Homomorphic Modular Reduction and Improved Bootstrapping for BGV Scheme

Ruiqi Li[1[0000−0002−7452−104X]] and Chunfu Jia[1,2[0000−0002−5588−9690]]

[1] College of Cyber Science, Nankai University, Tianjin 300350, China
lrq@mail.nankai.edu.cn
[2] Tianjin Key Laboratory of Network and Data Security Technology, Tianjin 300350, China
cfjia@nankai.edu.cn

Abstract. Bootstrapping is a crucial subroutine of fully homomorphic encryption (FHE), where a homomorphic encryption scheme evaluates its own decryption circuits. Homomorphic modular reduction is a crucial part of bootstrapping a BGV ciphertext.
In this paper, we investigate the homomorphic modular reduction technique. We propose a new homomorphic modular reduction algorithm based on the idea of "blind rotation". This new homomorphic modular reduction procedure requires no basic homomorphic operations, hence it has lower noise accumulation and more suitable for implementing. Furthermore, we also resort to the blind rotation to construct a new bootstrapping procedure for the BGV scheme. We analyze the noise performance and the computational complexity of our scheme. The results illustrate that our new bootstrapping scheme achieves low noise accumulation so that the lattice approximation factor for the underlying worst-case lattice assumption is smaller than Chen and Zhang's work. Meanwhile, the complexity of our bootstrapping scheme is comparable with their scheme.

Keywords: Homomorphic encryption · Bootstrapping · Modular Reduction.

1 Introduction

Fully homomorphic encryption (FHE) is an emerging cryptographic primitive that enables homomorphic computations on encrypted data without decryption. In 2009, Gentry [18, 19] proposed the blueprint for achieving fully homomorphism and constructed the first FHE scheme. After Gentry's breakthrough, many FHE schemes emerge [6, 5, 15, 3, 29, 24, 13].

Almost all existing HE schemes include "noise" in ciphertexts, and the "noise" accumulates during homomorphic operations. When the "noise" rises up to some extent, ciphertexts cannot be decrypted correctly. To address this problem, Gentry proposed a technique called bootstrapping in [18] to refresh ciphertexts. Generally, a bootstrapping procedure is to evaluate decryption function homomorphically on encryptions of secret key. Bootstrapping is a computationally

fairly expensive procedure, and it is the main bottleneck of making FHE schemes practical. Therefore, there are many studies aimed at improving the efficiency of bootstrapping to make FHE faster.

Currently, researches of improving bootstrapping are pursued following two main approaches. The first approach, studied in [2, 17, 7, 9, 23, 4, 31], is to present bootstrapping techniques for HE schemes based on the Gentry-Sahai-Waters (GSW) scheme [24]. These schemes try to decrease the cost of bootstrapping a single ciphertext as much as possible, even at the expense of having to perform bootstrapping after evaluating every gate of the circuit. However, the above schemes have some limitations that the bootstrapping procedure has to be performed for essentially every gate of the circuit, and do not support packing several messages into a single ciphertext. The only exception is [31], since it first packs a number of LWE ciphertexts into an RLWE ciphertext and then refresh it.

Another approach explored in [20–22, 1, 25, 26, 32, 14, 12, 11, 8] aims at providing bootstrapping techniques for FHE schemes that can pack several messages into one ciphertext and refresh them in parallel. At present, BGV (Brakerski-Gentry-Vaikuntanathan) HE schemes [5, 21, 16] are one class of the most efficient somewhat homomorphic encryption (SWHE) schemes that support batching. Since BGV schemes can encrypt a ring element rather than a single bit in one ciphertext, they naturally support packing a number of messages into independent "slots" and performing Single-Instruction-Multiple-Data (SIMD) operations using the techniques based on Chinese Remainder Theorem (CRT) [34]. Therefore, many bootstrapping techniques for BGV scheme are studied. While bootstrapping such kind of schemes may be costly, it can simultaneously refresh plenty of messages in a single bootstrapping execution. Though the bootstrapping schemes of the first approach can reduce the runtime of bootstrapping a single ciphertext as much as possible, bootstrapping methods of the batched BGV scheme still have much better amortized per-bit runtime.

