TOTA: Fully Homomorphic Encryption with Smaller Parameters and Stronger Security

Zhaomin Yang[1][2], Xiang Xie[1][2], Huajie Shen[3][⋆], Shiying Chen[3][⋆], and Jun Zhou[3]

[1] Shanghai Key Laboratory of Privacy-Preserving Computation
[2] MatrixElements Technologies
[3] East China Normal University

Abstract. We present fully homomorphic encryption schemes for fixed-point arithmetic with fixed precision. Our scheme achieves $\mathsf{IND\text{-}CPA^D}$ security and uses RLWE ring with dimension 2^{13} or less. Our techniques could also be extended to construct fully homomorphic encryption schemes for approximate numbers with $\mathsf{IND\text{-}CPA}$ security. The bootstrapping process of our $\mathsf{IND\text{-}CPA}$ scheme preserves about 39-bit precision with ring dimension 2^{13}, which is the first construction that preserves high precision while keeping the parameters small.
The core technique in this paper is a new and efficient functional bootstrapping algorithm that avoids the negacyclicity constraint of the evaluated functions, which enables us to extract bits blocks homomorphically. This new functional bootstrapping algorithm could be applied to BFV and TFHE schemes as well, and is of independent interest.

Keywords: Fully Homomorphic Encryption · Functional Bootstrapping · Approximate Encryption.

1 Introduction

Fully homomorphic encryption (FHE) [38] allows to perform arbitrary computation on encrypted data without decrypting the ciphertexts. Since the breakthrough work by Gentry [23], this field has been made a lot of progress both in theory and in practice. The core of Gentry's blueprint is a bootstrapping technique allows to refresh a ciphertext into a new one that encrypts the same message while containing smaller errors. Bootstrapping is the only known technique to construct FHE, and all existing constructions, e.g., [8,7,6,25,2,20,22,16], are based on this framework.

Different types of FHE schemes are proposed in the literature. BFV [6,21] and BGV [7] focus on homomorphic operations on finite field, FHEW [20] and TFHE [14,15,16] provide fast bootstrapping for binary operations, and CKKS [13] is dedicated for approximate numbers. In this paper, we focus on the fast bootstrapping procedure in FHEW, which is further improved in the TFHE scheme.

Functional bootstrapping, which is implicitly used in the TFHE scheme and formally introduced in [4], is a generalization of bootstrapping. It allows to refresh the error while performing some pre-determined function f on the encrypted message simultaneously. Functional bootstrapping turns out to be very useful, especially for oblivious inference in privacy-preserving machine learning. It is an effective method to deal with non-linear layers in machine learning models. Actually, [5,28] compute the sign function in the functional bootstrapping procedure to handle neural networks. However, the functions evaluated are subject to some constraints, it is an interesting problem to perform functions without the constraints in functional bootstrapping.

Let us first recall the main idea in FHEW. Based on the learning with errors problem [37], a ciphertext on m is of the form $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + m' + e \bmod q)$, where $\mathbf{a} \leftarrow \mathbb{Z}_q^N$, $\mathbf{s} \leftarrow \{0,1\}^N$ are uniformly random, e is

[⋆] Work done as an internship at Shanghai Key Laboratory of Privacy-Preserving Computation and MatrixElements Technologies.

small and $m' = \mathsf{encode}(m)$ for some encoding algorithm. One could compute $\mathsf{decode}(b - \langle \mathbf{a}, \mathbf{s} \rangle)$ to recover m, where decode is the corresponding decoding algorithm. For simplicity, we assume encode and decode both be the identity function. The FHEW bootstrapping leverages the ring structure of $\mathcal{R}_Q = \mathbb{Z}_Q[X]/(X^N + 1)$ for a larger modulus Q. A key observation is that the elements $\{1, X, X^2, ..., X^{N-1}, X^N, ..., X^{2N-1}\}$[4] in $\mathcal{R}_Q$ forms a multiplicative group of order $2N$. This allows to encrypt the elements in $\mathbf{a}$ and b in the exponent when $q = 2N$. Given a bootstrapping key that encrypts each element of $\mathbf{s}$, we could homomorphically compute an encryption of $v(X) \cdot X^{b - \langle \mathbf{a}, \mathbf{s} \rangle \bmod 2N} = v(X) \cdot X^{m'+e}$, where $v(X) \in \mathcal{R}_Q$ is some test vector. For a function $f : \mathbb{Z}_{2N} \mapsto \mathbb{Z}_Q$, we set the test vector as $v(X) = f(0) - f(N-1)X - \cdots - f(1)X^{N-1}$. Extract the constant term of $v(X) \cdot X^{m'+e}$ will result an LWE ciphertext that encrypts $f(m' + e)$. However, note that we could only embed N values in the coefficients and $X^N = -1$, the function f should satisfy that $f(x+N) = -f(x) \bmod Q$ for $0 \leq x < N$ to get the correct ciphertext. This property is called negacyclicity. This constraint heavily limits the usability of functional bootstrapping, and we will show later that functional bootstrapping for arbitrary functions will benefit us to solve some interesting problems and even improve the original bootstrapping.

In this paper, we focus on the CKKS-like scheme [13]. It provides an approximate encryption for fixed-point numbers. More specifically, the encoding function is set to $\mathsf{encode}(x) = \lfloor \Delta x \rceil$ for some large Δ and $x \in \mathbb{R}$. The decoding $\mathsf{decode}(x) = x/\Delta$ removes the scale and returns a real number. Although the decoding function is not exactly correct, the resulting message is very close to the input of the encoding function. The error occurs in the encryption and homomorphic operation is viewed as part of the approximation error of the fixed-point number. This is acceptable in applications where fixed-point or floating-point numbers are applied. Due to the high efficiency of the CKKS scheme, it is widely used in applications related to privacy-preserving machine learning.

For practicality reasons, we often consider the $\mathsf{IND\text{-}CPA}$ security of homomorphic encryption, which means the public key and ciphertext do not leak any information of the plaintext. A recent work by Li and Micciancio [35] pointed out that the $\mathsf{IND\text{-}CPA}$ security may not adequately capture security against passive adversaries when applied to approximate encryption. In fact, they present concrete attacks to the CKKS scheme assuming the existence of passive decryption oracle. The main reason behind these attacks is that the approximation error leaks information of the secret key.

Li and Micciancio [35] formally define a stronger notion called $\mathsf{IND\text{-}CPA^D}$ security. In addition to the original $\mathsf{IND\text{-}CPA}$ model, the adversary could passively access a decryption oracle. As demonstrated in [35], $\mathsf{IND\text{-}CPA^D}$ security is equivalent to $\mathsf{IND\text{-}CPA}$ security for homomorphic encryption schemes with exact decryption, while $\mathsf{IND\text{-}CPA^D}$ security is strictly stronger than $\mathsf{IND\text{-}CPA}$ for approximate homomorphic encryption schemes. A natural method to achieve $\mathsf{IND\text{-}CPA^D}$ security for CKKS is to use noise flooding. It adds a sufficiently large error to mask the original error (may leak information of secret key) in the decryption procedure. Although this method fills the security gap, it involves in using multi-precision modulus and large ring dimensions, especially when turning CKKS into an FHE scheme. To construct a CKKS-like scheme under $\mathsf{IND\text{-}CPA^D}$ security without noise flooding is left as an open problem in [35]. It is also open to construct CKKS-like FHE schemes under $\mathsf{IND\text{-}CPA^D}$ security without noise flooding.

Research on bootstrapping of the CKKS scheme is very active recently. The first bootstrapping for CKKS is presented in [12], all the follow-up works [10,27,33,34,32,3] apply the same framework to design improved bootstrapping for CKKS. In a nutshell, it requires to approximate the modular reduction function with large-depth homomorphic operations. The errors amplified by these operations will distort the plaintext significantly. The bootstrapping procedures can only achieve 20-bit precision until using the very recent techniques in [32]. [32] devises better approximate approaches to achieve about 40-bit precision with ring dimension 2^{16}. It is still unknown how to design bootstrapping for CKKS-like scheme with small parameters, say with dimension 2^{13} or less.

[4] Note that $X^N = -1$ in $\mathcal{R}_Q$.

1.1 Our Contributions

In this paper, we propose an efficient functional bootstrapping algorithm, which we call fully functional bootstrapping, that allows to perform arbitrary functions with in domain $\mathbb{Z}_N$ on the plaintext. We note that in our setting the ciphertext modulus is N, it is different from the case that restricting the plaintext in the interval $[0, N)$ with ciphertext modulus $2N$. Although the plaintext size is 1-bit smaller, our fully functional bootstrapping algorithm is simple and efficient, and is of independent interest.

Based on our fully functional bootstrapping algorithm, we construct a fully homomorphic encryption scheme TOTA with $\mathsf{IND\text{-}CPA^D}$ security for fixed-point arithmetic with fixed precision. By fixed precision, we mean that the precision always preserves after different operations, e.g., multiplication on fixed-point numbers. More specifically, given two fixed-point numbers x, y with precision p, the fixed-precision multiplication drops the last p bits of xy to ensure it still preserves p bits. In fact, we construct an efficient homomorphic truncation algorithm based on our fully functional bootstrapping algorithm with ring dimension 2^{13}. From another perspective, instead of encrypting approximate numbers, we perform operations on approximate circuits. Given a fixed-point or floating-point circuit, we first approximate it with a fixed-precision circuit and perform our FHE scheme with exact decryption on the new circuit.

Our techniques could be applied to approximate encryption schemes as well, which results in a totally different bootstrapping procedure for CKKS-like schemes. Our new bootstrapping technique does not need to approximate the modular reduction function and only introduces very small errors in the new ciphertext, which enables us to preserve high precision with small parameters. In fact, our FHE for approximate numbers preserves about 39-bit precision with ring dimension 2^{13}.

We implement TOTA from scratch with C++ and conduct a series of numerical experiments under different parameter sets which provide at least 127-bit security. For $\mathsf{IND\text{-}CPA^D}$ security, the truncation algorithm of TOTA runs in about $77s$ ($50s$) for 14-bit (12-bit) fixed precision. For $\mathsf{IND\text{-}CPA}$ security, the algorithm runs in about $68s$ ($8s$) for about 39-bit (11-bit) precision.

1.2 Overview of Our Techniques

Let us briefly describe our fully functional bootstrapping algorithm. Let $(\mathbf{a}, b) = (\mathbf{a}, -\langle \mathbf{a}, \mathbf{s}\rangle + m + e \bmod N)$ be a ciphertext with module N, where N is the ring dimension of the bootstrapping key. As discussed before, the FHEW bootstrapping technique encrypts $\mathbf{a}, b$ in the exponent and performs the decryption circuit homomorphically. However, we can not directly apply this method in this case, since the multiplicative group is of order $2N$, while the decryption circuit is performed in $\mathbb{Z}_N$.

To overcome this problem, we view $(\mathbf{a}, b)$ as elements in $\mathbb{Z}_{2N}$. A key observation is that we can write $b + \langle \mathbf{a}, \mathbf{s}\rangle = m + e + kN \bmod 2N$ for some $k \in \{0, 1\}$. In fact, k is the most significant bit of $m+e+kN$. Apply the original (functional) bootstrapping method in FHEW, we get a ciphertext of $-kN$ with modulus $2N$. Thus, we obtain a ciphertext of $m + e$ with modulus $2N$ by adding the two ciphertexts. Given any function f defined in $\mathbb{Z}_N$, we extend it to $F(x)$ defined in $\mathbb{Z}_{2N}$ by setting $F(x) = f(x)$ for $0 \le x < N$ and $F(x) = -f(x - N)$ for $N \le x < 2N$. Note that $m + e \in [0, N)$, we could perform the functional bootstrapping algorithm again on F and get the desired result. Our fully functional bootstrapping only involves two original functional bootstrapping procedures and one homomorphic addition. It could be applied to BFV-like and TFHE-like ciphertexts as well.

In order to construct FHE schemes with $\mathsf{IND\text{-}CPA^D}$ security, we first define fixed-precision arithmetic. Denote $\mathbb{FP}_p$ as a set of fixed-point numbers with p precision. Multiplication on $\mathbb{FP}_p$ is defined as $\mathsf{Trunc}_p(xy)$ for $x, y \in \mathbb{FP}_p$ by dropping the last p bits of xy. Note that the decryption will recover the exact $\mathbb{FP}_p$ value as long as the scale factor is sufficiently large and the errors do not distort the plaintext. In applications related to fixed-point or floating-point arithmetic, we first approximate the circuit with a fixed-precision circuit and then perform operations on $\mathbb{FP}_p$ with exact decryption.

It remains to design an efficient homomorphic truncation procedure. Actually, we design an extraction algorithm to extract bits blocks using our fully functional bootstrapping algorithm. Given a ciphertext of m under modulus q, assume q is a power of 2 and $N|q$. For an integer δ, one could drop the last δ bits of

m and then leftshift it by ℓ_e bits by switching the modulus from q to $(2^{\ell_e}q)/2^{\delta}$. The ℓ_e bits (all 0's) are reserved to eliminate the errors. Let $d = \log N - \ell_e$, we get d bits of m starting from position δ by taking the ciphertext modulo N. Then, invoking the fully functional bootstrapping with $f = 2^{\delta'} \cdot \lfloor x/2^{\ell_e} \rceil$ on the new ciphertext refreshes the errors and puts the d bits into position δ'. With careful noise analysis, we could set $d \approx \ell_e \approx (\log N)/2$.

With the bits extraction algorithm, we could truncate the message as follows. Given two ciphertexts of Δx and Δy, where Δ is a scale factor such that $\log \Delta > 2p$, the multiplication of the two ciphertexts results in a new ciphertext of $\Delta^2 xy$, where $xy \in \mathbb{FP}_{2p}$. Note that Δ is sufficiently large such that the error in the multiplication will not distort xy. We first extract the last p bits of xy by using the bits extraction algorithm starting from position $2\log\Delta - 2p$, then subtract it to get a ciphertext of $\mathsf{Trunc}_p(xy)$ with scale Δ^2 and large errors. Finally, we set the scale to Δ and refresh the errors by invoking the bits extraction algorithm again. The basic workflow of our truncation algorithm is given in Fig 1.

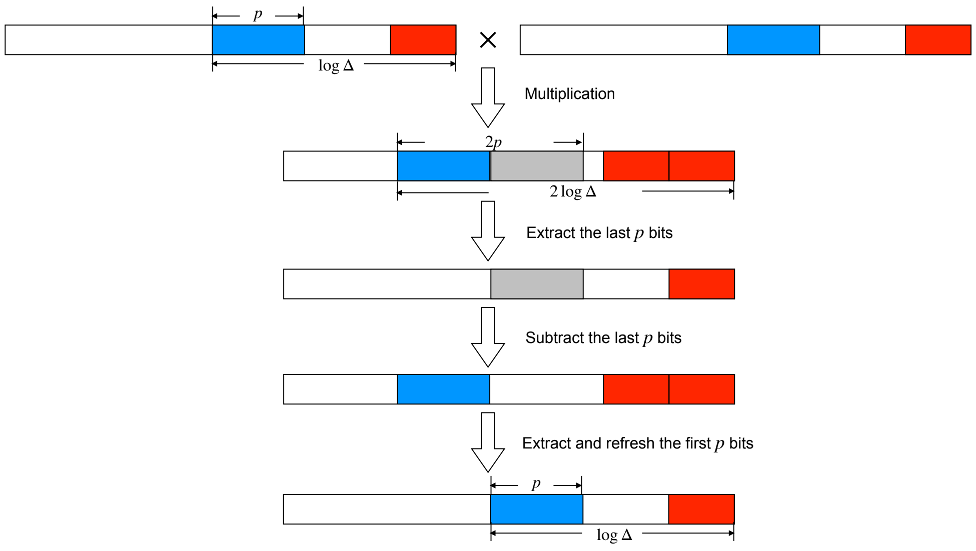
p
$\log\Delta$
$\times$
Multiplication
$2p$
$2\log\Delta$
Extract the last p bits
Subtract the last p bits
Extract and refresh the first p bits
p
$\log\Delta$

Fig. 1. Overview of Homomorphic Truncation

However, this technique incurs another problem. As depicted in the last step of Fig 1, it moves the bits blocks to the lower position. If the encrypted number is negative, which is represented in the two's complement form, this method will not preserve all the higher bits (e.g., all 1's) in the original plaintext which results in a wrong message in the final ciphertext. Note that there is no such problem when the plaintext is positive, because the higher bits are all 0's. To overcome this problem, we apply our fully functional bootstrapping algorithm to the last d-bit block with a carefully designed function. If the most significant bit of the last d-bit block is 0, the function outputs the d-bit block. Otherwise, it brings the higher bits to the output. Since the higher bits are publicly known (i.e., all 1's), the function could be defined in advance.

The bits extraction algorithm could be easily extended to bootstrap approximate encryption. In this case, $p \approx \log\Delta$ since approximation errors are allowed. We do not need to drop the last p bits, and could apply the bits extraction algorithm at a position close to $\log\Delta$ directly. The main difference is when invoking the bits extraction algorithm in the first d-bit block, there is not enough bits reserved to eliminate the errors. The extracted block may be different from the original one. Fortunately, we show that the whole plaintext extracted in this case is also very close to the original plaintext. The approximation errors in this case are dominated by the errors in our fully functional bootstrapping algorithm, which is very small. Thus, the approximate FHE is able to preserve high precision while keeping the ring dimension small.

1.3 Related and Concurrent Works

Li and Micciancio [35] introduce the notion of $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$, which captures the passive security of approximate homomorphic encryption. Functional bootstrapping is formally introduced in [4], applications and further improvements are proposed in [5,28,9,29,26,17]. The method of bootstrapping on CKKS scheme is developed by a series of works [12,10,27,33,34,32,3], which approximates the modular reduction function with polynomial evaluations.

Concurrent Works. Very recently, Chillotti et al. [18] and Kluczniak and Schild [31] present functional bootstrapping algorithms without the negacyclicity restriction independently. We note that our fully functional bootstrapping algorithm is different from theirs, and is simpler and more efficient.

Chillotti et al. [18] propose two methods to perform functional bootstrapping avoiding negacyclicity, which allow to evaluate any function on domain $[0, 2N)$. Both methods have to invoke at least two bootstrapping procedures and one multiplication by extending the BFV-like multiplication to TFHE. The BFV-like multiplication causes fast noisy growth in their scheme, it is also the main reason that they could not use small parameters as in TFHE and do not report concrete performance.

