Homomorphic Encryption for Arithmetic of Approximate Numbers

Jung Hee Cheon[1(✉)], Andrey Kim[1], Miran Kim[2], and Yongsoo Song[1]

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

Abstract. We suggest a method to construct a homomorphic encryption scheme for approximate arithmetic. It supports an approximate addition and multiplication of encrypted messages, together with a new *rescaling* procedure for managing the magnitude of plaintext. This procedure truncates a ciphertext into a smaller modulus, which leads to rounding of plaintext. The main idea is to add a noise following significant figures which contain a main message. This noise is originally added to the plaintext for security, but considered to be a part of error occurring during approximate computations that is reduced along with plaintext by rescaling. As a result, our decryption structure outputs an approximate value of plaintext with a predetermined precision.

We also propose a new batching technique for a RLWE-based construction. A plaintext polynomial is an element of a cyclotomic ring of characteristic zero and it is mapped to a message vector of complex numbers via complex canonical embedding map, which is an isometric ring homomorphism. This transformation does not blow up the size of errors, therefore enables us to preserve the precision of plaintext after encoding. In our construction, the bit size of ciphertext modulus grows linearly with the depth of the circuit being evaluated due to rescaling procedure, while all the previous works either require an exponentially large size of modulus or expensive computations such as bootstrapping or bit extraction. One important feature of our method is that the precision loss during evaluation is bounded by the depth of a circuit and it exceeds at most one more bit compared to unencrypted approximate arithmetic such as floating-point operations. In addition to the basic approximate circuits, we show that our scheme can be applied to the efficient evaluation of transcendental functions such as multiplicative inverse, exponential function, logistic function and discrete Fourier transform.

Keywords: Homomorphic encryption · Approximate arithmetic

1 Introduction

Homomorphic encryption (HE) is a cryptographic scheme that enables homomorphic operations on encrypted data without decryption. Many of HE schemes

T. Takagi and T. Peyrin (Eds.): ASIACRYPT 2017, Part I, LNCS 10624, pp. 409–437, 2017.
https://doi.org/10.1007/978-3-319-70694-8_15

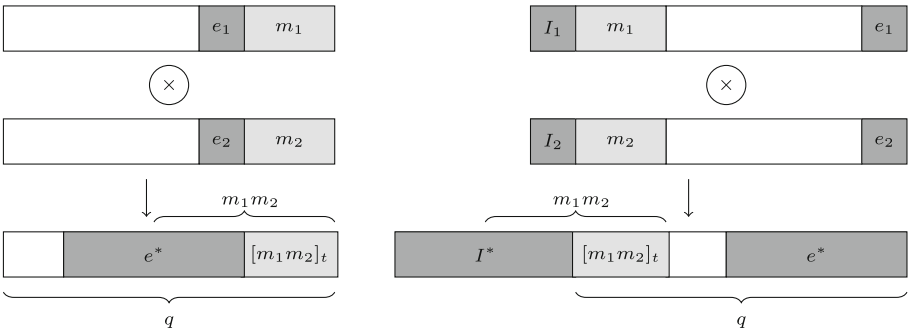

Fig. 1. Homomorphic multiplications of BGV-type HE schemes (left) and FV-type HE schemes (right)

(e.g. [2,4–7,12,13,18,19,21,25,26,33]) have been suggested following Gentry's blueprint [23]. HE can be applied to the evaluation of various algorithms on encrypted financial, medical, or genomic data [11,29,31,36,41].

Most of real-world data contain some errors from their true values. For instance, a measured value of quantity has an observational error from its true value and sampling error can be made as only a sample of the whole population is being observed in statistics. In practice, data should be discretized (quantized) to an approximate value such as floating-point number, in order to be represented by a finite number of bits in computer systems. In this case, an approximate value may substitute the original data and a small rounding error does not have too much effect on computation result. For the efficiency of approximate arithmetic, we store a few numbers of significant digits (e.g. most significant bits, MSBs) and carry out arithmetic operations between them. The resulting value should be rounded again by removing some inaccurate least significant bits (LSBs) to maintain the bit size of significant (mantissa).

Unfortunately this rounding operation has been considered difficult to perform on HE since it is not simply represented as a small-degree polynomial. Previous approaches to approximate arithmetic require similar multiplicative depth and complexity to the case of bootstrapping for extraction of MSBs [1,27]. Other methods based on exact integer operations [16,20] require an exponentially large bit size of ciphertext modulus with the depth of the circuit to ensure correctness.

We point out that the decryption structures of existing HE schemes are not appropriate for arithmetic of indiscreet spaces. For a plaintext modulus t and a ciphertext modulus q, BGV-type HE schemes [5,19,25,33] have a decryption structure of the form $\langle \boldsymbol{c}_i, sk \rangle = m_i + te_i \pmod q$. Therefore, the MSBs of $m_1 + m_2$ and $m_1 m_2$ are destroyed by inserted errors e_i during homomorphic operations. On the other hand, the decryption structure of FV-type HE schemes [2,4,22] is $\langle \boldsymbol{c}_i, sk \rangle = qI_i + (q/t)m_i + e_i$ for some I_i and e_i. Multiplication of two ciphertexts satisfies $\langle \boldsymbol{c}^*, sk \rangle = qI^* + (q/t)m_1 m_2 + e^*$ for $I^* = tI_1 I_2 + I_1 m_2 + I_2 m_1$ and $e^* \approx t(I_1 e_2 + I_2 e_1)$, so the MSBs of resulting message are also destroyed (see Fig. 1 for an illustration). HE schemes with matrix ciphertexts [21,26] support homomorphic operations over the integers

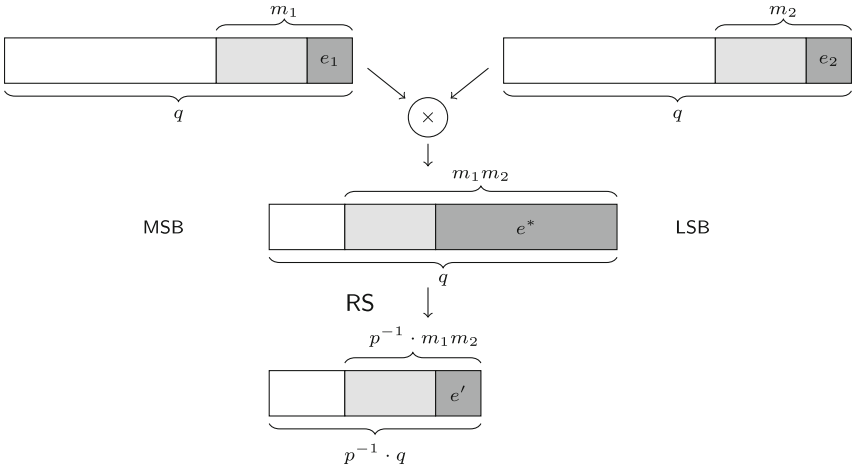

Fig. 2. Homomorphic multiplication and rescaling for approximate arithmetic

(or integral polynomials) but the error growth depends on the size of plaintexts. As a result, previous HE schemes are required to have an exponentially large ciphertext modulus with the depth of a circuit for approximate arithmetic.

Homomorphic Encryption for Approximate Arithmetic. The purpose of this paper is to present a method for efficient approximate computation on HE. The main idea is to treat an encryption noise as part of error occurring during approximate computations. That is, an encryption $\boldsymbol{c}$ of message m by the secret key sk will have a decryption structure of the form $\langle \boldsymbol{c}, sk \rangle = m + e \pmod{q}$ where e is a small error inserted to guarantee the security of hardness assumptions such as the learning with errors (LWE), the ring- LWE (RLWE) and the NTRU problems. If e is small enough compared to the message, this noise is not likely to destroy the significant figures of m and the whole value $m' = m + e$ can replace the original message in approximate arithmetic. One may multiply a scale factor to the message before encryption to reduce the precision loss from encryption noise.

For homomorphic operations, we always maintain our decryption structure small enough compared to the ciphertext modulus so that computation result is still smaller than q. However, we still have a problem that the bit size of message increases exponentially with the depth of a circuit without rounding. To address this problem, we suggest a new technique - called *rescaling* - that manipulates the message of ciphertext. Technically it seems similar to the modulus-switching method suggested by Brakerski and Vaikuntanatan [6], but it plays a completely different role in our construction. For an encryption $\boldsymbol{c}$ of m such that $\langle \boldsymbol{c}, sk \rangle = m + e \pmod{q}$, the rescaling procedure outputs a ciphertext $\lfloor p^{-1} \cdot \boldsymbol{c} \rceil \pmod{q/p}$, which is a valid encryption of m/p with noise about e/p. It reduces the size of ciphertext modulus and consequently removes the error located in the LSBs of messages, similar to the rounding step of fixed/floating-point arithmetic, while almost preserving the precision of plaintexts.

The composition of homomorphic operation and rescaling mimics the ordinary approximate arithmetic (see Fig. 2). As a result, the bit size of a required ciphertext modulus grows linearly with the depth of a circuit rather than exponentially. We also prove that this scheme is almost optimal in the sense of precision: precision loss of a resulting message is at most one bit more compared to unencrypted floating-point arithmetic.

Encoding Technique for Packing Messages. It is inevitable to encrypt a vector of multiple plaintexts in a single ciphertext for efficient homomorphic computation. The plaintext space of previous RLWE-based HE schemes is a cyclotomic polynomial ring $\mathbb{Z}_t[X]/(\Phi_M(X))$ of a finite characteristic. A plaintext polynomial could be decoded as a vector of plaintext values into a product of finite fields by a ring isomorphism [38,39]. An inserted error is placed separately from the plaintext space so it may be removed by using plaintext characteristic after carrying out homomorphic operations.

On the other hand, a plaintext of our scheme is an element of a cyclotomic ring of characteristic zero and it embraces a small error which is inserted from encryption to ensure the security or occurs during approximate arithmetic. Hence we adapt an *isometric ring homomorphism* - the complex canonical embedding map. It preserves the size of polynomials so that a small error in a plaintext polynomial is not blow up during encoding/decoding procedures.

Let $\mathbb{H} = \{(z_j)_{j\in\mathbb{Z}_M^*} : z_{-j} = \overline{z_j}, \forall j \in \mathbb{Z}_M^*\} \subseteq \mathbb{C}^{\Phi(M)}$ and let T be a subgroup of the multiplicative group $\mathbb{Z}_M^*$ satisfying $\mathbb{Z}_M^*/T = \{\pm 1\}$. The native plaintext space of our scheme is the set of polynomials in the cyclotomic ring $\mathcal{R} = \mathbb{Z}[X]/(\Phi_M(X))$ with magnitude bounded by ciphertext modulus. The decoding procedure first transforms a plaintext polynomial $m(X) \in \mathcal{R}$ into a complex vector $(z_j)_{j\in\mathbb{Z}_M^*} \in \mathbb{H}$ by the canonical embedding map σ and then sends it to a vector $(z_j)_{j\in T}$ using the natural projection $\pi\colon \mathbb{H} \to \mathbb{C}^{\phi(M)/2}$. The encoding method is almost the inverse of the decoding procedure, but a round-off algorithm is required for discretization so that the output becomes an integral polynomial. In short, our encoding function is given by

$$\begin{array}{ccccccc} \mathbb{C}^{\phi(M)/2} & \xrightarrow{\pi^{-1}} & \mathbb{H} & \xrightarrow{\lfloor\cdot\rceil_{\sigma(\mathcal{R})}} & \sigma(\mathcal{R}) & \xrightarrow{\sigma^{-1}} & \mathcal{R} \\ \boldsymbol{z} = (z_i)_{i\in T} & \longmapsto & \pi^{-1}(\boldsymbol{z}) & \longmapsto & \lfloor \pi^{-1}(\boldsymbol{z}) \rceil_{\sigma(\mathcal{R})} & \longmapsto & \sigma^{-1}\left(\lfloor \pi^{-1}(\boldsymbol{z}) \rceil_{\sigma(\mathcal{R})}\right) \end{array}$$

where $\lfloor\cdot\rceil_{\sigma(\mathcal{R})}$ denotes the rounding to a close element in $\sigma(\mathcal{R})$.

Homomorphic Evaluation of Approximate Arithmetic. One important feature of our method is that the precision loss during homomorphic evaluation is bounded by depth of a circuit and it is at most one more bit compared to unencrypted approximate arithmetic. Given encryptions of d messages with η bits of precision, our HE scheme of depth $\lceil \log d \rceil$ computes their product with $(\eta - \log d - 1)$ bits of precision in d multiplications while unencrypted approximate arithmetic such as floating-point multiplication can compute a significant with $(\eta - \log d)$ bits of precision. On the other hand, the previous methods require $\Omega(\eta^2 d)$ homomorphic computations by using bitwise encryption or need a large

plaintext space of bit size $\Omega(\eta d)$ unless relying on expensive computations such as bootstrapping or bit extraction.

In our scheme, the required bit size of the largest ciphertext modulus can be reduced down to $O(\eta \log d)$ by performing the rescaling procedure after multiplication of ciphertexts. The parameters are smaller than for the previous works and this advantage enables us to efficiently perform the approximate evaluation of *transcendental* functions such as the exponential, logarithm and trigonometric functions by the evaluation of their Taylor series expansion. In particular, we suggest a specific algorithm for computing the multiplicative inverse with reduced complexity, which enables the efficient evaluation of rational functions.

We verify our algorithms by implementation on a machine with an Intel Core i5 running at 2.9 GHz processor using a parameter set with 80-bit security level. It takes about 0.45 s for multiplicative inverse of ciphertext with 14 bits of precision, yielding an amortized rate of 0.11 ms per slot. We can also evaluate the exponential function using its Taylor expansion and it results in an amortized time per slots of 0.16 ms.

