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

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

We also propose a new batching method for RLWE-based construction. A plaintext polynomial of *characteristic zero* is mapped to a message vector of complex numbers via complex canonical embedding map which is an isometric ring homomorphism. The size of error is preserved during transformation and so it enables us to remove the errors after decryption procedure.

Our construction has the bit size of ciphertext modulus linear in the circuit depth due to rescaling procedure while all the previous works either require 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 circuit depth and it is at most one more bit compared with unencrypted approximate arithmetic such as floating-point operations. In addition to the basic approximate circuits, we show that our scheme can be applied to efficiently evaluate the 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 allows homomorphic operations on encrypted data without decryption. A lot of HE schemes (e.g., [17, 6, 7, 4, 5, 23, 24, 29, 2, 25, 12, 11, 18, 20]) have been suggested following Gentry's blueprint [22]. HE has a number of applications to homomorphically evaluate various algorithms on encrypted financial, medical, or genomic data [32, 27, 10, 36]. However, existing HE schemes just support arithmetic operations over a fixed modulus space and have a difficulty in evaluation of non-polynomial functions such as bit extraction and logical circuits. In particular, we will focus on arithmetic of approximate numbers in this paper.

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 a sampling error is caused by observing a sample of the whole population in statistics. In practice, a data value should be discretized (quantized) to an approximate value such as floating-point numbers to be represented by a fixed number of bits in computer systems while allowing a small error. In this case, an approximate value can substitute the original data, and this small error does not have too much effect on resulting value of computation.

For the efficient performance of approximate arithmetic, we store some significand of numbers (e.g., most significant bits, MSBs) and carry out homomorphic operations between them. The resulting values should be rounded by removing some useless least significant

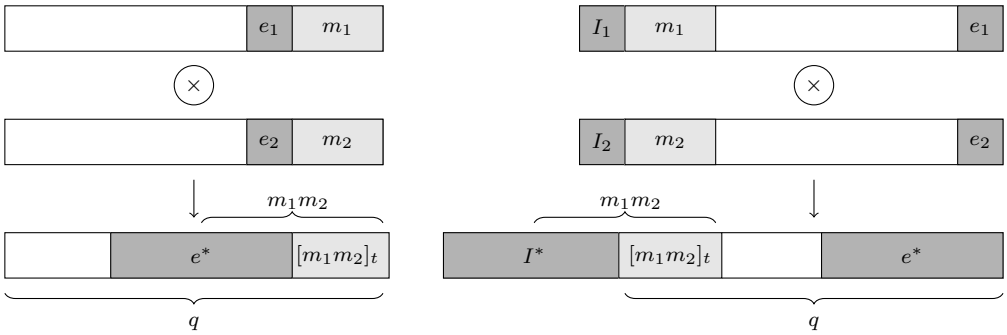

Fig. 1. Homomorphic multiplication of ciphertexts of BGV-type HE scheme (left) and scale-invariant type HE scheme (right)

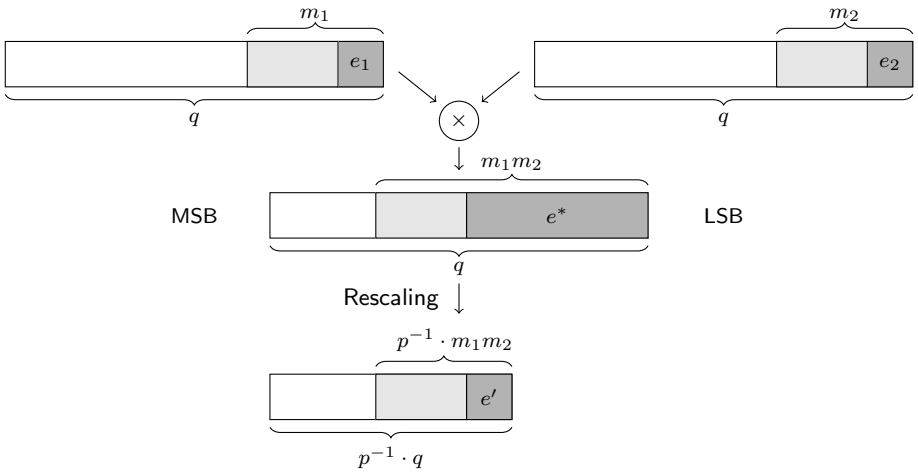

Fig. 2. Homomorphic multiplication of ciphertexts for approximate arithmetic

bits (LSBs) to restore the bitsize of significand. However, this rounding operation has been considered difficult to perform on HE. Previous approaches for approximate arithmetic have similar multiplicative depth and complexity to the case of bootstrapping for extraction of MSBs [1, 26]. The other methods based on exact integer operations [19, 15] require the bit size of ciphertext modulus to grow exponentially with circuit depth to ensure correctness.

We point out that the decryption structures of previous HE schemes are not appropriate for arithmetic over indiscrete spaces. First, the ciphertexts of BGV-type HE schemes [5, 23, 24, 29, 18] have a decryption structure of the form $\langle \boldsymbol{c}_i, sk \rangle = m_i + te_i \pmod q$, so the MSBs of $m_1 + m_2$ and $m_1 m_2$ are destroyed by inserted errors e_i after homomorphic operations. On the other hand, the scale-invariant type schemes [4, 21, 2] have the decryption structure $\langle \boldsymbol{c}_i, sk \rangle = qI_i + (q/t)m_i + e_i$ for some I_i and e_i. Multiplication of two ciphertexts satisfies $\langle \boldsymbol{c}^*, sk \rangle = qI^* + (q/t)m_1m_2 + e^*$ for $I^* = tI_1I_2 + I_1m_2 + I_2m_1$ and $e^* \approx t(I_1e_2 + I_2e_1)$, so the MSBs of resulting message is also destroyed. Moreover, since both BGV and scale-invariant type HE schemes have separated plaintext space from inserted errors, it is difficult to perform the rounding of message and the reduction of error at once after homomorphic operations of ciphertexts (See Figure 1 for an illustration). As a result, the current decryption structures of HE schemes require to have an exponentially growing ciphertext modulus on circuit depth and yield inefficiency of approximate arithmetic.

Homomorphic Encryption for Approximate Arithmetic. The purpose of this paper is to present a method to carry out approximate computations efficiently using HE schemes. The main idea of construction is to treat a noise from a hardness assumption of a scheme as part of

error occurred during approximate computations. More precisely, an encryption $\boldsymbol{c}$ of message m by the secret key sk will have a decryption structure of the form $\langle \mathbf{c}, sk \rangle = m + e \pmod{q}$ for some small error e which is 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 with the message, an addition of noise is not probable 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 caused by noise addition. We will tag some auxiliary informations to each ciphertext for correct understanding of decryption result and its precision.

In our construction, we always maintain the message $\langle \boldsymbol{c}, sk \rangle = m + e \pmod{q}$ small enough compared with the ciphertext modulus q so that it is still smaller than q after homomorphic operations. However, the bit size of message increases exponentially on circuit depth if homomorphic operations are performed on the same modulus. To overcome this problem, we provide a new tool, called *rescaling*, to manipulate the message of ciphertext. It is technically similar to the modulus-switching technique suggested by Brakerski and Vaikuntanatan [6], but it is simpler and 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 process outputs the vector $\lfloor p^{-1} \cdot \boldsymbol{c} \rceil \pmod{q/p}$ which is a valid encryption of m/p with noise $\approx e/p$. It reduces the size of ciphertext modulus and consequently removes the error located in the LSBs of messages, similar to the rounding step of ordinary (unencrypted) floating-point arithmetic. In this procedure, some additional error is generated by rounding but the precision of plaintext does not change.

The composition of homomorphic operation and rescaling (as in Figure 2) mimics the ordinary (unencrypted) floating-point arithmetic. As a result, the bit size of required ciphertext modulus grows linearly in the circuit depth rather than exponentially. We also prove that this scheme is almost optimal in the sense of precision: precision loss of resulting message is at most one bit more compared with unencrypted floating-point arithmetic. Our idea can be applied to most existing HE schemes based on the LWE problem [5], the RLWE problem [5, 24], and the NTRU problem [29].

Encoding Technique for Packing Messages. It is inevitable to encrypt a vector of multiple plaintexts in a single ciphertext for efficient homomorphic computation. The previous RLWE-based HE schemes use a plaintext space which is a ring of finite characteristic. An inserted error is placed on separated position from the plaintext space so it may be annihilated using plaintext characteristic after performing homomorphic operations. Then the output polynomial is decoded into a vector of plaintext values by encoding technique using the ring isomorphism to a product of modulo spaces [33, 34].

On the other hand, a plaintext of our scheme is in a characteristic zero space $\mathcal{R} = \mathbb{Z}[X]/(\Phi_M(X))$ and it embraces a small error for security, so the annihilation method is not applied to our case. We adapt another *isometric ring homomorphism* to preserve the size of error - the complex canonical embedding map. The small error in the plaintext polynomial is not blow up by encoding/decoding and can be removed by quantization after applying the ring homomorphism.