Kluczniak and Schild [31] present full domain functional bootstrapping to perform any function on $[0, 2N)$ without the restriction of negacyclicity. They split a function $F(x)$ defined on $[0, 2N)$ into two negacyclic functions F_0 and F_1, and then invoke multiple functional bootstrapping procedures to compute a GSW ciphertext of a bit b indicating which interval the message lies in. Finally, they apply another functional bootstrapping to compute $\mathsf{GSW}(1-b)\cdot\mathsf{LWE}(F_0(x))+\mathsf{GSW}(b)\cdot\mathsf{LWE}(F_1(x))$, which is a ciphertext of $F(m) = F_b(m)$. However, in order to get a GSW ciphertext of b, they have to run about $\log Q$ functional bootstrapping procedures, which is the main bottleneck of their scheme.

Recall that our fully functional bootstrapping focuses on functions on $[0, N)$. It only involves two bootstrapping procedures and one addition. Our fully functional bootstrapping could be easily applied to [18] and [31] to reduce parameters and improve efficiency.

A concurrent work by Kim et al. [30] proposes a new bootstrapping for CKKS without approximating the modular reduction function. They also apply the FHEW bootstrapping but with very different techniques from ours. Although they claim it is possible to set the ring dimension to 2^{14} or less, they do not report implementations of the algorithms at the time of this writing.

2 Preliminaries

2.1 Notations

All logarithms are base 2. We denote by $\langle \cdot, \cdot \rangle$ the inner product of two vectors. For a real number r, $\lfloor r \rceil, \lceil r \rceil, \lfloor r \rfloor$ denote the nearest round, upper round and floor round of r to integers, respectively. We use randomized rounding in our analysis and deterministic rounding in implementation as in [20]. We use $x \leftarrow D$ to denote the sampling of x according to distribution D. It denotes the sampling from the uniform distribution from D when D is a finite set.

For $q \in \mathbb{Z}$, we identify $\mathbb{Z}_q$ with the set $[-q/2, q/2) \cap \mathbb{Z}$ by default. We will point it out when $\mathbb{Z}_q$ lies in $[0, q)$. For $x \in \mathbb{Z}$, let $[x]_q \in \mathbb{Z}_q$ be such that $[x]_q \equiv x \pmod q$. Define the rings $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ and $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$, where N is a power of 2. We use bold $\mathbf{a} \in \mathbb{Z}_q^N$ for vectors, and lower case $a \in \mathcal{R}$ or $a \in \mathcal{R}_q$ for ring elements. Given a ring element $a \in \mathcal{R}$ or $a \in \mathcal{R}_q$, define the corresponding coefficient vector as $\mathbf{a} = (a_0, ..., a_{N-1})$, where $a_i \in \mathbb{Z}$ or $a_i \in \mathbb{Z}_q$ for $0 \leq i \leq N-1$ and $a = a_0 + a_1X + \cdots + a_{N-1}X^{N-1}$. Denote $\mathbf{a} = \mathsf{ToCoef}(a)$ and $a = \mathsf{ToRing}(\mathbf{a})$.

2.2 Fixed-Precision Numbers

In this paper, we use fixed-precision numbers to approximate fixed-point and floating-point numbers. In a nutshell, fixed-precision numbers always preserve some fixed precision even after different operations, say multiplication. Formally, we define the fixed-precision real numbers as follows.

Definition 1 (Fixed-Precision Numbers). *A set of fixed-precision numbers with p bits of precision is defined as*

$$\mathbb{FP}_p = \left\{ x \mid x = \pm\big(z + \sum_{i=1}^{p} b_i \cdot 2^{-i}\big); z \in \mathbb{Z}_{\geq 0}, b_i \in \{0,1\} \right\},$$

We also define operations on $\mathbb{FP}_p$. It is easy to verify that $x + y \in \mathbb{FP}_p$ for $x, y \in \mathbb{FP}_p$. It remains to define the multiplication operation. Before that, we first define the truncation function as $\mathsf{Trunc}_p(x) = \lfloor 2^p x \rfloor / 2^p$ for $x \in \mathbb{R}$. It preserves p-bit precision of the fractional part of x and drops the others. We define multiplication over $\mathbb{FP}_p$ as follows.

Definition 2. *For any $x, y \in \mathbb{FP}_p$, define the multiplication operation as $x \odot y = \mathsf{Trunc}_p(xy) \in \mathbb{FP}_p$.*

Note that $xy \in \mathbb{FP}_{2p}$, the truncation function drops the last p bits of the fractional part and moves the result back to $\mathbb{FP}_p$.

In our fully homomorphic encryption scheme, when dealing with fixed-point or floating-point arithmetic, we approximate it with fixed-precision arithmetic. Our FHE scheme supports homomorphic operations over $\mathbb{FP}_p$, the exact decryption procedure ensures the $\mathsf{IND\text{-}CPA^D}$ security.

2.3 Gaussian Distributions and (R)LWE Encryption

Let $\sigma > 0$, define the Gaussian function $\rho_\sigma(x) = \exp(-\pi x^2/\sigma^2)$ for $x \in \mathbb{R}$. Denoted by χ_σ the discrete Gaussian distribution on $\mathbb{Z}$ with variance σ^2 by sampling from continuous Gaussian probability distribution with density $\rho_\sigma(x)/\sigma$ and then rounding it to the nearest integer. Denoted by χ_σ^N the discrete Gaussian distribution on $\mathbb{Z}^N$ by sampling each element from χ_σ independently. We follow the heuristic approaches of error analysis in [19,24]. Hence, we will use 6σ as a high-probability bound on the size of elements sampled from χ_σ.

The Learning with Errors (LWE) problem is introduced by Regev [37]. It is extended to the ring version, which is called RLWE, by Lyubashevsky, Peikert and Regev [36]. Let $q > 0$ be an integer, $\mathbf{a} \leftarrow \mathbb{Z}_q^N$ and $\mathbf{s} \leftarrow \mathcal{D}^N$. For simplicity, we focus on $\mathcal{D}^N = \{0,1\}^N$. For a message $m \in \mathbb{Z}$, the (symmetric) LWE encryption of m under $\mathbf{s}$ is of the form $(\mathbf{a}, [-\langle \mathbf{a}, \mathbf{s}\rangle + m' + e]_q) \in \mathbb{Z}_q^{N+1}$, where $e \leftarrow \chi_\sigma$ and $m' = \mathsf{encode}(m)$. There always exists a $\mathsf{decode}(\cdot)$ function to recover m from $\mathsf{encode}(m) + e$. We assume the $\mathsf{encode}/\mathsf{decode}$ function be the identity function for simplicity. Let $\mathsf{LWE}_{\mathbf{s},q}^N(m)$ be the set of LWE ciphertexts of m. We say $\mathbf{ct} = (\mathbf{a}, b) \in \mathsf{LWE}_{\mathbf{s},q}^N(m)$ has error variance σ^2, if $[b + \langle \mathbf{a}, \mathbf{s}\rangle]_q - m$ has variance σ^2. Sometimes, we mix the use of $\mathsf{LWE}_{\mathbf{s},q}^N(m)$ and $\mathsf{LWE}_{\mathbf{s},q}^N(m+e)$ where e is the error term. If $\mathbf{ct} \in \mathsf{LWE}_{\mathbf{s},q}^N(m)$ with error term e, then $\mathbf{ct} \in \mathsf{LWE}_{\mathbf{s},q}^N(m+e)$ with error term 0.

Let $a \leftarrow \mathcal{R}_q$, $\mathbf{s} \leftarrow \mathcal{D}^N$ and $s = \mathsf{ToRing}(\mathbf{s})$. For a message $m \in \mathcal{R}$, the RLWE encryption of m under s is of the form $(a, -as + m' + e) \in \mathcal{R}_q \times \mathcal{R}_q$, where $e = \mathsf{ToRing}(\mathbf{e})$ and $\mathbf{e} \leftarrow \chi_\sigma^N$ and $m' = \mathsf{encode}(m)$. Similarly, we assume encode be the identity function. Let $\mathsf{RLWE}_{s,q}^N(m)$ be the set of RLWE ciphertexts of m. We say $\mathsf{ct} = (a, b) \in \mathsf{RLWE}_{s,q}^N(m)$ has error variance σ^2, if each coefficient of $(b + a \cdot s) - m \in \mathcal{R}_q$ has variance σ^2. In this paper, we only consider the "unpacking" version of RLWE encryption, which means $m = m_0 + 0 \cdot X + \cdots + 0 \cdot X^{N-1}$ for some $m_0 \in \mathbb{Z}$. Sometimes, we mix the use of $\mathsf{RLWE}_{s,q}^N(m)$ and $\mathsf{RLWE}_{s,q}^N(m+e)$ where e is the error term. If $\mathsf{ct} \in \mathsf{RLWE}_{s,q}^N(m)$ with error term e, then $\mathsf{ct} \in \mathsf{RLWE}_{s,q}^N(m+e)$ with error term 0.

Additionally, RGSW [25] ciphertexts are needed in our constructions. Let $\mathbf{G} \in \mathcal{R}_q^{2\ell \times 2}$ be a gadget matrix for some integer ℓ, the concrete definition is presented in Definition 3 in Appendix A. For a message $m \in \mathcal{R}$, sampling $\mathsf{ct}_0, ..., \mathsf{ct}_{2\ell-1} \in \mathsf{RLWE}_{s,q}^N(0)$, the RGSW ciphertext of m is of the form:

$$\begin{pmatrix} \mathsf{ct}_0 \\ \vdots \\ \mathsf{ct}_{2\ell-1} \end{pmatrix} + m \cdot \mathbf{G}.$$

Similar to LWE and RLWE encryptions, we let $\mathsf{RGSW}_{s,q}^N(m)$ be the set of RGSW ciphertexts. An RGSW ciphertext has error variance σ^2 if $\{\mathsf{ct}_i\}_{i=0}^{2\ell-1}$ have error variance σ^2.

2.4 Useful Algorithms

In this subsection, we list some useful algorithms which will be used in our constructions.

Extracting LWE from RLWE. Given a RLWE ciphertext, it is well-known to extract LWE ciphertexts. Since our RLWE ciphertexts only encrypt constant messages, the extract procedure is much simpler as described in Algorithm 1. We remark that the extract algorithm does not change the error variance of the ciphertext.

Algorithm 1 Extract (ct)

```
Input: $\mathsf{ct} = (a, b) = \mathsf{RLWE}_{s,q}^N(m)$.
Output: $\mathbf{ct} = \mathsf{LWE}_{\mathbf{s},q}^N(m_0)$, where $\mathbf{s} = \mathsf{ToCoef}(s)$.
1: Denote $a = a_0 + a_1X + \cdots a_{N-1}X^{N-1}$, $b = b_0 + b_1X + \cdots b_{N-1}X^{N-1}$
2: return $(a_0, -a_{N-1}, -a_{N-2}, \ldots, -a_1, b_0) \in \mathbb{Z}_q^{N+1}$
```

External Product. It allows to perform multiplication on a RGSW ciphertext and a RLWE ciphertext. The resulting ciphertext is a RLWE ciphertext encrypts the product of the messages. Specifically, adapted from [16] and [20], the external product between a RGSW ciphertext $\mathbf{C} \in \mathsf{RGSW}_{s,q}^N(m_1)$ and a RLWE ciphertext $\mathsf{ct} \in \mathsf{RLWE}_{s,q}^N(m_2)$ produces a RLWE ciphertext $\mathsf{ct}' \in \mathsf{RLWE}_{s,q}^N(m_1 \cdot m_2)$.

$$\mathbf{C} \boxdot \mathsf{ct} \to \mathsf{ct}' \in \mathsf{RLWE}_{s,q}^N(m_1 \cdot m_2)$$

The details and error analysis of external product are given in Lemma 1 in Appendix A.

Modulus Switching and Modular Reduction. Two approaches are given to change modulus of ciphertexts. Modulus switching is used to reduce the magnitude of ciphertext modulus and that of the noise. Modular reduction allows to compute the residue of the plaintext.

Algorithm 2 $\mathsf{ModS}_{q\to q'}(\mathbf{ct})$: Switching modulus between LWE ciphertexts

```
Input: $\mathbf{ct} = (\mathbf{a}, b) \in \mathsf{LWE}_{\mathbf{s},q}^N(m)$;
       $q'$ satisfies $q' < q$.
Output: $\mathbf{ct}' = (\mathbf{a}', b') \in \mathsf{LWE}_{\mathbf{s},q'}^N(m')$, where $m' = \lfloor (q'/q)m \rceil$.
1: Let $\mathbf{a}' = \lfloor (q'/q) \cdot \mathbf{a} \rceil$, $b' = \lfloor (q'/q) \cdot b \rceil$.
2: return $\mathbf{ct}' = (\mathbf{a}', b')$.
```

The correctness and error analysis are given in Lemma 2 in Appendix B. We note that the modulus switching procedure is crucial in our constructions to manage the errors. It is natural to extend to the ring case by performing the rounding operation on each coefficient of the ring elements. The error analysis is similar.

Modular reduction simply computes the residue of the ciphertext under a new and smaller modulus. The new ciphertext carries the residue of the original message.

Algorithm 3 $\mathsf{ModR}_{q\to q'}(\mathbf{ct})$: Modular reduction between LWE ciphertexts

```
Input: $\mathbf{ct} = (\mathbf{a}, b) = \mathsf{LWE}^N_{\mathbf{s},q}(m)$;
       $q'$ satisfies that $q'|q$.
Output: $\mathbf{ct}' = (\mathbf{a}', b') = \mathsf{LWE}^N_{s,q'}(m')$, where $m' = m \bmod q'$.
1: Let $\mathbf{a}' = \mathbf{a} \bmod q'$, $b' = b \bmod q'$.
2: return $\mathbf{ct}' = (\mathbf{a}', b')$.
```

The correctness of modular reduction is easy to verify since $q'|q$. This procedure will not change the error variance as long as $|m + e| < q'/2$, where e is the error term in $\mathbf{ct}$.

Key Switching and Dimension Switching. These two algorithms switch secret keys between two ciphertexts. To clarify the differences in this paper, key switching only focuses on the RLWE case and the dimension of the secret keys remains the same. Dimension switching is widely used in our fully functional bootstrapping procedure, it focuses on the LWE case and will change the dimension of the secret keys. Key switching is given in Algorithm 4, the correctness and error analysis are given in Lemma 3 in Appendix B.

Algorithm 4 $\mathsf{KeyS}_{s\to s'}(\mathsf{ct}, \mathsf{KSK})$: Switching keys between RLWE ciphertexts

```
Input: $\mathsf{ct} = (a, b) \in \mathsf{RLWE}^N_{s,q}(m)$;
       A key-switching key $\mathsf{KSK} \in \mathsf{RGSW}^N_{s',q}(s)$.
Output: $\mathsf{ct}' = (a', b') = \mathsf{RLWE}^N_{s',q}(m)$.
1: Let $(a', b') = (0, b) + \mathsf{KSK} \boxdot (0, a) \in \mathcal{R}_q \times \mathcal{R}_q$
2: return $\mathsf{ct}' = (a', b')$
```

For efficiency reasons, we apply the method proposed in [11] to switch dimensions for LWE ciphertexts while keeping the message unchanged. Given $n|N$ and $(\mathbf{a}, b) \in \mathsf{LWE}^N_{\mathbf{s},q}(m)$, define polynomials with degree n as

$$\widetilde{a}_i = a_{in} - a_{in+n-1}X - a_{in+n-2}X^2 - \cdots - a_{in+1}X^{n-1}, \text{ and}$$
$$\widetilde{s}_i = s_{in} + s_{in+1}X + \cdots + s_{in+n-2}X^{n-2} + s_{in+n-1}X^{n-1},$$

where $0 \le i \le N/n - 1$. View $\widetilde{a}_i$ and $\widetilde{s}_i$ as elements in $\mathbb{Z}_q[X]/(X^n + 1)$. Note that the constant term of $\sum_{i=0}^{N/n-1} \widetilde{a}_i \cdot \widetilde{s}_i$ is exactly $\langle \mathbf{a}, \mathbf{s} \rangle \in \mathbb{Z}_q$. The algorithm is given in Algorithm 5. The correctness and error analysis are given in Lemma 4 in Appendix B.

Algorithm 5 $\mathsf{DimS}_{N\to n}(\mathbf{ct}, \mathsf{KSK}^q_{s\to s'})$: Switching dimensions between LWE ciphertexts

```
Input: $\mathbf{ct} = (\mathbf{a}, b) = \mathsf{LWE}^N_{\mathbf{s},q}(m)$;
       A key-switching key $\mathsf{KSK}^q_{s\to s'} = \{\mathsf{KSK}^q_i \in \mathsf{RGSW}^n_{s',q}(\widetilde{s}_i)\}_{i=0}^{N/n-1}$;
Output: $\mathbf{ct}' = \mathsf{LWE}^n_{\mathsf{ToCoef}(s'),q}(m)$ under key $s' \in \mathbb{Z}_q[X]/(X^n + 1)$
1: Construct polynomials $\widetilde{a}_i$ from $\mathbf{a}$ as above, for $i \in \{0, \ldots, N/n - 1\}$
2: Let $\widetilde{\mathsf{ct}} = (0, b) + \sum_i \mathsf{KSK}^q_i \boxdot (0, \widetilde{a}_i)$,
3: return $\mathsf{Extract}(\widetilde{\mathsf{ct}})$
```

Evaluating Trace. Given $f(X) \in \mathcal{R}_q$, let $\varphi_i(x) : f(X) \mapsto f(X^i)$ for $i = 1, 3, 5, ..., 2N - 1$ be the Galois automorphisms. It is well-known that automorphisms could be homomorphically evaluated via key switching, which is listed as in Algorithm 6. The error analysis is exactly the same as the key switching procedure.