In a cloud-computing environment, a large amount of data is being generated and one needs to handle these huge data collections. Our scheme could be a practical solution for data analysis as it allows the encryption of much information in a single ciphertext so we can parallelize both space and computation together. For example, we improved the homomorphic evaluation of logistic function using a batching technique, which can be used in a disease prediction analysis. Our implementation homomorphically evaluated the degree seven Taylor polynomial of logistic function in about 0.13 ms per slot (and less than 0.54 s total) compared to 30 s and 1.8 s of evaluation time of [3,9] without parallelization, respectively.

Another example is evaluating discrete Fourier transform homomorphically using a *fast Fourier transform*(FFT) algorithm. We follow the encoding method of [15] for roots of unity in polynomial ring so that it does not consume ciphertext level during evaluation. We also apply our rescaling procedure for operations to Hadamard space and a batching technique, which results in a much smaller parameter and amortized evaluation time, respectively. We could process the standard processing (FFT-Hadamard product of two vectors-inverse FFT) of dimension 2^{13} in 22 min (0.34 s per slot) on a machine with four cores compared to 17 min of previous work [16] with six processors with no batching technique. Based on evaluation of discrete Fourier transform, we can securely compute the exact multiplication of integral polynomials by removing the fractional part of an approximate result. Likewise, our HE for approximate arithmetic can be applied to exact computation when the result has a specific format or property.

Follow-up. We provide an open-source implementation of our HE library (`HEAAN`) and algorithms in the C++ language. The source code is available at github [10]. We introduced `HEAAN` at a workshop for the standardization of HE hosted by Microsoft Research.[1]

[1] https://www.microsoft.com/en-us/research/event/homomorphic-encryption-standardization-workshop/.

There are some follow-up works on application of this paper to a secure control of cyber-physical system [28] and a gradient descent algorithm for privacy-preserving logistic regression of biomedical data [30].

Related Works. A substantial number of studies have concerned about the processing of real numbers over encryption. Jäschke and Armknecht [27] observed that a rational number can be approximated to an integer by multiplying with a power of two and rounding. An integer is encoded in a binary fashion, so that each bit is encrypted separately. The one performing homomorphic multiplication can bring the product to the required precision by simply discarding the ciphertexts which corresponds to the last LSBs. However, bitwise encryption causes a huge number of computation of ciphertexts for a single rounding operation. The other method is to scale them to integers, but a plaintext modulus is exponential in the length of message. For example, Arita and Nakasato [1] scale the fixed point numbers by a power of two and then represent them as scalars in a polynomial ring with an enlarged plaintext modulus. In order to realize homomorphic multiplication of encrypted fixed point numbers, it needs a right shift by a number equal to the precision. However, it requires a considerable amount of computations including a bit extraction operation.

On the other hand, Dowlin et al. [20] present an efficient method to represent fixed-point numbers, which are encoded as integral polynomials with coefficients in the range $(-\frac{1}{2}B, \frac{1}{2}B)$ using its base-B representation for an odd integer $B \geq 3$. Costache et al. [16] analyze the representations of [20] and compute the lower bound of plaintext modulus. However, exact arithmetic of fixed point numbers causes required the size of plaintext modulus to grow exponentially with the depth of a circuit.

Road-map. Section 2 briefly introduces notations and some preliminaries about algebras and the RLWE problem. Section 3 presents a homomorphic encryption scheme for approximate arithmetic and analyzes the noise growth during basic homomorphic operations. In Sect. 4, we suggest some algorithms to homomorphically evaluate typical approximate circuits, multiplicative inverse, exponential function, logistic function and discrete Fourier transform. We also compute the theoretical precision of the outputs. In Sect. 5, we perform the implementation of our scheme for the evaluations of circuits described in Sect. 4.

2 Preliminaries

2.1 Basic Notation

All logarithms are base 2 unless otherwise indicated. We denote vectors in bold, e.g. $\mathbf{a}$, and every vector in this paper will be a column vector. 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 a distribution D. It denotes the sampling

from the uniform distribution over D when D is a finite set. 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.

2.2 The Cyclotomic Ring and Canonical Embedding

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 cyclotomic ring $\mathcal{S} = \mathbb{R}[X]/(\Phi_M(X))$ of *real* polynomials will be represented as a polynomial $a(X) = \sum_{j=0}^{N-1} a_j X^j$ of degree less than N and identified with its coefficient vector $(a_0, \ldots, a_{N-1}) \in \mathbb{R}^N$. We define the relevant norms on the coefficient vector of a such as $\|a\|_\infty$ and $\|a\|_1$.

We 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 $a \in \mathbb{Q}[X]/(\Phi_M(X))$ into $\mathbb{C}^N$ is the vector of evaluation values of a at the roots of $\Phi_M(X)$. We naturally extend it to the set of real polynomials $\mathcal{S}$ so $\sigma(a)$ will be defined as $(a(\zeta_M^j))_{j \in \mathbb{Z}_M^*} \in \mathbb{C}^N$ for any $a \in \mathcal{S}$ where $\zeta_M = \exp(-2\pi i/M)$ denotes a primitive M-th roots of unity. The ℓ_∞-norm of $\sigma(a)$ is called the *canonical embedding norm* of a, denoted by $\|a\|_\infty^{\mathsf{can}} = \|\sigma(a)\|_\infty$. This measurement will be used to analyze the size of polynomials throughout this paper. The canonical embedding norm $\|\cdot\|_\infty^{\mathsf{can}}$ satisfies the following properties:

- For all $a, b \in \mathcal{S}$, we have $\|a \cdot b\|_\infty^{\mathsf{can}} \leq \|a\|_\infty^{\mathsf{can}} \cdot \|b\|_\infty^{\mathsf{can}}$.
- For all $a \in \mathcal{S}$, we have $\|a\|_\infty^{\mathsf{can}} \leq \|a\|_1$.
- There is a ring constant c_M depending only on M such that $\|a\|_\infty \leq c_M \cdot \|a\|_\infty^{\mathsf{can}}$ for all $a \in \mathcal{S}$.

The ring constant is obtained by $c_M = \|\mathsf{CRT}_M^{-1}\|_\infty$ where CRT_M is the CRT matrix for M, i.e., the Vandermonde matrix over the complex primitive M-th roots of unity, and the norm for a matrix $U = (u_{ij})_{0 \leq i,j < N}$ is defined by $\|U\|_\infty = \max_{0 \leq i < N} \left\{ \sum_{j=0}^{N-1} |u_{ij}| \right\}$. Refer [17] for a discussion of c_M.

2.3 Gaussian Distributions and RLWE Problem

We first define the space

$$\mathbb{H} = \{\boldsymbol{z} = (z_j)_{j \in \mathbb{Z}_M^*} \in \mathbb{C}^N : z_j = \overline{z_{-j}}, \forall j \in \mathbb{Z}_M^*\},$$

which is isomorphic to $\mathbb{R}^N$ as an inner product space via the unitary basis matrix

$$U = \begin{pmatrix} \frac{1}{\sqrt{2}} I & \frac{i}{\sqrt{2}} J \\ \frac{1}{\sqrt{2}} J & \frac{-i}{\sqrt{2}} I \end{pmatrix}$$

where I is the identity matrix of size $N/2$ and J is its reversal matrix.

For $r > 0$, we define the Gaussian function ρ_r: $\mathbb{H} \to (0, 1]$ as $\rho_r(\boldsymbol{z}) = \exp(-\pi\|\boldsymbol{z}\|_2^2/r^2)$. Denote by Γ_r the continuous Gaussian probability distribution whose density is given by $r^{-N} \cdot \rho_r(\boldsymbol{z})$. Now one can extend this to an elliptical Gaussian distribution $\Gamma_{\mathbf{r}}$ on $\mathbb{H}$ as follows: let $\mathbf{r} = (r_1, \ldots, r_N) \in (\mathbb{R}^+)^N$ be a vector of positive real numbers, then a sample from $\Gamma_{\mathbf{r}}$ is given by $U \cdot \boldsymbol{z}$ where each entry of $\boldsymbol{z} = (z_i)$ is chosen independently from the (one-dimensional) Gaussian distribution Γ_{r_i} on $\mathbb{R}$. This also gives a distribution $\Psi_{\mathbf{r}}$ on $\mathbb{Q}[X]/(\Phi_M(X)) \otimes \mathbb{R}$. That is, $\mathsf{CRT}_M^{-1} \cdot U \cdot \boldsymbol{z}$ gives us the coordinates with respect to the polynomial basis $1, X, X^2, \ldots, X^{N-1}$.

In practice, one can discritize the continuous Gaussian distribution $\Psi_{\mathbf{r}}$ by taking a valid rounding $\lfloor \Psi_{\mathbf{r}} \rceil_{\mathcal{R}^\vee}$. Refer [34,35] for explaining the methods in more details. We use this discrete distribution as the RLWE error distribution.

Here we define the RLWE distribution and decisional problem associated with it. Let $\mathcal{R}^\vee$ be the dual fractional ideal of $\mathcal{R}$ and write $\mathcal{R}_q^\vee = \mathcal{R}^\vee/q\mathcal{R}^\vee$. For a positive integer modulus $q \geq 2$, $s \in \mathcal{R}_q^\vee$, $\mathbf{r} \in (\mathbb{R}^+)^N$ and an error distribution $\chi := \lfloor \Psi_{\mathbf{r}} \rceil_{\mathcal{R}^\vee}$, we define $A_{N,q,\chi}(s)$ as the RLWE distribution obtained by sampling $a \leftarrow \mathcal{R}_q$ uniformly at random, $e \leftarrow \chi$ and returning $(a, a \cdot s + e) \in \mathcal{R}_q \times \mathcal{R}_q^\vee$.

The (decision) ring learning with errors, denoted by $\mathsf{RLWE}_{N,q,\chi}(\mathcal{D})$, is a problem to distinguish arbitrarily many independent samples chosen according to $A_{N,q,\chi}(s)$ for a random choice of s sampled from the distribution $\mathcal{D}$ over $\mathcal{R}^\vee$ from the same number of uniformly random and independent samples from $\mathcal{R}_q \times \mathcal{R}_q^\vee$.

3 Homomorphic Encryption for Approximate Arithmetic

In this section, we describe a method to construct a HE scheme for approximate arithmetic on encrypted data. Given encryptions of m_1 and m_2, this scheme allows us to securely compute encryptions of approximate values of $m_1 + m_2$ and $m_1 m_2$ with a predetermined precision. The main idea of our construction is to treat an inserted noise of RLWE problem as part of am error occurring during approximate computation. The most important feature of our scheme is the *rounding* operation of plaintexts. Just like the ordinary approximate computations using floating-point numbers, the rounding operation removes some LSBs of message and makes a trade-off between size of numbers and precision loss.

Our concrete construction is based on the BGV scheme [5] with a multiplication method by raising the ciphertext modulus [25], but our methodology can be applied to most of existing HE schemes. Appendix A shows a description of LWE-based HE scheme for approximate arithmetic.

3.1 Decryption Structure of Homomorphic Encryption for Approximate Arithmetic

Most of existing HE schemes perform operations on a modulo space such as $\mathbb{Z}_t$ and $\mathbb{Z}_t[X]/(\Phi_M(X))$. In other words, they aim to compute a ciphertext which encrypts some LSBs of a resulting message after homomorphic computation. For example, in the case of BGV-type schemes [5,25,33], plaintexts are placed

in the lowest bits of ciphertext modulus, that is, an encryption $\boldsymbol{c}$ of a message m with respect to a secret sk has a decryption structure of the form $\langle \boldsymbol{c}, sk \rangle = m + te \pmod{q}$. A multiplication of encryptions of m_1, m_2 preserves some LSBs of $m_1 m_2$ (i.e., $[m_1 m_2]_t$), while its MSBs (i.e., $\lfloor m_1 m_2 / t \rceil$) are destroyed by errors. On the other hand, FV-type schemes [2,4,22] put messages in the left-most bits of ciphertext modulus, so that their decryption structures satisfy $\langle \boldsymbol{c}, sk \rangle = \lfloor q/t \rceil \cdot m + e \pmod{q}$. However, the MSBs of the resulting message are also destroyed during homomorphic multiplication between $\langle \boldsymbol{c}_i, sk \rangle = q \cdot I_i + \lfloor q/t \rceil \cdot m_i + e_i$, each of which contains an additional error I_i in the left position of message.

Our goal is to carry out approximate arithmetic over encrypted data, or equivalently, compute the MSBs of a resulting message after homomorphic operations. The main idea is to add an encryption noise following significant figures of an input message. More precisely, our scheme has a decryption structure of the form $\langle \boldsymbol{c}, sk \rangle = m + e \pmod{q}$ for some small error e. We insert this encryption error to guarantee the security of scheme, but it will be considered as an error that arises during approximate computations. That is, the output of decryption algorithm will be treated as an approximate value of the original message with a high precision. The size of a plaintext will be small enough compared to the ciphertext modulus for homomorphic operations so that the result of an arithmetic computation is still smaller than the ciphertext modulus.

There are some issues that we need to consider more carefully. In unencrypted approximate computations, small errors may blow up when applying operations in succession, so it is valuable to consider the proximity of a calculated result to the exact value of an algorithm. Similarly, encrypted plaintexts in our scheme will contain some errors and they might be increased during homomorphic evaluations. Thus we compute an upper bound of errors and predict the precision of resulting values.