More precisely, a native plaintext of our scheme is a polynomial in the cyclotomic ring $\mathcal{R} = \mathbb{Z}[X]/(\Phi_M(X))$ with bounded magnitude by ciphertext modulus. Let S be a subgroup of $\mathbb{Z}_M^*$ satisfying $\mathbb{Z}_M^*/S = \{\pm 1\}$. Then there is a natural ring homomorphism ι from $\mathbb{C}^{\phi(M)/2}$ to $\mathbb{C}^{\phi(M)}$ defined as follows: for a complex vector $\boldsymbol{z} = (z_i)_{i \in S}$ and $j \in \mathbb{Z}_M^*$, the jth component of $\iota(\boldsymbol{z})$ is z_j if $j \in S$; otherwise it is $\overline{z_{-j}}$. For the purpose of packing, we first embed a $\phi(M)/2$-dimensional plaintext vector $\boldsymbol{z} = (z_i)_{i \in S}$ into $\mathbb{C}^{\phi(M)}$ via the map ι. Then it is multiplied by a predetermined scale factor Δ and discretized to $\sigma(\mathcal{R})$ by using a round-off algorithm. Finally it is converted to the corresponding polynomial in $\mathcal{R}$ via the inverse of the canonical

embedding map σ. In short, our encoding function is explicitly given by

$$\mathsf{Encode}: \begin{cases} \mathbb{C}^{\phi(M)/2} & \xrightarrow{\iota} & \mathbb{C}^{\phi(M)} & \xrightarrow{\lfloor \cdot \rceil} & \sigma(\mathcal{R}) & \xrightarrow{\sigma^{-1}} & \mathcal{R} \\ \boldsymbol{z} = (z_i)_{i \in S} & \longmapsto & \iota(\boldsymbol{z}) & \longmapsto \boldsymbol{m} = \lfloor \Delta \cdot \iota(\boldsymbol{z}) \rceil_{\sigma(\mathcal{R})} & & \longmapsto & \sigma^{-1}(\boldsymbol{m}) \end{cases}$$

Homomorphic Evaluation of Approximate Arithmetic. One important feature of our method is that the precision loss during homomorphic evaluation is bounded by circuit depth and it is at most one more bit compared with unencrypted approximate arithmetic. Given encryptions of messages $m_1, \ldots, m_d$ 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 significand 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 multiplications of ciphertexts. The parameters are smaller than for the previous works and this advantage enables to efficiently perform the approximate evaluation of *transcendental* functions such as the exponential, logarithm, and trigonometric functions through the evaluation of their Taylor series expansion. In particular, we suggest a specific algorithm for computing the multiplicative inverse with reduced complexity, which allows 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 and a parameter with the security level $\lambda = 80$. It takes about 0.69 seconds for multiplicative inverse of ciphertext with $\eta = 9$ bits of precision, yielding an amortized rate of 0.17 milliseconds per slot. We can also evaluate the exponential function using its Taylor expansion and it results in an amortized time per slots of 0.24 milliseconds.

In a cloud-computing environment, a large amount of data are generated and one needs to handle these huge data collections. Our scheme may be a practical solution for data analysis because it allows the encryption of a lot of 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 batching technique, which can be used in disease prediction analysis. Our implementation homomorphically evaluated the degree seven Taylor polynomial of logistic function in about 0.19 milliseconds per slot (and less than 0.79 seconds total) compared to 30 seconds and 1.8 seconds of evaluation time of [3] and [9] without parallelization, respectively.

Another example is to evaluate discrete Fourier transform homomorphically using a *fast Fourier transform* (FFT) algorithm. We follow the encoding method of [14] for roots of unity in polynomial ring so that it does not consume the ciphertext level during evaluation. We can also apply our rescaling procedure for operations on Hadamard space and batching technique, which results in much smaller parameter and amortized evaluation time, respectively. We can homomorphically process the standard processing (FFT-Hadamard product of two vectors-inverse FFT) of dimension 2^{13} in 1.06 seconds per slot (and 73 minutes total) compared to 17 minutes of previous work [15] on a machine with six Intel Xeon E5 2.7GHz processors with 64 GB RAM without parallelization. Based on evaluation of discrete Fourier transform, we can securely compute the exact multiplication of integral polynomials by removing the fractional part of approximate result. Likewise, our HE for approximate arithmetic can be applied to exact computation when the result has a specific format or property.

Related Works. A substantial number of studies have concerned with processing real numbers over encryption. Jäschke and Armknecht [26] 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 binary fashion, so that each of bit is encrypted separately. The one performing homomorphic multiplication can bring the product to the required precision by merely discarding the ciphertexts corresponding to the last LSBs. However, bitwise encryption causes 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 encode as the constant polynomial in a polynomial ring with 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. [19] 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. [15] analyze the representations of [19] and compute the lower bound of plaintext modulus. However, exact arithmetic of fixed point numbers causes the required the size of plaintext modulus to grow exponentially in circuit depth.

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 Section 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 Section 5, we perform the implementation of our scheme for the evaluations of circuits described in Section 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 is 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 distribution D. It denotes the uniform sampling when D is a finite set. For a set S, $U(S)$ denotes the uniform distribution on S. We let λ denote the security parameter throughout the paper: all known valid attacks against the cryptographic scheme under scope should take $\Omega(2^\lambda)$ bit operations.

2.2 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 $\mathcal{K}$ be a number field such that $\mathcal{K} = \mathbb{Q}[X]/(\Phi_M(X))$, and the ring of integers of $\mathcal{K}$ is $\mathcal{R} = \mathbb{Z}[X]/(\Phi_M(X))$. 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 a cyclotomic field $\mathcal{K}$ 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 $(a_0, \ldots, a_{N-1}) \in \mathbb{Z}^N$. We define $\|a\|_\infty$ and $\|a\|_1$ by the relevant norms on the coefficient vector of a.

Throughout this paper, we use the canonical embedding to measure the size of polynomials. Write $\mathbb{Z}_M^* = \{x \in \mathbb{Z}_M : \gcd(x, M) = 1\}$ for the group of units of $\mathbb{Z}_M$. Recall that the canonical embedding of $a \in \mathcal{K}$ into $\mathbb{C}^N$ is the vector of complex numbers $\sigma(a) = (a(\zeta_M^j))_{j \in \mathbb{Z}_M^*}$

for the complex primitive root of unity $\zeta_M = \exp(2\pi i/M)$. We call the ℓ_∞-norm of $\sigma(a)$ the *canonical embedding norm* of a, denoted by $\|a\|_\infty^{\mathsf{can}} = \|\sigma(a)\|_\infty$. The canonical embedding norm $\|\cdot\|_\infty^{\mathsf{can}}$ satisfies the following properties:

- For all $a, b \in \mathcal{K}$, we have $\|a \cdot b\|_\infty^{\mathsf{can}} \leq \|a\|_\infty^{\mathsf{can}} \cdot \|b\|_\infty^{\mathsf{can}}$
- For all $a \in \mathcal{K}$, 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 $a \in \mathcal{K}$.

The ring constant is defined 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. In particular, we have $c_M = 1$ when M is a power of two. Refer [16] for a discussion about c_M.

2.3 Gaussian Distributions and Ring-LWE Problem

We first define the space $\mathbb{H} \subseteq \mathbb{C}^N$ by

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

Note that the indices j are represented as elements of $\mathbb{Z}_M^*$. The space $\mathbb{H}$ is isomorphic to $\mathbb{R}^N$ as an inner product space via the unitary basis matrix

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

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

For $r > 0$, we define the Gaussian function $\rho_r : \mathbb{H} \to (0, 1]$ as $\rho_r(\boldsymbol{z}) = \exp(-\pi\|\boldsymbol{z}\|^2/r^2)$. Denote by Γ_r the continuous Gaussian probability distribution whose density is given by $r^{-N} \cdot \rho_r(\boldsymbol{z})$. Now one can define 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 $B \cdot \boldsymbol{z}$ 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 $\mathcal{K} \otimes \mathbb{R}$. That is, $\mathsf{CRT}_M^{-1} \cdot B \cdot \boldsymbol{z}$ 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 [30, 31] 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 $A_{s,\chi}$ as the RLWE distribution obtained by sampling $a \leftarrow \mathcal{R}_q$ uniformly at random, and $e \leftarrow \chi$ and returning $(a, a \cdot s + e) \in \mathcal{R}_q \times \mathcal{R}_q^\vee$.

The (decision) ring learning with errors, denoted by $\mathsf{RLWE}_{q,\chi}(\mathcal{D})$, is a problem to distinguish arbitrarily many independent samples chosen according to $A_{s,\chi}$ 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 based on the hardness of RLWE problem for approximate computation of ciphertexts. Given encryptions of m_1 and m_2, this scheme enables us to securely compute encryptions of approximate values of $m_1 + m_2$

and $m_1 m_2$ with the predetermined precision. The main idea of our construction is to treat an inserted noise of RLWE problem as an error occurred during approximate computation.

Most important feature of our scheme is the *rounding* operation of messages. Just like the ordinary approximate computations using floating-point numbers, the purpose of rounding operation is to remove the LSBs of message and make a trade-off between size of numbers and precision loss.

Our concrete construction in this section is based on the BGV scheme [5] with multiplication method by raising the ciphertext modulus [24], but the construction can be applied to the other LWE, RLWE and NTRU-based schemes such as [5, 25, 29]. Refer Appendix A to see the description of LWE-based HE scheme.