Algorithm 6 $\mathsf{HomAuto}(\mathsf{ct}, i, \mathsf{K})$: Evaluating an automorphism

```
Input: $\mathsf{ct} = (a, b) \in \mathsf{RLWE}^N_{s,q}(m)$;
        An integer $i \in \{1, 3, \dots, 2N-1\}$;
        An automorphism key $\mathsf{K} \in \mathsf{RGSW}^N_{s,q}(\varphi_i(s))$.
Output: $\mathsf{ct}' \in \mathsf{RLWE}^N_{s,q}(\varphi_i(m))$.
1: Compute $\varphi_i(a)$ and $\varphi_i(b)$;
2: return $\mathsf{ct}' \leftarrow \mathsf{KeyS}_{\varphi_i(s) \to s}((\varphi_i(a), \varphi_i(b)), \mathsf{K})$.
```

We will apply the method in [11] to transfer an LWE ciphertext into a RLWE ciphertext via evaluating the trace function homomorphically. For $a \in \mathcal{R}_q$, define the trace of a as the sum of all Galois transformations on it, i.e., $\mathsf{Trace}(a) = \sum_i \varphi_i(a)$, where $i \in \{1, 3, \cdots, 2N-1\}$. A key observation is that $\mathsf{Trace}(1) = N$ and $\mathsf{Trace}(X^i) = 0$ for $i \neq 0$. An efficient recursive homomorphic trace evaluation in [11] is described in Algorithm 7. We refer the details and correctness of this algorithm to [11].

Algorithm 7 $\mathsf{HomTrace}(\mathsf{ct}, \mathsf{AK})$: Evaluating the trace function

```
Input: $\mathsf{ct} \in \mathsf{RLWE}^N_{s,q}(m)$;
        Keys $\mathsf{AK} = \{\mathsf{K}_i \in \mathsf{RGSW}^N_{s,q}(\varphi_i(s))\}$, $i \in \{2^k + 1 | k = 1, \dots, \log N\}$.
Output: $\mathsf{ct}' \in \mathsf{RLWE}^N_{s,q}(\mathsf{Trace}(m))$.
1: $\mathsf{ct}' \leftarrow \mathsf{ct}$
2: for $i = N+1, \frac{N}{2}+1, \frac{N}{4}+1, \dots, 3$ do
3:     $\mathsf{ct}' \leftarrow \mathsf{ct}' + \mathsf{HomAuto}(\mathsf{ct}', i, \mathsf{K}_i)$
4: end for
5: return $\mathsf{ct}' \in \mathcal{R}_q \times \mathcal{R}_q$;
```

With the homomorphic trace algorithm, we could transfer an LWE into a RLWE with the following algorithm. The resulting RLWE ciphertext encrypts the message as the constant term.

Algorithm 8 $\mathsf{LWE\text{-}to\text{-}RLWE}(\mathbf{ct}, \mathsf{AK})$

```
Input: $(\mathbf{a}, b) \in \mathsf{LWE}^N_{s,q}(m)$;
        Keys $\mathsf{AK} = \{\mathsf{K}_i \in \mathsf{RGSW}^N_{s,q}(\varphi_i(s))\}$, $i \in \{2^k + 1 | k = 1, \dots, \log N\}$.
Output: $\mathsf{ct}' \in \mathsf{RLWE}^N_{s,q}(m)$.
1: Let $\hat{N} = N^{-1} \bmod q$
2: Let $a = (a_0, -a_{N-1}, ..., -a_1)$, and let $\mathsf{ct}^* = \hat{N} \cdot (a, b) \in \mathcal{R}_q \times \mathcal{R}_q$
3: return $\mathsf{ct}' = \mathsf{HomTrace}(\mathsf{ct}^*, \mathsf{AK})$.
```

The correctness is shown in [11], and we refer the readers to that paper. The error analysis of this algorithm is given in Lemma 5 in Appendix B.

3 Functional Bootstrapping

Functional bootstrapping is a generalized version of the original bootstrapping procedure, which enables to refresh the ciphertext while performing some pre-determined function on the plaintext, simultaneously. We follow the FHEW-like bootstrapping [20] and generalize it to handle multiple bits in a direct way. We note that the generalized functional bootstrapping procedure is still subject to the constraint that $f(x) = -f(x+N)$.

Consider an LWE ciphertext $\mathbf{ct} \in \mathsf{LWE}^{\widetilde{n}}_{\widetilde{\mathbf{s}},2N}(m)$. View it as a noiseless encryption of $\widetilde{m} = m + e$, where e is the error term of the original ciphertext. The functional bootstrapping procedure refreshes it into a ciphertext of $f(m+e)$ with error term independent of $\mathbf{ct}$. It seems not like the "bootstrapping" procedure in the literature, because the original error term is still in the output. We will explain that it is sufficient for our purpose to handle fixed-precision numbers.

Moreover, it is straightforward to remove the error term e. Consider the ciphertext is of the form $\mathbf{ct} \in \mathsf{LWE}^{\widetilde{n}}_{\widetilde{\mathbf{s}},2N}(\mathsf{encode}(m)+e)$, define an extended function $F = f \circ \mathsf{decode}$, where $\circ$ means the composition of the two functions. The functional bootstrapping on F will remove e, and the error term in the resulting ciphertext is independent of $\mathbf{ct}$.

Let $\mathbb{Z}_{2N}$ lie in $[0, 2N)$, we describe the functional bootstrapping procedure in Algorithm 9.

Algorithm 9 $\mathsf{FBS}_{(\widetilde{n},2N)\to(N,q)}(\mathbf{ct}, f, \mathsf{BK})$

```
Input: An LWE ciphertext $\mathbf{ct} = (\mathbf{a}, b) \in \mathsf{LWE}^{\widetilde{n}}_{\widetilde{\mathbf{s}},2N}(m)$ under key $\widetilde{\mathbf{s}} = (\widetilde{s}_0, \ldots, \widetilde{s}_{\widetilde{n}-1})$,
       where $[\langle \mathbf{a}, \widetilde{\mathbf{s}} \rangle + b]_{2N} = m + e =: \widetilde{m}$;
       A bootstrapping key $\mathsf{BK} = \{\mathsf{BK}_i \in \mathsf{RGSW}^N_{s,Q}(\widetilde{s}_i)\},\ 0 \le i \le \widetilde{n} - 1$;
       $f : \mathbb{Z}_{2N} \to \mathbb{Z}_q$ satisfying $f(x + N) = -f(x)$.
Output: $\mathbf{ct}' \in \mathsf{LWE}^N_{\mathbf{s},q}(f(\widetilde{m}))$, where $\mathbf{s} = \mathsf{ToCoef}(s)$.
1: Let $v(X) = \lfloor Qf(0)/q \rceil - \lfloor Qf(N-1)/q \rceil X - \cdots - \lfloor Qf(1)/q \rceil X^{N-1} \in \mathcal{R}_Q$
2: Let $\mathsf{acc} \leftarrow (0, X^b \cdot v(X)) \in \mathcal{R}_Q \times \mathcal{R}_Q$
3: for $i = 0$ to $\widetilde{n} - 1$ do
4:     $\mathsf{acc} = \mathsf{acc} + \mathsf{BK}_i \boxdot ((X^{a_i} - 1) \cdot \mathsf{acc})$.
5: end for
6: $\mathbf{ct}' = \mathsf{Extract}(\mathsf{acc})$                                   ▷ $\mathsf{LWE}^N_{\mathbf{s},Q}(\lfloor Qf(\widetilde{m})/q \rceil)$.
7: return $\mathbf{ct}' = \mathsf{ModS}_{Q\to q}(\mathbf{ct}')$
```

Denote $\mathsf{SimpleFBS}_{(\widetilde{n},2N)\to(N,q)}(\mathbf{ct}, f, \mathsf{BK})$ as a simplified version, which is exactly the same as the above $\mathsf{FBS}_{(\widetilde{n},2N)\to(N,q)}(\mathbf{ct}, f, \mathsf{BK})$ except running the modulus switching procedure in step 7, whose output $\mathbf{ct}'$ is still on modulus Q.

Remark 1. The functional bootstrapping procedure does not switch the secret key back to $\widetilde{\mathbf{s}}$. This is because in our main construction, the secret key is s, and $\widetilde{\mathbf{s}}$ is only viewed as an intermediate key for efficiency reasons.

Remark 2. For CKKS-like encryption schemes, they usually encounter the case when f is "somewhat continuous". For example, activate functions in machine learning models. The extra error e will not change the result too much, i.e., $f(\widetilde{m}) \approx f(m)$, which is acceptable in real applications.

Remark 3. For BFV-like encryption schemes, the input ciphertext is of the form in $\mathsf{LWE}^{\widetilde{n}}_{\widetilde{\mathbf{s}},2N}(\frac{2N}{t} \cdot m + e)$, where t is the plaintext module, and we assume $t|2N$ for simplicity. Algorithm 9 could handle BFV ciphertexts by extending $f(x)$ into $f(\lfloor \frac{t}{2N}(x) \rceil)$, and the the constraint of f is changed into $f(x) = -f(x + t/2)$.

Theorem 1. *Let* $\mathbf{ct} \in \mathsf{LWE}^{\widetilde{n}}_{\widetilde{\mathbf{s}},2N}(m)$ *with error term* e, $f : \mathbb{Z}_{2N} \to \mathbb{Z}_q$ *be any function satisfying* $f(x+N) = -f(x)$ *and* $Q \ge q$. *Then* $\mathbf{ct}'$ *output in Algorithm 9 is a ciphertext in* $\mathsf{LWE}^N_{\mathbf{s},q}(f(\widetilde{m}))$ *with error variance* σ^2, *where* $\widetilde{m} = m + e \in \mathbb{Z}_{2N}$.

Furthermore, Let the error variance in BK *be* σ^2_{bk}, *then*

$$\sigma^2 \le 2(q/Q)^2 \ell \widetilde{n} N g^2 \sigma^2_{\mathsf{bk}} + N/24 + 1/6,$$

where g *is defined as follows: If base power gadget is used in* BK, g *is the decomposition base. If RNS gadget is used then* $g = \max\{g_0, ..., g_{\ell-1}\}$.

Proof. In step 2, acc is a ciphertext in $\mathsf{RLWE}^N_{s,Q}(X^b \cdot v(X))$. In each iteration of step 4, the updated acc is a ciphertext in $\mathsf{RLWE}^N_{s,Q}(X^{b+\sum_{j=0}^{i} a_j \cdot s_j} \cdot v(X))$. Therefore, acc is a ciphertext in $\mathsf{RLWE}^N_{s,Q}(X^{b+\langle \mathbf{a}, \widetilde{\mathbf{s}} \rangle} \cdot v(X))$ after step 5. Since $[b + \langle \mathbf{a}, \widetilde{\mathbf{s}} \rangle]_{2N} = \widetilde{m}$, then we have $X^{b+\langle \mathbf{a}, \widetilde{\mathbf{s}} \rangle} \cdot v(X) = X^{\widetilde{m}} \cdot v(X)$. By the definition of $v(X)$

and $f(x+N) = -f(x)$, the constant term of $X^{\widetilde{m}} \cdot v(X)$ is $\lfloor Qf(\widetilde{m})/q \rceil$. Due to the modulus switching procedure, $\mathbf{ct}'$ is a ciphertext in $\mathsf{LWE}^N_{\mathbf{s},q}(f(\widetilde{m}))$.

Let σ_i^2 for $0 \le i \le \widetilde{n}-1$ be the error variances of acc in each iteration, according to Lemma 1 we have

$$\sigma_{i+1}^2 \le 2\ell N g^2 \sigma_{\mathsf{bk}}^2 + \widetilde{s}_i \cdot \sigma_i^2 \le 2\ell N g^2 \sigma_{\mathsf{bk}}^2 + \sigma_i^2.$$

Since $\sigma_0 = 0$, then the error variance in $\mathbf{ct}'$ is bounded by $2\ell \widetilde{n} N g^2 \sigma_{\mathsf{bk}}^2$. According to Lemma 2, we have $\sigma^2 \le 2(q/Q)^2 \ell \widetilde{n} N g^2 \sigma_{\mathsf{bk}}^2 + N/24 + 1/6$. $\square$

3.1 Fully Functional Bootstrapping

Since function f in Algorithm 9 has to satisfy the negacyclicity condition, it is not sufficient for our construction. This is because the functional bootstrapping procedure could not handle all the $\log(2N)$ bits and the most significant bit is out of control in this setting.

We propose the fully functional bootstrapping algorithm, which could deal with any function f, whose domain is $\mathbb{Z}_N$ (instead of $\mathbb{Z}_{2N}$), on a ciphertext in $\mathsf{LWE}^{\widetilde{n}}_{\widetilde{\mathbf{s}},N}(m)$ by reducing the modulus from $2N$ into N. We note that the homomorphic decryption process is still performed on the group with order $2N$.

In a nutshell, given a ciphertext in $(\mathbf{a}, b) \in \mathsf{LWE}^{\widetilde{n}}_{\widetilde{\mathbf{s}},N}(m)$, the fully functional bootstrapping procedure refresh it into a ciphertext in $\mathsf{LWE}^N_{\mathbf{s},q}(f(\widetilde{m}))$, for any $f : \mathbb{Z}_N \mapsto \mathbb{Z}_q$. The intuition of this algorithm is as follows. Viewing $(\mathbf{a}, b)$ as a ciphertext with modulus $2N$, then $[\langle \mathbf{a}, \widetilde{\mathbf{s}} \rangle + b]_{2N} = m + e + k \cdot N$, for $k \in \{0, 1\}$. We first apply the functional bootstrapping procedure in $(\mathbf{a}, b)$ to get a ciphertext of $-kN$, and add it to get a ciphertext of $m + e$ under modulus $2N$. Since $m + e$ is less than N, we could invoke the functional bootstrapping again on a function extended from f to obtain the required ciphertext. Details are given in Algorithm 10.

Algorithm 10 $\mathsf{FullyFBS}_{(\widetilde{n},N)\to(N,q)}(\mathbf{ct}, f, \mathsf{BK}, \widetilde{\mathsf{BK}}, \mathsf{KSK}^{\widetilde{Q}}_{s\to\widetilde{s}})$

```
Input: ct = $(\mathbf{a}, b) \in \mathsf{LWE}^{\widetilde{n}}_{\widetilde{\mathbf{s}},N}(m)$ under key $\widetilde{\mathbf{s}} = (\widetilde{s}_0, \ldots, \widetilde{s}_{\widetilde{n}-1})$;
        A bootstrapping key $\mathsf{BK} = \{\mathsf{BK}_i \in \mathsf{RGSW}^N_{s,Q}(\widetilde{s}_i)\}_{i=0}^{\widetilde{n}-1}$;
        A bootstrapping key $\widetilde{\mathsf{BK}} = \{\widetilde{\mathsf{BK}}_i \in \mathsf{RGSW}^N_{s,\widetilde{Q}}(\widetilde{s}_i)\}_{i=0}^{\widetilde{n}-1}$;
        A key-switching key $\mathsf{KSK}^{\widetilde{Q}}_{s\to\widetilde{s}}$;
        Any function $f : \mathbb{Z}_N \mapsto \mathbb{Z}_q$.
Output: $\mathbf{ct}' \in \mathsf{LWE}^N_{\mathbf{s},q}(f(\widetilde{m} + \widetilde{e}))$ for some $\widetilde{e}$, where $\mathbf{s} = \mathsf{ToCoef}(s)$ and $\widetilde{m} = [b + \langle \mathbf{a}, \widetilde{\mathbf{s}} \rangle]_N$.
 1: Lift $(\mathbf{a}, b)$ into $\mathbb{Z}_{2N}$ to get $[\langle \mathbf{a}, \widetilde{\mathbf{s}} \rangle + b]_{2N} = m + e + kN$ with $k \in \{0, 1\}$. Viewing $\mathbf{ct} \in \mathsf{LWE}^{\widetilde{n}}_{\widetilde{\mathbf{s}},2N}(m + e + kN)$.
 2: Perform $\mathsf{fix} = \mathsf{SimpleFBS}_{(\widetilde{n},2N)\to(N,2N)}(\mathbf{ct}, f_c, \widetilde{\mathsf{BK}})$ with function $f_c(x) = N/2$ for $x \in [0, N)$, $f_c(x) = -N/2$ for $x \in [N, 2N)$.Obtaining $\mathsf{fix} \in \mathsf{LWE}^N_{\mathbf{s},\widetilde{Q}}((-1)^k \cdot \widetilde{Q}/4)$.
 3: $\mathsf{fix} = \mathsf{ModS}_{\widetilde{Q}\to 2N}(\mathsf{DimS}_{N\to\widetilde{n}}(\mathsf{fix}, \mathsf{KSK}^{\widetilde{Q}}_{s\to\widetilde{s}}))$
 4: let $\mathsf{fix} = \mathsf{fix} + (\mathbf{0}, -N/2)$.                                  ▷ $\mathsf{fix} \in \mathsf{LWE}^{\widetilde{n}}_{\widetilde{\mathbf{s}},2N}(-kN)$.
 5: Let $\mathbf{ct}'' = [\mathbf{ct} + \mathsf{fix}]_{2N} \in \mathsf{LWE}^{\widetilde{n}}_{\widetilde{\mathbf{s}},2N}(\widetilde{m} + \widetilde{e})$                ▷ $\widetilde{e}$ is the error term in fix.
 6: Let $F(x) = f(x)$ for $0 \le x < N$, and $F(x) = -f(x - N)$ otherwise.
 7: Perform $\mathbf{ct}' = \mathsf{FBS}_{(\widetilde{n},2N)\to(N,q)}(\mathbf{ct}'', F, \mathsf{BK})$                 ▷ $\mathbf{ct}' \in \mathsf{LWE}^N_{\mathbf{s},q}(f(\widetilde{m} + \widetilde{e}))$.
```

Remark 4. We note that in step 2, the functional bootstrapping procedure does not switch the modulus from $\widetilde{Q}$ into $2N$, which will be done in step 3. But we still use $\left\lfloor \widetilde{Q} f_c(i)/(2N) \right\rceil$ in the lookup table in step 2. The reason of doing this is to control the errors, since modulus switching will reduce the magnitude of the errors.

Remark 5. We use a smaller modulus $\widetilde{Q}$ in $\widetilde{\mathsf{BK}}$ for efficiency reasons. $\widetilde{Q}$ could be much smaller than Q. We note that if RNS gadget matrix is used, $\widetilde{\mathsf{BK}}$ will not increase the key size, because it is only part of BK.