However, in the BGV scheme, the noise of ciphertext grows quadratically after every homomorphic multiplication, therefore these schemes essentially incur quasi-polynomial noise when decryption circuits have polynomial multiplication levels, and consequently require worst-case lattice assumptions with super-polynomial approximation factors. [14] is an exception among these works, which utilizes the techniques of [2] to refresh BGV ciphertexts. These techniques allow their construction to achieve worst-case assumptions with polynomial approximation factors.

The decryption function of BGV scheme can be represented as

$$\mathbf{Dec}(c, sk) = \mu + te \bmod (q, \Phi_m(X)) \tag{1}$$

where c is the encryption of message μ under the secret key sk, t is a plaintext modulus, q is a modulus and $\Phi_m(X)$ is a cyclotomic polynomial modulus. Therefore, a bootstrapping procedure of BGV scheme consists of two steps: homomorphic polynomial arithmetic and homomorphic modular reduction. This indicates that we can make homomorphic modular reduction algorithm more practical so as to obtain a bootstrapping procedure which has better performance.

Our Results. In this paper, we investigate the homomorphic modular reduction algorithm in the procedure of bootstrapping a BGV ciphertext. We propose a new homomorphic modular reduction algorithm based on the idea of blind rotation used in the FHEW-like scheme [17, 30]. Our new algorithm is suitable for all RLWE-based HE schemes whose decryption circuits satisfy Equation 1. We obtain a new bootstrapping procedure for BGV. We analyze the noise growth and the computational complexity. Theoretic analysis results show that our scheme can bootstrap BGV ciphertexts with polynomial noise and has lower noise accumulation than Chen and Zhang's work [14], hence we can achieve smaller lattice approximation factor for the underlying worst-case lattice assumption. Meanwile, the computational complexity is comparable with that of [14].

2 Preliminaries

In this paper, we use lower case letters to denote scalers including integers, reals, e.g. a, and use italic bold lower case letters to denote polynomials, e.g. $\boldsymbol{a}$. We use $\overrightarrow{a}$ to denote a vector. We write the ceiling, floor and rounding functions as $\lceil\cdot\rceil$, $\lfloor\cdot\rfloor$, and $\lfloor\cdot\rceil$, respectively. For integers n, t, $[n]_t$ represents the reduction of n modulo t, and $[n]_t \in (-\lfloor t/2\rfloor, \lfloor t/2\rfloor]$.

We use $a \leftarrow \chi$ to denote sampling a according to the distribution χ, and use $U(S)$ to denote a uniform distribution whose support is a finite set S. We denote by χ_{key} a ternary distribution, which samples a value from $\{-1, 0, 1\}$. χ_{err} is used to denote a discrete Gaussian distribution with a standard deviation σ_{err}. All logarithms in this paper are base two, unless stated otherwise.

2.1 Cyclotomic Rings

In this paper, BGV schemes and our bootstrapping procedure are supposed to perform over power-of-two cyclotomic rings.

Let N be a power of two, we denote the $2N$-th cyclotomic ring by $\mathcal{R} := \mathbb{Z}[X]/(X^N + 1)$ and its quotient ring by $\mathcal{R}_q := \mathcal{R}/q\mathcal{R}$. For a polynomial $\boldsymbol{a} \in \mathcal{R}$, we write $\boldsymbol{a} = a_0 + a_1X + \cdots a_{N-1}X^{N-1}$, and denote its coefficient vector by $\overrightarrow{a} = (a_0, a_1, \ldots, a_{N-1})$. We denote the ℓ_∞ norm of $\boldsymbol{a}$ as $\|\boldsymbol{a}\|_\infty = \|\overrightarrow{a}\|_\infty = \max_{0\le i<N}\{|a_i|\}$. There exists a constant $\delta_{\mathcal{R}}$ such that $\|\boldsymbol{a}\cdot\boldsymbol{b}\|_\infty \le \delta_{\mathcal{R}}\|\boldsymbol{a}\|_\infty\|\boldsymbol{b}\|_\infty$ for any $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{R}$, and we use the bound $\delta_{\mathcal{R}} = 2\sqrt{N}$ for $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$.