3.1 Decryption Structure of Homomorphic Encryption for Approximate Arithmetic

Most of existing HE schemes just compute the ciphertext which represents some LSBs of the resulting message after homomorphic computation. For example, in the non-scale invariant schemes such as BGV [5, 24] and the basic NTRU scheme [29], the messages are placed in the lower bits, that is, the resulting ciphertext $\boldsymbol{c}$ has the decryption structure of the form $\langle \boldsymbol{c}, sk \rangle = m + te \pmod{q}$ where sk is the secret vector. The multiplication of two ciphertexts $\boldsymbol{c}_1, \boldsymbol{c}_2$ only gives some LSBs of $m_1 m_2$ (i.e., $[m_1 m_2]_t$), while some of its MSBs (i.e., $\lfloor m_1 m_2 / t \rceil$) is destroyed by the error from homomorphic operation. Their scale invariant versions [4, 21, 2] put the message in the upper bits such that $\langle \boldsymbol{c}, sk \rangle = \lfloor q/t \rceil \cdot m + e \pmod{q}$. The MSBs of resulting messages is also destroyed by multiplication because scale invariant schemes multiply the values $\langle \boldsymbol{c}, sk \rangle = q \cdot I + \lfloor q/t \rceil \cdot m + e$ which contains some additional noise in the upper part of the message.

Our goal is to compute encryptions of MSBs of the resulting message after homomorphic operations, or equivalently, carry out approximate arithmetic of ciphertexts. The main idea is to place an encryption noise in the rightmost position of an input message. More precisely, for a fixed constant Δ, our decryption structure has the form $\langle \boldsymbol{c}, sk \rangle = \Delta \cdot z + e \bmod q$, where $\boldsymbol{c}$ encrypts the message z under the secret key sk and e is a small error. The error polynomial e is usually inserted to guarantee the security of HE schemes, but it will be considered as an error that arises during the process of approximate computations. That is, the decryption structure $\Delta \cdot z + e$ of the ciphertext $\boldsymbol{c}$ will be treated as an approximate value of given message multiplied by scale factor Δ. The value obtained from decryption structure will be small enough compared with the ciphertext modulus similar to the cases of the BGV and the NTRU schemes, so that one can make the resulting value still smaller than ciphertext modulus and preserve its MSBs after several homomorphic operations.

There are some important issues that we need to consider very carefully. In unencrypted floating-point arithmetic, small errors can also grow as operations are performed in succession, so it is valuable to consider the proximity of the computed result to the exact value of an algorithm. Similarly, our messages will contain small errors located at their LSBs and they can increase during homomorphic evaluation. Thus we should compute the upper bound and the precision of resulting values. Another issue is the size management of messages. If we want to evaluate a circuit of multiplicative depth L in the fixed modulus, then the size of resulting value will be exponential in L. This naive approach is too inefficient for practical usage because it causes a huge ciphertext modulus. In our scheme, intermediate values are divided by the base using rescaling technique. It enables to discard the useless LSBs of messages while making error be still relatively small compared with the messages. This method maintains the size of messages almost same and makes the required ciphertext modulus linear in L.

3.2 Plaintext Encoding for Packing

The SIMD operations in HE system allows each ciphertext to represent several independent plaintexts, so it enables parallel processing in computation. In practice, we take advantage of SIMD techniques to parallelize both space and computation time together. In the previous works, the RLWE-based HE schemes use a plaintext space which is a ring of finite characteristic. A small error, which is multiplied by the characteristic of the ring, is inserted into the plaintext for security. It is annihilated by the characteristic after performing homomorphic operations and then the output polynomial is decoded into a vector of plaintext values by the CRT-based encoding technique [33, 34].

On the other hand, a plaintext is in characteristic zero space and it embraces an error for security, so the annihilation method does not work for our scheme. We adapt another isometric ring homomorphism to preserve the size of error - the canonical embedding map $\sigma : \mathcal{K} = \mathbb{Q}[X]/(\Phi_M(X)) \to \mathbb{C}^N$. More precisely, the native plaintext space of our scheme can be viewed as the set of polynomials $m(X) \in \mathcal{R} = \mathbb{Z}[X]/(\Phi_M(X))$ of size bounded by ciphertext modulus. It is easy to see that the image of $m(X)$ under the canonical embedding map is contained in the subring $\mathbb{H} = \{\boldsymbol{z} = (z_j)_{j\in\mathbb{Z}_M^*} : z_j = \overline{z_{-j}}, \forall j \in \mathbb{Z}_M^*\}$ of $\mathbb{C}^N$. Let S be a subgroup of $\mathbb{Z}_M^*$ satisfying $\mathbb{Z}_M^*/S = \{\pm 1\}$. Note that the cyclic subgroup generated by three $S = \langle 3\rangle$ satisfies this condition when M is a power of two. Then there is a natural ring homomorphism ι from $\mathbb{H}$ to $\mathbb{C}^N$ defined by

$$\iota(\boldsymbol{z})[j] = \begin{cases} z_j, & j \in S \\ \overline{z_{-j}}, & j \notin S \end{cases}$$

for an input vector $\boldsymbol{z} = (z_i)_{i\in S}$ and an index $j \in \mathbb{Z}_M^*$. Hence the encoding of a plaintext vector $\boldsymbol{z} = (z_i)_{i\in S}$ can be obtained by $z(X) = \sigma^{-1} \circ \iota(\boldsymbol{z})$. Note that $\|z(X)\|_\infty^{\mathsf{can}} = \|\boldsymbol{z}\|_\infty$.

We need to discritize $\iota(\boldsymbol{z})$ to $\sigma(\mathcal{R})$ so that the encoding function is applied to RLWE-based HE schemes. 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 is to recover $\boldsymbol{m} \in \sigma(\mathcal{R})$, denoted by $\boldsymbol{m} = \lfloor \iota(\boldsymbol{z}) \rceil_{\sigma(\mathcal{R})}$, such that the size $\|\iota(\boldsymbol{z}) - \boldsymbol{m}\|_\infty^{\mathsf{can}}$ of rounding error is not too large. There are several round-off algorithms to do that, e.g., by coordinate-wise randomized rounding. See [31] for details. Combining with the multiplication with the scale factor in previous subsection, a vector of plaintext values $\boldsymbol{z} \in \mathbb{C}^{N/2}$ can be encoded into $\mathcal{R}$ as follows:

- $\mathsf{Encode}(\boldsymbol{z})$. For a $(N/2)$-dimensional vector $\boldsymbol{z} = (z_i)_{i\in S}$ of complex numbers, the encoding procedure first transforms it into the vector $\iota(\boldsymbol{z}) \in \mathbb{H} \subseteq \mathbb{C}^N$ and computes its discretization $\boldsymbol{m} = \lfloor \Delta \cdot \iota(\boldsymbol{z}) \rceil_{\sigma(\mathcal{R})}$ in $\mathbb{H}$ after multiplying the scale factor Δ. Return the corresponding integral polynomial $m(X) = \sigma^{-1}(\boldsymbol{m}) \in \mathcal{R}$.
- $\mathsf{Decode}(m)$. For an input polynomial $m \in \mathcal{R}$, compute the vector $\boldsymbol{m} = \sigma(m(X)) \in \mathbb{H}$ and output the corresponding scaled vector $\boldsymbol{z} = \Delta^{-1} \cdot \iota^{-1}(\boldsymbol{m}) \in \mathbb{C}^{N/2}$, i.e., the entry of $\boldsymbol{z}$ of index $i \in S$ is $z_i = \Delta^{-1} \cdot m(\zeta_M^i)$.

Multiplication by the scaling factor Δ is optional but it is helpful to preserve the precision of plaintexts. The encoding and decoding algorithms can be viewed as an approximate homomorphism from $\mathbb{C}^{N/2}$ to $\mathcal{R}$ and its approximate inverse, respectively.

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 play a role as base for scaling in approximate computations. We also choose

the parameter $M = M(\lambda, q_L)$ for cyclotomic polynomial. For any level $0 \le \ell \le L$, 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 $(\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Add}, \mathsf{Mult})$ with error bounds B_{clean} and $B_{\mathsf{mult}}(\ell)$. 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.
- $\mathsf{Encrypt}_{pk}(m, \ell)$. For a given polynomial $m \in \mathcal{R}$ and a level $0 \le \ell \le L$, output a ciphertext $\boldsymbol{c} \in \mathcal{R}_{q_\ell}^k$. An encryption $\boldsymbol{c}$ of m at level ℓ will satisfy $\langle \boldsymbol{c}, sk \rangle = m + e \pmod{q_\ell}$ for some small e. The constant B_{clean} denotes an encryption bound, i.e., error polynomial of a fresh ciphertext satisfies $\|e\|_\infty^{\mathsf{can}} \le B_{\mathsf{clean}}$ with an overwhelming probability.
- $\mathsf{Decrypt}_{sk}(\boldsymbol{c})$. For a ciphertext $\boldsymbol{c}$ at level ℓ, output the polynomial $m' \leftarrow \langle \boldsymbol{c}, sk \rangle \pmod{q_\ell}$ for the secret key sk.

In the field of science and engineering, a measurement of a quantity is used in computation of continuous functions instead of its (unknown) true value. In the case of computer science, we also use the floating-point representation for big number arithmetics to increase the speed of operations and reduce the storage amount.

Unlike the most of existing schemes, our scheme does not have a fixed or separate plaintext space from an inserted error. It may seem strange because the output $m' = m + e$ of its decryption circuit is slightly different from the original message m. However, the output can be considered to be an approximate value of the original message in approximate computation if $\|e\|_\infty^{\mathsf{can}} / \|m\|_\infty^{\mathsf{can}}$ is small enough.