Remark 6. Algorithm 10 could be extended for BFV-like ciphertext similarly as in Remark 3. The plaintext modulus is reduced into $t/2$. Namely, the ciphertext is in $\mathsf{LWE}^{\widetilde{n}}_{\widetilde{\mathbf{s}},N}(\frac{N}{t/2}\cdot m+e)$ and function $f:\mathbb{Z}_{t/2}\mapsto\mathbb{Z}_q$ could be any functions. Step 2 remains unchanged, and $f(x)$ in step 6 is replaced by $f(\lfloor\frac{t}{2N}\cdot x\rceil)$. In this case, the error terms $e,\widetilde{e}$ in Algorithm 10 are removed.

Theorem 2. *Let* $\mathbf{ct}\in\mathsf{LWE}^{\widetilde{n}}_{\widetilde{\mathbf{s}},N}(m)$ *with error term* e*,* $f:\mathbb{Z}_N\to\mathbb{Z}_q$ *be any function and* $Q\geq\widetilde{Q}\geq q>2N$*. Then* $\mathbf{ct}'$ *output in Algorithm 10 is a ciphertext in* $\mathsf{LWE}^{N}_{\mathbf{s},q}(f([\widetilde{m}+\widetilde{e}]_N))$ *with error variance* σ'^2*, where* $\widetilde{m}+\widetilde{e}=m+e+\widetilde{e}\in\mathbb{Z}$ *and* $\widetilde{e}$ *is some error term with variance* $\widetilde{\sigma}^2$*.*

Furthermore, Suppose BK *contains* 2ℓ *RLWE ciphertexts with error variance* σ^2_{bk}*,* $\widetilde{\mathsf{BK}}$ *contains* $2\widetilde{\ell}$ *RLWE ciphertexts with error variance* $\widetilde{\sigma}^2_{\mathsf{bk}}$*,* $\mathsf{KSK}^{\widetilde{Q}}_{s\to s'}$ *contains* ℓ_{ksk} *RLWE ciphertexts with error variance* σ^2_{ksk}*. Then*

$$\widetilde{\sigma}^2\leq(2N/\widetilde{Q})^2(N\ell_{\mathsf{ksk}}g^2_{\mathsf{ksk}}\sigma^2_{\mathsf{ksk}}+2\widetilde{\ell}\widetilde{n}N\widetilde{g}^2\widetilde{\sigma}^2_{\mathsf{bk}})+\widetilde{n}/24+1/6,$$
$$\sigma'^2\leq2(q/Q)^2\ell\widetilde{n}Ng^2\sigma^2_{\mathsf{bk}}+N/24+1/6,$$

where g *is defined as follows: If base power gadget is used in* BK*,* g *is the decomposition base. If RNS gadget is used then* $g=\max\{g_0,...,g_{\ell-1}\}$*.* $\widetilde{g},g_{\mathsf{ksk}}$ *are similarly defined in* $\widetilde{\mathsf{BK}}$ *and* $\mathsf{KSK}^{\widetilde{Q}}_{s\to s'}$*.*

Proof. Since $m+e\in\mathbb{Z}_N$, then $m+e+N\in[N,2N)$. Then k identifies $m+e+kN$ belongs to $[0,N)$ or $[N,2N)$.Since f_c is negacyclic, due to Theorem 1 and $\mathsf{SimpleFBS}$, $\mathsf{fix}\in\mathsf{LWE}^{N}_{\mathbf{s},\widetilde{Q}}((-1)^k\cdot\widetilde{Q}/4)$ in step 2, where $\mathbf{s}=\mathsf{ToCoef}(s)$. After dimension switching and modulus switching, fix is transferred into a ciphertext in $\mathsf{LWE}^{\widetilde{n}}_{\widetilde{\mathbf{s}},2N}((-1)^k\cdot N/2)$. Note that $(-1)^k-1=-2k$ for $k\in\{0,1\}$, $\mathbf{ct}''$ is a ciphertext in $\mathsf{LWE}^{\widetilde{n}}_{\widetilde{\mathbf{s}},2N}(m+e)$. Due to Algorithm 9 and $m+e+\widetilde{e}\in\mathbb{Z}$, $\mathbf{ct}'$ is a ciphertext in $\mathsf{LWE}^{N}_{\mathbf{s},q}(f([\widetilde{m}+\widetilde{e}]_N))$, where $\widetilde{e}$ is the error term in $\mathbf{ct}''$.

According to the analysis of Theorem 1, the error variance of fix in step 2 is $\sigma^2_{\mathsf{fix}}\leq2\widetilde{\ell}\widetilde{n}N\widetilde{g}^2\widetilde{\sigma}^2_{\mathsf{bk}}$ because we avoid modulus switching in the functional bootstrapping procedure. After the dimension switching procedure, the error variance is less than $N\ell_{\mathsf{ksk}}g^2_{\mathsf{ksk}}\sigma^2_{\mathsf{ksk}}+2\widetilde{\ell}\widetilde{n}N\widetilde{g}^2\widetilde{\sigma}^2_{\mathsf{bk}}$ according to Lemma 4. The modulus switching procedure updates the error variance of fix into

$$\sigma^2_{\mathsf{fix}}\leq(2N/\widetilde{Q})^2(N\ell_{\mathsf{ksk}}g^2_{\mathsf{ksk}}\sigma^2_{\mathsf{ksk}}+2\widetilde{\ell}\widetilde{n}N\widetilde{g}^2\widetilde{\sigma}^2_{\mathsf{bk}})+\widetilde{n}/24+1/6$$

which is also the error variance of $\mathbf{ct}''$, i.e., the variance of $\widetilde{e}$.

According to Theorem 1, we have

$$\sigma'^2\leq2(q/Q)^2\ell\widetilde{n}Ng^2\sigma^2_{\mathsf{bk}}+N/24+1/6.$$

This completes the proof. □

Remark 7. Thanks to the modulus switching procedure, we will keep the error variance $\widetilde{\sigma}^2$ and σ'^2 very close to $\widetilde{n}$ and N, respectively.

4 Homomorphic Truncation for Fixed-Precision Numbers

In this section, we describe our algorithms to homomorphically evaluate multiplication of fixed-precision numbers. Namely, we show how to homomorphically evaluate the truncation procedure. The key ingredient is a bits extraction algorithm that heavily relies on the fully functional bootstrapping procedure.

The bits extraction procedure enables to homomorphically extract the bits blocks of the plaintext from LWE ciphertexts and reduces the errors simultaneously. Given positive integers d,δ,δ', the bits extraction procedure homomorphically extracts d bits at the position δ (counting from the least significant bit) of the message, and puts it into the position δ'. With this algorithm, one could easily drop the undesired bits and refresh the effective bits. Let $\mathbb{Z}_{2^d}$ lie in $[0, 2^d)$, assume q is a power of 2. The algorithm is described in Algorithm 11.

Algorithm 11 $\mathsf{BitsExtract}^{d}_{(\mathsf{s},N)}(\mathbf{ct}, \delta, \delta', \mathsf{BK}, \widetilde{\mathsf{BK}}, \mathsf{KSK}^{\widetilde{Q}}_{s\to\widetilde{s}}, \mathsf{KSK}^{q}_{s\to\widetilde{s}})$

```
Parameters: $d = \log N - \ell_e$, where $\ell_e$ is to be determined;
            $\widetilde{Q}$: an intermediate modulus with $Q > \widetilde{Q} > q$;
            $\widetilde{s}$: an intermediate secret key derived from $\mathbf{s}$;
            $\widetilde{n}$: an intermediate dimension.
Input: $\mathbf{ct} \in \mathsf{LWE}^{N}_{\mathsf{s},q}(m)$, where $2^{\delta} | m$, and error term $|e| < 2^{\delta-1}$;
       $\delta$: the position being extracted;
       $\delta'$: the target position;
       A bootstrapping key $\mathsf{BK} = \{\mathsf{BK}_i \in \mathsf{RGSW}^{N}_{s,Q}(\widetilde{s}_i)\}_{i=0}^{\widetilde{n}-1}$;
       A bootstrapping key $\widetilde{\mathsf{BK}} = \{\widetilde{\mathsf{BK}}_i \in \mathsf{RGSW}^{N}_{s,\widetilde{Q}}(\widetilde{s}_i)\}_{i=0}^{\widetilde{n}-1}$;
       Two key-switching keys $\mathsf{KSK}^{\widetilde{Q}}_{s\to\widetilde{s}}$, $\mathsf{KSK}^{q}_{s\to\widetilde{s}}$.
Output: $\mathbf{ct}' \in \mathsf{LWE}^{N}_{\mathsf{s},q}(m')$ , where $m' = 2^{\delta'}[m^*]_{2^d}$, and $m^* = m/2^{\delta}$;
 1: Let $r = \log q - \delta + \ell_e$.
 2: Let $\mathbf{ct}^* = \mathsf{DimS}_{N\to\widetilde{n}}(\mathbf{ct}, \mathsf{KSK}^{q}_{s\to\widetilde{s}})$
 3: Let $\widetilde{\mathbf{ct}} = \mathsf{ModR}_{2^r\to N}(\mathsf{ModS}_{q\to 2^r}(\mathsf{ct}^*))$      ▷ $\widetilde{\mathbf{ct}} = \mathsf{LWE}^{\widetilde{n}}_{\widetilde{\mathsf{s}},N}(2^{\ell_e}[m^*]_{2^d})$
 4: Perform $\mathsf{FullyFBS}_{(\widetilde{n},N)\to(N,q)}(\widetilde{\mathbf{ct}}, f, \mathsf{BK}, \widetilde{\mathsf{BK}}, \mathsf{KSK}^{\widetilde{Q}}_{s\to\widetilde{s}})$ with $f$ being
        $f : \mathbb{Z}_N \to \mathbb{Z}$
        $x \mapsto 2^{\delta'} \cdot \left[\lfloor x/2^{\ell_e} \rceil\right]_{2^d}$,
    obtaining $\mathbf{ct}' = \mathsf{LWE}^{N}_{\mathsf{s},q}(m')$;
 5: return $\mathbf{ct}'$.
```

Remark 8. As shown in our implementation, we could extract about $(\log N)/2$ bits once. $\ell_e \approx (\log \widetilde{n})/2$ bits are reserved to handle the errors to make sure the effective bits in the plaintext are not destroyed. Algorithm 11 could be extended for BFV-like ciphertexts, the analysis is similar to Remark 3 and 6.

Remark 9. Note that the constraint on m such that $2^{\delta} | m$ is dedicated for our exact truncation algorithm, which is required for $\mathsf{IND\text{-}CPA^D}$ security. This constraint could be removed if only approximate truncation is needed, as in the $\mathsf{IND\text{-}CPA}$ case.

Remark 10. In our setting, we also need to preserve the higher bits when move the bits blocks into lower positions. This is crucial in our truncation algorithm, when handling negative numbers. Thus, define an algorithm $\mathsf{SignBitsExtract}^{d}_{(\mathsf{s},N)}$ be the same as in Algorithm 11 except that f is defined as

$$f(x) = \begin{cases} 2^{\delta'} \cdot \left[\lfloor x/2^{\ell_e} \rceil\right]_{2^d} & , x \in [0, \frac{N}{2} - 2^{\ell_e - 1}) \cup [N - 2^{\ell_e - 1}, N) \\ q - 2^{\delta' + d} + 2^{\delta'} \cdot \left[\lfloor x/2^{\ell_e} \rceil\right]_{2^d} & , x \in [\frac{N}{2} - 2^{\ell_e - 1}, N - 2^{\ell_e - 1}). \end{cases}$$

If the most significant bit of the extracted d bits is 1 (i.e., negative), then the extraction algorithm will preserve the two's complement representation of this d-bits number.

Theorem 3. *Suppose* BK *contains* 2ℓ *RLWE ciphertexts with error variance* σ^2_{bk}, $\widetilde{\mathsf{BK}}$ *contains* $2\widetilde{\ell}$ *RLWE ciphertexts with error variance* $\widetilde{\sigma}^2_{\mathsf{bk}}$, $\mathsf{KSK}^{\widetilde{Q}}_{s\to s'}$ *contains* $2\ell_{\mathsf{ksk}}$ *RLWE ciphertexts with error variance* σ^2_{ksk}, $\mathsf{KSK}^{q}_{s\to s'}$ *contains* $2\ell'_{\mathsf{ksk}}$ *RLWE ciphertexts with error variance* σ'^2_{ksk}. *If base power gadget is used in* BK, *g is the decomposition base. If RNS gadget is used then* $g = \max\{g_0, ..., g_{\ell-1}\}$. $\widetilde{g}, g_{\mathsf{ksk}}, g'_{\mathsf{ksk}}$ *are similarly defined in* $\widetilde{\mathsf{BK}}$, $\mathsf{KSK}^{\widetilde{Q}}_{s\to s'}$ *and* $\mathsf{KSK}^{q}_{s\to s'}$.

Let $\mathbf{ct} \in \mathit{LWE}^{\widetilde{n}}_{\widetilde{\mathsf{s}},q}(m)$ *with* $2^{\delta}|m$ *and error variance* σ^2*,* q *is a power of* 2*. Let*

$$\tau = 2^{2(\ell_e-\delta)}(N\ell'_{\mathsf{ksk}}g'^2_{\mathsf{ksk}}\sigma'^2_{\mathsf{ksk}} + \sigma^2) + \left(\frac{2N}{\widetilde{Q}}\right)^2(N\ell_{\mathsf{ksk}}g^2_{\mathsf{ksk}}\sigma^2_{\mathsf{ksk}} + 2\widetilde{\ell}\widetilde{n}N\widetilde{g}^2\widetilde{\sigma}^2_{\mathsf{bk}}) + \frac{\widetilde{n}}{12} + \frac{1}{3}.$$

If $6\sqrt{\tau} + \frac{1}{2} < 2^{\ell_e-1}$*. Then* $\mathbf{ct}'$ *output in Algorithm 11 is a ciphertext in* $\mathit{LWE}^{N}_{\mathsf{s},q}(m')$ *with error variance* σ'^2 *and satisfies*

$$\sigma'^2 \leq 2(q/Q)^2\ell\widetilde{n}Ng^2\sigma^2_{\mathsf{bk}} + N/24 + 1/6.$$

Proof. According to Lemma 4, the error variance $(\sigma^*)^2$ of $\mathbf{ct}^*$ is bounded by $N\ell'_{\mathsf{ksk}}g'^2_{\mathsf{ksk}}\sigma'^2_{\mathsf{ksk}} + \sigma^2$. After running the modulus switching procedure in step 3, $\mathbf{ct}^*$ is transferred into a ciphertext encrypting $\left\lfloor \frac{2^r}{q}m \right\rceil$ and the error variance is bounded by $(2^r/q)^2 \cdot (\sigma^*)^2 + \widetilde{n}/24 + 1/6$. We have

$$\begin{aligned}\lfloor (2^r/q)m \rceil &= \frac{2^r}{q}\cdot m + e' \\ &= \frac{2^{\log q + \ell_e - \delta}}{q}\cdot m + e' = 2^{\ell_e}\cdot m^* + e'\end{aligned}$$

where e' is the rounding error. Let e'' be the error term after modulus switching, and this error term stays the same after modular reduction as long as $|e''| \leq N$, which is guaranteed by the chosen parameters. Denote $e^* = e' + e''$. Since $d = \log N - \ell_e$, then $(2^{\ell_e}\cdot m^*) \mod N = 2^{\ell_e}\cdot [m^*]_{2^d}$. Thus $\widetilde{\mathbf{ct}}$ is a ciphertext in $\mathsf{LWE}^{\widetilde{n}}_{\widetilde{\mathsf{s}},N}(2^{\ell_e}[m^*]_{2^d} + e^*)$ with error term 0.

Due to Algorithm 10 and Theorem 2, we know that $\mathbf{ct}'$ is a ciphertext in $\mathsf{LWE}^{N}_{\mathsf{s},q}(f([2^{\ell_e}[m^*]_{2^d} + e^* + \widetilde{e}]_N))$ with error variance $\sigma'^2 \leq 2(q/Q)^2\ell\widetilde{n}Ng^2\sigma^2_{\mathsf{bk}} + N/24 + 1/6$. Where $\widetilde{e}$ has variance

$$\widetilde{\sigma}^2 \leq (2N/\widetilde{Q})^2(N\ell_{\mathsf{ksk}}g^2_{\mathsf{ksk}}\sigma^2_{\mathsf{ksk}} + 2\widetilde{\ell}\widetilde{n}N\widetilde{g}^2\widetilde{\sigma}^2_{\mathsf{bk}}) + \widetilde{n}/24 + 1/6.$$

Therefore, we have $|e'' + \widetilde{e}| \leq 6\sqrt{\tau}$, which implies $|e^* + \widetilde{e}| \leq |e'' + \widetilde{e}| + 1/2 < 2^{\ell_e-1}$. By the definition of f, we have for some $k \in \mathbb{Z}$ such that

$$\begin{aligned} f\left(\left[2^{\ell_e}[m^*]_{2^d} + e^* + \widetilde{e}\right]_N\right) &= f\left(2^{\ell_e}[m^*]_{2^d} + e^* + \widetilde{e} + kN\right) \\ &= 2^{\delta'}\cdot\left[\left\lfloor (2^{\ell_e}[m^*]_{2^d} + e^* + \widetilde{e} + kN)/2^{\ell_e}\right\rceil\right]_{2^d} \\ &= 2^{\delta'}\cdot\left[[m^*]_{2^d} + k\cdot 2^d + \left\lfloor (e^* + \widetilde{e})/2^{\ell_e}\right\rceil\right]_{2^d} \\ &= 2^{\delta'}\cdot\left[[m^*]_{2^d} + k\cdot 2^d\right]_{2^d} = 2^{\delta'}\cdot[m^*]_{2^d}.\end{aligned}$$

This completes the proof. □

Corollary 1. *Assuming the conditions hold in Theorem 3. Then the extraction algorithm defined in Remark 10* $\mathsf{SignBitsExtract}^{d}_{(\mathsf{s},N)}(\mathbf{ct}, \delta, \delta', \mathsf{BK}, \widetilde{\mathsf{BK}}, \mathit{KSK}^{\widetilde{Q}}_{s\to\widetilde{s}}, \mathit{KSK}^{q}_{s\to\widetilde{s}})$ *outputs a ciphertext* $\mathbf{ct}'$ *encrypts* m' *such that* $m' = 2^{\delta'}[m^*]_{2^d}$*, if* $0 \leq 2^{\ell_e}[m^*]_{2^d} < N/2$*, and* $m' = q - 2^{\delta'+d} + 2^{\delta'}[m^*]_{2^d}$*, if* $N/2 \leq 2^{\ell_e}[m^*]_{2^d} < N$*.*

Proof. Similar to the analysis of Theorem 3, we have the ciphertext $\widetilde{\mathbf{ct}}$ encrypts plaintext $2^{\ell_e}[m^*]_{2^d}$ with error term $e^* + \widetilde{e}$, where $|e^* + \widetilde{e}| < 2^{\ell_e-1}$. The fully functional bootstrapping returns a ciphertext of $f([2^{\ell_e}[m^*]_{2^d} + e^* + \widetilde{e}]_N)$ with f defined in Remark 10.

