pi i m_j}{q}\right)$ in the corresponding plaintext slots.

Extraction of the imaginary part. We take two input ciphertexts encrypting the values $\frac{q}{2\pi}\exp\left(\frac{2\pi i m_j}{q}\right)$ in their plaintext slots for $0 \le j < N$. We extract their imaginary parts as $\frac{q}{2\pi}\sin\left(\frac{2\pi m_j}{q}\right) \approx m_j$ by using the relation

$$\sin\left(\frac{2\pi m_j}{q}\right) = \frac{1}{2}\left(\exp\left(\frac{2\pi i m_j}{q}\right) - \exp\left(\frac{-2\pi i m_j}{q}\right)\right)$$

and applying the evaluation method of slot-wise conjugation described in Section 4.2.

Switching back to the coefficient representation. The final step is to pack all the coefficients m_j in the plaintext slots of two ciphertexts back in a single ciphertext. This procedure is exactly the inverse of the CoeffToSlot transformation. That is, when given two ciphertexts that encrypt the vectors $\boldsymbol{z}_0 = (m_0, \ldots, m_{\frac{N}{2}-1})$ and $\boldsymbol{z}_1 = (m_{\frac{N}{2}}, \ldots, m_{N-1})$, we aim to generate an encryption of $m(X)$. Since the plaintext vector $\boldsymbol{z} = \tau(m)$ of $m(X)$ satisfies the identity $\boldsymbol{z} = U \cdot \boldsymbol{m} = U_0 \cdot \boldsymbol{z}_0 + U_1 \cdot \boldsymbol{z}_1$, this transformation is also represented as a linear transformation over the plaintext vectors.

Our bootstrapping process returns an encryption of $m(X)$ with a ciphertext modulus $Q_1 < Q_0$, which is much larger enough than the initial modulus q to allow us to perform further homomorphic operations on the ciphertext.

We can perform the final two steps together by pre-computing the composition of linear transformations. The (inverse) linear transformation step consumes only one level for scalar multiplications but requires a number of slot rotations. On the other hand, EvalExp performs homomorphic evaluation of the polynomial $P_r(\cdot)$, which is the most levels consuming part of the recryption but requires relatively small computational cost from our recursive evaluation strategy: linear with the depth.

5.2 Recryption with Sparsely Packed Ciphertexts

As mentioned in Section 4.4, the use of sparsely packed ciphertexts has an advantage in some applications, in that it reduces the complexity of linear transformation steps. However, the recryption of sparsely packed ciphertexts requires an additional step before the CoeffToSlot step.

Let $n \ge 2$ be a divisor of N and let $Y = X^{N/n}$ as in Section 4.4. Assume that we are given an encryption ct of $m(Y) \in \mathbb{Z}[Y]/(Y^n+1)$ such that $\langle \mathsf{ct}, \mathsf{sk}\rangle \approx q \cdot I(X) + m(Y)$ for some $I(X) = I_0 + I_1 \cdot X + \cdots + I_{N-1} \cdot X^{N-1} \in \mathcal{R}$. Then the ModRaise step returns an encryption of $q \cdot I(X) + m(Y)$ which is not a polynomial of Y. We aim to generate an encryption of $q \cdot \tilde{I}(Y) + m(Y)$ for some $\tilde{I}(Y) \in \mathbb{Z}[Y]/(Y^n+1)$ before the next CoeffToSlot step. It proceeds as described in Algorithm 2.

Note that the monomial X^k vanishes by the homomorphism $X \mapsto X - X^{n+1} + X^{2n+1} - \cdots - X^{N-n+1}$ if k is not divisible by (N/n); otherwise, it is multiplied by the constant (N/n). Hence $q \cdot I(X) + m(Y)$ is mapped to $(N/n) \cdot (q \cdot \tilde{I}(Y) + m(Y))$ by this homomorphism where $\tilde{I}(Y) = I_0 + I_{N/n} \cdot Y + \cdots + I_{N-(N/n)} \cdot Y^{n-1}$. For an efficient evaluation of this homomorphism, Algorithm 2 uses the rotation operation repeatedly to fill the same value in the plaintext slots of index $j \pmod{n/2}$ for each $j = 0, \ldots, n/2 - 1$.