This notion of approximate encryption has been already partially used in previous work, for example, additional information for multiplication in [6, 4, 5, 11] and intermediate values in squashed decryption circuit in [17, 12] are encrypted in the same way. Note that these inaccurate ciphertexts work correctly because the target circuits require approximate computations allowing small errors.

- $\mathsf{Add}(\boldsymbol{c}_1, \boldsymbol{c}_2)$. For a pair of ciphertexts $(\boldsymbol{c}_1, \boldsymbol{c}_2)$ at the same level ℓ, output the ciphertext $\boldsymbol{c}_{\mathsf{add}} \leftarrow \boldsymbol{c}_1 + \boldsymbol{c}_2 \pmod{q_\ell}$. The message (or noise) of output ciphertext can be bounded by the sum of two message (or noise, resp.) bounds of input ciphertexts.
- $\mathsf{Mult}_{evk}(\boldsymbol{c}_1, \boldsymbol{c}_2)$. For a pair of ciphertexts $(\boldsymbol{c}_1, \boldsymbol{c}_2)$ at the same level ℓ, output a ciphertext $\boldsymbol{c}_{\mathsf{mult}} \in \mathcal{R}_{q_\ell}^k$ of level ℓ. The multiplication of ciphertexts will satisfy $\langle \boldsymbol{c}_{\mathsf{mult}}, sk \rangle = \langle \boldsymbol{c}_1, sk \rangle \cdot \langle \boldsymbol{c}_2, sk \rangle + e_{\mathsf{mult}} \pmod{q_\ell}$ for some polynomial $e_{\mathsf{mult}} \in \mathcal{R}$ with $\|e_{\mathsf{mult}}\|_\infty^{\mathsf{can}} \le B_{\mathsf{mult}}(\ell)$.

We may adapt the techniques of existing HE schemes over the ring $\mathcal{R}$ to construct our HE scheme for approximate arithmetic. For example, the ring-based BGV scheme [5], its modification with multiplication by raising ciphertext modulus [24] ($k = 2$), or the NTRU scheme [29] ($k = 1$) can be us-d as a base scheme for our construction. The complexity of encryption, decryption, and homomorphic operations are similar to that of basic HE scheme in the same parameter setting.

Our scheme has its own distinct and unique characteristics represented by the following *rescaling* procedure. If every homomorphic operation is done in a single level, the bit size of result increases exponentially on the depth of target circuit and the ciphertext modulus should be exponentially large. The rescaling procedure plays an important role in managing the magnitude of messages and achieving the efficiency of approximate computations.

- $\mathsf{Rescaling}_{\ell \to \ell'}(\mathbf{c})$. 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 $\ell' = \ell - 1$.

If an input ciphertext of m has a decrypted value $\langle \boldsymbol{c}, sk \rangle = m + e \pmod{q_\ell}$, then the output ciphertext $\boldsymbol{c}'$ of rescaling procedure satisfies $\langle \boldsymbol{c}', sk \rangle = \frac{q_{\ell'}}{q_\ell} m + (\frac{q_{\ell'}}{q_\ell} e + e_{\mathsf{scale}}) \pmod{q_{\ell'}}$ for the vector $\boldsymbol{\tau} = \frac{q_{\ell'}}{q_\ell} \boldsymbol{c} - \boldsymbol{c}'$ and the error polynomial $e_{\mathsf{scale}} = \langle \boldsymbol{\tau}, sk \rangle$ which is bounded by some constant B_{scale} independent of ciphertext $\boldsymbol{c}$. Hence, the resulting ciphertext is an encryption of $\frac{q_{\ell'}}{q_\ell} m$ with a noise bounded by $\frac{q_{\ell'}}{q_\ell} \|e\|_\infty^{\mathsf{can}} + B_{\mathsf{scale}}$.

This procedure is technically similar to modulus-switching of previous HE schemes for reduction of error, but it has a different role in our construction. After the size of a message increases by homomorphic operations, it rounds the message and reduces the size of significand by removing inaccurate LSBs as in usual approximate computations using floating-point numbers or scientific notation. The integer p works just like a base of rescaling procedure. The magnitude of messages can be maintained almost the same during computations and thus the size of largest ciphertext modulus grows linearly in the circuit depth.

Security. The security of our scheme relies on the security of base HE scheme. Since encryption of message is done by encrypting real/imaginary part using base HE scheme, our scheme is IND-CPA secure under the assumption that the underlying HE scheme is IND-CPA secure.

Tagged Informations. The size of messages may increase after homomorphic operations and they have effects on the size of multiplication noise. In the implementation of our scheme, each of ciphertext will be tagged with its level, bounds of message, and noise together in order to manage the magnitude of message and noise dynamically. Hence, a full ciphertext will be of the form $(\boldsymbol{c}, \ell, \nu, B)$ for a ciphertext vector $\boldsymbol{c} \in \mathcal{R}_{q_\ell}^k$, the current 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 with respect to the canonical embedding norm. Table 1 shows the full description of our scheme and homomorphic operations for ciphertexts with tagged informations.

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

Table 1. Description of our scheme

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 operations. There are two candidates: simple modular reduction and the Rescaling procedure. The way of transformation 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 Rescaling procedure changes the plaintext from m to $\frac{q_{\ell'}}{q_\ell} m$ as in Fig. 3. In the remaining part of this paper, a homomorphic operation between ciphertext at different levels without specific description of modulus switching method means the homomorphic operation after simple modulus reduction to the smaller modulus.

3.4 Concrete Construction based on the BGV Scheme

The performance of our construction and the noise growth depend on the choice of basic HE scheme. Moreover, a more precise noise estimation can be done if we choose a specific one. We

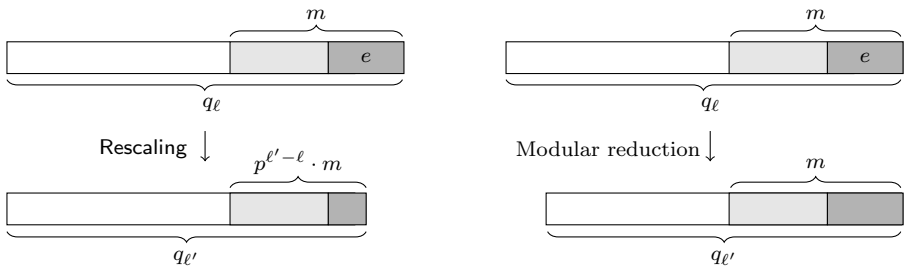

Fig. 3. Rescaling and simple modular reduction

take the BGV scheme [5] with multiplication method by raising ciphertext modulus [24] as the underlying scheme of our concrete construction and implementation. From Costache and Smart's comparison [13], it seems to be the most efficient among the existing RLWE-based schemes in the most of case.

We adapt some distributions over the cyclotomic rings $\mathcal{R}$ and $\mathcal{R}_q$. The following distributions are defined by coefficient vectors in $\mathbb{Z}^N$ or $\mathbb{Z}_q^N$, but they will be identified to distributions over the polynomial ring as mentioned above.

- $\mathcal{U}_q$ is the uniform distribution over $\mathbb{Z}_q^N$.
- For a real $\sigma > 0$, an element according to distribution $\mathcal{DG}_q(\sigma^2)$ is generated by drawing its entries independently from the discrete Gaussian distribution with zero-mean and variance σ^2.
- For a real $0 \leq \rho \leq 1$, the distribution $\mathcal{ZO}(\rho)$ draws each entry in the vector from $\{0, \pm 1\}$, with probability $\rho/2$ for each of -1 and +1, and probability being zero $1 - \rho$.
- For an integer $0 \leq h \leq N$, the distribution $\mathcal{HWT}(h)$ chooses a vector uniformly from $\{0, \pm 1\}^N$, under the condition that it has exactly h nonzero entries.

For simplicity of analysis, we will use *power-of-two* cyclotomic rings. When M is a power of two, the dual ideal $\mathcal{R}^\vee = N^{-1} \cdot \mathcal{R}$ of $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ is a simply a scaling of the ring. In this case, the RLWE problem is informally described by transforming samples $(a, b = a \cdot s + e) \in \mathcal{R}_q \times \mathcal{R}_q^\vee$ to $(a, b = a \cdot s' + e') \in \mathcal{R}_q \times \mathcal{R}_q$ 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.

Another advantage of power-of-two 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 Section 2.2 are mutually orthogonal, the encoding of plaintext can be efficiently done by rounding coefficients to nearest integers after multiplication with the matrix CRT_M^{-1}. We will also choose the ring of Gaussian integers $\mathbb{Z}[i]$ as a candidate of discrete subspace of $\mathbb{C}$ for implementation.