If $0 \leq 2^{\ell_e}[m^*]_{2^d} < N/2$, then $0 \leq 2^{\ell_e}[m^*]_{2^d} \leq 2^{\ell_e}(2^{d-1} - 1) = N/2 - 2^{\ell_e}$. Since $|e^* + \widetilde{e}| < 2^{\ell_e-1}$, then

$$\left[2^{\ell_e}[m^*]_{2^d} + e^* + \widetilde{e}\right]_N \in [0, \frac{N}{2} - 2^{\ell_e-1}) \cup [N - 2^{\ell_e-1}, N),$$

$f([2^{\ell_e}[m^*]_{2^d} + e^* + \widetilde{e}]_N) = 2^{\delta'}[m^*]_{2^d}$ in this case.

If $N/2 \leq 2^{\ell_e}[m^*]_{2^d} < N$, then $N/2 \leq 2^{\ell_e}[m^*]_{2^d} \leq 2^{\ell_e}(2^d - 1) = N - 2^{\ell_e}$. Since $|e^* + \widetilde{e}| < 2^{\ell_e - 1}$, then

$$\left[2^{\ell_e}[m^*]_{2^d} + e^* + \widetilde{e}\right]_N \in [\frac{N}{2} - 2^{\ell_e - 1}, N - 2^{\ell_e - 1}),$$

$f([2^{\ell_e}[m^*]_{2^d} + e^* + \widetilde{e}]_N) = q - 2^{\delta' + d} + 2^{\delta'}[m^*]_{2^d}$ in this case. This completes the proof. □

Corollary 2. *Assuming the conditions hold in Theorem 3, except that the input message $m \nmid 2^\delta$ in Algorithm 11. Let $m = \bar{m} \cdot 2^\delta + \hat{m}$, where $\bar{m}$ and $\hat{m}$ are integers and $\hat{m} = [m]_{2^\delta}$. Then,* $\mathbf{ct}'$ *is a ciphertext of $2^{\delta'} \cdot [\bar{m} + \bar{e}]_{2^d}$ with error variance σ'^2 for some integer $\bar{e} \in \{0, 1\}$. Further, we have*

$$\sigma'^2 = 2(q/Q)^2 \ell \widetilde{n} N g^2 \sigma_{\mathsf{bk}}^2 + N/24 + 1/6.$$

In other words, if $\bar{m} \in \mathbb{Z}_{2^d}$, then the only possible case that $[\bar{m} + \bar{e}]_{2^d} \neq \bar{m} + \bar{e}$ is $\bar{m} = 2^d - 1$.

Proof. Similar to Theorem 3, the message m^* is now defined as $m^* = \bar{m} + \hat{m}/2^\delta$. Then the fully functional bootstrapping procedure in Algorithm 11 results in a ciphertext that encrypts

$$2^{\delta'}\left[\left\lfloor \bar{m} + \hat{m}/2^\delta + (e^* + \widetilde{e})/2^{\ell_e} \right\rceil\right]_{2^d} = 2^{\delta'}\left[\bar{m} + \left\lfloor \hat{m}/2^\delta + (e^* + \widetilde{e})/2^{\ell_e} \right\rceil\right]_{2^d}$$

Let $\bar{e} = \lfloor \hat{m}/2^\delta + (e^* + \widetilde{e})/2^{\ell_e} \rceil$. Note that $0 < \hat{m}/2^\delta < 1$ and $|e^* + \widetilde{e}|/2^{\ell_e} < 1/2$, then $-1/2 < \hat{m}/2^\delta + (e^* + \widetilde{e}/2^{\ell_e}) < 3/2$. We have $\bar{e} \in \{0, 1\}$. The analysis of σ' is the same as in Theorem 3. □

4.1 Homomorphic Truncation

With algorithm 11, we could perform the truncation by repeatedly extracting the last d bits of the current message m and subtract it from m. Recall that the truncation algorithm takes as input an integer $\Delta^2 \cdot m$, where $m \in \mathbb{FP}_{2p}$ and $\log \Delta > 2p$, and outputs $\Delta \cdot \mathsf{Trunc}_p(m)$. It drops the last p bits of m, and rescale the factor from Δ^2 into Δ.

In the algorithm, we assume $d \mid p$ and $d \mid (\log q - 2\log \Delta + p)$ for simplicity, the general case is essentially the same. The input of this algorithm takes as input a RLWE ciphertext that only encrypts the scaled fixed-precision number as the constant term. It is easy to extend the packing version of RLWE ciphertexts. Details are given in Algorithm 12.

Algorithm 12 $\mathsf{HomTrunc}_p(\mathsf{ct}, \mathsf{BK}, \widetilde{\mathsf{BK}}, \mathsf{KSK}^{\widetilde{Q}}_{s\to\widetilde{s}}, \mathsf{KSK}^{q}_{s\to\widetilde{s}}, \mathsf{AK})$

```
Parameters: $d = \log N - \ell_e$;
            $p$: The amount of bits being dropped, assuming $d|p$.
Input: $\mathsf{ct} \in \mathsf{RLWE}^N_{s,q}(\Delta^2 m)$, where $m \in \mathbb{FP}_{2p}$, and the error satisfies $|e| < \frac{\Delta^2}{2^{2p+1}}$;
       A bootstrapping key $\mathsf{BK} = \{\mathsf{BK}_i \in \mathsf{RGSW}^N_{s,Q}(\widetilde{s}_i)\}_{i=0}^{\widetilde{n}-1}$;
       A bootstrapping key $\widetilde{\mathsf{BK}} = \{\widetilde{\mathsf{BK}}_i \in \mathsf{RGSW}^N_{s,\widetilde{Q}}(\widetilde{s}_i)\}_{i=0}^{\widetilde{n}-1}$;
       Two key-switching keys $\mathsf{KSK}^{\widetilde{Q}}_{s\to\widetilde{s}}$, $\mathsf{KSK}^{q}_{s\to\widetilde{s}}$;
       An automorphism key $\mathsf{AK}$.
Output: $\mathsf{ct}' \in \mathsf{RLWE}^N_{s,q}(\Delta m')$, where $m' = \mathsf{Trunc}_p(m) \in \mathbb{FP}_p$
 1: Let $\mathsf{acc} = \mathsf{Extract}(\mathsf{ct})$                                  ▷ $\mathsf{acc} \in \mathsf{LWE}^N_{s,q}(\Delta^2 m)$
 2: Denote $\eta = 2\log\Delta - 2p$
 3: for $i = 0, 1, \ldots, \frac{p}{d} - 1$ do
 4:     Let $\mathsf{cbits} = \mathsf{BitsExtract}^d_{(s,N)}(\mathsf{acc}, \eta + i\cdot d, \eta + i\cdot d, \mathsf{BK}, \widetilde{\mathsf{BK}}, \mathsf{KSK}^{\widetilde{Q}}_{s\to\widetilde{s}}, \mathsf{KSK}^{q}_{s\to\widetilde{s}})$
 5:     $\mathsf{acc} \leftarrow \mathsf{acc} - \mathsf{cbits}$
 6: end for                                                                ▷ Drop the last $p$ bits.
 7: Let $\mathbf{ct}^* = (\mathbf{0}, 0) \in \mathsf{LWE}^N_{s,Q}(0)$, $\kappa = \eta + p$ and $\gamma = \log q - d$.
 8: for $i = 0, 1, ..., (\log q - \kappa)/d - 2$ do
 9:     Let $\mathsf{cbits} = \mathsf{BitsExtract}^d_{(s,N)}(\mathsf{acc}, \kappa + i\cdot d, \kappa - \log\Delta + i\cdot d, \mathsf{BK}, \widetilde{\mathsf{BK}}, \mathsf{KSK}^{\widetilde{Q}}_{s\to\widetilde{s}}, \mathsf{KSK}^{q}_{s\to\widetilde{s}})$
                ▷ Here in BitsExtract, SimpleFBS is applied, i.e. $\mathsf{cbits} \in \mathsf{LWE}^N_{s,Q}$.
10:     $\mathsf{acc} \leftarrow \mathsf{acc} - \mathsf{ModS}_{Q\to q}(\Delta\cdot\mathsf{cbits})$
11:     $\mathbf{ct}^* \leftarrow \mathbf{ct}^* + \mathsf{cbits}$
12: end for                                                                ▷ Reconstruct the higher bits.
13: $\mathsf{cbits} = \mathsf{SignBitsExtract}^d_{(s,N)}(\mathsf{acc}, \gamma, \gamma - \log\Delta, \mathsf{BK}, \widetilde{\mathsf{BK}}, \mathsf{KSK}^{\widetilde{Q}}_{s\to\widetilde{s}}, \mathsf{KSK}^{q}_{s\to\widetilde{s}})$.
                                            ▷ Preserve the two's complement representation.
14: $\mathbf{ct}^* \leftarrow \mathbf{ct}^* + \mathsf{cbits}$.
15: return $\mathsf{ct}' = \mathsf{ModS}_{Q\to q}(\mathsf{LWE\text{-}to\text{-}RLWE}(\mathbf{ct}^*, \mathsf{AK}))$
```

Remark 11. In step 9, only the SimpleFBS procedure is applied, and the modulus switching procedure is applied in step 10. This slight modification is to manage the errors and keep them small.

Remark 12. It is easy to modify Algorithm 12 into a simplified version $\mathsf{HomTrunc}'_p$. Its input ciphertext is $\mathsf{ct} \in \mathsf{RLWE}^N_{s,q}(\Delta m)$ with $m \in \mathbb{FP}_p$ and the plaintext in the output ciphertext is exactly the same. In other words, it only refreshes the errors. In this algorithm, it only performs step 7-14, the position $\kappa - \log\Delta + i\cdot d$ is replaced with $\kappa + i\cdot d$ and the $\mathsf{SignBitsExtract}^d_{(s,N)}$ algorithm is replaced with the original $\mathsf{BitsExtract}^d_{(s,N)}$ algorithm. This is because we do not need to drop bits blocks, and the position of the effective bits are not changed.

Remark 13. Algorithm 12 could be extended to the coefficient packing version of RLWE ciphertexts. The message is a ring element whose coefficients are in the form of $\Delta^2 m_i$ for $0 \le i \le N-1$ and $m_i \in \mathbb{FP}_{2p}$. It is straightforward to extract the coefficient from the RLWE ciphertext and get N LWE ciphertexts each encrypt $\Delta^2 m_i$. For each LWE ciphertext, perform step 2 to step 14 in parallel and then apply the repacking techniques in [11] to pack the N LWE ciphertexts into one RLWE ciphertext. In this case, the complexity will increase N times.

Theorem 4. *Suppose* BK *contains* 2ℓ *RLWE ciphertexts with error variance* σ^2_{bk}, $\widetilde{\mathsf{BK}}$ *contains* $2\widetilde{\ell}$ *RLWE ciphertexts with error variance* $\widetilde{\sigma}^2_{\mathsf{bk}}$, $\mathsf{KSK}^{\widetilde{Q}}_{s\to s'}$ *contains* $2\ell_{\mathsf{ksk}}$ *RLWE ciphertexts with error variance* σ^2_{ksk}, $\mathsf{KSK}^{q}_{s\to s'}$ *contains* $2\ell'_{\mathsf{ksk}}$ *RLWE ciphertexts with error variance* σ'^2_{ksk}. AK *contains* $2\ell_{\mathsf{ak}}$ *RLWE ciphertexts with error variance* σ^2_{ak}. *If base power gadget is used in* BK, *g is the decomposition base. If RNS gadget is* *used then* $g = \max\{g_0, ..., g_{\ell-1}\}$. $\widetilde{g}, g_{\mathsf{ksk}}, g'_{\mathsf{ksk}}$ *and* g_{ak} *are similarly defined in* $\widetilde{\mathsf{BK}}$, $\mathit{KSK}^{\widetilde{Q}}_{s\to s'}$, $\mathit{KSK}^{q}_{s\to s'}$ *and* AK.

If $\mathit{ct} \in \mathit{RLWE}^N_{s,q}(\Delta^2 m)$ *with error variance* σ^2, *where* $m \in \mathbb{FP}_{2p}$. *Let*

$$\tau_{1,k} = 2^{2(\ell_e - 2\log\Delta + 2p - i\cdot d)}\Big(N\ell'_{\mathsf{ksk}} g'^2_{\mathsf{ksk}} \sigma'^2_{\mathsf{ksk}} + k\cdot\big(2(\frac{q}{Q})^2 \ell\widetilde{n}Ng^2\sigma^2_{\mathsf{bk}} + \frac{N}{24} + \frac{1}{6}\big) + \sigma^2\Big) + \left(\frac{2N}{\widetilde{Q}}\right)^2 (N\ell_{\mathsf{ksk}} g^2_{\mathsf{ksk}}\sigma^2_{\mathsf{ksk}} + 2\widetilde{\ell}\widetilde{n}N\widetilde{g}^2\widetilde{\sigma}^2_{\mathsf{bk}}) + \frac{\widetilde{n}}{12} + \frac{1}{3},$$

where $0 \le k \le p/d - 1$. *Let*

$$\tau_{2,t} = 2^{2(\ell_e - \log\Delta - t\cdot d)}\Big(N\ell'_{\mathsf{ksk}} g'^2_{\mathsf{ksk}} \sigma'^2_{\mathsf{ksk}} + \frac{p}{d}\big(2(\frac{q}{Q})^2 \ell\widetilde{n}Ng^2\sigma^2_{\mathsf{bk}} + \frac{N}{24} + \frac{1}{6}\big) + t\cdot\big(2(\frac{q\Delta}{Q})^2 \ell\widetilde{n}Ng^2\sigma^2_{\mathsf{bk}} + \frac{N}{24} + \frac{1}{6}\big) + \sigma^2\Big) + \left(\frac{2N}{\widetilde{Q}}\right)^2 (N\ell_{\mathsf{ksk}} g^2_{\mathsf{ksk}}\sigma^2_{\mathsf{ksk}} + 2\widetilde{\ell}\widetilde{n}N\widetilde{g}^2\widetilde{\sigma}^2_{\mathsf{bk}}) + \frac{\widetilde{n}}{12} + \frac{1}{3},$$

where $0 \le t \le \frac{\log q - \kappa}{d} - 1$.

Suppose $6\sqrt{\max\{\{\tau_{1,k}\}_k, \{\tau_{2,t}\}_t\}} + 1/2 < 2^{\ell_e - 1}$, q *is a power of* 2, $d|p$, $d|(\log q - k)$ *and* $\log\Delta > p$. *Then* ct' *output in Algorithm* 12 *is a ciphertext in* $\mathit{RLWE}^N_{s,q}(\Delta m')$ *with error variance* σ'^2, *where* $m' = \mathsf{Trunc}_p(m)$. *Moreover,*

$$\sigma'^2 \le \left(\frac{q}{Q}\right)^2 \cdot \left(\ell_{\mathsf{ak}} N^2 g^2_{\mathsf{ak}} \sigma^2_{\mathsf{ak}} + \frac{\log q - \kappa}{d} \cdot 2\ell\widetilde{n}Ng^2\sigma^2_{\mathsf{bk}}\right) + \frac{N}{24} + \frac{1}{6}.$$

Proof. Since $m \in \mathbb{FP}_{2p}$ and $\log\Delta > p$, we know that $2^\eta|\Delta^2 m$. We consider two cases: $\Delta^2 m \ge 0$ and $\Delta^2 m < 0$. For the first case, split $\Delta^2 m \in \mathbb{Z}_{\ge 0}$ into bits blocks with each block d bits,

$$\Delta^2 m = \underbrace{2^\eta m_0 + 2^{\eta+d} m_1 + \cdots + 2^{\eta+p-d} m_{\frac{p}{d}-1}}_{p \text{ bits being dropped}} + \underbrace{2^{\eta+p} m_{\frac{p}{d}} + \cdots + 2^{\eta+p+\left(\frac{\log q - \eta - p}{d} - 1\right)\cdot d} \cdot m_{\frac{\log q - \eta}{d} - 1}}_{\Delta^2 m' \text{ being reserved}},$$

where $0 \le m_i < 2^d$. Consider the loop from step 3 to step 6. For $0 \le i \le p/d - 1$, let cbits_i be the output of the i-th iteration of step 4 and σ_i^2 be the error variance. Let acc_i be the output of the i-th iteration of step 5 and $\widetilde{\sigma}_i^2$ be the error variance. We claim cbits_i is a ciphertext of $2^{\eta+i\cdot d}\cdot m_i$ and $\sigma_i^2 \le 2(q/Q)^2\ell\widetilde{n}Ng^2\sigma^2_{\mathsf{bk}} + N/24 + 1/6$, acc_i is a ciphertext of $\Delta^2 m - \sum_{j=0}^{i} 2^{\eta+j\cdot d}\cdot m_j$ and $\widetilde{\sigma}_i^2 \le \sigma^2 + (i+1)\cdot\sigma_0^2$.

According to the chosen parameters and Theorem 3, cbits_0 is a ciphertext of $2^\eta \cdot m_0$ and $\sigma_0^2 \le 2(q/Q)^2\ell\widetilde{n}Ng^2\sigma^2_{\mathsf{bk}} + N/24 + 1/6$, acc_0 is a ciphertext of $\Delta^2 m - 2^\eta\cdot m_0$ and $\widetilde{\sigma}_0^2 \le \sigma^2 + \sigma_0^2$.