Algorithm 2 Homomorphic evaluation of the partial-sum procedure

procedure PARTIALSUM($\mathsf{ct} \in \mathcal{R}_q^2, n|N, n \geq 2$)
  $\mathsf{ct}' \leftarrow \mathsf{ct} \pmod q$
  for $j = 0$ to $\log(N/n) - 1$ **do**
    $\mathsf{ct}_j \leftarrow \mathsf{Rot}(\mathsf{ct}'; 2^j \cdot (n/2)) \pmod q$
    $\mathsf{ct}' \leftarrow \mathsf{Add}(\mathsf{ct}', \mathsf{ct}_j) \pmod q$
  end for
  return ct'
end procedure

As mentioned before, the main advantage of sparsely packed ciphertexts is that the COEFFTOSLOT step can be represented as relatively small matrices multiplications with only a single encryption of the coefficients vector of $t(Y) = q \cdot \tilde{I}(Y) + m(Y)$ while fully-packed slots need two ciphertexts for the COEFFTOSLOT step. In some more details, for the plaintext vector $\boldsymbol{w} \in \mathbb{C}^{n/2}$ of $t(Y)$, the desired ciphertext can be computed by $\frac{1}{n}(\overline{U'}^T \cdot \boldsymbol{w} + U'^T \cdot \overline{\boldsymbol{w}})$ where

$$U' = \begin{bmatrix} 1 & \xi_0 & \xi_0^2 & \cdots & \xi_0^{n-1} \\ 1 & \xi_1 & \xi_1^2 & \cdots & \xi_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \xi_{\frac{n}{2}-1} & \xi_{\frac{n}{2}-1}^2 & \cdots & \xi_{\frac{n}{2}-1}^{n-1} \end{bmatrix},$$

as in Section 4.1. We can either generate two ciphertexts encrypting two plaintext vectors $(w_0, \ldots, w_{n/2-1})$ and $(w_{n/2}, \ldots, w_{n-1})$ by separating U' into two square matrices, or compute a single encryption of $(w_0, \ldots, w_{n-1})$ with n plaintext slots when $n < N$. In the latter case, EVALEXP and IMGEXT perform the same operations as in the fulled-packed ciphertexts, but the memory and the computational cost are reduced by half since we can work on a single ciphertext. The final SLOTTOCOEFF step is also expressed as a linear transformation over n-dimensional vector.

5.3 Noise Estimation of Recryption

In this section, we describe each step of recryption procedure with a noise analysis. We start with an upper bound K of $\|I\|_\infty$. Since each coefficient of a ciphertext $\mathsf{ct} = (c_0, c_1)$ is an element of $\mathbb{Z}_q$ and the signed binary secret key s has exactly h nonzero coefficients, each coefficient of $\langle \mathsf{ct}, \mathsf{sk} \rangle = c_0 + c_1 s$ can be considered to be the sum of $(h+1)$ elements in $\mathbb{Z}_q$, which is bounded by $\frac{q}{2}(h+1)$. Hence all the coefficient of $I(X) = \lfloor \frac{1}{q} \langle \mathsf{ct}, \mathsf{sk} \rangle \rceil$ is bounded by $\frac{1}{2}(h+1) \approx \frac{1}{2}\|s\|_1$. In practice, the coefficients of c_i look like a random variable over the interval $\mathbb{Z}_q$ and a coefficient of $\frac{1}{q}\langle \mathsf{ct}, \mathsf{sk} \rangle$ behaves as the sum of $(h+1)$-numbers of i.i.d. uniform random variables over the interval $(-\frac{1}{2}, \frac{1}{2})$. This heuristic assumption gives us a smaller bound as $K = O(\sqrt{h})$ for $\|I\|_\infty$.

We now consider the error growth during homomorphic evaluation of a linear transformation. As noted in Section 4.3, it induces two types of errors such as from the rounding of a scalar polynomial and the key-switching operation. We multiply a sufficiently large scaling factor of $\Delta = O(q)$ to scalar polynomials, so that the precision of rounded polynomials becomes larger than that of an input plaintext. Then we do not need to consider the rounding errors because they have no effect on the precision of the resulting plaintext. The second type of error is added to a plaintext when we apply the key-switching technique for some functionalities such as rotation or conjugation. From Lemma 4, the key-switching error is bounded by $P^{-1} \cdot q \cdot B_{\mathsf{ks}} + B_{\mathsf{rs}} \approx B_{\mathsf{rs}}$ since we set a ciphertext modulus q much smaller than

P. During matrix multiplication, key-switching errors are multiplied with diagonal vectors of matrix. Since infinite norms of diagonal vectors of $\frac{1}{N}U_0$ and $\frac{1}{N}U_1$ are exactly one, the total error of second type is bounded by $O(B_{\mathsf{rs}})$.