- $\mathsf{KeyGen}(1^\lambda)$.
  - 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 $\sigma = \sigma(\lambda, q_L)$ appropriately for $\mathsf{RLWE}_{P \cdot q_L, \sigma}(\mathcal{HWT}(h))$ that achieves at least 2^λ security.
  - Sample $s \leftarrow \mathcal{HWT}(h)$, $a \leftarrow \mathcal{U}_{q_L}$ and $e \leftarrow \mathcal{DG}_{q_L}(\sigma^2)$. 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 $b \leftarrow -as + e \pmod{q_L}$.
  - Let $s' \leftarrow s^2$. Sample $a' \leftarrow \mathcal{U}_{P \cdot q_L}$ and $e' \leftarrow \mathcal{DG}_{P \cdot q_L}(\sigma^2)$. Set the evaluation key as $evk \leftarrow (b', a') \in \mathcal{R}_{P \cdot q_L}^2$ where $b' \leftarrow -a's + e' + Ps' \pmod{P \cdot q_L}$.
- $\mathsf{Encode}(\boldsymbol{z})$. For a $(N/2)$-dimensional vector $\boldsymbol{z} = (z_j)_{j \in S} \in \mathbb{Z}[i]^{N/2}$ of Gaussian integers, compute the polynomial $z(X) = \sigma^{-1}(\iota(\boldsymbol{z}))$ where $\iota : \mathbb{C}^{N/2} \to \mathbb{H}$ is the ring isomorphism defined in Section 3.2. Return the closest polynomial of scaled polynomial $m(X) = \lfloor \Delta \cdot z(X) \rceil \in \mathcal{R}$ by rounding the coefficients of $\Delta \cdot z(X)$ to the nearest integers.

- $\mathsf{Decode}(m)$. For an input polynomial $m(X) \in \mathcal{R}$, compute the corresponding vector $\boldsymbol{m} = \iota^{-1} \circ \sigma(m) = (m(\zeta_M^j))_{j \in S} \in \mathbb{C}^{N/2}$ and return the vector $\boldsymbol{z} = \lfloor \Delta^{-1} \cdot \boldsymbol{m} \rceil$ in $\mathbb{Z}[i]^{N/2}$ by rounding its entries to the closest Gaussian integers after scaling, i.e., the entry of $\boldsymbol{z}$ of index $j \in S$ is $z_j = \left\lfloor \Delta^{-1} \cdot m(\zeta_M^j) \right\rceil \in \mathbb{Z}[i]$.
- $\mathsf{Encrypt}(m)$. Sample $v \leftarrow \mathcal{ZO}(0.5)^2$ and $e_0, e_1 \leftarrow \mathcal{DG}_{q_L}(\sigma^2)$. Output $v \cdot pk + (m + e_0, e_1) \in \mathcal{R}_{q_L}^2$.
- $\mathsf{Decrypt}(\boldsymbol{c})$. For $\boldsymbol{c} = (c_0, c_1)$, output $c_0 + c_1 \cdot s \pmod{q_\ell}$.
- $\mathsf{Add}(\boldsymbol{c}_1, \boldsymbol{c}_2)$. For $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{R}_{q_\ell}^2$, output $\boldsymbol{c}_{\mathsf{add}} \leftarrow \boldsymbol{c}_1 + \boldsymbol{c}_2 \pmod{q_\ell}$.
- $\mathsf{Mult}(\boldsymbol{c}_1, \boldsymbol{c}_2)$. For $\boldsymbol{c}_1 = (b_1, a_1), \boldsymbol{c}_2 = (b_2, a_2) \in \mathcal{R}_{q_\ell}^2$, let $(d_0, d_1, d_2) = (b_1 b_2,\ a_1 b_2 + a_2 b_1, a_1 a_2) \in \mathcal{R}_{q_\ell}^3$. Output $\boldsymbol{c}_{\mathsf{mult}} \leftarrow (d_0, d_1) + \left\lfloor \frac{1}{P}(d_2 \cdot evk \pmod{Pq_\ell}) \right\rceil \in \mathcal{R}_{q_\ell}^2$.
- $\mathsf{Rescaling}_{\ell \to \ell'}(\mathbf{c})$. For a ciphertext $\boldsymbol{c} \in \mathcal{R}_{q_\ell}^2$ at level ℓ, output $\boldsymbol{c}' \leftarrow \left\lfloor \frac{q_{\ell'}}{q_\ell} \boldsymbol{c} \right\rceil \in \mathcal{R}_{q_{\ell'}}^2$.

For noise estimation of this scheme, we need to bound the size (canonical embedding norm) of polynomials sampled from the distributions used in scheme description. We will follow the heuristic approach in [24, 13]. 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 $\mathcal{U}_q$, $\mathcal{DG}_q(\sigma^2)$, 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 $16\sigma_1\sigma_2$.

In the rest of paper, we allow non-integral polynomial plaintext for convenience of analysis so that a ciphertext $(\boldsymbol{c}, \ell, \nu, B)$ will be called a valid encryption of $m \in \mathcal{K}$ if $\boldsymbol{c} \in \mathcal{R}_{q_\ell}^2$, $\|m\|_\infty^{\mathsf{can}} \le \nu$, and $\langle \boldsymbol{c}, sk \rangle = m + e \pmod{q_\ell}$ for some polynomial $e \in \mathcal{K}$ with $\|e\|_\infty^{\mathsf{can}} \le B$. Refer Appendix B for the detail proofs of following lemmas.

Lemma 1 (Encoding and Encryption). *The encryption noise is bounded by* $B_{\mathsf{clean}} = 8\sqrt{2}\sigma N + 6\sigma\sqrt{N} + 16\sigma\sqrt{hN}$. *If* $\boldsymbol{c} \leftarrow \mathsf{Encrypt}(m)$ *is an encryption of* $m \leftarrow \mathsf{Encode}(\boldsymbol{z})$ *for some vector* $\boldsymbol{z} \in \mathbb{Z}[i]^{N/2}$ *with a scaling factor* $\Delta > N + 2B_{\mathsf{clean}}$, *then* $\boldsymbol{z}' \leftarrow \mathsf{Decode}(\mathsf{Decrypt}_{sk}(\boldsymbol{c}))$ *is equal to* $\boldsymbol{z}$.

Lemma 2 (Rescaling). *Let* $(\boldsymbol{c}, \ell, \nu, B)$ *be an encryption of* $m \in \mathcal{K}$. *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* $\boldsymbol{c}' \leftarrow \left\lfloor \frac{q_{\ell'}}{q_\ell} \boldsymbol{c} \right\rceil$ *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* $m_i \in \mathcal{R}$ *for* $i = 1, 2$, *and let* $\boldsymbol{c}_{\mathsf{add}} \leftarrow \mathsf{Add}(\boldsymbol{c}_1, \boldsymbol{c}_2)$ *and* $\boldsymbol{c}_{\mathsf{mult}} \leftarrow \mathsf{Mult}_{pk}(\boldsymbol{c}_1, \boldsymbol{c}_2)$. *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* $B_{\mathsf{mult}}(\ell) = B_{\mathsf{Ks}} \cdot q_\ell / P + B_{\mathsf{scale}}$.

Automorphisms and Permutations over the Plaintext Slots. It is known that the Galois group $\mathcal{Gal} = \mathcal{Gal}(\mathbb{Q}(\zeta_M)/\mathbb{Q})$ consists of the mappings $\kappa_k : m(X) \mapsto m(X^k) \pmod{\Phi_M(X)}$ 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 [23], applying the transformation κ_k to the polynomials is very useful for the permutation on the vector of plaintext values.

In our scheme, the vector of plaintext values encoded in a polynomial $m(X) \in \mathcal{R}$ is the evaluations points in the subgroup $S = \{i \in \mathbb{Z}_M^* : 0 < i < M/2\}$ of $\mathbb{Z}_M^*$, i.e., $\sigma(m) = (m(\zeta_M^i))_{i \in S}$. Note that S is a cyclic group of order $N/2$ with a generator 3. For any $i, j \in S$, 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 $\boldsymbol{z} = (z_i)_{i \in S} \in \mathbb{C}^{N/2}$ with the corresponding polynomial $m(x) = \sigma^{-1}(\iota(\boldsymbol{z}))$, if $k = j^{-1} \cdot i \pmod M \in S$ and $m' = \kappa_k(m)$, then we have $z'_j = m'(\zeta_M^j) = m(\zeta_M^{jk}) = m(\zeta_M^i) = z_i$, so 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 $m \in \mathcal{R}$, we denote $\kappa_k(\boldsymbol{c})$ the vector obtained by applying κ_k to the entries of ciphertext $\boldsymbol{c}$. It follows from [23] that $\kappa_k(\boldsymbol{c})$ is a valid encryption of $\kappa_k(m)$ with the secret key $\kappa_k(sk)$. 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 key sk.

Relative Error. The decrypted value of a ciphertext is an approximate value of plaintext, so the noise growth from homomorphic operations may cause a negative effect such as loss of significance. Hence it requires to dynamically manage the bound of noise of ciphertexts for a correct understanding of decrypted value. 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 $\beta_i = B_i/\nu_i$ has 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 the circuits commonly used in practical applications and analyze the noise of output ciphertext using our concrete initiation based on BGV. We start with the homomorphic evaluations of some 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. We deduce the bounds on the noise growth during evaluation. 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 $\boldsymbol{c}' \leftarrow \mathsf{Rescaling}_{\ell\to\ell'}(\mathsf{Mult}(\boldsymbol{c}_1, \boldsymbol{c}_2))$ satisfies the inequality $\beta' \le \beta_1 + \beta_2 + \beta_*$. We will discuss about the choice of β_* and validity of this assumption at the end of Section 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 $m \in \mathcal{R}$. 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.*

Note that constant addition decreases the relative error from B/ν down to $B/(\nu + \|a\|_\infty^{\mathsf{can}})$, while the constant multiplication preserves it (if $\|a\|_\infty^{\mathsf{can}} \neq 0$). Now we describe an algorithm to evaluate the power polynomial x^d for a power of two degree d. For simplicity, we assume that the bound ν of message m is equal to the base p.