Suppose cbits_i and acc_i satisfies the above properties, we have $\widetilde{\sigma}_i^2 \le \sigma^2 + (i+1)\cdot\sigma_0^2$ for $0 < i < p/d - 1$. Then due to the chosen parameters and the constraint of $\{\tau_{1,k}\}_k$, it is easy to verify that the constraint of τ in Theorem 3 is satisfied. Invoke Algorithm 11 on acc_i will obtain cbits_{i+1}, which is a ciphertext of

$$2^{\eta+(i+1)\cdot d}\cdot\left[\frac{\Delta^2\cdot m - \sum_{j=0}^{i} 2^{\eta+j\cdot d}\cdot m_j}{2^{\eta+(i+1)\cdot d}}\right]_{2^d} = 2^{\eta+(i+1)\cdot d}\cdot m_{i+1},$$

and the error variance $\sigma_{i+1}^2 = \sigma_0^2 \le 2(q/Q)^2\ell\widetilde{n}Ng^2\sigma^2_{\mathsf{bk}} + N/24 + 1/6$ due to Theorem 3. Therefore, acc_{i+1} is a ciphertext of $\Delta^2 m - \sum_{j=0}^{i+1} 2^{\eta+j\cdot d}\cdot m_j$ and the error variance $\widetilde{\sigma}_{i+1}^2 \le \sigma^2 + (i+2)\sigma_0^2$.

After step 6, acc is a ciphertext of $\Delta^2 m'$ with error variance less than $\sigma^2 + (p/d)\cdot\sigma_0^2$. The following steps will refresh acc into a ciphertext $\mathbf{ct}^*$ of $\lfloor (Q/q)\Delta m' \rceil$ with error variance independent of σ. The analysis from step 8 to step 14 is similar as above, because the $\mathsf{SignBitsExtract}$ algorithm outputs the same as $\mathsf{BitsExtract}$. For $0 \le i \le (\log q - \kappa)/d - 1$, the i-th iteration of step 9 outputs a ciphertext cbits_i which encrypts $\left\lfloor (Q/q) 2^{\log\Delta - p + i\cdot d} \cdot m_{\frac{p}{d}+i} \right\rceil$, this is because the modulus switching procedure is not performed in the functional bootstrapping procedure. The i-th iteration of step 10 outputs a ciphertext of $\Delta^2 m' - \sum_{j=0}^{i} 2^{\eta+p+j\cdot d} m_{\frac{p}{d}+j} = \Delta^2 m' - \Delta(\sum_{j=0}^{i} 2^{\log\Delta - p + j\cdot d} m_{\frac{p}{d}+j})$. Therefore, the final $\mathbf{ct}^*$ is a ciphertext of $\lfloor (Q/q)\Delta m' \rceil$. The LWE-to-RLWE algorithm and modulus switching procedure convert $\mathbf{ct}^*$ into an RLWE ciphertext that encrypts $\Delta m'$.

For the case that $\Delta^2 m < 0$, the message is in the form $q + \Delta^2 m > 0$. Since $2^\eta | q$, then $2^\eta | (q + \Delta^2 m)$. In this case, the most significant bit of the highest d-bits in step 13 is 1. Due to Corollary 1, the message contains an additional term $q - 2^{\gamma - \log\Delta + d}$. Combining with the analysis of the first case, the resulting message is of the form

$$\begin{aligned} q - 2^{\gamma-\log\Delta+d} + \Delta\Big(\mathsf{Trunc}_p\Big(\frac{q+\Delta^2 m}{\Delta^2}\Big)\Big) &= q - \frac{q}{\Delta} + \Delta \cdot \left\lfloor 2^p \Big(\frac{q+\Delta^2 m}{\Delta^2}\Big) \right\rfloor / 2^p \\ &= q + \Delta \left\lfloor 2^p \cdot m \right\rfloor / 2^p = q + \Delta \cdot \mathsf{Trunc}_p(m), \end{aligned}$$

which is desired.

Considering the error variance, due to Theorem 3 without using modulus switching, we know that the error variance of the final $\mathbf{ct}^*$ satisfies $(\sigma^*)^2 \le (\log q - \kappa)/d \cdot 2\ell\tilde{n}Ng^2\sigma_{\mathsf{bk}}^2$ which is independent of σ. By Lemma 5 and 2, the LWE-to-RLWE and modulus switching procedures ensure that ct' is a ciphertext of $\Delta m'$ with error variance

$$\sigma'^2 \le \left(\frac{q}{Q}\right)^2 \cdot \left(\ell_{\mathsf{ak}} N^2 g_{\mathsf{ak}}^2 \sigma_{\mathsf{ak}}^2 + \frac{\log q - \kappa}{d} \cdot 2\ell\tilde{n}Ng^2\sigma_{\mathsf{bk}}^2\right) + \frac{N}{24} + \frac{1}{6}.$$

It remains to prove that the parameters are satisfied the constraints in Theorem 3 when invoking the bits extraction procedure in step 9. After step 6, acc is a ciphertext of $\Delta^2 m'$ with error variance less than $\sigma^2 + (p/d)\cdot\sigma_0^2$. In the i-th iteration, the error variance of acc is less than

$$\sigma^2 + (p/d)\sigma_0^2 + i \cdot \Big(2\big(q\Delta/Q\big)^2 \ell\tilde{n}Ng^2\sigma_{\mathsf{bk}}^2 + N/24 + 1/6\Big).$$

By the choice of $\{\tau_{2,t}\}_t$ for $0 \le t \le (\log q - \kappa)/d - 1$, the constraint in Theorem 3 is satisfied. This completes the proof. $\square$

Approximate Homomorphic Truncation. The homomorphic truncation algorithm is dedicated for fixed-precision numbers to achieve IND-$\mathsf{CPA}^{\mathsf{D}}$ security. When approximate encryption for fixed-point or floating-point numbers is considered, we could modify the homomorphic truncation algorithm into an approximate version. The resulting fully homomorphic encryption scheme is only IND-CPA secure.

In the exact homomorphic truncation algorithm, p is much smaller than $\log\Delta$. This is because we have to reserve enough space for the errors that will not destroy the effective bits. In the worst case, the bit length of the error produced in the homomorphic operations could not exceed $2\log\Delta - 2p$. In the approximate homomorphic truncation algorithm, p could be very close to $\log\Delta$. The bits extraction algorithm begins at the position $2\log\Delta - p$ instead of $2\log\Delta - 2p$, this is where the approximate value occurs. Details are given in Algorithm 13.

Algorithm 13 $\mathsf{ApproxTrunc}_p(\mathsf{ct}, \mathsf{BK}, \widetilde{\mathsf{BK}}, \mathsf{KSK}^{\widetilde{Q}}_{s\to\widetilde{s}}, \mathsf{KSK}^{q}_{s\to\widetilde{s}}, \mathsf{AK})$

```
Parameters: $d = \log N - \ell_e$;
            $p$: A parameter related to precision, assuming $d|p$.
Input: $\mathsf{ct} \in \mathsf{RLWE}^N_{s,q}(\lfloor \Delta^2 m \rceil)$, where $m \in \mathbb{R}$;
       A bootstrapping key $\mathsf{BK} = \{\mathsf{BK}_i \in \mathsf{RGSW}^N_{s,Q}(\widetilde{s}_i)\}_{i=0}^{\widetilde{n}-1}$;
       A bootstrapping key $\widetilde{\mathsf{BK}} = \{\widetilde{\mathsf{BK}}_i \in \mathsf{RGSW}^N_{s,\widetilde{Q}}(\widetilde{s}_i)\}_{i=0}^{\widetilde{n}-1}$;
       Two key-switching keys $\mathsf{KSK}^{\widetilde{Q}}_{s\to\widetilde{s}}$, $\mathsf{KSK}^{q}_{s\to\widetilde{s}}$;
       An automorphism key $\mathsf{AK}$.
Output: $\mathsf{ct}' \in \mathsf{RLWE}^N_{s,q}(\Delta m')$, where $\Delta m' \in \mathbb{Z}$ and $m' \approx m$.
 1: Let $\mathsf{acc} = \mathsf{Extract}(\mathsf{ct})$                ▷ $\mathsf{acc} \in \mathsf{LWE}^N_{\mathrm{s},q}(\lfloor \Delta^2 m \rceil)$
 2: Denote $\eta = 2\log\Delta - 2p$ and $\kappa = \eta + p$, $\gamma = \log q - d$.
 3: Let $\mathbf{ct}^* = (\mathbf{0}, 0) \in \mathsf{LWE}^N_{\mathrm{s},Q}(0)$.
 4: for $i = 0, 1, ..., (\log q - \kappa)/d - 2$ do
 5:     Let $\mathsf{cbits} = \mathsf{BitsExtract}^d_{(\mathrm{s},N)}(\mathsf{acc}, \kappa + i\cdot d, \kappa - \log\Delta + i \cdot d, \mathsf{BK}, \widetilde{\mathsf{BK}}, \mathsf{KSK}^{\widetilde{Q}}_{s\to\widetilde{s}}, \mathsf{KSK}^{q}_{s\to\widetilde{s}})$
            ▷ Here in $\mathsf{BitsExtract}$, $\mathsf{SimpleFBS}$ is applied, i.e. $\mathsf{cbits} \in \mathsf{LWE}^N_{\mathrm{s},Q}$.
 6:     $\mathsf{acc} \leftarrow \mathsf{acc} - \mathsf{ModS}_{Q\to q}(\Delta \cdot \mathsf{cbits})$
 7:     $\mathbf{ct}^* \leftarrow \mathbf{ct}^* + \mathsf{cbits}$
 8: end for                                              ▷ Reconstruct the higher bits.
 9: $\mathsf{cbits} = \mathsf{SignBitsExtract}^d_{(\mathrm{s},N)}(\mathsf{acc}, \gamma, \gamma - \log\Delta, \mathsf{BK}, \widetilde{\mathsf{BK}}, \mathsf{KSK}^{\widetilde{Q}}_{s\to\widetilde{s}}, \mathsf{KSK}^{q}_{s\to\widetilde{s}})$.
            ▷ Preserve the two's complement representation.
10: $\mathbf{ct}^* \leftarrow \mathbf{ct}^* + \mathsf{cbits}$.
11: return $\mathsf{ct}' = \mathsf{ModS}_{Q\to q}(\mathsf{LWE\text{-}to\text{-}RLWE}(\mathbf{ct}^*, \mathsf{AK}))$
```

Remark 14. Algorithm 13 could be extended to a simplified version $\mathsf{ApproxTrunc}'_p$. Its input ciphertext is $\mathsf{ct} \in \mathsf{RLWE}^N_{s,q}(\lfloor \Delta m \rceil)$ with $m \in \mathbb{R}$ and the output ciphertext is of the form $\mathsf{ct}' \in \mathsf{RLWE}^N_{s,q}(\Delta m')$. In this algorithm, the position $\kappa - \log\Delta + i \cdot d$ is replaced with $\kappa + i \cdot d$ and the $\mathsf{SignBitsExtract}^d_{(\mathrm{s},N)}$ algorithm is replaced with the original $\mathsf{BitsExtract}^d_{(\mathrm{s},N)}$ algorithm.

Theorem 5. *Let all the conditions holds as in Theorem 4, then ct' is a ciphertext of $\Delta m'$ with error variance σ'^2. Moreover, we have $|\Delta m' - \Delta m| \le \frac{\Delta}{2^p} + \frac{1}{\Delta}$, and*

$$\sigma'^2 \le \left(\frac{q}{Q}\right)^2 \cdot \left(\ell_{\mathsf{ak}} N^2 g^2_{\mathsf{ak}} \sigma^2_{\mathsf{ak}} + \frac{\log q - \kappa}{d} \cdot 2\ell\widetilde{n} N g^2 \sigma^2_{\mathsf{bk}}\right) + \frac{N}{24} + \frac{1}{6}.$$

Proof. Similar to the proof of Theorem 4, we could write $\lfloor \Delta^2 m \rceil$ in the form

$$\lfloor \Delta^2 m \rceil = m^* + \underbrace{2^{\eta+p} m_{\frac{p}{d}} + \cdots + 2^{\eta+p+\left(\frac{\log q - \eta - p}{d} - 1\right)\cdot d} \cdot m_{\frac{\log q - \eta}{d} - 1}}_{\Delta^2 m' \text{ being reserved}},$$

where $0 \le m^* < 2^{\eta+p}$, $0 \le m_{\frac{p}{d}+i\cdot d} < 2^d$ for $0 \le i < (\log q - \eta - p)/d$.

Denote $\mathcal{M}$ as the event that

$$\exists\ i \in \left[0, \frac{\log q - \eta - p}{d}\right),\ \text{s.t.}\ \forall\ 0 \le j \le i,\ m_{\frac{p}{d}+j\cdot d} = 2^d - 1.$$

If $\mathcal{M}$ does not happen, i.e., there is no consecutive $m_{\frac{p}{d}+i\cdot d}$'s that equal to $2^d - 1$ starting from the first block, then according to Corollary 2 and Theorem 4, we have the output ciphertext encrypts $\Delta m'$. Thus $|\lfloor \Delta^2 m \rceil - \Delta^2 m'| = m^* < 2^{\eta+p}$, which implies $|\Delta^2 m - \Delta^2 m'| < 2^{\eta+p} + |\epsilon| < 2^{\eta+p} + 1/2$, where ϵ is the rounding error. We get that $|\Delta m - \Delta m'| < \frac{\Delta}{2^p} + \frac{1}{2\Delta}$.

If $\mathcal{M}$ happens for some i^*, we have $m_{\frac{p}{d}+j\cdot d} = 2^d - 1$ for all $0 \le j \le i^*$. Due to Corollary 2, we know that the BitsExtract algorithm will obtain ciphertexts of 0 when extracting $m_{\frac{p}{d}+j\cdot d}$ for $0 \le j \le i^*$. The extracted ciphertext of the block $m_{\frac{p}{d}+(i^*+1)\cdot d}$ is of the form $(m_{\frac{p}{d}+(i^*+1)\cdot d}+1)$. Since the (i^*+1)-th block does not overflow, then all the remaining message blocks will be extracted exactly. This is because subtracting $(m_{\frac{p}{d}+(i^*+1)\cdot d}+1)$ will reserve enough 0's (in fact $d-1$ 0's) for the next extraction. Thus we have

$$\begin{aligned}\left|\lfloor\Delta^2 m\rceil - \Delta^2 m'\right| &= \left|m^* + \sum_{j=0}^{i^*}(2^{\eta+p+j\cdot d})(2^d-1) - 2^{\eta+p+(i^*+1)\cdot d}\right|\\ &= \left|m^* + 2^{\eta+p}(2^d-1)\sum_{j=0}^{i^*}2^{j\cdot d} - 2^{\eta+p+(i^*+1)\cdot d}\right|\\ &= \left|m^* + 2^{\eta+p}\cdot 2^{(i^*+1)\cdot d} - 2^{\eta+p} - 2^{\eta+p+(i^*+1)\cdot d}\right|\\ &= \left|m^* - 2^{\eta+p}\right|\end{aligned}$$

Since $0 \le m^* < 2^{\eta+p}$, thus we have $|\Delta m - \Delta m'| < \frac{\Delta}{2^p} + \frac{1}{2\Delta}$. The analysis of σ' is exactly the same as in Theorem 4. This completes the proof. $\square$

Remark 15. Note that $p \approx \log \Delta$ in this case, and the resulting message m' is very close to m. Therefore, the bootstrapping error dominates the approximation error in this procedure. This is the main reason our IND-CPA scheme preserves high precision while keeping the dimension small.

Remark 16. Note that in the approximate truncation algorithm, the modulus q is not required to be a power of 2 since approximate errors are allowed. One could choose q be a product of distinct prime factors and close to a power of 2.

5 The TOTA Algorithms

We are ready to describe our FHE scheme for fixed-precision numbers and for fixed-point numbers. Since our approximate version is very similar, we describe them in one algorithm and point out the differences.

The fully homomorphic encryption TOTA consists of the following five algorithms: (TOTA.KeyGen, TOTA.Enc, TOTA.Dec, TOTA.Add, TOTA.Mult). The plaintext space is defined in $\mathbb{FP}_p$ or $\mathbb{R}$.

- TOTA.KeyGen(1^λ)
  - It takes as input the security parameter λ, chooses integers $N, q, Q, \widetilde{Q}, \widetilde{n}, \Delta$, where N ,$\widetilde{n}$, q and Δ are powers of 2. Let χ_σ be the error distribution for encryption and $\sigma > 0$ be the standard variance .
  - It samples a secret key $\mathbf{s} \leftarrow \{0,1\}^N$, $a \leftarrow \mathcal{R}_q$ and $e \leftarrow \chi_\sigma$. Let $s = \mathsf{ToRing}(\mathbf{s})$, set the secret key $\mathsf{SK} = s$ and the public key as
$$\mathsf{PK} = (a, -as+e) \in \mathcal{R}_q^2.$$
    It generates a key switching key $\mathsf{RLK} \in \mathsf{RGSW}^N_{s,q}(s^2)$, which will be used to relinearize the ciphertext. It also generates an automorphism key $\mathsf{AK} = \{\mathsf{K}_i \in \mathsf{RGSW}^N_{s,Q}(\varphi_i(s))\}$, $i \in \{2^k+1 | k = 1, \dots, \log N\}$.
  - It samples an intermediate key $\widetilde{\mathbf{s}} = \{\widetilde{s}_0, \dots, \widetilde{s}_{\widetilde{n}-1}\} \leftarrow \{0,1\}^{\widetilde{n}}$ and let $\widetilde{s} = \mathsf{ToRing}(\widetilde{\mathbf{s}})$, generates bootstrapping keys $\mathsf{BK} = \{\mathsf{BK}_i \in \mathsf{RGSW}^N_{s,Q}(\widetilde{s}_i)\}_{i=0}^{\widetilde{n}-1}$, $\widetilde{\mathsf{BK}} = \{\widetilde{\mathsf{BK}}_i \in \mathsf{RGSW}^N_{s,\widetilde{Q}}(\widetilde{s}_i)\}_{i=0}^{\widetilde{n}-1}$, and two switching keys $\mathsf{KSK}^{\widetilde{Q}}_{s\to\widetilde{s}}$ and $\mathsf{KSK}^{q}_{s\to\widetilde{s}}$ as defined in Algorithm 5.

  Let $\mathsf{EK} = (\mathsf{BK}, \widetilde{\mathsf{BK}}, \mathsf{KSK}^{\widetilde{Q}}_{s\to\widetilde{s}}, \mathsf{KSK}^{q}_{s\to\widetilde{s}}, \mathsf{AK}, \mathsf{RLK})$, it outputs $(\mathsf{SK}, \mathsf{PK}, \mathsf{EK})$.