In the EvalExp step, we take two ciphertexts to be homomorphically evaluated by the approximate polynomial $P_r(\cdot)$ of the complex exponential polynomial. Each component of the corresponding plaintext slots contains $t_j + e_j$ for some small error e_j such that $|e_j| \leq O(B_{\mathsf{rs}})$. Hence, the error between the desired value $E(t_j)$ and the resulting plaintext of EvalExp can be measured by

$$\begin{aligned} |E(t_j) - P_r(t_j + e_j)| &\leq |E(t_j) - E(t_j + e_j)| + |E(t_j + e_j) - P_r(t_j + e_j)| \\ &\leq \frac{\Delta \cdot 2\pi}{q}|e_j| + \frac{\Delta \cdot 2^r}{\sqrt{2\pi(d_0+1)}}\left(\frac{e\pi K}{2^{r-1}(d_0+1)}\right)^{d_0+1}, \end{aligned}$$

since $E(\cdot)$ is analytic and $|E'(\cdot)| \leq \Delta \cdot 2\pi/q$. The second term can be bounded by $O(1)$ and so it is negligible when $d_0 = O(1)$ and $r = O(\log(qK))$ as described in Section 3.2. By combining the error bound as $|e_j| \leq O(B_{\mathsf{rs}})$ of the previous step, we deduce that the output ciphertexts of EvalExp encrypt $E(t_j) = \Delta \cdot \exp\left(\frac{2\pi i t_j}{q}\right)$ in their plaintext slots with errors bounded by $O(B_{\mathsf{rs}})$.

The imaginary part of $E(t)$ is $\frac{\Delta \cdot 2\pi}{q}S(t) = \Delta \cdot \sin\left(\frac{2\pi t}{q}\right)$, which is an approximation of $\frac{\Delta \cdot 2\pi}{q}m$ with an error bounded by $O\left(q \cdot \frac{|m|^3}{q^3}\right)$ for $m = [t]_q$. We set a small bound on an input plaintext (e.g. $|m| \leq q^{2/3}$) so that an approximation error does not destroy the significant digits of the plaintext. Hence the ImgExt step does not change the magnitude of bootstrapping error.

Finally, in the SlotToCoeff step, the plaintext vectors are multiplied with the matrices U_0 and U_1, and their diagonal vectors have the size of one. The size of error in the resulting plaintext is bounded by $O(N \cdot B_{\mathsf{rs}})$ since it is the sum of N errors of size $O(B_{\mathsf{rs}})$. In practice, under the heuristic assumption that these errors behaves as independent Gaussian distributions, we get a reduced error bound as $O(\sqrt{N} \cdot B_{\mathsf{rs}})$.

In summary, for an input ciphertext $\mathsf{ct} \in \mathcal{R}_q^2$ satisfying $\langle \mathsf{ct}, \mathsf{sk} \rangle = m \pmod q$, our bootstrapping process returns a ciphertext ct' such that $\langle \mathsf{ct}', \mathsf{sk} \rangle = m + e \pmod{Q_1}$ for a modulus $Q_1 \gg q$ and some error e with $\|e\|_\infty^{\mathsf{can}} \leq O(\sqrt{N} \cdot B_{\mathsf{rs}})$. It consists of the initial/final linear transformations and the evaluation of the complex exponential function, so the total depth (number of levels consumed) for bootstrapping is $O(\log(Kq)) = O(\log \lambda)$. The linear transformations require and $O(\sqrt{N})$ rotations, while the evaluation of the exponential function needs $r = O(\log(Kq)) = O(\log \lambda)$ homomorphic multiplications, which is linear with the depth of the decryption formula. As described above, the total complexity can be significantly reduced when we handle a sparsely packed ciphertext with $(n/2)$ slots: the linear transformations require $O(\sqrt{n})$ rotations and the evaluation complexity of the exponential function is reduced by half.

6 Implementation

In this section, we suggest some parameter sets for our bootstrapping procedure with experimental results. Our implementation is based on the `HEAAN` library [13] implementing the HE scheme of Cheon et al. [14]. The source code is publicly available at `github` [12].