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

procedure POWER2($\boldsymbol{c} \in \mathcal{R}_{q_\ell}^2, d = 2^r$)
  $\boldsymbol{c}_0 \leftarrow \boldsymbol{c}$
  for $j = 1$ to r **do**
    $\boldsymbol{c}_j \leftarrow \mathsf{Rescaling}(\mathsf{Mult}(\boldsymbol{c}_{j-1}, \boldsymbol{c}_{j-1}))$
  end for
  return $\boldsymbol{c}_r$
end procedure

For an input polynomial $m \in \mathcal{R}$ of size $\|m\|_\infty^{\mathsf{can}} \leq p$, 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 checks the correctness of Algorithm 1 and computes the relative error of the output ciphertext.

Lemma 5 (Power Polynomial of Power-of-Two Degree). *Let $(\boldsymbol{c}, \ell, p, \beta_0 \cdot p)$ be an encryption of m. 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 $\beta_d \leq d \cdot \beta_0 + (d-1) \cdot \beta_*$.*

Proof. We inductively show that $(\boldsymbol{c}_j, \ell - j, p, \beta_{2^j} \cdot p) \leftarrow \mathsf{Rescaling}(\mathsf{Mult}(\boldsymbol{c}_{j-1}, \boldsymbol{c}_{j-1}))$ is an encryption of m^{2^j}/p^{2^j-1} for some real number $\beta_{2^j} \leq 2 \cdot \beta_{2^{j-1}} + \beta_*$. 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 $\beta_{2^r} \leq 2^r \cdot \beta_0 + (2^r - 1) \cdot \beta_*$. □

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 $p \cdot f(m/p) = m^d/p^{d-1}$. Refer Appendix B for detail description of algorithms and proofs.

Lemma 6 (Power Polynomial). *Let $(\boldsymbol{c}, \ell, p, B)$ be an encryption of m 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 $\beta_d \leq d \cdot \beta_0 + (d-1) \cdot \beta_*$.*

Theorem 1 (General 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. Then Algorithm 4 outputs a valid encryption $(\boldsymbol{c}', \ell - \lceil \log d \rceil, M_f, \beta_d \cdot M_f)$ of $p \cdot f(m/p)$ for $M_f = p \cdot \sum_{j=0}^{d} \|a_j\|_\infty^{\mathsf{can}}$ and for some real number $\beta_d \leq d \cdot \beta_0 + (d-1) \cdot \beta_*$.*

If the relative error of input ciphertext satisfies $\beta_0 \leq \beta_*$, the relative error of resulting ciphertext is bounded by $\beta_d \leq d \cdot \beta_0 + (d-1) \cdot \beta_* \leq 2d \cdot \beta_0$. Hence, the precision loss is bounded

by $(\log d+1)$ bits that is comparable to loss of significance occurring in unencrypted numerical computations. The evaluation of polynomial of degree d can be done in d homomorphic multiplication 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.

Let us return to the assumption $\beta_1\beta_2 + (B_{\mathsf{mult}}(\ell) + p^{\ell-\ell'} \cdot B_{\mathsf{scale}})/(\nu_1\nu_2) \leq \beta_*$. 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 $2^{L+1} \cdot \beta_*^2$ during evaluation. The condition $2^{L+1} \cdot \beta_*^2 \leq \frac{1}{2}\beta_*$, or equivalently $\beta_* \leq 2^{-L-2}$, seems to be natural; otherwise the relative error becomes $2^{L+1} \cdot \beta_* \geq 2^{-1}$ after evaluation, so the decrypted result will have almost no information. 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 $\nu' \geq p$ 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 $\beta_* \geq p^{-1} \cdot B_{\mathsf{clean}} = \Omega(p^{-1} \cdot \sigma N)$.

4.2 Approximate Polynomials and Multiplicative Inverse

In this subsection, we 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 $R_d(x) = f(x) - T_d(x)$. Theorem 1 can be employed 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 is an approximate value of $p^{u+1} \cdot f(m/p)$. The bound of error is obtained by summing the error occurred during evaluation, the rounding error, and the remainder term $p^{u+1} \cdot R_d(m/p)$. In the case of RLWE based constructions, we should consider the corresponding vector $\sigma(m) = (m_j)_{j \in S}$ and convergence of series in each of slots.

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 $R_d(x) = \exp(x) - T_d(x)$ 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 corresponding vector $\boldsymbol{m} = (\sigma(m_j))_{j \in S}$. With the input ciphertext $\boldsymbol{c}$ and the polynomial $\lfloor p^u \cdot T_d \rceil (x)$, Algorithm 4 output an encryption of $p^{u+1} \cdot T_d(m/p)$. It is obtained by following the proof of Theorem 1 that the error of resulting ciphertext is bounded by

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

If we write $\exp(m/p) := \sigma^{-1} \circ \iota((\exp(m_j/p))_{j \in S})$, then this 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.

Let us consider another example: logistic function $f(x) = \frac{\exp(x)}{1+\exp(x)}$ that is widely used in statistics, neural networks, and machine learning as the probability function. For example, logistic regression is used for 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 [35]. Homomorphic evaluation of the logistic function $\frac{\exp(x)}{1+\exp(x)} = (1 + \exp(-x))^{-1}$ can be done by composition of the multiplicative inverse and exponential functions or direct computation Taylor decomposition.

Now we adapt a specific algorithm described in [8] to get a better complexity for multiplicative inverse. Assuming that a complex number x satisfies $|\hat{x}| \leq 1/2$ for $\hat{x} = 1 - x$, then we get

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

Note that $|\hat{x}^{2^r}| \leq 2^{-2^r}$, and this product 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.

Returning to our subject, we assume that a complex number m_j satisfies $|\hat{m_j}| \leq p/2$ for $\hat{m_j} = p - m_j$. The standard approach starts by normalizing those numbers to be in the unit interval by setting $x = m_j/p$. Since we cannot multiply fractions over encrypted data, the precision point has to move to the left for each term of (1). We multiply both sides of the equation (1) by p^{2^r}, and then it yields

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

Therefore, the product $p^{-2^r}\cdot\prod_{i=0}^{r-1}(p^{2^i}+\hat{m_j}^{2^i})$ can be seen as the approximate inverse of m_j with 2^r bits of precision. Algorithm 2 takes an encryption of $\hat{m} = \sigma^{-1}((\hat{m_j})_{j\in\mathbb{Z}_M^*})$ as an input and outputs an encryption of its scaled multiplicative inverse $p^2\cdot\sigma^{-1}(\iota((m_j^{-1})_{j\in S}))$ by evaluating the polynomial $\prod_{i=0}^{r-1}(p^{2^i}+\hat{m}^{2^i})$. The precision of resulting ciphertext and optimal number r of iterations will be analyzed in the following theorem.

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

```
1: procedure INVERSE(c ∈ R²_{q_ℓ}, r)
2:     p ← (p, 0)
3:     c_0 ← c
4:     v_1 ← p + c_0 (mod q_{ℓ−1})
5:     for j = 1 to r − 1 do
6:         c_j ← Rescaling(Mult(c_{j−1}, c_{j−1}))
7:         v_{j+1} ← Rescaling(Mult(v_j, p + c_j))
8:     end for
9:     return v_r
10: end procedure
```

Theorem 2 (Inverse Function). *Let $(\boldsymbol{c}, \ell, p/2, B_0 = \beta_0\cdot p/2)$ be an encryption of $\hat{m}\in\mathcal{K}$ 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 $\beta \leq \beta_0 + r\beta_*$.*

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 $(\boldsymbol{c}_j, \ell-j, 2^{-2^j}\cdot p, \beta_j\cdot 2^{-2^j}\cdot p)$ are valid encryptions of $\hat{m}^{2^j}/p^{2^j-1}$ for some real numbers $\beta_j \leq 2^j\cdot(\beta_0+\beta_*)$, and so $(\boldsymbol{p}+\boldsymbol{c}_j, \ell-j, (1+2^{-2^j})p, \beta_j\cdot 2^{-2^j}\cdot p)$ are valid encryptions of $p+\hat{m}^{2^j}/p^{2^j-1} = (p^{2^j}+\hat{m}^{2^j})/p^{2^j-1}$ with relative errors $\beta_j' \leq \beta_j/(2^{2^j}+1) \leq 2^j\cdot(\beta_0+\beta_*)/(2^{2^j}+1)$, 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 $\prod_{i=0}^{j-1}(p^{2^i}+\hat{m}^{2^i})/p^{2^j-2} = p\cdot\prod_{i=0}^{j-1}(1+(\hat{m}/p)^{2^i})$ for some real number $\beta''_j \le \sum_{i=0}^{j-1}\beta'_i + (j-1)\cdot\beta_*$. Note that the message bound 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 forms 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 real $\beta \le \beta_0 + r\cdot\beta_*$. □

Let $m^{-1}(X) := \sigma^{-1}(\iota((m_j^{-1})_{j\in S}))$ be the polynomial corresponding to the vector obtained by slot-wise multiplicative inversion from the vector $\iota^{-1}(\sigma(m)) = (m_j)_{j\in S}$. Then the output ciphertext $(\boldsymbol{v}_r, \ell-r, 2p, \beta\cdot 2p)$ of previous theorem can be also viewed as an encryption of $p^2\cdot m^{-1}$. The bound of error 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 $p^2\cdot m^{-1}$. Its relative error is $\beta + 2^{-2^r} \le \beta_0 + r\beta_* + 2^{-2^r}$, which is minimized when $r\beta_* \approx 2^{-2^r}$. Specifically, the choice of $r = \lceil \log\log \beta_*^{-1}\rceil$ makes this value bounded by $\beta_0 + r\beta_* + 2^{-2^r} \le \beta_0 + 2r\beta_* = \beta_0 + 2\lceil\log\log\beta_*^{-1}\rceil\cdot\beta_*$. 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 number r of iterations 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 value $0 < \epsilon < 1$. Applying Theorem 2, 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 $\epsilon^{2^r}\cdot 2p$. The optimal number of iteration 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. Let $W_d(y) = (y^{i\cdot j})_{0\le i,j<d}$ be the Vandermonde matrix generated by $\{y^i : 0 \le i < d\}$. 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. [14] 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.