– $\mathsf{TOTA.Enc}(\mathsf{PK},m)$. Given $m \in \mathbb{FP}_p$ (or $m \in \mathbb{R}$), set $m' = \Delta \cdot m$ (or $m' = \lfloor \Delta m \rceil$). Sample two $\mathcal{R}$ elements $e_0, e_1 \leftarrow \chi_\sigma$ and $\mathbf{u} \leftarrow \{0,1\}^N$, set $u = \mathsf{ToRing}(\mathbf{u})$. Output the ciphertext as $\mathsf{PK} \cdot u + (e_0, e_1 + m')$.

– $\mathsf{TOTA.Dec}(\mathsf{SK}, \mathsf{ct})$. Given a ciphertext $\mathsf{ct} = (a, b)$, if the plaintext is defined in $\mathbb{FP}_p$, it outputs

$$m = \frac{1}{2^p} \left\lfloor \frac{2^p}{\Delta} (b + a \cdot s)_0 \right\rceil \in \mathbb{FP}_p.$$

If the plaintext is defined in $\mathbb{R}$, it outputs $m = \lfloor (b + as)_0 / \Delta \rceil \in \mathbb{R}$. Where $(\cdot)_0$ denotes the constant term of the ring element.

– $\mathsf{TOTA.Add}(\mathsf{EK}, \mathsf{ct}_1, \mathsf{ct}_2)$. Given two ciphertexts $\mathsf{ct}_1 = (a_1, b_1)$,$\mathsf{ct}_2 = (a_2, b_2)$. It outputs $\mathsf{ct}' = (a_1 + a_2, b_1 + b_2)$. If the plaintext is defined in $\mathbb{FP}_p$, it outputs the refreshed ciphertext $\mathsf{ct} = \mathsf{HomTrunc}'_p(\mathsf{ct}', \mathsf{BK}, \widetilde{\mathsf{BK}}, \mathsf{KSK}^{\widetilde{Q}}_{s \to \widetilde{s}}, \mathsf{KSK}^{q}_{s \to \widetilde{s}}, \mathsf{AK})$. If the plaintext is defined in $\mathbb{R}$, it outputs the refreshed ciphertext $\mathsf{ct} = \mathsf{ApproxTrunc}'_p(\mathsf{ct}', \mathsf{BK}, \widetilde{\mathsf{BK}}, \mathsf{KSK}^{\widetilde{Q}}_{s \to \widetilde{s}}, \mathsf{KSK}^{q}_{s \to \widetilde{s}}, \mathsf{AK})$.

– $\mathsf{TOTA.Mult}(\mathsf{EK}, \mathsf{ct}_1, \mathsf{ct}_2)$. Given two ciphertexts $\mathsf{ct}_1 = (a_1, b_1)$,$\mathsf{ct}_2 = (a_2, b_2)$. It computes $(f, g, h) = (a_1 a_2, a_1 b_2 + a_2 b_1, b_1 b_2)$. Relinearize the ciphertext by setting $\mathsf{ct}' = \mathsf{KeyS}_{s^2 \to s}((0, f), \mathsf{RLK}) + (g, h)$.If the plaintext is defined in $\mathbb{FP}_p$, it outputs $\mathsf{ct} = \mathsf{HomTrunc}_p(\mathsf{ct}', \mathsf{BK}, \widetilde{\mathsf{BK}}, \mathsf{KSK}^{\widetilde{Q}}_{s \to \widetilde{s}}, \mathsf{KSK}^{q}_{s \to \widetilde{s}}, \mathsf{AK})$. If the plaintext is in $\mathbb{R}$, it outputs $\mathsf{ct} = \mathsf{ApproxTrunc}_p(\mathsf{ct}', \mathsf{BK}, \widetilde{\mathsf{BK}}, \mathsf{KSK}^{\widetilde{Q}}_{s \to \widetilde{s}}, \mathsf{KSK}^{q}_{s \to \widetilde{s}}, \mathsf{AK})$.

Theorem 6. *Suppose* BK *contains* 2ℓ *RLWE ciphertexts with error variance* σ^2_{bk}, $\widetilde{\mathsf{BK}}$ *contains* $2\widetilde{\ell}$ *RLWE ciphertexts with error variance* $\widetilde{\sigma}^2_{\mathsf{bk}}$, $\mathsf{KSK}^{\widetilde{Q}}_{s \to s'}$ *contains* $2\ell_{\mathsf{ksk}}$ *RLWE ciphertexts with error variance* σ^2_{ksk}, $\mathsf{KSK}^{q}_{s \to s'}$ *contains* $2\ell'_{\mathsf{ksk}}$ *RLWE ciphertexts with error variance* σ'^2_{ksk}, RLK *contains* ℓ_{rlk} *RLWE ciphertexts with error variance* σ^2_{rlk}, AK *contains* $2\ell_{\mathsf{ak}}$ *RLWE ciphertexts with error variance* σ^2_{ak}. *If base power gadget is used in* BK, *g is the decomposition base. If RNS gadget is used then* $g = \max\{g_0, ..., g_{\ell-1}\}$. $\widetilde{g}, g_{\mathsf{ksk}}, g'_{\mathsf{ksk}}$, g_{ak} *and* g_{rlk} *are similarly defined in* $\widetilde{\mathsf{BK}}$, $\mathsf{KSK}^{\widetilde{Q}}_{s \to s'}$, $\mathsf{KSK}^{q}_{s \to s'}$, AK *and* RLK.

Let σ_0^2 *be the error variance of ciphertexts in TOTA.* $\tau_{1,k}$ *and* $\tau_{2,t}$ *are defined as in Theorem 4 except that* σ^2 *is replaced by* $3\Delta^2\beta^2\sigma_0^2 + \ell_{\mathsf{rlk}} N g^2_{\mathsf{rlk}} \sigma^2_{\mathsf{rlk}}$. *Suppose*

$$\sigma_0^2 < \max\left\{ \frac{\Delta^2}{144 \cdot 2^{2p}}, \quad \frac{\Delta^2}{108\beta^2 2^{4p+2}} - \frac{\ell_{\mathsf{rlk}} N g^2_{\mathsf{rlk}} \sigma^2_{\mathsf{rlk}}}{3\Delta^2\beta^2} \right\}, \ (N+1) \cdot \sigma^2 \leq \sigma_0^2,$$

$$\left(\frac{q}{Q}\right)^2 \cdot \left(\ell_{\mathsf{ak}} N^2 g^2_{\mathsf{ak}} \sigma^2_{\mathsf{ak}} + \frac{\log q - \kappa}{d} \cdot 2\ell \widetilde{n} N g^2 \sigma^2_{\mathsf{bk}} \right) + \frac{N}{24} + \frac{1}{6} \leq \sigma_0^2.$$

Then TOTA defined in $\mathbb{FP}_p$ *decrypts the ciphertext correctly and is fully homomorphic.*

Proof. Let $\mathsf{ct} = (a, b) \in \mathsf{RLWE}^N_{s,q}(\Delta m)$ be a fresh ciphertext. It is easy to verify that $b + as = e_0 s + eu + e_1 + \Delta m$. Denote $e^* = e_0 s + eu + e_1$, we have the variance of e^* is less than $(N+1)\sigma^2 \leq \sigma_0^2$. When invoking the decryption algorithm on ct, we have $\mathsf{TOTA.Dec}(\mathsf{SK}, \mathsf{ct}) = \left\lfloor \frac{2^p}{\Delta}(b + a \cdot s)_0 \right\rceil / 2^p = \left\lfloor 2^p m + \frac{2^p}{\Delta} e^* \right\rceil / 2^p = 2^p m + \left\lfloor \frac{2^p}{\Delta} e^* \right\rceil$. Since $\sigma_0^2 \leq \Delta^2/(144 \cdot 2^{2p})$, then $|e^*| \leq 6\sigma_0$ and $\left| \frac{2^p}{\Delta} e \right| < 1/2$. $\mathsf{TOTA.Dec}(\mathsf{SK}, \mathsf{ct}) = m \in \mathbb{FP}_p$.

Because addition is similar and simple, we only consider multiplication here. Let $\mathsf{ct}_1 \in \mathsf{RLWE}^N_{s,q}(\Delta m_1)$, $\mathsf{ct}_2 \in \mathsf{RLWE}^N_{s,q}(\Delta m_2)$ be fresh ciphertexts with error term e_1, e_2, respectively. Denote the multiplication ciphertext $\mathsf{ct}_{\mathsf{mult}} = \mathsf{TOTA.Mult}(\mathsf{EK}, \mathsf{ct}_1, \mathsf{ct}_2)$. Before relinearizing the ciphertext, the error term of the multiplication is of the form $\Delta m_1 e_2 + \Delta m_2 e_1 + e_1 e_2$, and the last term is much smaller. Heuristically, the variance is less than $3\Delta^2\beta^2\sigma_0^2$. Due to Lemma 3, the error variance $\sigma^2_{\mathsf{relin}}$ of the relinearized product ciphertext ct' satisfies $\sigma^2_{\mathsf{relin}} \leq 3\Delta^2\beta^2\sigma_0^2 + \ell_{\mathsf{rlk}} N g^2_{\mathsf{rlk}} \sigma^2_{\mathsf{rlk}}$. In order to invoke $\mathsf{HomTrunc}_p$ and decrypt the exact message, the error term in ct' will not distort the first $2p$ bits of the fraction part. By the setting of σ_0, it ensures that $6\sigma_{\mathsf{relin}} \leq \Delta^2/2^{2p+1}$.

By the setting of $\tau_{1,k}$, $\tau_{2,t}$ and Theorem 4, we have $\mathsf{ct}_{\mathsf{mult}}$ is a ciphertext of $\mathsf{Trunc}_p(m_1 m_2)$ with error variance σ'^2. It satisfies

$$\sigma'^2 \leq \left(\frac{q}{Q}\right)^2 \cdot \left(\ell_{\mathsf{ak}} N^2 g_{\mathsf{ak}}^2 \sigma_{\mathsf{ak}}^2 + \frac{\log q - \kappa}{d} \cdot 2\ell \widetilde{n} N g^2 \sigma_{\mathsf{bk}}^2\right) + \frac{N}{24} + \frac{1}{6} \leq \sigma_0.$$

This completes the proof. □

Remark 17. The analysis for the approximate encryption is similar, we omit the detailed description.

Corollary 3. *Suppose all the conditions hold as in Theorem 6,* EK *and* PK *in* TOTA *are indistinguishable from uniformly random, then* TOTA *performed on* $\mathbb{FP}_p$ *and* $\mathbb{R}$ *achieves* $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ *and* $\mathsf{IND\text{-}CPA}$ *security, respectively .*

Proof. It is straightforward from [35]. □

6 Parameters and Implementation

We choose concrete parameters for TOTA and implement it with C++ from scratch. All the parameters are chosen to provide at least 127-bit security according to [1]. For simplicity, we set the magnitude of all the messages to be less than 1. The parameters are given in Table 1. Set I and II are chosen for $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ security and Set III,IV and V are chosen for $\mathsf{IND\text{-}CPA}$ security[5].

Note that we could choose the dimension N to 2^{13} and 2^{12} for $p = 14$ and $p = 12$ in the $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ case, respectively. In the $\mathsf{IND\text{-}CPA}$ case, we could use $N = 2^{13}$ and preserve about 39-bit precision. This is the first construction of this kind.

Param.	N	q	σ	$\widetilde{n}$	Δ	BK	$\widetilde{\mathsf{BK}}$	$\mathsf{KSK}^{\widetilde{Q}}_{s\to\widetilde{s}}$	$\mathsf{KSK}^{q}_{s\to\widetilde{s}}$	AK	p/precision	d	ℓ_e
Set I	2^{13}	2^{109}	3.2	2^{11}	2^{51}	$Q : \{54, 55, 50, 50\}$ $g : \mathsf{rns}$ $\sigma_{\mathsf{bk}} : 3.2$	$\widetilde{Q} : \{54, 55\}$ $\widetilde{g} : \mathsf{rns}$ $\widetilde{\sigma}_{\mathsf{bk}} : 3.2$	$g_{\mathsf{ksk}} : 2^7$ $\sigma_{\mathsf{ksk}} : 3.2$	$g'_{\mathsf{ksk}} : 2^7$ $\sigma'_{\mathsf{ksk}} : 3.2$	$g_{\mathsf{ak}} : \mathsf{rns}$ $\sigma_{\mathsf{ak}} : 3.2$	14	7	6
Set II	2^{12}	2^{54}	3.2	2^{11}	2^{24}	$Q : \{54, 54\}$ $g : 2^{20}$ $\sigma_{\mathsf{bk}} : 3.2$	$\widetilde{Q} : \{54, 54\}$ $\widetilde{g} : 2^{20}$ $\widetilde{\sigma}_{\mathsf{bk}} : 3.2$	$g_{\mathsf{ksk}} : 2$ $\sigma_{\mathsf{ksk}} : 3.2$	$g'_{\mathsf{ksk}} : 2$ $\sigma'_{\mathsf{ksk}} : 3.2$	$g_{\mathsf{ak}} : 2^{20}$ $\sigma_{\mathsf{ak}} : 3.2$	12	6	6
Set III	2^{11}	$\{33\}$	3.2	2^{10}	2^{16}	$Q : \{54\}$ $g_{\mathsf{bk}} : 2^4$ $\sigma_{\mathsf{bk}} : 3.2$	$\widetilde{Q} : \{54\}$ $\widetilde{g}_{\mathsf{bk}} : 2^4$ $\widetilde{\sigma}_{\mathsf{bk}} : 3.2$	$g_{\mathsf{ksk}} : 2$ $\sigma_{\mathsf{ksk}} : 3.2 \times 2^6$	$g'_{\mathsf{ksk}} : 2$ $\sigma'_{\mathsf{ksk}} : 3.2 \times 2^6$	$g_{\mathsf{ak}} : 2$ $\sigma_{\mathsf{ak}} : 3.2$	≈ 11	6	5
Set IV	2^{12}	$\{54\}$	3.2	2^{11}	2^{26}	$Q : \{54, 54\}$ $g_{\mathsf{bk}} : 2^{20}$ $\sigma_{\mathsf{bk}} : 3.2$	$\widetilde{Q} : \{54, 54\}$ $\widetilde{g}_{\mathsf{bk}} : 2^{20}$ $\widetilde{\sigma}_{\mathsf{bk}} : 3.2$	$g_{\mathsf{ksk}} : 2^{14}$ $\sigma_{\mathsf{ksk}} : 3.2$	$g'_{\mathsf{ksk}} : 2^{14}$ $\sigma'_{\mathsf{ksk}} : 3.2$	$g_{\mathsf{ak}} : 2^{20}$ $\sigma_{\mathsf{ak}} : 3.2$	≈ 21	6	6
Set V	2^{13}	$\{54, 54\}$	3.2	2^{11}	2^{53}	$Q : \{54, 54, 50, 50\}$ $g_{\mathsf{bk}} : \mathsf{rns}$ $\sigma_{\mathsf{bk}} : 3.2$	$\widetilde{Q} : \{54, 54\}$ $\widetilde{g}_{\mathsf{bk}} : \mathsf{rns}$ $\widetilde{\sigma}_{\mathsf{bk}} : 3.2$	$g_{\mathsf{ksk}} : 2^{14}$ $\sigma_{\mathsf{ksk}} : 3.2$	$g'_{\mathsf{ksk}} : 2^{14}$ $\sigma'_{\mathsf{ksk}} : 3.2$	$g_{\mathsf{ak}} : \mathsf{rns}$ $\sigma_{\mathsf{ak}} : 3.2$	≈ 39	7	6

Table 1. Parameters for TOTA with different security. The numbers of the modulus mean the bit length of each prime factor. rns means that we choose the largest generator in the RNS representation. Set I and II set $p = 14$ and $p = 12$, respectively. Set III, IV and V achieves precision about $11, 21$ and 39, respectively. Note that in the $\mathsf{IND\text{-}CPA}$ case, p in the algorithm is not the final precision.

We also test all the parameter sets and the concrete performance is given in Table 2. The experiments are run on a machine with Intel(R) Core(TM) i7-10510U 2304 MHz CPU and 16GB RAM.

As shown in Table 2, we could truncate the plaintext homomorphically in about $77s/50s$ for Set I/II and in about $8s/23s/68s$ for Set III/IV/V. The key size of our construction is relatively large. One could use the methods in [30] to compact the key size and perform the functional bootstrapping procedure on the fly. We leave it to the future work.

[5] Note that the theoretical analysis of the parameters is relatively loose, the concrete parameters chosen are tighter. We also note that q is not required to be a power of 2 in the $\mathsf{IND\text{-}CPA}$ case.

Security	Parameter Set	Truncation time (s)	Public Key Size (GB)
$\mathsf{IND\text{-}CPA}^{\mathsf{D}}$	Set I	77.6	≈ 8.006
	Set II	50.3	≈ 3.007
$\mathsf{IND\text{-}CPA}$	Set III	8.52	≈ 0.878
	Set IV	23.5	≈ 3.001
	Set V	68.3	≈ 8.004

Table 2. Concrete Performance of Different Parameter Sets

7 Conclusions

We propose TOTA, a fully homomorphic encryption for fixed-precision arithmetic with $\mathsf{IND\text{-}CPA}^{\mathsf{D}}$ security. Along with that, we also extend it into an FHE scheme for approximate encryption with $\mathsf{IND\text{-}CPA}$ security. The resulting approximate FHE preserves high precision with only small ring dimension. The underlying core technique is a new fully functional bootstrapping methods that enables to perform arbitrary function with in domain $[0, N)$. We believe our fully functional bootstrapping is of independent interest.