Let $\boldsymbol{a} = a_0 + a_1X + \cdots a_{N-1}X^{N-1}$ be a polynomial in $\mathcal{R}$ and m be a positive integer less than N. Notice that

$$\begin{aligned}\boldsymbol{a}\cdot X^m &= a_0X^m + \cdots + a_{N-1-m}X^{N-1} + a_{N-m}X^N + \cdots + a_{N-1}X^{N-1+m}\\ &= -a_{N-m} - \cdots - a_{N-1}X^{m-1} + a_0X^m + \cdots + a_{N-1-m}X^{N-1} \bmod X^N + 1\end{aligned}$$

The above equation implies that $\boldsymbol{a}\cdot X^m$ is a cyclic rotation of $\boldsymbol{a}$ with the cycled entries negated, and that the ℓ_∞-norm of $\boldsymbol{a}\cdot X^m$ is equal to that of $\boldsymbol{a}$.

2.2 (Ring) LWE Problems and Ciphertexts

We now introduce the LWE problem and the ring-LWE problem. Firstly, we bring in the definition of B-bounded distribution.

Definition 1 (B-bounded distribution). *A distribution ensemble $\{\chi_n\}_{n\in\mathbb{N}}$, supported over the integers or polynomial rings, is called B-bounded if*

$$\Pr_{e\leftarrow\chi_n}[\|e\|_\infty > B] = \mathrm{negl}(n).$$

The LWE problem was firstly introduced by Regev in [33].

Definition 2 (Decisional LWE (DLWE)[33]). *For security parameter λ, let $n = n(\lambda), q = q(\lambda)$ be integers, and let $\chi = \chi(\lambda)$ be a distribution over $\mathbb{Z}$. The decisional LWE problem (denoted by $DLWE_{n,q,\chi}$) is to distinguish the following two distributions: In the first distribution, one first draws a secret vector $\boldsymbol{s} \in \mathbb{Z}_q^n$ uniformly, and then samples tuples $(b_i, \boldsymbol{a}_i) \in \mathbb{Z}_q \times \mathbb{Z}_q^n$ by choosing $\boldsymbol{a}_i \leftarrow \mathbb{Z}_q^n$ uniformly at random and a noise term $e_i \leftarrow \chi$, and setting $b_i = \langle \boldsymbol{a}_i, \boldsymbol{s}\rangle + e_i$. In the second distribution, one samples $(b_i, \boldsymbol{a}_i)$ uniformly from $\mathbb{Z}_q^{n+1}$.*

The RLWE problem was firstly introduced by Lyubashevsky et al. in [28].

Definition 3 (Decisional RLWE (DRLWE)[28]). *Let K be a number field and R be the ring of integers of K. $R^\vee \subset K$ is the dual fractional ideal of R. Let χ be a distribution over $K_\mathbb{R} = K \otimes_\mathbb{Q} \mathbb{R}$. The decisional version of RLWE problem (denoted by $DRLWE_{q,\chi}$) is to distinguish the following two distributions: In the first distribution, one first draws $s \leftarrow R_q$ uniformly at random, and samples (a, b) by sampling $a \leftarrow R_q$ uniformly, and a noise term $e \leftarrow \chi$, and setting $b = a \cdot s + e$. In the second distribution, one samples (a, b) uniformly over $R_q \times K \otimes_\mathbb{Q} \mathbb{R}$.*

The theorem below captures reductions from ideal lattice GapSVP (or GapSIVP) to RLWE, and we state the result in terms of B-bounded distributions.