Different from [14], our RLWE-based HE scheme supports batch technique based on the complex canonical embedding map as described in Section 3.2. In the slot of index $k \in \mathbb{Z}_M^*$, 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 batch scheme is still meaningful because the evaluation result of whole pipeline consisting of DFT, Hadamard operations, and inverse DFT is independent of index $k \in \mathbb{Z}_M^*$, even though $W_d(Y)$ corresponds to the DFT matrices generate by different primitive d-th roots of unity.

It is obtained 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 B/ν of input ciphertexts is preserved. Our FFT algorithm takes twice longer than the results of [14] in the same parameter setting due to complex structure, but the amortized time is much better thanks to our own message packing method. 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 restore the magnitude of messages and reduce the required level for whole pipeline.

The fast polynomial multiplication using the FFT algorithm is a typical example which computes the exact value using approximate arithmetic. Particularly in 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. This paradigm of *exact computation with approximate arithmetic* is another direction to apply our HE in practice.

5 Implementation Results

In this section we describe how to select parameters for evaluating arithmetic circuits described in Section 4. We also provide the implementation results with concrete parameters. Every experiment in our paper was performed on a machine with an Intel Core i5 running at 2.9 GHz processor without multi-threading method using a parameter set with 80-bit security level.

We need to set the ring dimension N to should satisfy the required security condition $N \geq \frac{\lambda+110}{7.2} \log(P \cdot q_L)$ to get λ-bit security level. [28, 24] 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. Each coefficient of the secret key is chosen at random from $\{0, \pm 1\}$ and we set 64 as the number of nonzero coefficients in the secret key.

Evaluation of Typical Circuits. In Table 2, we present the parameter setting and performance results for computing the product of ciphertexts, the 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 Section 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 evaluation of the circuit x^{1024} with an input of 22-bit precision is hard to be implemented in practice over previous methods. Meanwhile, our scheme can compute this circuit over 2^{14} slots simultaneously in about 8.53 seconds, yielding an amortized rate of 0.52 milliseconds per slot. Computation of the multiplicative inverse x^{-1} is done by evaluating the polynomial up to degree 16 as described in Algorithm 2. It gives an amortized time per slots of about 0.17 milliseconds. In 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 0.24 milliseconds.

Significancen Loss. In Section 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 Figure 4, we depict bit precisions of

Table 2. Implementation results for homomorphic evaluation of typical circuits

Function	N	$\lceil \log q \rceil$	$\lceil \log p \rceil$	Consumed levels	Bit precision of input	Total time	Amortized time
x^{16}	2^{13}	150	30	4	15	0.43s	0.10ms
x^{1024}	2^{15}	440	40	10	22	8.53s	0.52ms
x^{-1}	2^{13}	150	25	5	9	0.69s	0.17ms
$\exp(x)$	2^{13}	175	35	5	20	0.98s	0.24ms

output ciphertexts for 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 circuit depth is 4 (or resp. 10).

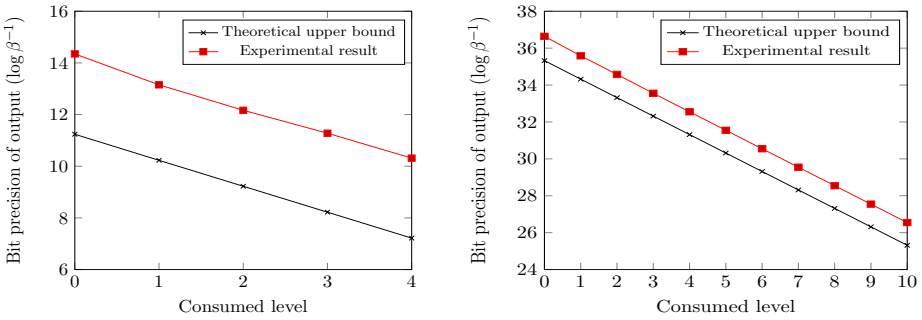

Fig. 4. The variation of bit precision of ciphertexts when $(N, \log p, \log q) = (2^{13}, 30, 150)$ and $(2^{15}, 40, 440)$

Logistic Function. Suppose that we are given by the logistic regression function $f(x) = \frac{\exp(x)}{1+\exp(x)}$. It can be approximated by a 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 and ring degree N 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 circuit depth which strictly limits the depth of implementable circuits. On the other hand, the rescaling procedure in our scheme has the advantage that it significantly reduces the size of parameters (e.g., $(\log p, \log q) = (35, 175)$).

The parallelized computation of logistic function is especially important in real world applications such as statistic analysis using multiple data. In previous approach, each slot of plaintext space should have a larger degree than encoded polynomials so they could support only a few number 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.

Discrete Fourier Transform. With the parameters $N = 2^{13}$ and $\log p = 50$, we encrypt coefficients of polynomials and homomorphically evaluate the standard processing (FFT-Hadamard product of two vectors-inverse FFT) in 73 minutes (amortized 1.06 seconds per

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

Method	N	$\lceil \log q \rceil$	Degree of polynomial	Amortization amount	Total time	Amortized time
[3]	2^{14}	512	7	-	$> 30s$	-
[9]	17430	370	7	-	$1.8s$	-
Ours	2^{13}	175	7	2^{12}	0.79s	0.19ms
	2^{14}	210	9	2^{13}	2.36s	0.29ms

slot) when $d = 2^{13}$. We could reduce down the evaluation time to 19 minutes (amortized 0.28 seconds per slot) by adapting the multi-threading method on a machine with four cores, compared to 17 minutes of the previous work [14] 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 $N = 2^{14}$ and $\log p = 50$ in amortized time of 1.10 seconds.

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

Method	d	N	$\lceil \log q \rceil$	Degree of Hadamard operation	Amortization amount	Total time	Amortized time
[14][3]	2^4	2^{13}	150	2	-	0.46s	-
	2^{13}	2^{14}	192	2	-	17min	-
Ours[4]	2^4	2^{13}	100	2	2^{12}	0.88s	0.22ms
	2^{13}	2^{13}	100	2	2^{12}	19min	0.28s
	2^{13}	2^{14}	200	8	2^{13}	2.5h	1.10s

References

1. S. Arita and S. Nakasato. Fully homomorphic encryption for point numbers. Cryptology ePrint Archive, Report 2016/402, 2016. `http://eprint.iacr.org/`.
2. J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In *Cryptography and Coding*, pages 45–64. Springer, 2013.
3. J. W. Bos, K. Lauter, and M. Naehrig. Private predictive analysis on encrypted medical data. *Journal of biomedical informatics*, 50:234–243, 2014.
4. Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, 2012.
5. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *Proc. of ITCS*, pages 309–325. ACM, 2012.
6. Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, FOCS'11, pages 97–106. IEEE Computer Society, 2011.
7. Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from Ring-LWE and security for key dependent messages. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011.