6.1 Parameter Selection

We adapt the estimator of Albrecht at el. [1] to guarantee the concrete security of the proposed parameters. All the parameter sets achieve at least 80-bit security level against the known attacks of the LWE problem.

The key-switching keys have the largest modulus in the `HEAAN` scheme and the modulus Q_0 (after ModRaise) has half the bit size of the modulus. We use the discrete Gaussian distribution of standard deviation $\sigma = 3.2$ to sample error polynomials and set the Hamming weight $h = 64$ of the secret key $s(X)$. The parameters $d_0 = O(1)$ and $r = O(\log q)$ were chosen asymptotically in the above sections, but in practice, we set the parameter experimentally based on the bootstrapping error. We take the degree 7 Taylor expansion as an initial approximation of the exponential function and choose a sufficiently large number of iterations r to maintain the precision of output plaintexts.

The parameter $\log p$ is the bit size of plaintexts and the plaintext precision denotes the number of significant bits of plaintexts after bootstrapping. The before and after levels L_0, L_1 are obtained by dividing $\log Q_0$ and $\log Q_1$ by $\log p$, respectively. The whole parameter sets are described in Table 1.

Parameter	$\log N$	$\log p$	$\log q$	r	$\log Q_0$	L_0	$\log Q_1$	L_1	Plaintext precision
Set-I	15	23	29	6	620	26	202	8	8 bits
Set-II		27	37	7		22	64	2	12 bits
Set-III	16	31	41	7	1240	40	631	20	16 bits
Set-IV		39	54	9		31	344	8	24 bits

Table 1. Parameter sets

We present some specific examples of input and output plaintexts. For simplicity, we show the real parts of the plaintext values in four slots. All the values are divided by a factor of p for a clear interpretation.

```
Before bootstrapping : [0.777898 0.541580 0.603675 0.822638]
 After bootstrapping : [0.777435 0.541021 0.603023 0.822321]
```

It shows that the error vectors is bounded by $2^{-11} \cdot p$, i.e., the bootstrapping procedure with the parameter set I outputs a ciphertext of 10 bits of precision.

```
Before bootstrapping : [0.516015 0.772621 0.939175 0.345987]
 After bootstrapping : [0.516027 0.772614 0.939172 0.346001]
```

Similarly, the second example shows that the bootstrapping error with the parameter set II is bounded by $2^{-16} \cdot p$ and the output plaintext has 15 bits of precision.

6.2 Experimental Results

We show the performance of our bootstrapping procedure based on the proposed parameter sets. All the experimentations were performed as a single hyperthread on a 2.10 GHz Intel Xeon E5-2620. The experimental results are summarized in Table 2. The linear transformation includes the three steps - PartialSum, CoeffToSlot, and SlotToCoeff.

Parameter	Number of slots	Linear trans.	EvalExp	Total time	Amortized time
Set-I	1	12.3 s	12.3 s	24.6 s	24.6 s
	32	48.6 s		60.9 s	1.9 s
	64	82.8 s		95.1 s	1.5 s
	128	139.2 s		151.5 s	1.2 s
Set-II	1	14.1 s	12.5 s	26.6 s	26.6 s
	32	46.0 s		58.5 s	1.8 s
	64	77.1 s		89.6 s	1.4 s
	128	127.3 s		139.8 s	1.1 s
Set-III	1	64 s	63 s	127 s	127 s
	32	218 s		281 s	8.8 s
	64	343 s		406 s	6.3 s
	128	528 s		591 s	4.6 s
Set-IV	1	58 s	68 s	126 s	126 s
	32	200 s		268 s	8.4 s
	64	307 s		375 s	5.9 s
	128	456 s		524 s	4.1 s

Table 2. Bootstrapping timings with parameter sets I to IV

The amortized time is obtained by dividing the total bootstrapping time by the number of plaintext slots.

The linear transformations take a longer time as the number of plaintext slots grows while the complexity of the EvalExp step remains stable. Therefore, we can make a trade-off between the latency time and the amortized time by changing the number of slots. Fig. 3 illustrates the trend of evaluation timings for each of the bootstrapping phases (parameter set I).