The management of the size of messages is another issue. If we compute a circuit of multiplicative depth L without rounding of messages, then the bit size of an output value will exponentially grow with L. This naive method is inappropriate for practical usage because it causes a huge ciphertext modulus. To resolve this problem, we suggest a new technique which divides intermediate values by a base. It allows us to discard some inaccurate LSBs of a message while an error is still kept relatively small compared to the message. This method leads to maintain the size of messages almost same and make the required ciphertext modulus linear in the depth L.

3.2 Plaintext Encoding for Packing

The batching technique in HE system allows us to encrypt multiple messages in a single ciphertext and enables a parallel processing in SIMD manner. In practice, we take its advantage to parallelize computations and reduce the memory and complexity. A ring of finite characteristic has been used as a plaintext space in the previous RLWE-based HE schemes. A small error, which is located in a separated place in a ciphertext modulus, is inserted to ensure security and it may be removed after carrying out homomorphic operations. Then an output

polynomial is decoded into a message vector with respect to the CRT-based encoding technique [38,39]. Meanwhile, a plaintext of our scheme is a polynomial contained in a ring of characteristic zero and it embraces am error for security, so an inserted error cannot be removed after decryption.

Intuition. A native plaintext space of our RLWE-based construction can be understood as the set of polynomials $m(X) \in \mathcal{S}$ such that $\|m\|_{\infty}^{\mathsf{can}} \ll q$. The roots of a cyclotomic polynomial $\Phi_M(X)$ are the complex primitive roots of unity in the extension field $\mathbb{C}$. We evaluate a plaintext polynomial at these roots in order to transform it into a vector of complex numbers, so the (extended) canonical embedding map $\sigma\colon \mathcal{S} \to \mathbb{C}^N$ plays a role of decoding algorithm.

For technical details, we first point out that the image of canonical embedding map is the subring $\mathbb{H} = \{(z_j)_{j\in\mathbb{Z}_M^*} : z_j = \overline{z_{-j}}\}$ of $\mathbb{C}^N$. Let T be a multiplicative subgroup of $\mathbb{Z}_M^*$ satisfying $\mathbb{Z}_M^*/T = \{\pm 1\}$. Then $\mathbb{H}$ can be identified with $\mathbb{C}^{N/2}$ via the natural projection π, defined by $(z_j)_{j\in\mathbb{Z}_M^*} \mapsto (z_j)_{j\in T}$. Then our decoding algorithm is to transform an arbitrary polynomial $m(X) \in \mathcal{R}$ into a complex vector $\boldsymbol{z}$ such that $\boldsymbol{z} = \pi \circ \sigma(m) \in \mathbb{C}^{N/2}$.

The encoding algorithm is defined as the inverse of decoding procedure. Specifically, it encodes an input vector $\boldsymbol{z} = (z_i)_{i\in T}$ in a polynomial $m(X) = \sigma^{-1} \circ \pi^{-1}(\boldsymbol{z})$ where $\pi^{-1}(\boldsymbol{z})[j]$ is z_j if $j \in T$, and $\overline{z_{-j}}$ otherwise. Note that the encoding/decoding algorithms are isometric ring isomorphisms between $(\mathcal{S}, \|\cdot\|_{\infty}^{\mathsf{can}})$ and $(\mathbb{C}^{N/2}, \|\cdot\|_{\infty})$, so the size of plaintexts and errors are preserved via these transformations.

Since $\pi^{-1}(\boldsymbol{z})$ might not be contained in the image of canonical embedding map, we need to discritize $\pi^{-1}(\boldsymbol{z})$ to an element of $\sigma(\mathcal{R})$. Recall that $\mathcal{R}$ has a $\mathbb{Z}$-basis $\{1, X, \ldots, X^{N-1}\}$ and it yields a rank-N ideal lattice $\sigma(\mathcal{R})$ having basis $\{\sigma(1), \sigma(X), \ldots, \sigma(X^{N-1})\}$. The goal of rounding process is to find a vector, denoted by $\lfloor \pi^{-1}(\boldsymbol{z}) \rceil_{\sigma(\mathcal{R})}$, with a rounding error $\|\pi^{-1}(\boldsymbol{z}) - \lfloor \pi^{-1}(\boldsymbol{z}) \rceil_{\sigma(\mathcal{R})}\|_{\infty}$. There are several round-off algorithms including the coordinate-wise randomized rounding. See [35] for details.

A rounding error may destroy the significant figures of a message during encoding procedure. Hence we recommend to multiply a scaling factor $\Delta \geq 1$ to a plaintext before rounding in order to preserve its precision. Our encoding/decoding algorithms are explicitly given as follows:

- $\mathsf{Ecd}(\boldsymbol{z}; \Delta)$. For a $(N/2)$-dimensional vector $\boldsymbol{z} = (z_i)_{i\in T}$ of complex numbers, the encoding procedure first expands it into the vector $\pi^{-1}(\boldsymbol{z}) \in \mathbb{H}$ and computes its discretization to $\sigma(\mathcal{R})$ after multiplying a scaling factor Δ. Return the corresponding integral polynomial $m(X) = \sigma^{-1}(\lfloor \Delta \cdot \pi^{-1}(\boldsymbol{z}) \rceil_{\sigma(\mathcal{R})}) \in \mathcal{R}$.
- $\mathsf{Dcd}(m; \Delta)$. For an input polynomial $m \in \mathcal{R}$, output the vector $\boldsymbol{z} = \pi \circ \sigma(\Delta^{-1} \cdot m)$, i.e., the entry of $\boldsymbol{z}$ of index $j \in T$ is $z_j = \Delta^{-1} \cdot m(\zeta_M^j)$.

As a toy example, let $M = 8$ (i.e., $\Phi_8(X) = X^4 + 1$) and $\Delta = 64$. Let $T = \{\zeta_8, \zeta_8^3\}$ for the root of unity $\zeta_8 = \exp(-2\pi i/8)$. For a given vector $\boldsymbol{z} = (3+4i, 2-i)$, the corresponding real polynomial $\frac{1}{4}(10 + 4\sqrt{2}X + 10X^2 + 2\sqrt{2}X^3)$ has evaluation values $3+4i$ and $2-i$ at ζ_8 and ζ_8^3, respectively. Then the output

of encoding algorithm is $m(X) = 160+91X+160X^2+45X^3 \leftarrow \mathsf{Ecd}(\boldsymbol{z}; \Delta)$, which is the closest integral polynomial to $64 \cdot \frac{1}{4}(10 + 4\sqrt{2}X + 10X^2 + 2\sqrt{2}X^3)$. Note that $64^{-1} \cdot (m(\zeta_8), m(\zeta_8^3)) \approx (3.0082 + 4.0026i, 1.9918 - 0.9974i)$ is approximate to the input vector $\boldsymbol{z}$ with a high precision.

3.3 Leveled Homomorphic Encryption Scheme for Approximate Arithmetic

The purpose of this subsection is to construct a leveled HE scheme for approximate arithmetic. For convenience, we fix a base $p > 0$ and a modulus q_0, and let $q_\ell = p^\ell \cdot q_0$ for $0 < \ell \leq L$. The integer p will be used as a base for scaling in approximate computation. For a security parameter λ, we also choose a parameter $M = M(\lambda, q_L)$ for cyclotomic polynomial. For a level $0 \leq \ell \leq L$, a ciphertext of level ℓ is a vector in $\mathcal{R}_{q_\ell}^k$ for a fixed integer k. Our scheme consists of five algorithm $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add}, \mathsf{Mult})$ with constants B_{clean} and $B_{\mathsf{mult}}(\ell)$ for noise estimation. For convenience, we will describe a HE scheme over the polynomial ring $\mathcal{R} = \mathbb{Z}[X]/(\Phi_M(X))$.

- $\mathsf{KeyGen}(1^\lambda)$. Generate a secret value sk, a public information pk for encryption, and a evaluation key evk.
- $\mathsf{Enc}_{pk}(m)$. For a given polynomial $m \in \mathcal{R}$, output a ciphertext $\boldsymbol{c} \in \mathcal{R}_{q_L}^k$. An encryption $\boldsymbol{c}$ of m will satisfy $\langle \boldsymbol{c}, sk \rangle = m + e \pmod{q_L}$ for some small e. The constant B_{clean} denotes an encryption bound, i.e., error polynomial of a fresh ciphertext satisfies $\|e\|_\infty^{\mathsf{can}} \leq B_{\mathsf{clean}}$ with an overwhelming probability.
- $\mathsf{Dec}_{sk}(\boldsymbol{c})$. For a ciphertext $\boldsymbol{c}$ at level ℓ, output a polynomial $m' \leftarrow \langle \boldsymbol{c}, sk \rangle \pmod{q_\ell}$ for the secret key sk.

Unlike the most of existing schemes, our scheme does not have a separate plaintext space from an inserted error. An output $m' = m + e$ of decryption algorithm is slightly different from the original message m, but it can be considered to be an approximate value for approximate computations when $\|e\|_\infty^{\mathsf{can}}$ is small enough compared to $\|m\|_\infty^{\mathsf{can}}$. The intuition of approximate encryption has been partially used previously, for example, a switching key for homomorphic multiplication in [4–6,12] or an evaluation key for the squashed decryption circuit in [13,18] are encrypted in a similar way.

The algorithms for homomorphic operations are required to satisfy the following properties.

- $\mathsf{Add}(\boldsymbol{c}_1, \boldsymbol{c}_2)$. For given encrypts of m_1 and m_2, output an encryption of $m_1 + m_2$. An error of output ciphertext is bounded by sum of two errors in input ciphertexts.
- $\mathsf{Mult}_{evk}(\boldsymbol{c}_1, \boldsymbol{c}_2)$. For a pair of ciphertexts $(\boldsymbol{c}_1, \boldsymbol{c}_2)$, output a ciphertext $\boldsymbol{c}_{\mathsf{mult}} \in \mathcal{R}_{q_\ell}^k$ which satisfies $\langle \boldsymbol{c}_{\mathsf{mult}}, sk \rangle = \langle \boldsymbol{c}_1, sk \rangle \cdot \langle \boldsymbol{c}_2, sk \rangle + e_{\mathsf{mult}} \pmod{q_\ell}$ for some polynomial $e_{\mathsf{mult}} \in \mathcal{R}$ with $\|e_{\mathsf{mult}}\|_\infty^{\mathsf{can}} \leq B_{\mathsf{mult}}(\ell)$.

We may adapt the techniques of existing HE schemes over the ring $\mathcal{R}$ to construct a HE scheme for approximate arithmetic. For example, the ring-based

BGV scheme [5], its variant with multiplication by raising ciphertext modulus [25] ($k = 2$), or the NTRU scheme [33] ($k = 1$) can be used as a base scheme. Our scheme has its own distinct and unique characteristic represented by the following *rescaling* procedure.

- $\mathsf{RS}_{\ell\to\ell'}(\mathbf{c})$. For a ciphertext $\boldsymbol{c} \in \mathcal{R}_{q_\ell}^k$ at level ℓ and a lower level $\ell' < \ell$, output the ciphertext $\boldsymbol{c}' \leftarrow \left\lfloor \frac{q_{\ell'}}{q_\ell}\boldsymbol{c} \right\rceil$ in $\mathcal{R}_{q_{\ell'}}^k$, i.e., $\boldsymbol{c}'$ is obtained by scaling $\frac{q_{\ell'}}{q_\ell}$ to the entries of $\boldsymbol{c}$ and rounding the coefficients to the closest integers. We will omit the subscript $\ell \to \ell'$ when $\ell' = \ell - 1$.

For an input ciphertext $\boldsymbol{c}$ of a message m such that $\langle \boldsymbol{c}, sk\rangle = m + e \pmod{q_\ell}$, the output ciphertext $\boldsymbol{c}'$ of rescaling procedure satisfies $\langle \boldsymbol{c}', sk\rangle = \frac{q_{\ell'}}{q_\ell}m + (\frac{q_{\ell'}}{q_\ell}e + e_{\mathsf{scale}}) \pmod{q_{\ell'}}$. Let $\boldsymbol{\tau} = \frac{q_{\ell'}}{q_\ell}\boldsymbol{c} - \boldsymbol{c}'$ and assume that an error polynomial $e_{\mathsf{scale}} = \langle \boldsymbol{\tau}, sk\rangle$ is bounded by some constant B_{scale}. Then the output ciphertext becomes an encryption of $\frac{q_{\ell'}}{q_\ell}m$ with a noise bounded by $\frac{q_{\ell'}}{q_\ell}\|e\|_\infty^{\mathsf{can}} + B_{\mathsf{scale}}$.

Technically this procedure is similar to the modulus-switching algorithm [5], but it has a completely different role in our construction. The rescaling algorithm divides a plaintext by an integer to remove some inaccurate LSBs as a rounding step in usual approximate computations using floating-point numbers or scientific notation. The magnitude of messages can be maintained almost the same during homomorphic evaluation, and thus the required size of the largest ciphertext modulus grows linearly with the depth of the circuit being evaluated.

Tagged Informations. A homomorphic operation has an effect on the size of plaintext and the growth of message and noise. Each ciphertext will be tagged with bounds of a message and an error in order to dynamically manage their magnitudes. Hence, a full ciphertext will be of the form $(\boldsymbol{c}, \ell, \nu, B)$ for a ciphertext vector $\boldsymbol{c} \in \mathcal{R}_{q_\ell}^k$, a level $0 \le \ell \le L$, an upper bound $\nu \in \mathbb{R}$ of message and an upper bound $B \in \mathbb{R}$ of noise. Table 1 shows the full description of our scheme and homomorphic operations for ciphertexts with tagged information.