References

1. Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *J. Math. Cryptol.*, pages 169–203, 2015.
2. Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *Advances in Cryptology - CRYPTO*, pages 297–314. Springer, 2014.
3. Jean-Philippe Bossuat, Christian Mouchet, Juan Ramón Troncoso-Pastoriza, and Jean-Pierre Hubaux. Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. In *Advances in Cryptology - EUROCRYPT*, pages 587–617. Springer, 2021.
4. Christina Boura, Nicolas Gama, Mariya Georgieva, and Dimitar Jetchev. Simulating homomorphic evaluation of deep learning predictions. In *Cyber Security Cryptography and Machine Learning - Third International Symposium, CSCML 2019*, pages 212–230. Springer, 2019.
5. Florian Bourse, Michele Minelli, Matthias Minihold, and Pascal Paillier. Fast homomorphic evaluation of deep discretized neural networks. In *Advances in Cryptology - CRYPTO*, pages 483–512. Springer, 2018.
6. Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Advances in Cryptology - CRYPTO*, pages 868–886. Springer, 2012.
7. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Innovations in Theoretical Computer Science 2012*, pages 309–325. ACM, 2012.
8. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS*, pages 97–106. IEEE Computer Society, 2011.
9. Sergiu Carpov, Malika Izabachène, and Victor Mollimard. New techniques for multi-value input homomorphic evaluation and applications. In *Topics in Cryptology - CT-RSA*, pages 106–126. Springer, 2019.
10. Hao Chen, Ilaria Chillotti, and Yongsoo Song. Improved bootstrapping for approximate homomorphic encryption. In *Advances in Cryptology - EUROCRYPT*, pages 34–54. Springer, 2019.
11. Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient homomorphic conversion between (ring) LWE ciphertexts. In *Applied Cryptography and Network Security - 19th International Conference, ACNS*, pages 460–479. Springer, 2021.
12. Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In *Advances in Cryptology - EUROCRYPT*, pages 360–384. Springer, 2018.
13. Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology - ASIACRYPT*, pages 409–437. Springer, 2017.

14. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *Advances in Cryptology - ASIACRYPT 2016*, pages 3–33, 2016.
15. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In *Advances in Cryptology - ASIACRYPT*, pages 377–408. Springer, 2017.
16. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: fast fully homomorphic encryption over the torus. *J. Cryptol.*, 33(1):34–91, 2020.
17. Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. Cryptology ePrint Archive, Report 2021/091, 2021.
18. Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. Improved programmable bootstrapping with larger precision and efficient arithmetic circuits for tfhe. Cryptology ePrint Archive, Report 2021/729, 2021.
19. Ana Costache and Nigel P. Smart. Which ring based somewhat homomorphic encryption scheme is best? In *Topics in Cryptology - CT-RSA*, pages 325–340. Springer, 2016.
20. Léo Ducas and Daniele Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology - EUROCRYPT*, pages 617–640. Springer, 2015.
21. Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012.
22. Nicolas Gama, Malika Izabachène, Phong Q. Nguyen, and Xiang Xie. Structural lattice reduction: Generalized worst-case to average-case reductions and homomorphic cryptosystems. In *Advances in Cryptology - EUROCRYPT*, pages 528–558. Springer, 2016.
23. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC*, pages 169–178. ACM, 2009.
24. Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In *Advances in Cryptology - CRYPTO*, pages 850–867. Springer, 2012.
25. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology - CRYPTO*, pages 75–92. Springer, 2013.
26. Antonio Guimarães, Edson Borin, and Diego F. Aranha. Revisiting the functional bootstrap in TFHE. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(2):229–253, 2021.
27. Kyoohyung Han and Dohyeong Ki. Better bootstrapping for approximate homomorphic encryption. In Stanislaw Jarecki, editor, *Topics in Cryptology - CT-RSA*, pages 364–390. Springer, 2020.
28. Malika Izabachène, Renaud Sirdey, and Martin Zuber. Practical fully homomorphic encryption for fully masked neural networks. In *Cryptology and Network Security - 18th International Conference, CANS*, pages 24–36. Springer, 2019.
29. Wen jie Lu, Zhicong Huang, Cheng Hong, Yiping Ma, and Hunter Qu. Pegasus: Bridging polynomial and non-polynomial evaluations in homomorphic encryption. Cryptology ePrint Archive, Report 2020/1606, 2020.
30. Andrey Kim, Maxim Deryabin, Jieun Eom, Rakyong Choi, Yongwoo Lee, Whan Ghang, and Donghoon Yoo. General bootstrapping approach for rlwe-based homomorphic encryption. Cryptology ePrint Archive, Report 2021/691, 2021.
31. Kamil Kluczniak and Leonard Schild. Fdfb: Full domain functional bootstrapping towards practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2021/1135, 2021.
32. Joon-Woo Lee, Eunsang Lee, Yongwoo Lee, Young-Sik Kim, and Jong-Seon No. High-precision bootstrapping of RNS-CKKS homomorphic encryption using optimal minimax polynomial approximation and inverse sine function. In *Advances in Cryptology - EUROCRYPT*, pages 618–647. Springer, 2021.
33. Yongwoo Lee, Joon-Woo Lee, Young-Sik Kim, and Jong-Seon No. Near-optimal polynomial for modulus reduction using l2-norm for approximate homomorphic encryption. *IEEE Access*, 8:144321–144330, 2020.
34. Yongwoo Lee, Joonwoo Lee, Young-Sik Kim, HyungChul Kang, and Jong-Seon No. High-precision and low-complexity approximate homomorphic encryption by error variance minimization. Cryptology ePrint Archive, Report 2020/1549, 2020.
35. Baiyu Li and Daniele Micciancio. On the security of homomorphic encryption on approximate numbers. In *Advances in Cryptology - EUROCRYPT*, pages 648–677. Springer, 2021.

36. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Advances in Cryptology - EUROCRYPT*, pages 1–23. Springer, 2010.
37. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 84–93. ACM, 2005.
38. Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

Appendix

A Details on RGSW Ciphertext and External Product

The gadget matrix consists of multiple gadget vectors. Let $\mathbf{g} = \{g_0, ..., g_\ell\}$ be a gadget vector. Two types of gadget vectors are considered. If base power is used, i.e., $\mathbf{g} = \{1, g, g^2, ...g^{\ell-1}\}$, we call it a base power gadget, denoted as $\mathbf{g}_{\mathsf{power}}$. For some $Q = q_0 \times q_1 \times \cdots \times q_{\ell-1}$ with distinct primes, we call $\mathbf{g}$ a RNS gadget, which is denoted as $\mathbf{g}_{\mathsf{rns}}$, if $g_i = [(Q/q_i)^{-1}]_{q_i}$ for $0 \le i \le \ell - 1$.

Definition 3 (Gadget Matrix). *Let $\mathbf{g} = \{g_0, g_1, \ldots, g_{l-1}\} \subset \mathbb{Z}$ be the gadget vector, then the gadget matrix $\mathbf{G}$ corresponding to it is the following matrix in $\mathcal{R}^{2l\times 2}$:*

$$\begin{pmatrix} g_0 & 0 \\ \vdots & \vdots \\ g_{l-1} & 0 \\ 0 & g_0 \\ \vdots & \vdots \\ 0 & g_{l-1} \end{pmatrix}$$

Also we define the decomposition procedure corresponding to the gadget basis.

Definition 4. *Let $\mathbf{g}$ be a gadget vector, $f \in \mathcal{R}$, then define*

$$\mathtt{Decomp}_{\mathbf{g}}(f) = (f_0, \ldots, f_{l-1}) \in \mathcal{R}^l,$$

such that $\sum g_i f_i = f$. More specifically, if $\mathbf{g} = \mathbf{g}_{\mathsf{power}}$, then $(f_0, ..., f_{\ell-1})$ is the power decomposition on base g, i.e., $\sum_{i=0}^{\ell-1} f_i \cdot g^i = f$. If $\mathbf{g} = \mathbf{g}_{\mathsf{rns}}$, then $(f_0, ..., f_{\ell-1})$ is the RNS decomposition, i.e., $f_i = f \mod q_i$.

For $(a, b) \in \mathcal{R} \times \mathcal{R}$, define

$$\mathtt{Decomp}_{\mathbf{g}}((a, b)) = (\mathtt{Decomp}_{\mathbf{g}}(a),\ \mathtt{Decomp}_{\mathbf{g}}(b)) \in \mathcal{R}^{2\ell}$$

It is easy to verify that

$$\mathtt{Decomp}_{\mathbf{g}}((a, b)) \cdot \mathbf{G} = (a, b)$$

Definition 5 (External Product). *Let $\mathbf{C} \in \mathsf{RGSW}^N_{s,q}(m_1)$, and $\mathbf{b} \in \mathsf{RLWE}^N_{s,q}(m_2)$, then the external product on them is*

$$\mathbf{C} \boxdot \mathbf{b} = \mathtt{Decomp}_{\mathbf{g}}(\mathbf{b}) \cdot \mathbf{C}$$

Lemma 1. *Let $\mathbf{C} \in \mathsf{RGSW}^N_{s,q}(m_1)$ with error variance σ_1^2, $\mathbf{b} \in \mathsf{RLWE}^N_{s,q}(m_2)$ with error variance σ_2^2. Let $\mathsf{ct} = \mathbf{C} \boxdot \mathbf{b}$, then $\mathsf{ct} \in \mathsf{RLWE}^N_{s,q}(m_1 \cdot m_2)$ with error variance $\sigma_{\mathsf{ext}}^2 \le 2\ell N g^2 \sigma_1^2 + \|m_1\|_2^2 \sigma_2^2$. Where g is defined as follows: If base power gadget is used, g is the decomposition base. If RNS gadget is used then $g = \max\{g_0, ..., g_{\ell-1}\}$.*

Proof. Let $\mathbf{b} = (a, b)$, and e be the error term of $\mathbf{b}$. Let $\{e_i\}_{i=0}^{2\ell-1}$ be the error terms of RLWE samples in $\mathbf{C}$. Let $(a_0, ...a_{\ell-1}, b_0, ..., b_{\ell-1}) = \mathtt{Decomp}_{\mathbf{g}}((a, b))$. We have $\mathbf{C} \boxdot \mathbf{b} = \mathsf{ct}' + m_1 \cdot (a, b)$, where $\mathsf{ct}' \in \mathsf{RLWE}^N_{s,q}(0)$. Since m_2 is the message of (a, b), then $\mathbf{C} \boxdot \mathbf{b} \in \mathsf{RLWE}^N_{s,q}(m_1 \cdot m_2)$.

The error term of $\mathbf{C} \boxdot \mathbf{b}$ is of the form $\sum_{i=0}^{\ell-1}(a_i \cdot e_i + b_i \cdot e_{i+\ell}) + m_1 \cdot e$. Simple analysis shows that the variance σ_{ext}^2 of the error term is bounded by $2\ell N g^2 \sigma_1^2 + \|m_1\|_2^2 \sigma_2^2$. ☐

B Noise Analysis

Lemma 2. *Let* $\mathbf{ct} \in \mathsf{LWE}^N_{\mathbf{s},q}(m)$ *with error variance* σ^2*, and* $\mathbf{ct}' = \mathsf{ModS}_{q\to q'}(ct)$*, then* $\mathbf{ct}' \in \mathsf{LWE}^N_{\mathbf{s},q'}(\lfloor (q'/q)\cdot m\rceil)$ *with error variance* $\sigma^2_{\mathrm{ms}} = (q'/q)^2\sigma^2 + N/24 + 1/6$.

Proof. Let $(\mathbf{a}, b) = \mathsf{LWE}^N_{\mathbf{s},q}(m)$ under key $\mathbf{s}$, where $\langle \mathbf{a}, \mathbf{s}\rangle + b = m + e \in \mathbb{Z}_q$. Let $(\mathbf{a}', b') = \mathbf{ct}' = \mathsf{ModS}_{q\to q'}(\mathbf{ct})$, then

$$\begin{aligned}\langle \lfloor (q'/q)\cdot \mathbf{a}\rceil, \mathbf{s}\rangle + \lfloor (q'/q)\cdot b\rceil &= \langle (q'/q)\cdot \mathbf{a}, \mathbf{s}\rangle + (q'/q)\cdot b + \langle \mathbf{r}, \mathbf{s}\rangle + \epsilon_1 \in \mathbb{Z}\\ &= (q'/q)\cdot (m+e) + \langle \mathbf{r}, \mathbf{s}\rangle + \epsilon_1\\ &= (\lfloor (q'/q)\cdot m\rceil) + (q'/q)\cdot e + \langle \mathbf{r}, \mathbf{s}\rangle + \epsilon_1 + \epsilon_2\\ &:= (\lfloor (q'/q)\cdot m\rceil) + e'\end{aligned}$$

where $\mathbf{r} := (q'/q)\mathbf{a} - \lfloor (q'/q)\cdot \mathbf{a}\rceil$ and $\|\mathbf{r}\|_\infty < 1/2, |\epsilon_1| < 1/2, |\epsilon_2| < 1/2$.

Similar to [20], the variance of $\langle \mathbf{r}, \mathbf{s}\rangle$ is $\|\mathbf{s}\|_1^2/12$ according to the central limit heuristic and the randomness of $\mathbf{a}$. Then by the distribution of $\mathbf{s}$ we have the variance of e' being $(q'/q)^2\sigma^2 + (N/2)/12 + 1/12 + 1/12 = (q'/q)^2\sigma^2 + N/24 + 1/6$, where σ^2 is the variance of e. □

Lemma 3. *Let* $\mathsf{ct} \in \mathsf{RLWE}^N_{s,q}(m)$ *with error variance* σ^2*,* KSK *be a* RGSW *ciphertext with error variance* σ^2_{ksk}*. Then* $\mathsf{ct}' = \mathsf{KeyS}_{s\to s'}(\mathsf{ct}, \mathsf{KSK})$ *is a ciphertext in* $\mathsf{RLWE}^N_{s',q}(m)$ *with error variance* σ'^2*, and*

$$\sigma'^2 \le \ell N g^2 \sigma^2_{\mathsf{ksk}} + \sigma^2,$$

where g *is defined as follows: If base power gadget is used in* BK*,* g *is the decomposition base. If RNS gadget is used then* $g = \max\{g_0, ..., g_{\ell-1}\}$.

Proof. The key switching procedure is essentially an external product except only ℓ RLWE ciphertexts are needed. For correctness, (a', b') in Algorithm 4 is of the form $\widetilde{\mathsf{ct}} + (a, b)$, where $\widetilde{\mathsf{ct}} \in \mathsf{RLWE}^N_{s',q}(0)$. Thus $\mathsf{ct} \in \mathsf{RLWE}^N_{s',q}(m)$. The error term of ct' is $\sum_{i=0}^{\ell} a_i \cdot e_i$, where $(a_0, ..., a_{\ell-1})$ is the decomposition of a, and $(e_0, ..., e_{\ell-1})$ are error term of the first RLWE ciphertext in KSK. Similar to Lemma 1, the error variance of ct satisfies $\sigma'^2 \le \ell N g^2\sigma^2_{\mathsf{ksk}} + \sigma^2$. This completes the proof. □

Lemma 4. *Let* $\mathbf{ct} \in \mathsf{LWE}^N_{\mathbf{s},q}(m)$ *with error variance* σ^2*,* $\mathsf{KSK}_{s\to s'}$ *be a set of* RGSW *ciphertexts with error variance* σ^2_{ksk}*. Then* $\mathbf{ct}' = \mathsf{DimS}_{N\to n'}(\mathbf{ct}, \mathsf{KSK}_{s\to s'})$ *is a ciphertext in* $\mathsf{LWE}^{n'}_{\mathbf{s}',q}(m)$ *with error variance* σ'^2*, and*

$$\sigma'^2 \le \ell N g^2 \sigma^2_{\mathsf{ksk}} + \sigma^2,$$

where g *is defined as follows: If base power gadget is used in* BK*,* g *is the decomposition base. If RNS gadget is used then* $g = \max\{g_0, ..., g_{\ell-1}\}$.

Proof. The correctness directly follows from the fact that the constant term of $\sum_{i=0}^{N/n-1} \widetilde{a}_i \cdot \widetilde{s}_i$ is the same as $\langle \mathbf{a}, \mathbf{s}\rangle \in \mathbb{Z}_q$. The analysis of the error variance is similar to Lemma 3. It involves in N/n key switching procedures in the ring $\mathbb{Z}_q[X]/(X^n+1)$, and the extract procedure will not change the error variance. Therefore $\sigma'^2 \le (N/n)(\ell n g^2 \sigma^2_{\mathsf{ksk}}) + \sigma^2 = \ell N g^2\sigma^2_{\mathsf{ksk}} + \sigma^2$. □

Lemma 5. *Let* $\mathbf{ct} \in \mathsf{LWE}^N_{\mathbf{s},q}(m)$ *with error variance* σ^2*,* AK *be a set of* RGSW *ciphertexts with error variance* σ^2_{ak}*. Then* $\mathsf{ct}' = \mathsf{LWE\text{-}to\text{-}RLWE}(\mathbf{ct}, \mathsf{AK})$ *is a ciphertext of* $\mathsf{RLWE}^N_{s,q}(m)$ *with error variance* σ'^2*, and*

$$\sigma'^2 \le \ell N^2 g^2 \sigma^2_{\mathsf{ak}} + \sigma^2,$$

where g *is defined as follows: If base power gadget is used in* AK*,* g *is the decomposition base. If RNS gadget is used then* $g = \max\{g_0, ..., g_{\ell-1}\}$.

Proof. In Algorithm 8, denote $[\langle \mathbf{a}, \mathbf{s} \rangle + b]_q = m + e =: \widetilde{m}$, then ct^* can be seen as a noiseless ciphertext on $m^* \in \mathcal{R}_q$ whose constant term is $\hat{N}\widetilde{m}$. The process of evaluating trace invokes $\log N$ times $\mathsf{HomAuto}$ procedures, and each involves a key switching procedure. This brings a new noise of variance $\sigma^2_{\mathsf{ext}} \le \ell N g^2 \sigma^2_{\mathsf{ak}}$. Let the error variance of the output ciphertext in step 3 is σ_j^2 in the j-th iteration, for $1 \le j \le \log N$. We have $\sigma^2_{j+1} = \sigma^2_j + (\sigma^2_j + \sigma^2_{\mathsf{ext}})$ and $\sigma^2_1 = 0$. Thus $\sigma^2_{\log N} = \sum_{i=0}^{\log N - 1} 2^i \cdot \sigma^2_{\mathsf{ext}} \le N\sigma^2_{\mathsf{ext}}$.

The resulting message of trace is $[N \cdot \hat{N}\widetilde{m}]_q = \widetilde{m} = m + e$. If we view m as the plaintext, then the noise of Algorithm 8 is of variance $\sigma'^2 \le \ell N^2 g^2 \sigma^2_{\mathsf{ak}} + \sigma^2$. $\square$