8. G. S. Çetin, Y. Doröz, B. Sunar, and W. J. Martin. An investigation of complex operations with word-size homomorphic encryption. Cryptology ePrint Archive, Report 2015/1195, 2015. http://eprint.iacr.org/2015/1195.
9. J. H. Cheon, J. Jung, J. Lee, and K. Lee. Privacy-preserving computations of predictive medical models with minimax approximation and non-adjacent form. To appear in WAHC 2017.
10. J. H. Cheon, M. Kim, and K. Lauter. Homomorphic computation of edit distance. In *International Conference on Financial Cryptography and Data Security*, pages 194–212. Springer, 2015.
11. J. H. Cheon and D. Stehlé. Fully homomophic encryption over the integers revisited. In *Advances in Cryptology–EUROCRYPT 2015*, pages 513–536. Springer, 2015.
12. J.-S. Coron, T. Lepoint, and M. Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In *Public-Key Cryptography–PKC 2014*, pages 311–328. Springer, 2014.
13. A. Costache and N. P. Smart. Which ring based somewhat homomorphic encryption scheme is best? In *Cryptographers Track at the RSA Conference*, pages 325–340. Springer, 2016.
14. A. Costache, N. P. Smart, and S. Vivek. Faster homomorphic evaluation of discrete fourier transforms. Cryptology ePrint Archive, Report 2016/1019, 2016. http://eprint.iacr.org/2016/1019.
15. A. Costache, N. P. Smart, S. Vivek, and A. Waller. Fixed point arithmetic in SHE schemes. Cryptology ePrint Archive, Report 2016/250, 2016. http://eprint.iacr.org/2016/250.
16. I. Damgård, V. Pastro, N. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology–CRYPTO 2012*, pages 643–662. Springer, 2012.
17. M. v. Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010.
18. Y. Doröz, Y. Hu, and B. Sunar. Homomorphic aes evaluation using the modified ltv scheme. *Designs, Codes and Cryptography*, 80(2):333–358, 2016.
19. N. Dowlin, R. Gilad, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. Manual for using homomorphic encryption for bioinformatics. 2015. http://research.microsoft.com/pubs/258435/ManualHE.pdf.
20. L. Ducas and D. Micciancio. Fhew: Bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology–EUROCRYPT 2015*, pages 617–640. Springer, 2015.
21. J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.
22. C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. http://crypto.stanford.edu/craig.
23. C. Gentry, S. Halevi, and N. P. Smart. Fully homomorphic encryption with polylog overhead. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 465–482. Springer, 2012.
24. C. Gentry, S. Halevi, and N. P. Smart. Homomorphic evaluation of the AES circuit. In R. Safavi-Naini and R. Canetti, editors, *Advances in Cryptology - CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer, 2012.
25. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013*, pages 75–92. Springer, 2013.
26. A. Jäschke and F. Armknecht. Accelerating homomorphic computations on rational numbers. In *International Conference on Applied Cryptography and Network Security*, pages 405–423. Springer, 2016.
27. K. Lauter, A. López-Alt, and M. Naehrig. Private computation on encrypted genomic data. In *International Conference on Cryptology and Information Security in Latin America*, pages 3–27. Springer, 2014.
28. R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. In *Topics in Cryptology–CT-RSA 2011*, pages 319–339. Springer, 2011.
29. A. López-Alt, E. Tromer, and V. Vaikuntanathan. 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*, pages 1219–1234. ACM, 2012.
30. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *Advances in Cryptology - EUROCRYPT 2010*, pages 1–23, 2010.
31. V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for ring-lwe cryptography. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 35–54. Springer, 2013.
32. M. Naehrig, K. Lauter, and V. Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 113–124. ACM, 2011.
33. N. P. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In P. Nguyen and D. Pointcheval, editors, *PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer, 2010.
34. N. P. Smart and F. Vercauteren. Fully homomorphic SIMD operations. *Designs, codes and cryptography*, 71(1):57–81, 2014.

35. B. P. Tabaei and W. H. Herman. A multivariate logistic regression equation to screen for diabetes development and validation. *Diabetes Care*, 25(11):1999–2003, 2002.
36. S. Wang, Y. Zhang, W. Dai, K. Lauter, M. Kim, Y. Tang, H. Xiong, and X. Jiang. Healer: Homomorphic computation of exact logistic regression for secure rare disease variants analysis in gwas. *Bioinformatics*, 32(2):211–218, 2016.

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 a B_{err}-bounded error distribution $\chi = \chi(\lambda, q_L)$ appropriately for LWE problem of parameter (n, q_L, χ) that achieves at least 2^λ security. 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 $A \leftarrow \mathbb{Z}_{q_L}^{\tau \times N}$, $\boldsymbol{e} \leftarrow \mathcal{DG}_{q_L}(\sigma^2)^\tau$ 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 $\boldsymbol{e}' \leftarrow \mathcal{DG}_{q_L}(\sigma^2)^{N^2\lceil \log q_L \rceil}$, and let $\boldsymbol{b}' \leftarrow -A'\boldsymbol{s}' + \boldsymbol{e}'$. Set the evaluation key as $evk \leftarrow (\boldsymbol{b}', A') \in \mathbb{Z}_{q_L}^{N^2\lceil \log q_L \rceil \times (N+1)}$.
- $\mathsf{Encrypt}(m)$. 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 $\boldsymbol{c}_{\mathsf{mult}} \leftarrow evk^T \cdot \boldsymbol{c}' \pmod{q_\ell}$.
- $\mathsf{Rescaling}_{\ell \to \ell'}(\mathbf{c})$. 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 Algorithms and Proofs

Proof of Lemma 1.

Proof. We choose $v \leftarrow \mathcal{ZO}(0.5)^2$ and $e_0, e_1 \leftarrow \mathcal{DG}_{q_L}(\sigma^2)$, 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 $m = \mathsf{Encode}(\boldsymbol{z})$ can be seen as a valid encryption of $\Delta \cdot \sigma^{-1}(\iota(\boldsymbol{z}))$ 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 of decoding algorithm. □

Proof of Lemma 2.

Proof. There is a polynomial $e \in \mathcal{R}$ such that $\langle \boldsymbol{c}, sk \rangle = m + e \pmod{q_\ell}$ and $\|e\|_\infty^{\mathsf{can}} \le B$. 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 $e_i \in \mathcal{R}$ such that $\|e_i\|_\infty^{\mathsf{can}} \le B_i$. Let $(d_0, d_1, d_2) = (b_1 b_2, a_1 b_2 + a_2 b_1, a_1 a_2)$. This vector can be viewed as a level-ℓ encryption of $m_1 m_2$ with an error $m_1 \cdot e_2 + m_2 \cdot e_1 + e_1 \cdot e_2$ 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 $e'' = P^{-1} \cdot d_2 e'$ 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 $m_1 m_2$ with an error bounded by

$$\|m_1 e_2 + m_2 e_1 + e_1 e_2 + e''\|_\infty^{\mathsf{can}} + B_{\mathsf{scale}} \le \nu_1 B_2 + \nu_2 B_1 + B_1 B_2 + B_{\mathsf{Ks}} \cdot q_\ell / P + B_{\mathsf{scale}},$$

as desired. □

Proof of Lemma 4.

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 $\|a \cdot e\|_\infty^{\mathsf{can}} \le \|a\|_\infty^{\mathsf{can}} \cdot B$. □

Proof of Lemma 6.

Algorithm 3 Power polynomial $f(x) = x^d$

```
procedure POWER(c ∈ R_{q_ℓ}^2, d)
    Compute the binary representation d = 2^{r_1} + ··· + 2^{r_l} with r_1 < ··· < r_l = ⌊log d⌋.
    c_1 ← POWER2(c, 2^{r_1})
    v_1 ← c_1
    for j = 2 to l do
        c_j ← POWER2(c_{j−1}, 2^{r_j − r_{j−1}})
        v_j ← Rescaling(Mult(c_j, v_{j−1}))
    end for
    return v_l
end procedure
```

Proof. The case $l = 1$ is done by Lemma 5, so we assume $l \ge 2$. It follows from the proof of Lemma 5 that Algorithm 3 forms the ciphertexts $(\boldsymbol{c}_j, \ell - r_j, p, \beta_{2^{r_j}} \cdot p)$ which are valid encryptions of the messages $m^{2^{r_j}}/p^{2^{r_j}-1}$ for some real numbers $\beta_{2^{r_j}} \le 2^{r_j}(\beta_0 + \beta_*) - \beta_*$. Using the induction on j, we can demonstrate that $(\boldsymbol{v}_j, \ell - r_j - 1, p, \beta_{2^{r_1}+\cdots+2^{r_j}} \cdot p)$ is a valid encryption of $m^{2^{r_1}+\cdots+2^{r_j}}/p^{2^{r_1}+\cdots+2^{r_j}-1}$ for some real number $\beta_{2^{r_1}+\cdots+2^{r_j}} \le (2^{r_1} + \cdots + 2^{r_j}) \cdot (\beta_0 + \beta_*) - \beta_*$. We conclude the proof by taking $j = l$. □

Algorithm 4 Polynomial function $f(x) = \sum_{j=0}^{d} a_j x^j$

procedure POLYNOMIAL($\boldsymbol{c} \in \mathcal{R}_{q\ell}^2, f(x) = \sum_{j=0}^{d} a_j x^j, a_j \in \mathcal{R}$)
  $\boldsymbol{v} \leftarrow p \cdot (a_0, 0)$
  for $j = 1$ to d **do**
    $\boldsymbol{v} \leftarrow \mathsf{Add}(\boldsymbol{v}, a_j \cdot \text{POWER}(\boldsymbol{c}, j))$
  end for
  return $\boldsymbol{v}$
end procedure

Proof of Theorem 1.

Proof. For $j = 1, \ldots, d$, let $\boldsymbol{w}_j \leftarrow \text{POWER}(\boldsymbol{c}, j)$ and $\boldsymbol{u}_j \leftarrow a_j \cdot \boldsymbol{w}_j$. It follows from Lemma 6 and Lemma 4 that $(\boldsymbol{w}_j, \ell - \lceil \log j \rceil, p, \beta_j \cdot p)$ and $(\boldsymbol{u}_j, \ell - \lceil \log j \rceil, \|a_j\|_\infty^{\mathsf{can}} \cdot p, \beta_j \cdot \|a_j\|_\infty^{\mathsf{can}} \cdot p)$ are valid encryptions of m^j/p^{j-1} and $a_j \cdot m^j/p^{j-1}$, respectively, for some $\beta_j \le j \cdot \beta_0 + (j-1) \cdot \beta_*$. Therefore, the output $\boldsymbol{v}$ of Algorithm 4 is a valid encryption of $pa_0 + \sum_{j=1}^{d} a_j \cdot m^j/p^{j-1} = p \cdot f(m/p)$, with the message bound $p \cdot \sum_{j=0}^{d} \|a_j\|_\infty^{\mathsf{can}} = p \cdot \|f\|_1$ and a relative error bounded by $\max\{\beta_j : 1 \le j \le d\} \le d \cdot \beta_0 + (d-1) \cdot \beta_*$. □