Table 1. Description of our scheme

Enc_{pk} :	$m \mapsto (\mathbf{c}, L, \nu, B_{\mathsf{clean}})$ for some $\nu \ge \|m\|_\infty^{\mathsf{can}}$
Dec_{sk} :	$(\boldsymbol{c}, \ell, \nu, B) \mapsto (\langle \boldsymbol{c}, sk\rangle \pmod{q_\ell}, B)$
$\mathsf{RS}_{\ell\to\ell'}$:	$(\mathbf{c}', \ell, \nu, B) \mapsto (\mathbf{c}, \ell', p^{\ell'-\ell}\cdot\nu, p^{\ell'-\ell}\cdot B + B_{\mathsf{scale}})$
Add :	$((\boldsymbol{c}_1, \ell, \nu_1, B_1), (\boldsymbol{c}_2, \ell, \nu_2, B_2)) \mapsto (\mathbf{c}_{\mathsf{add}}, \ell, \nu_1 + \nu_2, B_1 + B_2)$
Mult_{evk} :	$((\mathbf{c}_1, \ell, \nu_1, B_1), (\mathbf{c}_2, \ell, \nu_2, B_2))$
	$\mapsto (\mathbf{c}_{\mathsf{mult}}, \ell, \nu_1\nu_2, \nu_1 B_2 + \nu_2 B_1 + B_1 B_2 + B_{\mathsf{mult}})$

Homomorphic Operations of Ciphertexts at Different Levels. When given encryptions $\boldsymbol{c}, \boldsymbol{c}'$ of m, m' belong to the different levels ℓ and $\ell' < \ell$, we should bring a ciphertext $\boldsymbol{c}$ at a larger level ℓ to the smaller level ℓ' before

homomorphic operation. There are two candidates: simple modular reduction and the RS procedure. It should be chosen very carefully by considering the scale of messages because the simple modular reduction $\boldsymbol{c} \mapsto \boldsymbol{c} \pmod{q_{\ell'}}$ preserves the plaintext while RS procedure changes the plaintext from m to $\frac{q_{\ell'}}{q_\ell} m$ as in Fig. 3. Throughout this paper, we perform simple modulus reduction to the smaller modulus before computation on ciphertexts at different levels unless stated otherwise.

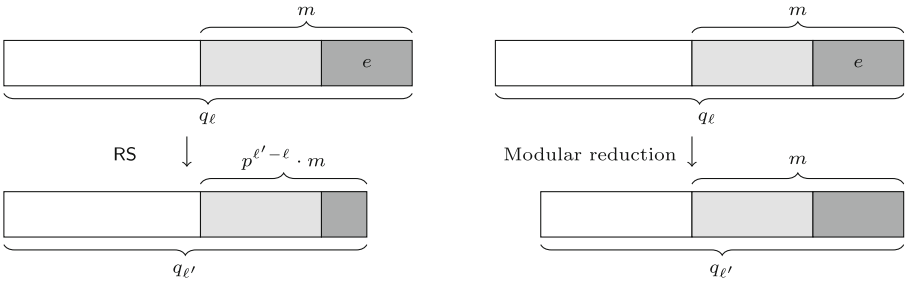

Fig. 3. Rescaling and simple modular reduction

3.4 Concrete Construction of RLWE-based HE Scheme

The performance of our construction and the noise growth depend on the base HE scheme. Moreover, a more accurate noise estimation can be done if we choose a specific one. We take the BGV scheme [5] with multiplication method by raising ciphertext modulus [25] as the underlying scheme of our concrete construction and implementation. From Costache and Smart's comparison [14], it seems to be the most efficient among the existing RLWE-based schemes.

For security and simplicity, we will use *power-of-two* degree cyclotomic rings. In this case, the dual ideal $\mathcal{R}^\vee = N^{-1} \cdot \mathcal{R}$ of $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ is simply a scaling of the ring. The RLWE problem is informally described by transforming samples $(a, b = a \cdot s' + e') \in \mathcal{R}_q \times \mathcal{R}_q^\vee$ into $(a, b = a \cdot s + e) \in \mathcal{R}_q \times \mathcal{R}_q$ where $s = s' \cdot N \in \mathcal{R}$ and $e = e' \cdot N \in \mathcal{R}$, so that the coefficients of e can be sampled independently from the discrete Gaussian distribution.

We will also choose the ring of Gaussian integers $\mathbb{Z}[i]$ as a discrete subspace of $\mathbb{C}$ for implementation. Another advantage of power-of-two degree cyclotomic rings is the efficient rounding operation $\lfloor \cdot \rceil_{\mathcal{R}^\vee}$ in dual fractional ideal $\mathcal{R}^\vee$. Since the columns of matrix CRT_M defined in Sect. 2.2 are mutually orthogonal, the encoding of plaintext can be efficiently done by rounding coefficients to the nearest integers after multiplication with the matrix CRT_M^{-1}.

We adopt the notation of some distributions on from [25]. For a real $\sigma > 0$, $\mathcal{DG}(\sigma^2)$ samples a vector in $\mathbb{Z}^N$ by drawing its coefficient independently from the discrete Gaussian distribution of variance σ^2. For an positive integer h, $\mathcal{HWT}(h)$ is the set of signed binary vectors in $\{0, \pm 1\}^N$ whose Hamming weight is exactly h. For a real $0 \leq \rho \leq 1$, the distribution $\mathcal{ZO}(\rho)$ draws each entry

in the vector from $\{0, \pm 1\}^N$, with probability $\rho/2$ for each of -1 and $+1$, and probability being zero $1 - \rho$.

- $\mathsf{KeyGen}(1^\lambda)$.
  - Given the security parameter λ, choose a power-of-two $M = M(\lambda, q_L)$, an integer $h = h(\lambda, q_L)$, an integer $P = P(\lambda, q_L)$ and a real value $\sigma = \sigma(\lambda, q_L)$.
  - Sample $s \leftarrow \mathcal{HWT}(h)$, $a \leftarrow \mathcal{R}_{q_L}$ and $e \leftarrow \mathcal{DG}(\sigma^2)$. Set the secret key as $sk \leftarrow (1, s)$ and the public key as $pk \leftarrow (b, a) \in \mathcal{R}_{q_L}^2$ where $b \leftarrow -as + e \pmod{q_L}$.
  - Sample $a' \leftarrow \mathcal{R}_{P \cdot q_L}$ and $e' \leftarrow \mathcal{DG}(\sigma^2)$. Set the evaluation key as $evk \leftarrow (b', a') \in \mathcal{R}_{P \cdot q_L}^2$ where $b' \leftarrow -a's + e' + Ps^2 \pmod{P \cdot q_L}$.
- $\mathsf{Ecd}(\boldsymbol{z}; \Delta)$. For a $(N/2)$-dimensional vector $\boldsymbol{z} = (z_j)_{j \in T} \in \mathbb{Z}[i]^{N/2}$ of Gaussian integers, compute the vector $\lfloor \Delta \cdot \pi^{-1}(\boldsymbol{z}) \rceil_{\sigma(\mathcal{R})}$. Return its inverse with respect to canonical embedding map.
- $\mathsf{Dcd}(m; \Delta)$. For an input polynomial $m(X) \in \mathcal{R}$, compute the corresponding vector $\pi \circ \sigma(m)$. Return the closest vector of Gaussian integers $\boldsymbol{z} = (z_j)_{j \in T} \in \mathbb{Z}[i]^{N/2}$ after scaling, i.e., $z_j = \left\lfloor \Delta^{-1} \cdot m(\zeta_M^j) \right\rceil$ for $j \in T$.
- $\mathsf{Enc}_{pk}(m)$. Sample $v \leftarrow \mathcal{ZO}(0.5)$ and $e_0, e_1 \leftarrow \mathcal{DG}(\sigma^2)$. Output $v \cdot pk + (m + e_0, e_1) \pmod{q_L}$.
- $\mathsf{Dec}_{sk}(\boldsymbol{c})$. For $\boldsymbol{c} = (b, a)$, output $b + a \cdot s \pmod{q_\ell}$.
- $\mathsf{Add}(\boldsymbol{c}_1, \boldsymbol{c}_2)$. For $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{R}_{q_\ell}^2$, output $\boldsymbol{c}_{\mathsf{add}} \leftarrow \boldsymbol{c}_1 + \boldsymbol{c}_2 \pmod{q_\ell}$.
- $\mathsf{Mult}_{evk}(\boldsymbol{c}_1, \boldsymbol{c}_2)$. For $\boldsymbol{c}_1 = (b_1, a_1), \boldsymbol{c}_2 = (b_2, a_2) \in \mathcal{R}_{q_\ell}^2$, let $(d_0, d_1, d_2) = (b_1 b_2,\ a_1 b_2 + a_2 b_1, a_1 a_2) \pmod{q_\ell}$. Output $\boldsymbol{c}_{\mathsf{mult}} \leftarrow (d_0, d_1) + \lfloor P^{-1} \cdot d_2 \cdot evk \rceil \pmod{q_\ell}$.
- $\mathsf{RS}_{\ell \to \ell'}(\boldsymbol{c})$. For $\boldsymbol{c} \in \mathcal{R}_{q_\ell}^2$, output $\boldsymbol{c}' \leftarrow \left\lfloor \frac{q_{\ell'}}{q_\ell} \boldsymbol{c} \right\rceil \in \pmod{q'_\ell}$.

Throughout this paper, we use non-integral polynomial as plaintext for convenience of analysis, so that a ciphertext $(\boldsymbol{c} \in \mathcal{R}_{q_\ell}^2, \ell, \nu, B)$ will be called a valid encryption of $m \in \mathcal{S}$ if $\|m\|_\infty^{\mathsf{can}} \le \nu$ and $\langle \boldsymbol{c}, sk \rangle = m + e \pmod{q_\ell}$ for some polynomial $e \in \mathcal{S}$ with $\|e\|_\infty^{\mathsf{can}} \le B$. The following lemmas give upper bounds on noise growth after encryption, rescaling and homomorphic operations. See Appendix B for proofs.

Lemma 1 (Encoding and Encryption). *Encryption noise is bounded by* $B_{\mathsf{clean}} = 8\sqrt{2}\sigma N + 6\sigma\sqrt{N} + 16\sigma\sqrt{hN}$. *If* $\boldsymbol{c} \leftarrow \mathsf{Enc}_{pk}(m)$ *and* $m \leftarrow \mathsf{Ecd}(\boldsymbol{z}; \Delta)$ *for some* $\boldsymbol{z} \in \mathbb{Z}[i]^{N/2}$ *and* $\Delta > N + 2B_{\mathsf{clean}}$, *then* $\mathsf{Dcd}(\mathsf{Dec}_{sk}(\boldsymbol{c})) = \boldsymbol{z}$.

Lemma 2 (Rescaling). *Let* $(\boldsymbol{c}, \ell, \nu, B)$ *be an encryption of* $m \in \mathcal{S}$. *Then* $(\boldsymbol{c}', \ell', p^{\ell'-\ell} \cdot \nu, p^{\ell'-\ell} \cdot B + B_{\mathsf{scale}})$ *is a valid encryption of* $p^{\ell'-\ell} \cdot m$ *for* $\boldsymbol{c}' \leftarrow \mathsf{RS}_{\ell \to \ell'}(\boldsymbol{c})$ *and* $B_{\mathsf{scale}} = \sqrt{N/3} \cdot (3 + 8\sqrt{h})$.

Lemma 3 (Addition/Multiplication). *Let* $(\boldsymbol{c}_i, \ell, \nu_i, B_i)$ *be encryptions of* $m_i \in \mathcal{S}$ *for* $i = 1, 2$, *and let* $\boldsymbol{c}_{\mathsf{add}} \leftarrow \mathsf{Add}(\boldsymbol{c}_1, \boldsymbol{c}_2)$ *and* $\boldsymbol{c}_{\mathsf{mult}} \leftarrow \mathsf{Mult}_{evk}(\boldsymbol{c}_1, \boldsymbol{c}_2)$. *Then* $(\boldsymbol{c}_{\mathsf{add}}, \ell, \nu_1 + \nu_2, B_1 + B_2)$ *and* $(\boldsymbol{c}_{\mathsf{mult}}, \ell, \nu_1 \nu_2, \nu_1 B_2 + \nu_2 B_1 + B_1 B_2 + B_{\mathsf{mult}}(\ell))$ *are valid encryptions of* $m_1 + m_2$ *and* $m_1 m_2$, *respectively, where* $B_{\mathsf{ks}} = 8\sigma N/\sqrt{3}$ *and* $B_{\mathsf{mult}}(\ell) = P^{-1} \cdot q_\ell \cdot B_{\mathsf{ks}} + B_{\mathsf{scale}}$.

Permutations over the Plaintext Slots. It is known that the Galois group $\mathcal{G}\mathsf{al} = \mathcal{G}\mathsf{al}(\mathbb{Q}(\zeta_M)/\mathbb{Q})$ consists of the mappings κ_k: $m(X) \mapsto m(X^k) \pmod{\Phi_M(X)}$ for a polynomial $m(X) \in \mathcal{R}$ and all k co-prime with M, and that it is isomorphic to $\mathbb{Z}_M^*$. As describe in [24], applying the transformation κ_k to the polynomials is very useful for the permutation on a vector of plaintext values.