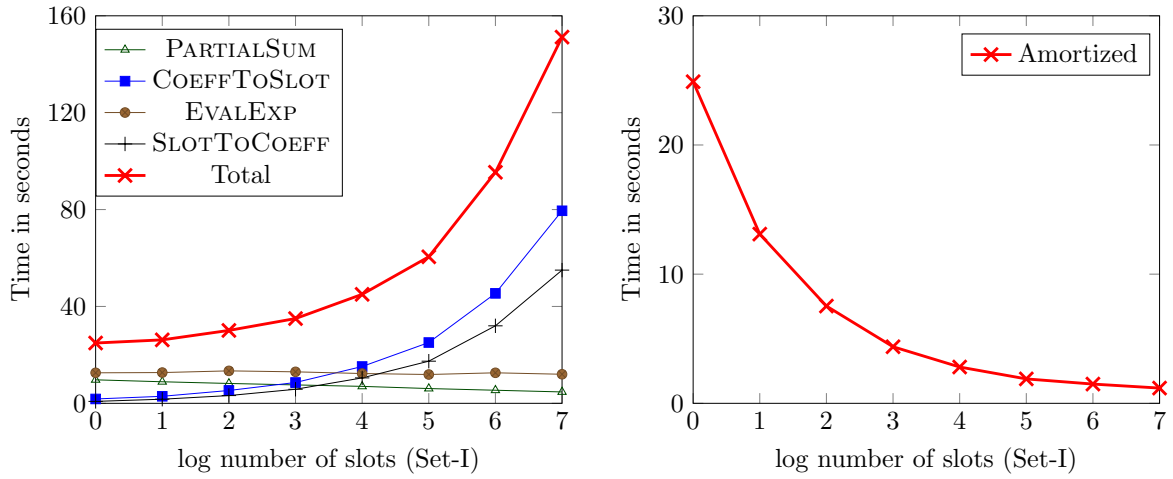

Fig. 3. Tendency of real (left) and amortized (right) bootstrapping timings

7 Conclusion

In this paper, we suggested a method to recrypt a ciphertext of the `HEAAN` scheme. The performance of our bootstrapping procedure was significantly improved by adapting a trigonometric approximation of the modular reduction function. The linear transformation turns out to be the most time-consuming part, but we used almost the same method as in [28, 33]. It would be an interesting open problem to find an efficient algorithm to evaluate the linear transformations approximately.

Acknowledgments. We would like to thank the anonymous EUROCRYPT'18 reviewers for useful comments. This work was partially supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.B0717-16-0098, Development of homomorphic encryption for DNA analysis and biometry authentication). M. Kim was supported by the National Institute of Health (NIH) under award number U01EB023685.

References

1. M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.
2. J. Alperin-Sheriff and C. Peikert. Faster bootstrapping with polynomial error. In *Advances in Cryptology–CRYPTO 2014*, pages 297–314. Springer, 2014.
3. G. Bonnoron, L. Ducas, and M. Fillinger. Large fhe gates from tensored homomorphic accumulator. Cryptology ePrint Archive, Report 2017/996, 2017. `https://eprint.iacr.org/2017/996`.
4. C. Bonte, C. Bootland, J. W. Bos, W. Castryck, I. Iliashenko, and F. Vercauteren. Faster homomorphic function evaluation using non-integral base encoding. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 579–600. Springer, 2017.
5. J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In *Cryptography and Coding*, pages 45–64. Springer, 2013.
6. Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Advances in Cryptology–CRYPTO 2012*, pages 868–886. Springer, 2012.
7. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *Proc. of ITCS*, pages 309–325. ACM, 2012.
8. Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, FOCS'11, pages 97–106. IEEE Computer Society, 2011.
9. Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from Ring-LWE and security for key dependent messages. In *Advances in Cryptology–CRYPTO 2011*, pages 505–524. Springer, 2011.
10. H. Chen, K. Laine, R. Player, and Y. Xia. High-precision arithmetic in homomorphic encryption. Cryptology ePrint Archive, Report 2017/809, 2017. `https://eprint.iacr.org/2017/809`.
11. J. H. Cheon, J.-S. Coron, J. Kim, M. S. Lee, T. Lepoint, M. Tibouchi, and A. Yun. Batch fully homomorphic encryption over the integers. In *Advances in Cryptology–EUROCRYPT*, volume 7881, pages 315–335. Springer, 2013.
12. J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song. Implementation of boostrapping for HEAAN, 2017. `https://github.com/kimandrik/HEAANBOOT`.
13. J. H. Cheon, A. Kim, M. Kim, and Y. Song. Implementation of HEAAN, 2016. `https://github.com/kimandrik/HEAAN`.
14. J. H. Cheon, A. Kim, M. Kim, and Y. Song. 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*, pages 409–437. Springer, 2017.
15. J. H. Cheon, M. Kim, and K. Lauter. Homomorphic computation of edit distance. In *International Conference on Financial Cryptography and Data Security*, pages 194–212. Springer, 2015.
16. J. H. Cheon and D. Stehlé. Fully homomophic encryption over the integers revisited. In *Advances in Cryptology–EUROCRYPT 2015*, pages 513–536. Springer, 2015.
17. I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security*, pages 3–33. Springer, 2016.
18. I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster packed homomorphic operations and efficient circuit bootstrapping for tfhe. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Application of Cryptology and Information Security*, pages 377–408. Springer, 2017.
19. J.-S. Coron, T. Lepoint, and M. Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In *Public-Key Cryptography–PKC 2014*, pages 311–328. Springer, 2014.
20. A. Costache, N. P. Smart, and S. Vivek. Faster homomorphic evaluation of discrete fourier transforms. In *International Conference on Financial Cryptography and Data Security*, pages 517–529. Springer, 2017.
21. A. Costache, N. P. Smart, S. Vivek, and A. Waller. Fixed-point arithmetic in she schemes. In *International Conference on Selected Areas in Cryptography*, pages 401–422. Springer, 2016.
22. M. v. Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology–EUROCRYPT 2010*, pages 24–43. Springer, 2010.
23. N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. Manual for using homomorphic encryption for bioinformatics. *Proceedings of the IEEE*, 105(3):552–567, 2017.
24. L. Ducas and D. Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology–EUROCRYPT 2015*, pages 617–640. Springer, 2015.
25. J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.
26. C. Gentry et al. Fully homomorphic encryption using ideal lattices. In *STOC*, volume 9, pages 169–178, 2009.