For example, a plaintext polynomial $m(X)$ is decoded into a vector of evaluations at the specific points, i.e., $(m(\zeta_M^j))_{j\in T}$ for a subgroup T of $\mathbb{Z}_M^*$ satisfying $\mathbb{Z}_M^*/T = \{\pm 1\}$. For any $i, j \in T$, there is an element $\kappa_k \in \mathcal{G}\mathsf{al}$ which sends an element in the slot of index i to an element in the slot of index j. That is, for a vector of plaintext values $\boldsymbol{z} = (z_j)_{j\in T} \in \mathbb{C}^{N/2}$ with the corresponding polynomial $m(X) = \sigma^{-1} \circ \pi^{-1}(\boldsymbol{z})$, if $k = j^{-1} \cdot i \pmod{M}$ and $m' = \kappa_k(m)$, then we have $z'_j = m'(\zeta_M^j) = m(\zeta_M^{jk}) = m(\zeta_M^i) = z_i$. Hence the element in the slot of index j of m' is the same as that in the slot of index i of m.

Given an encryption $\boldsymbol{c}$ of a message $m \in \mathcal{R}$ with a secret key $sk = (1, s)$, we denote $\kappa_k(\boldsymbol{c})$ the vector obtained by applying κ_k to the entries of ciphertext $\boldsymbol{c}$. It follows from [24] that $\kappa_k(\boldsymbol{c})$ is a valid encryption of $\kappa_k(m)$ with respect to the secret $\kappa_k(s)$. In addition, the key-switching technique can be applied to the ciphertext $\kappa_k(\boldsymbol{c})$ in order to get an encryption of the same message with respect to the original secret s.

Relative Error. The decryption result of a ciphertext is an approximate value of plaintext, so the noise growth from homomorphic operations may cause some negative effect such as loss of significance. Hence it needs to dynamically manage the bound of noise of ciphertexts for a correct understanding of the outputs. A full ciphertext $(\boldsymbol{c}, \ell, \nu, B)$ contains upper bounds of plaintext and noise, but sometimes it is convenient to consider the relative error defined by $\beta = B/\nu$.

For example, it is easy to see that the addition of ciphertexts with relative errors $\beta_i = B_i/\nu_i$ produces a ciphertext with a relative error bounded by $\max_i\{\beta_i\}$. In other case, if we multiply two ciphertexts $(\boldsymbol{c}_1, \ell, \nu_1, B_1), (\boldsymbol{c}_2, \ell, \nu_2, B_2)$ and scale down to a lower level ℓ' (as floating-point multiplication does), it produces a ciphertext at level ℓ' with a relative error

$$\beta' = \beta_1 + \beta_2 + \beta_1\beta_2 + \frac{B_{\mathsf{mult}}(\ell) + p^{\ell-\ell'} \cdot B_{\mathsf{scale}}}{\nu_1\nu_2}$$

from Lemmas 2 and 3. This relative error is very close to $\beta_1 + \beta_2$ similar to the case of unencrypted floating-point multiplication under an appropriate choice of parameter and level.

4 Homomorphic Evaluation of Approximate Arithmetic

In this section, we describe some algorithms for evaluating some circuits commonly used in practical applications and analyze error growth of an output ciphertext based on our concrete construction. We start with the homomorphic evaluations of typical circuits such as addition and multiplication by constants,

monomial, and polynomial. These can be extended to approximate series for analytic functions such as multiplicative inverse and exponential function. The required parameters and precision of results will be also analyzed together.

For the convenience of analysis, we will assume that the term $\beta_1\beta_2 + (B_{\mathsf{mult}}(\ell) + p^{\ell-\ell'} \cdot B_{\mathsf{scale}})/(\nu_1\nu_2)$ is always bounded by a fixed constant β_*, so the relative error of ciphertext $\boldsymbol{c}' \leftarrow \mathsf{RS}_{\ell\to\ell'}(\mathsf{Mult}(\boldsymbol{c}_1, \boldsymbol{c}_2))$ satisfies the inequality $\beta' \le \beta_1 + \beta_2 + \beta_*$. We will discuss about the choice of β_* and check the validity of this assumption at the end of Sect. 4.1.

4.1 Polynomial Functions

The goal of this subsection is to suggest an algorithm for evaluating an arbitrary polynomial, and analyze its complexity and precision of output ciphertext. We start with the constant addition and multiplication functions $f(x) = x + a$ and $f(x) = ax$ for a constant $a \in \mathcal{R}$.

Lemma 4 (Addition/Multiplication by Constant). *Let $(\boldsymbol{c}, \ell, \nu, B)$ be an encryption of $m \in \mathcal{S}$. For a constant $a \in \mathcal{R}$, let $\boldsymbol{c}_{\mathsf{a}} \leftarrow \boldsymbol{c} + (a, 0) \pmod{q_\ell}$ and $\boldsymbol{c}_{\mathsf{m}} \leftarrow a \cdot \boldsymbol{c} \pmod{q_\ell}$. Then $(\boldsymbol{c}_{\mathsf{a}}, \ell, \nu + \|a\|_\infty^{\mathsf{can}}, B)$ and $(\boldsymbol{c}_{\mathsf{m}}, \ell, \|a\|_\infty^{\mathsf{can}} \cdot \nu, \|a\|_\infty^{\mathsf{can}} \cdot B)$ are valid encryptions of $m + a$ and am, respectively.*

Proof. There is a polynomial $e \in \mathcal{R}$ such that $\langle \boldsymbol{c}, sk\rangle = m + e \pmod{q_\ell}$ and $\|e\|_\infty^{\mathsf{can}} \le B$. It is obvious that $\langle \boldsymbol{c}_{\mathsf{a}}, sk\rangle = a + \langle \boldsymbol{c}, sk\rangle = (a + m) + e \pmod{q_\ell}$. We also have $\langle \boldsymbol{c}_{\mathsf{m}}, sk\rangle = a \cdot (m + e) = am + ae \pmod{q_\ell}$ and $\|a \cdot e\|_\infty^{\mathsf{can}} \le \|a\|_\infty^{\mathsf{can}} \cdot B$. □

Now we describe an algorithm to evaluate the power polynomial x^d for a power of two integer d. For simplicity, we assume that the bound ν of message m is equal to the base p.

For an input polynomial $m \in \mathcal{R}$ of size $\|m\|_\infty^{\mathsf{can}} \le p$, Algorithm 1 repeatedly performs the rescaling procedure after each of squaring step to maintain the size of message, thus the output of Algorithm 1 is an encryption of the scaled value $p \cdot f(m/p) = m^d/p^{d-1}$. The following lemma explains the correctness of Algorithm 1 and gives the relative error of the output ciphertext.

Algorithm 1. Power polynomial $f(x) = x^d$ of power-of-two degree

```
1: procedure POWER(c ∈ R²_{q_ℓ}, d = 2^r)
2:     c_0 ← c
3:     for j = 1 to r do
4:         c_j ← RS(Mult(c_{j−1}, c_{j−1}))
5:     end for
6:     return c_r
7: end procedure
```

Lemma 5. *Let $(\boldsymbol{c}, \ell, p, \beta_0 \cdot p)$ be an encryption of $m \in \mathcal{S}$ and d be a power-of-two integer. Then Algorithm 1 outputs a valid encryption $(\boldsymbol{c}_r, \ell - r, p, \beta_d \cdot p)$ of m^d/p^{d-1} for some real number $\beta_d \le d \cdot \beta_0 + (d-1) \cdot \beta_*$.*

Proof. We argue by induction on j. It is easy to see that $(\boldsymbol{c}_j, \ell - j, p, \beta_{2^j} \cdot p)$ is an encryption of m^{2^j}/p^{2^j-1} for some real number $\beta_{2^j} \le 2 \cdot \beta_{2^{j-1}} + \beta_*$. After r iterations, it produces an encryption $(\boldsymbol{c}_r, \ell - r, p, \beta_{2^r} \cdot p)$ of m^{2^r}/p^{2^r-1} for some β_{2^r} such that $\beta_{2^r} \le 2^r \cdot \beta_0 + (2^r - 1) \cdot \beta_*$. □

Algorithm 1 can be extended to an algorithm which evaluates an arbitrary polynomial. Similar to the previous case, this extended algorithm outputs an encryption of the scaled value $p \cdot f(m/p) = m^d/p^{d-1}$.

Lemma 6. *Let $(\boldsymbol{c}, \ell, p, B)$ be an encryption of $m \in \mathcal{S}$ and let d be a positive integer. Then one can compute a valid encryption $(\boldsymbol{c}', \ell - \lceil \log d \rceil, p, \beta_d \cdot p)$ of m^d/p^{d-1} for some real number $\beta_d \le d \cdot \beta_0 + (d-1) \cdot \beta_*$.*

Lemma 7 (Polynomial). *Let $f(x) = \sum_{j=0}^{d} a_j x^j$ be a nonzero polynomial of coefficients a_j in $\mathcal{R}$ and of degree d. Let $(\boldsymbol{c}, \ell, p, \beta_0 \cdot p)$ be an encryption of $m\mathcal{S}$. Then one can compute a valid encryption $(\boldsymbol{c}', \ell - \lceil \log d \rceil, M_f, \beta_d \cdot M_f)$ of $p \cdot f(m/p)$ for $M_f = p \cdot \sum_{j=0}^{d} \|a_j\|_\infty^{can}$ and for some real number $\beta_d \le d \cdot \beta_0 + (d-1) \cdot \beta_*$.*

If the relative error of input ciphertext satisfies $\beta_0 \le \beta_*$, the relative error of the resulting ciphertext is bounded by $\beta_d \le d \cdot \beta_0 + (d-1) \cdot \beta_* \le 2d \cdot \beta_0$. Hence, the precision loss is bounded by $(\log d + 1)$ bits, which is comparable to loss of significance occurring in unencrypted numerical computations. The evaluation of polynomial of degree d can be done in d homomorphic multiplications between ciphertext of depth $r = \lceil \log d \rceil$ by computing the encryptions of $m, m^2/p, \ldots, m^d/p^{d-1}$ simultaneously. We may apply the Paterson-Stockmeyer algorithm [37] to the evaluation procedure. Then a degree d polynomial can be evaluated using $O(\sqrt{d})$ multiplications, which gives a similar upper bound on relative error as the naive approach.

Let us return to the assumption $\beta_1\beta_2 + (B_{\mathsf{mult}}(\ell) + p^{\ell-\ell'} \cdot B_{\mathsf{scale}})/(\nu_1\nu_2) \le \beta_*$. We will choose β_* as an upper bound of relative errors of fresh ciphertexts in our scheme. After evaluation of circuits of depth less than $(L-1)$, the resulting ciphertext will have a relative error less than $2^L \cdot \beta_*$. It means that the first term $\beta_1\beta_2$ will be bounded by $2^{L+1} \cdot \beta_*^2$ after evaluation. The condition $2^{L+1} \cdot \beta_*^2 \le \frac{1}{2}\beta_*$, or equivalently $\beta_* \le 2^{-L-2}$, seems to be natural; otherwise the relative error becomes $2^{L+1} \cdot \beta_* \ge 2^{-1}$ after evaluation, so the decryption result will have almost no information. Thus we have $\beta_1 + \beta_2 \le \frac{1}{2}\beta_*$. The second term is equal to $(p^{\ell'-\ell} \cdot B_{\mathsf{mult}}(\ell) + B_{\mathsf{scale}})/\nu'$ where $\nu' = p^{\ell'-\ell} \cdot \nu_1\nu_2$ is the message bound of new ciphertext obtained by rescaling after multiplication. The numerator is asymptotically bounded by $p^{\ell'-\ell} \cdot B_{\mathsf{mult}}(\ell) + B_{\mathsf{scale}} = O(N)$. If the message bound always satisfies $\nu' \ge p$ as in our algorithms, the second term is $(B_{\mathsf{mult}}(\ell) + p^{\ell-\ell'} \cdot B_{\mathsf{scale}})/(\nu_1\nu_2) = O(p^{-1} \cdot N)$ which is smaller than a half of relative error of fresh ciphertext because $\beta_* \ge p^{-1} \cdot B_{\mathsf{clean}} = \Omega(p^{-1} \cdot \sigma N)$.

4.2 Approximate Polynomials and Multiplicative Inverse

We now homomorphically evaluate complex analytic functions $f(x)$ using their Taylor decomposition $f(x) = T_d(x) + R_d(x)$ for $T_d(x) = \sum_{j=0}^{d} \frac{f^{(j)}(0)}{j!} x^j$ and $R_d(x) = f(x) - T_d(x)$. Lemma 7 can be utilized to evaluate the rounded polynomial of scaled Taylor expansion $\lfloor p^u \cdot T_d \rceil(x)$ of $f(x)$ for some non-negative integers u and d, which outputs an approximate value of $p^{u+1} \cdot f(m/p)$. The bound of error is obtained by aggregating the error occurring during evaluation, the rounding error and the error of the remainder term $p^{u+1} \cdot R_d(m/p)$. In the case of RLWE-based constructions, we should consider the corresponding plaintext vector $\pi \circ \sigma(m) = (z_j)_{j \in T}$ and convergence of series in each slot.

As an example, the exponential function $f(x) = \exp(x)$ has the Taylor polynomial $T_d(x) = \sum_{j=0}^{d} \frac{1}{j!} x^j$ and the remaining term is bounded by $|R_d(x)| \leq \frac{e}{(d+1)!}$ when $|x| \leq 1$. Assume that we are given an encryption $(\boldsymbol{c}, \ell, p, \beta_0 \cdot p)$ of m. With the input ciphertext $\boldsymbol{c}$ and the polynomial $\lfloor p^u \cdot T_d \rceil(x)$, one can compute an encryption of $p^{u+1} \cdot T_d(m/p)$. We see that an error of the resulting ciphertext is bounded by

$$dp + p^{u+1} \cdot \sum_{j=1}^{d} \frac{1}{j!}(j \cdot \beta_0 + (j-1)\beta_*) \leq dp + p^{u+1} \cdot (e\beta_0 + \beta_*).$$

If we write $\exp(m/p) := \sigma^{-1} \circ \pi^{-1}((\exp(z_j/p))_{j \in T})$, the output ciphertext can be also viewed as an encryption of $p^{u+1} \cdot \exp(m/p)$ of the form $(\boldsymbol{c}', \ell - \lceil \log d \rceil, \nu', B')$ for $\nu' = p^{u+1} \cdot e$ and $B' = dp + p^{u+1} \cdot (e\beta_0 + \beta_* + \frac{e}{(d+1)!})$, and its relative error is bounded by $\beta' \leq (\beta_0 + \beta_* \cdot e^{-1}) + (p^{-u} \cdot d \cdot e^{-1} + \frac{1}{(d+1)!})$. If $\beta_0 \geq \beta_*$, then we may take integers d and u satisfying $(d+1)! \geq 4\beta_0^{-1}$ and $p^u \geq 2\beta_0^{-1} \cdot d$ to make the relative error less than $2\beta_0$. In this case, the precision loss during evaluation of exponential function is less than one bit.

In the case of multiplicative inverse, we adopt an algorithm described in [8] to get a better complexity. Assuming that a complex number x satisfies $|\hat{x}| \leq 1/2$ for $\hat{x} = 1 - x$, we get

$$x(1+\hat{x})(1+\hat{x}^2)(1+\hat{x}^{2^2}) \cdots (1+\hat{x}^{2^{r-1}}) = 1 - \hat{x}^{2^r}. \tag{1}$$

Note that $|\hat{x}^{2^r}| \leq 2^{-2^r}$, and it converges to one as r goes to infinity. Hence, $\prod_{j=0}^{r-1}(1+\hat{x}^{2^j}) = x^{-1}(1-\hat{x}^{2^r})$ can be considered as an approximate multiplicative inverse of x with 2^r bits of precision.

For homomorphic evaluation, we change a scale and assume that a complex number z_j satisfies $|\hat{z_j}| \leq p/2$ for $\hat{z_j} = p - z_j$. The standard approach starts by normalizing those numbers to be in the unit interval by setting $x = z_j/p$. Since we cannot multiply fractions over encrypted data, the precision point should move to the left for each term of (1). That is, we multiply both sides of the Eq. (1) by p^{2^r} and then it yields

$$z_j(p+\hat{z_j})(p^{2^1}+\hat{z_j}^{2^1})(p^{2^2}+\hat{z_j}^{2^2}) \cdots (p^{2^{r-1}}+\hat{z_j}^{2^{r-1}}) = p^{2^r} - \hat{z_j}^{2^r}.$$

Therefore, the product $p^{-2^r} \cdot \prod_{i=0}^{r-1}(p^{2^i} + \hat{z_j}^{2^i})$ can be seen as the approximate inverse of z_j with 2^r bits of precision. Let $\hat{\boldsymbol{z}} = (\hat{z_j})_{j\in T}$ and $\boldsymbol{z}^{-1} = (z_j^{-1})_{j\in T}$. Algorithm 2 takes an encryption of $\hat{m} = \sigma^{-1} \circ \pi^{-1}(\hat{\boldsymbol{z}})$ as an input and outputs an encryption of its scaled multiplicative inverse $p^2 \cdot (\sigma^{-1} \circ \pi^{-1}(\boldsymbol{z}^{-1}))$ by evaluating the polynomial $\prod_{j=0}^{r-1}(p^{2^j} + \hat{m}^{2^j})$. The precision of the resulting ciphertext and the optimal iterations number r will be analyzed in the following lemma.

Algorithm 2. Inverse function $f(x) = x^{-1}$

procedure INVERSE($\boldsymbol{c} \in \mathcal{R}^2_{q_\ell}, r$)
  $\boldsymbol{p} \leftarrow (p, 0)$
  $\boldsymbol{c}_0 \leftarrow \boldsymbol{c}$
  $\boldsymbol{v}_1 \leftarrow \boldsymbol{p} + \boldsymbol{c}_0 \pmod{q_{\ell-1}}$
  for $j = 1$ to $r - 1$ **do**
    $\boldsymbol{c}_j \leftarrow \mathsf{RS}(\mathsf{Mult}(\boldsymbol{c}_{j-1}, \boldsymbol{c}_{j-1}))$
    $\boldsymbol{v}_{j+1} \leftarrow \mathsf{RS}(\mathsf{Mult}(\boldsymbol{v}_j, \boldsymbol{p} + \boldsymbol{c}_j))$
  end for
  return $\boldsymbol{v}_r$
end procedure

Lemma 8 (Multiplicative Inverse). *Let $(\boldsymbol{c}, \ell, p/2, B_0 = \beta_0 \cdot p/2)$ be an encryption of $\hat{m} \in \mathcal{S}$ and let $m = p - \hat{m}$. Then Algorithm 2 outputs a valid encryption $(\boldsymbol{v}_r, \ell - r, 2p, \beta \cdot 2p)$ of $m' = p \cdot \prod_{i=0}^{r-1}(1 + (\hat{m}/p)^{2^i})$ for some $\beta \le \beta_0 + r\beta_*$.*

Proof. From Lemma 4, $(\boldsymbol{v}_1, \ell - 1, 3p/2, B_0)$ is a valid encryption of $p + \hat{m}$ and its relative error is $\beta'_1 = \beta_0/3$. It also follows from Lemma 5 that $(\boldsymbol{c}_j, \ell - j, 2^{-2^j} \cdot p, \beta_j \cdot 2^{-2^j} \cdot p)$ is a valid encryption of $\hat{m}^{2^j}/p^{2^j-1}$ for some real number $\beta_j \le 2^j \cdot (\beta_0 + \beta_*)$, and so $(\boldsymbol{p} + \boldsymbol{c}_j, \ell - j, (1 + 2^{-2^j})p, \beta_j \cdot 2^{-2^j} \cdot p)$ is a valid encryption of $p + \hat{m}^{2^j}/p^{2^j-1} = (p^{2^j} + \hat{m}^{2^j})/p^{2^j-1}$ with a relative error $\beta'_j \le \beta_j/(2^{2^j} + 1) \le 2^j \cdot (\beta_0 + \beta_*)/(2^{2^j} + 1)$, respectively.

Using the induction on j, we can show that

$$\left(\boldsymbol{v}_j, \ell - j, p \cdot \prod_{i=0}^{j-1}(1 + 2^{-2^i}), \beta''_j \cdot p \cdot \prod_{i=0}^{j-1}(1 + 2^{-2^i})\right)$$

is a valid encryption of $\prod_{i=0}^{j-1}(p^{2^i} + \hat{m}^{2^i})/p^{2^j-2} = p \cdot \prod_{i=0}^{j-1}(1 + (\hat{m}/p)^{2^i})$ with a relative error $\beta''_j \le \sum_{i=0}^{j-1} \beta'_i + (j-1) \cdot \beta_*$. Note that the message is bounded by $p \cdot \prod_{i=0}^{j-1}(1 + 2^{-2^i}) = (2p) \cdot (1 - 2^{-2^j}) < 2p$ and the relative error satisfies

$$\beta''_j \le \left(\sum_{i=0}^{j-1} \frac{2^i}{2^{2^i} + 1}\right) \cdot (\beta_0 + \beta_*) + (j-1) \cdot \beta_* \le \beta_0 + j \cdot \beta_*$$

from the fact that $\sum_{i=0}^{\infty} \frac{2^i}{2^{2^i}+1} = 1$. Therefore, the output $\boldsymbol{v}_r$ of Algorithm 2 represents a valid encryption $(\boldsymbol{v}_r, \ell - r, 2p, \beta \cdot 2p)$ of $m' = p \cdot \prod_{i=0}^{r-1}(1 + (\hat{m}/p)^{2^i})$ for some $\beta \le \beta_0 + r \cdot \beta_*$. □

Let $m^{-1}(X) := \sigma^{-1} \circ \pi^{-1}(\boldsymbol{z}^{-1})$ be the polynomial in $\mathcal{S}$ corresponding to $\boldsymbol{z}^{-1}$. The output ciphertext $(\boldsymbol{v}_r, \ell - r, 2p, \beta \cdot 2p)$ of the previous lemma can be also viewed as an encryption of $p^2 \cdot m^{-1}$. The error bound is increased by the convergence error $\|p^2 \cdot m^{-1} - m'\|_{\infty}^{\mathsf{can}} = \|p^2 \cdot m^{-1} \cdot (\hat{m}/p)^{2^r}\|_{\infty}^{\mathsf{can}} \le 2^{-2^r} \cdot 2p$. Therefore, the ciphertext $(\boldsymbol{v}_r, \ell - r, 2p, (\beta + 2^{-2^r}) \cdot 2p)$ is a valid encryption of m' and its relative error is $\beta + 2^{-2^r} \le \beta_0 + r\beta_* + 2^{-2^r}$, which is minimized when $r\beta_* \approx 2^{-2^r}$. Namely, $r = \lceil \log\log \beta_*^{-1} \rceil$ yields the inequality $\beta_0 + r\beta_* + 2^{-2^r} \le \beta_0 + 2r\beta_* = \beta_0 + 2\lceil \log\log \beta_*^{-1} \rceil \cdot \beta_*$. Thus the precision loss during evaluation of multiplicative inverse is less than one bit if $2\lceil \log\log \beta_*^{-1} \rceil \cdot \beta_* \le \beta_0$.

The optimal iterations number r can be changed upon more/less information about the magnitude of $\hat{m}$. Assume that we have an encryption of message $\hat{m}$ whose size is bounded by $\|\hat{m}\|_{\infty}^{\mathsf{can}} \le \epsilon p$ for some $0 < \epsilon < 1$. By applying Lemma 8, we can compute an encryption of $p \cdot \prod_{i=0}^{r-1}(1 + (\hat{m}/p)^{2^i}) = (p^2 \cdot m^{-1}) \cdot (1 - (\hat{m}/p)^{2^r})$ with a relative error $\beta \le \beta_0 + r\beta_*$, which is an approximate value of $p^2 \cdot m^{-1}$ with an error bounded by $\epsilon^{2^r} \cdot 2p$. Then the optimal iterations number is $r \approx \log\log \beta_*^{-1} - \log\log \epsilon^{-1}$ and the relative error becomes $\beta \le \beta_0 + 2\lceil (\log\log \beta_*^{-1} - \log\log \epsilon^{-1}) \rceil \cdot \beta_*$ when $r = \lceil (\log\log \beta_*^{-1} - \log\log \epsilon^{-1}) \rceil$.

4.3 Fast Fourier Transform

Let d be a power of two integer and consider the complex primitive d-th root of unity $\zeta_d = \exp(2\pi i/d)$. For a complex vector $\boldsymbol{u} = (u_0, \ldots, u_{d-1})$, its discrete Fourier transform (DFT) is defined by the vector $\boldsymbol{v} = (v_0, \ldots, v_{d-1}) \leftarrow \mathrm{DFT}(\boldsymbol{u})$ where $v_k = \sum_{j=0}^{d-1} \zeta_d^{jk} \cdot u_j$ for $k = 0, \ldots, d-1$. The DFT has a numerous applications in mathematics and engineering such as signal processing technology. The basic idea is to send the data to Fourier space, carry out Hadamard operations and bring back the computation result to a original domain via the inverse DFT. We denote by $W_d(z) = (z^{j \cdot k})_{0 \le j,k < d}$ the Vandermonde matrix generated by $\{z^k \colon 0 \le k < d\}$. The DFT of $\boldsymbol{u}$ can be evaluated by the matrix multiplication $\mathrm{DFT}(\boldsymbol{u}) = W_d(\zeta_d) \cdot \boldsymbol{u}$, but the complexity of DFT can be reduced down to $O(d \log d)$ using FFT algorithm by representing the DFT matrix $W_d(\zeta_d)$ as a product of sparse matrices.

Recently, Costache et al. [15] suggested an encoding method which sends the complex d-th root of unity to the monomial $Y = X^{M/d}$ over cyclotomic ring $\mathcal{R} = \mathbb{Z}[X]/(\Phi_M(X))$ for cryptosystem. Then homomorphic evaluation of DFT is simply represented as a multiplication of the matrix $W_d(Y)$ to a vector of ciphertexts over polynomial ring.

On the other hand, our RLWE-based HE scheme can take advantage of batch technique as described in Sect. 3.2. In the slot of index $k \in T$, the monomial $Y = X^{M/d}$ and matrix $W_d(Y)$ are converted into ζ_d^k and the DFT matrix $W_d(\zeta_d^k)$, respectively, depending on primitive root of unity ζ_d^k. However,

our batching scheme is still meaningful because the evaluation result of whole pipeline consisting of DFT, Hadamard operations, and inverse DFT is independent of index k, even though $W_d(Y)$ corresponds to the DFT matrices generated by different primitive d-th roots of unity.

It follows from the property of ordinary FFT algorithm that if $(\boldsymbol{c}_i, \ell, \nu, B)$ is an encryption of u_i for $i = 0, \ldots, d-1$ and $\boldsymbol{v} = (v_0, \ldots, v_{d-1}) \leftarrow W_d(Y) \cdot \boldsymbol{u}$, then the output of FFT algorithm using $X^{M/d}$ instead of ζ_d forms valid encryptions $(\boldsymbol{c}'_i, \ell, \sqrt{d} \cdot \nu, \sqrt{d} \cdot B)$. Note that the precision of input ciphertexts is preserved as B/ν. Our FFT algorithm takes a similar time with [15] in the same parameter setting, but the amortized time is much smaller thanks to our own plaintext packing technique. In the evaluation of whole pipeline DFT-Hadamard multiplication-inverse DFT, one may scale down the transformed ciphertexts by $\sqrt{d}$ before Hadamard operations to maintain the magnitude of messages and reduce the required levels for whole pipeline.