27. C. Gentry and S. Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In *Advances in Cryptology–EUROCRYPT 2011*, pages 129–148. Springer, 2011.
28. C. Gentry, S. Halevi, and N. P. Smart. Better bootstrapping in fully homomorphic encryption. In *Public Key Cryptography–PKC 2012*, pages 1–16. Springer, 2012.
29. C. Gentry, S. Halevi, and N. P. Smart. Fully homomorphic encryption with polylog overhead. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 465–482. Springer, 2012.
30. C. Gentry, S. Halevi, and N. P. Smart. Homomorphic evaluation of the AES circuit. In *Advances in Cryptology–CRYPTO 2012*, pages 850–867. Springer, 2012.
31. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013*, pages 75–92. Springer, 2013.
32. S. Halevi and V. Shoup. Algorithms in HElib. In *Advances in Cryptology–CRYPTO 2014*, pages 554–571. Springer, 2014.
33. S. Halevi and V. Shoup. Bootstrapping for HElib. In *Advances in Cryptology–EUROCRYPT 2015*, pages 641–670. Springer, 2015.
34. J. Hoffstein and J. Silverman. Optimizations for ntru. *Public-Key Cryptography and Computational Number Theory, Warsaw*, pages 77–88, 2001.
35. J. Kim, C. Lee, H. Shim, J. H. Cheon, A. Kim, M. Kim, and Y. Song. Encrypting controller using fully homomorphic encryption for security of cyber-physical systems. *IFAC-PapersOnLine*, 49(22):175–180, 2016.
36. M. Kim, Y. Song, and J. H. Cheon. Secure searching of biomarkers through hybrid homomorphic encryption scheme. *BMC medical genomics*, 10(2):42, 2017.
37. M. Kim, Y. Song, S. Wang, Y. Xia, and X. Jiang. Secure logistic regression based on homomorphic encryption. *To appear in JMIR medical informatics*, 2018.
38. K. Lauter, A. López-Alt, and M. Naehrig. Private computation on encrypted genomic data. In *International Conference on Cryptology and Information Security in Latin America*, pages 3–27. Springer, 2014.
39. M. Naehrig, K. Lauter, and V. Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 113–124. ACM, 2011.
40. M. S. Paterson and L. J. Stockmeyer. On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM Journal on Computing*, 2(1):60–66, 1973.
41. S. Wang, Y. Zhang, W. Dai, K. Lauter, M. Kim, Y. Tang, H. Xiong, and X. Jiang. Healer: Homomorphic computation of exact logistic regression for secure rare disease variants analysis in GWAS. *Bioinformatics*, 32(2):211–218, 2016.