The fast polynomial multiplication using the FFT algorithm is a typical example that computes the exact value using approximate arithmetic. In particular for the case of integral polynomials, the exact multiplication can be recovered from its approximate value since we know that their multiplication is also an integral polynomial. Likewise, when the output of a circuit has a specific format or property, it is possible to get the exact computation result from its sufficiently close approximation.

5 Implementation Results

In this section we describe how to select parameters for evaluating arithmetic circuits described in Sect. 4. We also provide implementation results with concrete parameters. Our implementation is based on the NTL C++ library running over GMP. Every experimentation was performed on a machine with an Intel Core i5 running at 2.9 GHz processor using a parameter set with 80-bit security level.

We need to set the ring dimension N that satisfies the security condition $N \geq \frac{\lambda+110}{7.2} \log(P \cdot q_L)$ to get λ-bit security level. [25,32] We note that $P \cdot q_L$ is the largest modulus to generate evaluation key and it suffices to assume that P is approximately equal to q_L. In our implementation, we used the Gaussian distribution of standard deviation $\sigma = 3.2$ to sample error polynomials, and set $h = 64$ as the number of nonzero coefficients in a secret key $s(X)$.

Evaluation of Typical Circuits. In Table 2, we present the parameter setting and performance results for computing a power of a ciphertext, the multiplicative inverse of a ciphertext and exponential function. The average running times are only for ciphertext operations, excluding encryption and decryption procedures. As described in Sect. 3.4, each ciphertext can hold $N/2$ plaintext slots and one can perform the computation in parallel in each slot. Here the amortized running time means a relative time per slot.

The homomorphic evaluation of the circuit x^{1024} with an input of 36-bit precision is hard to be implemented in practice over previous methods. Meanwhile,

our scheme can compute this circuit simultaneously over 2^{14} slots in about 7.46 s, yielding an amortized rate of 0.43 ms per slot. Computation of the multiplicative inverse is done by evaluating the polynomial up to degree 8 as described in Algorithm 2. It gives an amortized time per slots of about 0.11 ms. In the case of exponential function, we used terms in its Taylor expansion up to degree 8 and it results in an amortized time per slots of 0.16 ms.

Table 2. Implementation results for homomorphic evaluation of typical circuits

Function	N	$\log q$	$\log p$	Consumed levels	Input precision	Total time	Amortized time
x^{16}	2^{13}	155	30	4	14 bits	0.31 s	0.07 ms
x^{-1}						0.45 s	0.11 ms
$\exp(x)$						0.65 s	0.16 ms
x^{1024}	2^{15}	620	56	10	36 bits	7.46 s	0.43 ms

Significance Loss. In Sect. 4, we analyzed the theoretical upper bounds on the growth of relative errors during evaluations. We can see from experimental result that initial precision is about 4 bits greater than theoretic bound of precision since we multiply 16 to the variance of encryption error to get a high probability bound. In Fig. 4, we depict bit precisions of output ciphertexts during the evaluation of homomorphic multiplications (e.g. x^{16} for the left figure and x^{1024} for the right figure). We can actually check that both theoretic bound and experimental result of precision loss during homomorphic multiplications is less than 4.1 (or resp. 10.1) when the depth of the circuit is 4 (or resp. 10).

Logistic Function. Let us consider the logistic function $f(x) = (1 + \exp(-x))^{-1}$, which is widely used in statistics, neural networks and machine

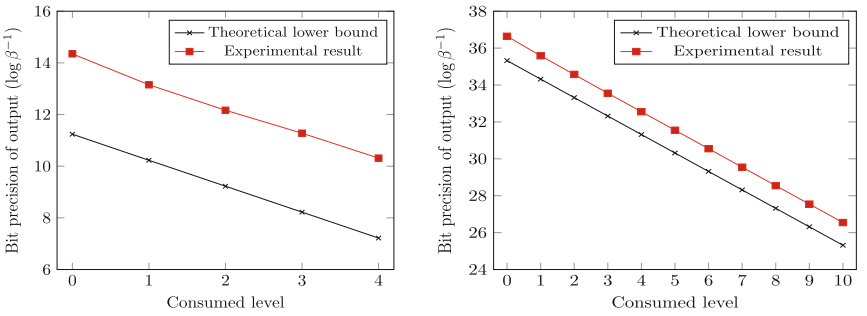

Fig. 4. The variation of bit precision of ciphertexts when $(f(x), N, \log p, \log q) = (x^{16}, 2^{13}, 30, 155)$ and $(x^{1024}, 2^{15}, 56, 620)$

learning as a probability function. For example, logistic regression is used for a prediction of the likelihood to have a heart attack in an unspecified period for men, as indicated in [3]. It was also used as a predictive equation to screen for diabetes, as described in [40]. This function can be approximated by its Taylor series

$$f(x) = \frac{1}{2} + \frac{1}{4}x - \frac{1}{48}x^3 + \frac{1}{480}x^5 - \frac{17}{80640}x^7 + \frac{31}{1451520}x^9 + O(x^{11}).$$

In [3,9], every real number is scaled by a predetermined factor to transform it as a binary polynomial before computation. The plaintext modulus t should be set large enough so that no reduction modulo t occurs in the plaintext space. The required bit size of plaintext modulus exponentially increases on the depth of the circuit, which strictly limits the performance of evaluation. On the other hand, the rescaling procedure in our scheme has the advantage that it significantly reduces the size of parameters (e.g. $(\log p, \log q) = (30, 155)$).

The parallelized computation for logistic function is especially important in real world applications such as statistic analysis using multiple data. In previous approaches, each slot of plaintext space should represent a larger degree than encoded polynomials so they could support only a few numbers of slots. On the other hand, we provide a parallelization method with an amortization amount independent from target circuit and get a better amortized time of evaluation (Table 3).

Table 3. Comparison of implementation results for homomorphic evaluation of logistic function

Method	N	$\log q$	Polynomial degree	Amortization amount	Total time	Amortized time
[3]	2^{14}	512	7	-	>30 s	-
[9]	17430	370	7	-	1.8 s	-
Ours	2^{13}	155	7	2^{12}	0.54 s	0.13 ms
	2^{14}	185	9	2^{13}	0.78 s	0.09 ms

Discrete Fourier Transform. With the parameters $(N, \log p) = (2^{13}, 50)$, we encrypt coefficients of polynomials and homomorphically evaluate the standard processing (FFT-Hadamard product of two vectors-inverse FFT) in 73 min (amortized 1.06 s per slot) when $d = 2^{13}$. We could reduce down the evaluation time to 22 min (amortized 0.34 s per slot) by adapting the multi-threading method on a machine with four cores, compared to 17 min of the previous work [15] on six cores. Since the rescaling procedure of transformed ciphertexts enables us to efficiently carry out higher degree Hadamard operations in Fourier space, the gap of parameter and running time between our scheme and previous methods grows very quickly as degree N and the depth of Hadamard operation increase. For instance, we also homomorphically evaluate the product

of 8 polynomials, using pipeline consisting of FFT-Hadamard product of eight vectors-inverse FFT with parameters $(N, \log q) = (2^{14}, 250)$ in amortized time of 1.76 s (Table 4).

Table 4. Comparison of implementation results for homomorphic evaluation of a full image processing pipeline

Method	d	N	$\log q$	Degree of Hadamard operation	Amortization amount	Total time	Amortized time
[15][a]	2^4	2^{13}	150	2	-	0.46 s	-
	2^{13}	2^{14}	192	2	-	17 min	-
Ours[b]	2^4	2^{13}	120	2	2^{12}	0.88 s	0.21 ms
	2^{13}	2^{13}	130	2	2^{12}	22 min	0.34 s
	2^{13}	2^{14}	250	8	2^{13}	4 h	1.76 s

[a]This experiment was done on a machine with six Intel Xeon E5 2.7 GHz processors with 64 GB RAM.
[b]This experiment was done on a machine with four Intel Core i7 2.9 GHz processors with 16 GB RAM.

6 Conclusion

In this work, we presented a homomorphic encryption scheme which supports an approximate arithmetic over encryption. We also introduced a new batching technique for packing much information in a single ciphertext, so we could achieve practical performance advantage by parallelism. Another benefit of our scheme is the rescaling procedure, which enables us to preserve the precision of the message after approximate computation. Furthermore, it leads to reduce the size of ciphertext significantly so the scheme can be a reasonable solution for computation over large integers.

The primary open problem is finding way to convert our scheme to a fully homomorphic scheme using bootstrapping. The modulus of ciphertext decreases with homomorphic operations and our scheme can no longer support homomorphic computation at the lowest level. To overcome this problem, we aim to transform an input ciphertext into an encryption of almost the same plaintext with a much larger modulus.

Further improvement of our implementations are possible by other optimizations. We would like to enhance them to take advantage of Number Theoretical Transform (NTT) for fast polynomial multiplication.

Acknowledgments. This work was partially supported by IT R&D program of MSIP/KEIT (No. B0717-16-0098) and Samsung Electronics Co., Ltd. (No. 0421-20150074). The fourth author was supported by National Research Foundation of Korea (NRF) Grant funded by the Korean Government (No. NRF-2012H1A2A1049334). We would like to thank Kristin Lauter, Damien Stehlé and an anonymous ASIACRYPT referee for useful comments.

A LWE-based Construction

We start by adapting some notations from [5] to our context. Let n and q be positive integers. For a vector $\boldsymbol{x} \in \mathbb{Z}_q^N$, its bit decomposition and power of two are defined by $\mathsf{BD}(\boldsymbol{x}) = (\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{\lceil \log q \rceil - 1}) \in \{0,1\}^{N\lceil \log q \rceil}$ with $\boldsymbol{x} = \sum_{i=0}^{\lceil \log q \rceil - 1} 2^i \boldsymbol{u}_i$, and $\mathcal{P}_2(\boldsymbol{x}) = (\boldsymbol{x}, \ldots, 2^{\lceil \log q \rceil - 1}\boldsymbol{x})$. Then we can see that $\langle \mathsf{BD}(\boldsymbol{x}), \mathcal{P}_2(\boldsymbol{y}) \rangle = \langle \boldsymbol{x}, \boldsymbol{y} \rangle$. We also recall the definition of tensor product $\boldsymbol{u} \otimes \boldsymbol{v} = (u_1 v_1, u_1 v_2, \ldots, u_1 v_m, \ldots, u_n v_1, \ldots, u_n v_m)$ on the vector space $\mathbb{R}^n \times \mathbb{R}^m$, and its relation with the inner product $\langle \boldsymbol{u} \otimes \boldsymbol{v}, \boldsymbol{u}' \otimes \boldsymbol{v}' \rangle = \langle \boldsymbol{u}, \boldsymbol{u}' \rangle \cdot \langle \boldsymbol{v}, \boldsymbol{v}' \rangle$.

- $\mathsf{KeyGen}(1^\lambda)$
  - Take an integer p and q_0. Let $q_\ell = p^\ell \cdot q_0$ for $\ell = 1, \ldots, L$. Choose the parameters $N = N(\lambda, q_L)$ and an error distribution $\chi = \chi(\lambda, q_L)$ appropriately for LWE problem of parameter (N, q_L, χ). Let $\tau = 2(N+1)\lceil \log q_L \rceil$. Output the parameters $params = (n, q_L, \chi, \tau)$.
  - Sample $\boldsymbol{s} \leftarrow \mathcal{HWT}(h)$ and set the secret key as $sk \leftarrow (1, \boldsymbol{s}) \in \mathbb{Z}_{q_L}^{N+1}$. For $1 \le i \le \tau$, sample $A \leftarrow \mathbb{Z}_{q_L}^{\tau \times N}$, $\boldsymbol{e} \leftarrow \chi^\tau$ and let $\boldsymbol{b} \leftarrow -A\boldsymbol{s} + \boldsymbol{e} \pmod{q_L}$. Set the public key as $pk \leftarrow (\boldsymbol{b}, A) \in \mathbb{Z}_{q_L}^{\tau \times (N+1)}$.
  - Let $\boldsymbol{s}' \leftarrow \mathcal{P}_2(\boldsymbol{s} \otimes \boldsymbol{s})$. Sample $A' \leftarrow \mathbb{Z}_{q_L}^{N^2\lceil \log q_L \rceil \times N}$ and $\boldsymbol{e}' \leftarrow \chi^{N^2\lceil \log q_L \rceil}$, and let $\boldsymbol{b}' \leftarrow -A'\boldsymbol{s}' + \boldsymbol{e}'$. Set the evaluation key as $evk \leftarrow (\boldsymbol{b}', A') \in \mathbb{Z}_{q_L}^{N^2\lceil \log q_L \rceil \times (N+1)}$.
- $\mathsf{Enc}(m)$. For an integer $m \in \mathbb{Z}$, sample a vector $\boldsymbol{r} \leftarrow \{0,1\}^\tau$. Output $\boldsymbol{c} \leftarrow (m, \boldsymbol{0}) + pk^T \cdot \boldsymbol{r} \in \mathbb{Z}_{q_L}^{N+1}$.
- $\mathsf{Add}(\boldsymbol{c}_1, \boldsymbol{c}_2)$. For $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathbb{Z}_{q_\ell}^{N+1}$, output $\boldsymbol{c}_{\mathsf{add}} \leftarrow \boldsymbol{c}_1 + \boldsymbol{c}_2 \pmod{q_\ell}$.
- $\mathsf{Mult}(\boldsymbol{c}_1, \boldsymbol{c}_2)$. For $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathbb{Z}_{q_\ell}^{N+1}$, let $\boldsymbol{c}' \leftarrow \mathsf{BD}(\boldsymbol{c}_1 \otimes \boldsymbol{c}_2)$. Output $\boldsymbol{c}_{\mathsf{mult}} \leftarrow evk^T \cdot \boldsymbol{c}' \pmod{q_\ell}$.
- $\mathsf{RS}_{\ell \to \ell'}(\mathbf{c})$. For a ciphertext $\boldsymbol{c} \in \mathbb{Z}_{q_\ell}^{N+1}$ at level ℓ, output the ciphertext $\boldsymbol{c}' \leftarrow \left\lfloor \frac{q_{\ell'}}{q_\ell} \boldsymbol{c} \right\rceil \in \mathbb{Z}_{q_{\ell'}}^{N+1}$.

B Noise Estimations

We follow the heuristic approach in [14,25]. Assume that a polynomial $a(X) \in \mathcal{R} = \mathbb{Z}[X]/(\Phi_M(X))$ is sampled from one of above distributions, so its nonzero entries are independently and identically distributed. Since $a(\zeta_M)$ is the inner product of coefficient vector of a and the fixed vector $(1, \zeta_M, \ldots, \zeta_M^{N-1})$ of Euclidean norm $\sqrt{N}$, the random variable $a(\zeta_M)$ has variance $V = \sigma^2 N$, where σ^2 is the variance of each coefficient of a. Hence $a(\zeta_M)$ has the variances $V_U = q^2 N/12$, $V_G = \sigma^2 N$ and $V_Z = \rho N$, when a is sampled from $U(\mathcal{R}_q)$, $\mathcal{DG}(\sigma^2)$ and $\mathcal{ZO}(\rho)$, respectively. In particular, $a(\zeta_M)$ has the variance $V_H = h$ when $a(X)$ is chosen from $\mathcal{HWT}(h)$. Moreover, we can assume that $a(\zeta_M)$ is distributed similarly to a Gaussian random variable over complex plane since it is a sum of many independent and identically distributed random variables. Every evaluations at root of unity ζ_M^j share the same variance. Hence, we will use 6σ

as a high-probability bound on the canonical embedding norm of a when each coefficient has a variance σ^2. For a multiplication of two independent random variables close to Gaussian distributions with variances σ_1^2 and σ_2^2, we will use a high-probability bound $16\sigma_1\sigma_2$.

Proof of Lemma 1.

Proof. We choose $v \leftarrow \mathcal{ZO}(0.5)^2$ and $e_0, e_1 \leftarrow \mathcal{DG}(\sigma^2)$, then set $\boldsymbol{c} \leftarrow v \cdot pk + (m + e_0, e_1)$. The bound B_{clean} of encryption noise is computed by the following inequality:

$$\begin{aligned}\|\langle \boldsymbol{c}, sk\rangle - m \pmod{q_L}\|_\infty^{\mathsf{can}} &= \|v \cdot e + e_0 + e_1 \cdot s\|_\infty^{\mathsf{can}} \\ &\le \|v \cdot e\|_\infty^{\mathsf{can}} + \|e_0\|_\infty^{\mathsf{can}} + \|e_1 \cdot s\|_\infty^{\mathsf{can}} \\ &\le 8\sqrt{2} \cdot \sigma N + 6\sigma\sqrt{N} + 16\sigma\sqrt{hN}.\end{aligned}$$

For a vector $\boldsymbol{z} \in \mathbb{Z}[i]^{N/2}$, an encryption of $m = \mathsf{Ecd}(\boldsymbol{z}; \Delta)$ is also a valid encryption of $\Delta \cdot \sigma^{-1} \circ \pi^{-1}(\boldsymbol{z})$ with an increased error bound $B' = B_{\mathsf{clean}} + N/2$. If $\Delta^{-1} \cdot B' < 1/2$, then this error polynomial is removed by the rounding operation in decoding algorithm. □

Proof of Lemma 2.

Proof. It is satisfied that $\langle \boldsymbol{c}, sk\rangle = m + e \pmod{q_\ell}$ for some polynomial $e \in \mathcal{S}$ such that $\|e\|_\infty^{\mathsf{can}} \le B$. The output ciphertext $\boldsymbol{c}' \leftarrow \left\lfloor \frac{q_{\ell'}}{q_\ell} \boldsymbol{c} \right\rceil$ satisfies $\langle \boldsymbol{c}', sk\rangle = \frac{q_{\ell'}}{q_\ell}(m+e) + e_{\mathsf{scale}} \pmod{q_{\ell'}}$ for the rounding error vector $\boldsymbol{\tau} = (\tau_0, \tau_1) = \boldsymbol{c}' - \frac{q_{\ell'}}{q_\ell}\boldsymbol{c}$ and the error polynomial $e_{\mathsf{scale}} = \langle \boldsymbol{\tau}, sk\rangle = \tau_0 + \tau_1 \cdot s$.

We may assume that each coefficient of τ_0 and τ_1 in the rounding error vector is computationally indistinguishable from the random variable in the interval $\frac{q_{\ell'}}{q_\ell}\mathbb{Z}_{q_\ell/q_{\ell'}}$ with variance $\approx 1/12$. Hence, the magnitude of scale error polynomial is bounded by $\|e_{\mathsf{scale}}\|_\infty^{\mathsf{can}} \le \|\tau_0\|_\infty^{\mathsf{can}} + \|\tau_1 \cdot s\|_\infty^{\mathsf{can}} \le 6\sqrt{N/12} + 16\sqrt{hN/12}$, as desired. □

Proof of Lemma 3.

Proof. Let $\boldsymbol{c}_i = (b_i, a_i)$ for $i = 1, 2$. Then $\langle \boldsymbol{c}_i, sk\rangle = m_i + e_i \pmod{q_\ell}$ for some polynomials $e_i \in \mathcal{S}$ such that $\|e_i\|_\infty^{\mathsf{can}} \le B_i$. Let $(d_0, d_1, d_2) = (b_1b_2, a_1b_2 + a_2b_1, a_1a_2)$. This vector can be viewed as a level-ℓ encryption of m_1m_2 with an error $m_1e_2 + m_2e_1 + e_1e_2$ with respect to the secret vector $(1, s, s^2)$. It follows from Lemma 2 that the ciphertext $\boldsymbol{c}_{\mathsf{mult}} \leftarrow (d_0, d_1) + \left\lfloor P^{-1} \cdot (d_2 \cdot evk \pmod{Pq_\ell})\right\rceil$ contains an additional error $e'' = P^{-1} \cdot d_2e'$ and a rounding error bounded by B_{scale}. We may assume that d_2 behaves as a uniform random variable on $\mathcal{R}_{q_\ell}$, so $P\|e''\|_\infty^{\mathsf{can}}$ is bounded by $16\sqrt{Nq_\ell^2/12}\sqrt{N\sigma^2} = 8N\sigma q_\ell/\sqrt{3} = B_{\mathsf{ks}} \cdot q_\ell$. Therefore, $\boldsymbol{c}_{\mathsf{mult}}$ is an encryption of m_1m_2 with an error bounded by

$$\|m_1e_2 + m_2e_1 + e_1e_2 + e''\|_\infty^{\mathsf{can}} + B_{\mathsf{scale}} \le \nu_1B_2 + \nu_2B_1 + B_1B_2 + P^{-1} \cdot q_\ell \cdot B_{\mathsf{ks}} + B_{\mathsf{scale}},$$

as desired. □

References

1. Arita, S., Nakasato, S.: Fully homomorphic encryption for point numbers. In: Chen, K., Lin, D., Yung, M. (eds.) Inscrypt 2016. LNCS, vol. 10143, pp. 253–270. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54705-3_16
2. Bos, J.W., Lauter, K., Loftus, J., Naehrig, M.: Improved security for a ring-based fully homomorphic encryption scheme. In: Stam, M. (ed.) IMACC 2013. LNCS, vol. 8308, pp. 45–64. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45239-0_4
3. Bos, J.W., Lauter, K., Naehrig, M.: Private predictive analysis on encrypted medical data. J. Biomed. Inform. **50**, 234–243 (2014)
4. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_50
5. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Proceedings of ITCS, pp. 309–325. ACM (2012)
6. Brakerski, Z., Vaikuntanathan. V.: Efficient fully homomorphic encryption from (standard) LWE. In: Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, pp. 97–106. IEEE Computer Society (2011)
7. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_29
8. Çetin, G.S., Doröz, Y., Sunar, B., Martin, W.J.: An investigation of complex operations with word-size homomorphic encryption. Cryptology ePrint Archive, Report 2015/1195 (2015). http://eprint.iacr.org/2015/1195
9. Cheon, J.H., Jung, J., Lee, J., Lee, K.: Privacy-preserving computations of predictive medical models with minimax approximation and non-adjacent form. In: WAHC 2017 (2017, to appear)
10. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Implementation of HEA-AN (2016). https://github.com/kimandrik/HEAAN
11. Cheon, J.H., Kim, M., Lauter, K.: Homomorphic computation of edit distance. In: Brenner, M., Christin, N., Johnson, B., Rohloff, K. (eds.) FC 2015. LNCS, vol. 8976, pp. 194–212. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48051-9_15
12. Cheon, J.H., Stehlé, D.: Fully homomophic encryption over the integers revisited. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 513–536. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_20
13. Coron, J.-S., Lepoint, T., Tibouchi, M.: Scale-invariant fully homomorphic encryption over the integers. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 311–328. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_18
14. Costache, A., Smart, N.P.: Which ring based somewhat homomorphic encryption scheme is best? In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 325–340. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29485-8_19
15. Costache, A., Smart, N.P., Vivek, S.: Faster homomorphic evaluation of discrete fourier transforms. Cryptology ePrint Archive, Report 2016/1019 (2016). http://eprint.iacr.org/2016/1019

16. Costache, A., Smart, N.P., Vivek, S., Waller, A.: Fixed point arithmetic in SHE schemes. Cryptology ePrint Archive, Report 2016/250 (2016). http://eprint.iacr.org/2016/250
17. Damgård, I., Pastro, V., Smart, N., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 643–662. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_38
18. Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_2
19. Doröz, Y., Hu, Y., Sunar, B.: Homomorphic AES evaluation using the modified LTV scheme. Des. Codes Crypt. **80**(2), 333–358 (2016)
20. Dowlin, N., Gilad-Bachrach, R., Laine, K., Lauter, K., Naehrig, M., Wernsing, J.: Manual for using homomorphic encryption for bioinformatics. Proc. IEEE **105**(3), 552–567 (2017)
21. Ducas, L., Micciancio, D.: FHEW: bootstrapping homomorphic encryption in less than a second. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 617–640. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_24
22. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive 2012/144 (2012)
23. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University (2009). http://crypto.stanford.edu/craig
24. Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 465–482. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_28
25. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_49
26. 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.) CRYPTO 2013. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_5
27. Jäschke, A., Armknecht, F.: Accelerating homomorphic computations on rational numbers. In: Manulis, M., Sadeghi, A.-R., Schneider, S. (eds.) ACNS 2016. LNCS, vol. 9696, pp. 405–423. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39555-5_22
28. Kim, J., Lee, C., Shim, H., Cheon, J.H., Kim, A., Kim, M., Song, Y.: Encrypting controller using fully homomorphic encryption for security of cyber-physical systems. IFAC-PapersOnLine **49**(22), 175–180 (2016)
29. Kim, M., Song, Y., Cheon, J.H.: Secure searching of biomarkers through hybrid homomorphic encryption scheme. BMC Med. Genomics **10**(2), 42 (2017)
30. Kim, M., Song, Y., Wang, S., Xia, Y., Jiang, X.: Privacy-preserving logistic regression based on homomorphic encryption. preprint
31. Lauter, K., López-Alt, A., Naehrig, M.: Private computation on encrypted genomic data. In: Aranha, D.F., Menezes, A. (eds.) LATINCRYPT 2014. LNCS, vol. 8895, pp. 3–27. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16295-9_1

32. Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 319–339. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19074-2_21
33. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, pp. 1219–1234. ACM (2012)
34. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_1
35. Lyubashevsky, V., Peikert, C., Regev, O.: A toolkit for ring-LWE cryptography. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 35–54. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_3
36. Naehrig, M., Lauter, K., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop, pp. 113–124. ACM (2011)
37. Paterson, M.S., Stockmeyer, L.J.: On the number of nonscalar multiplications necessary to evaluate polynomials. SIAM J. Comput. **2**(1), 60–66 (1973)
38. Smart, N.P., Vercauteren, F.: Fully homomorphic encryption with relatively small key and ciphertext sizes. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 420–443. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13013-7_25
39. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. Des. Codes Crypt. **71**(1), 57–81 (2014)
40. Tabaei, B.P., Herman, W.H.: A multivariate logistic regression equation to screen for diabetes development and validation. Diab. Care **25**(11), 1999–2003 (2002)
41. Wang, S., Zhang, Y., Dai, W., Lauter, K., Kim, M., Tang, Y., Xiong, H., Jiang, X.: Healer: homomorphic computation of exact logistic regression for secure rare disease variants analysis in GWAS. Bioinformatics **32**(2), 211–218 (2016)