Theorem 1 (Adapted from [14]). *Let R be the m-th cyclotomic ring, and $n = \phi(m)$. Let $q = q(n)$, $q \equiv 1 \bmod m$ be an integer and $B = \omega(\sqrt{n \log n})$. Let χ be a B-bounded distribution. There is a polynomial time quantum reduction from $n^{\omega(1)}q/B$-approximate SVP on ideal lattices in R to $DRLWE_{q,\chi}$.*

In the following, we introduce two forms of ciphertexts based on LWE problem and RLWE problem.

Definition 4 (LWE ciphertexts). *Let n, q be positive integers. An LWE ciphertext of $m \in \mathbb{Z}$ under the secret key $\overrightarrow{s} \in \mathbb{Z}^n$ is defined as*

$$\boldsymbol{LWE}_{\overrightarrow{s},q}(m, e) := (\overrightarrow{a}, b) = (\overrightarrow{a}, -\langle \overrightarrow{a}, \overrightarrow{s}\rangle + e + m) \in \mathbb{Z}_q^{n+1}$$

where $\overrightarrow{a} \leftarrow U(\mathbb{Z}_q^n)$ and e is a small error.

Definition 5 (Ring-LWE ciphertexts). *Let Q be a positive integer and N be a power of 2. An RLWE ciphertext of $\boldsymbol{m} \in \mathcal{R}$ under the secret key $\boldsymbol{s} \in \mathcal{R}$ is defined as*

$$\boldsymbol{RLWE}_{\boldsymbol{s},Q}(\boldsymbol{m}) := (\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a}, -\boldsymbol{a} \cdot \boldsymbol{s} + \boldsymbol{e} + \boldsymbol{m}) \in \mathcal{R}_Q^2$$

where $\boldsymbol{a} \leftarrow U(\mathcal{R}_Q)$ and $\boldsymbol{e}$ is a small error polynomial.

Sample Extraction. A RLWE ciphertext consists of two polynomials with N coefficients in $\mathcal{R}$ and it is easy to homomorphically extract a coefficient as a scalar LWE ciphertext with the same key. Let $(\boldsymbol{a}, \boldsymbol{b}) = \mathbf{RLWE}_{\boldsymbol{s},q}(\boldsymbol{m}, \boldsymbol{e})$ be an RLWE ciphertext. Multiplication of two polynomials $\boldsymbol{a}$ and $\boldsymbol{s}$ in $\mathcal{R}$ can be written as:

$$\boldsymbol{s} \cdot \boldsymbol{a} = \sum_{i=0}^{N-1} \left(\sum_{j=0}^{i} s_j \cdot a_{i-j} - \sum_{j=i+1}^{N-1} s_j \cdot a_{i-j+N} \right) X^i$$

Let $\overrightarrow{s} = (s_0, \ldots, s_{N-1}) \in \mathbb{Z}^N$ be a vector of coefficients of $\boldsymbol{s}$. We can extract LWE ciphertexts $\mathbf{LWE}_{\overrightarrow{s},q}(m_i, e_i) = (\overrightarrow{a}^{(i)}, b_i)$ for $0 \leq i < N$ from $\boldsymbol{a}$, where

$$\overrightarrow{a}^{(i)} = (a_i, a_{i-1}, \ldots, a_0, -a_{N-1}, -a_{N-2}, \ldots, -a_{i+1}).$$

We denote this procedure as $(\overrightarrow{a}^{(i)}, b_i) \leftarrow \mathbf{Extract}_i((\boldsymbol{a}, \boldsymbol{b}))$, and we simply write $\mathbf{Extract}((\boldsymbol{a}, \boldsymbol{b}))$ when $i = 0$.

2.3 BGV Scheme

The BGV scheme [5, 20] is one of frequently-used RLWE-based HE schemes. In the BGV scheme, n is the degree of underlying cyclotomic polynomial, t is the plaintext modulus, and q is the coefficient modulus. In this paper, we assume that the BGV scheme is defined over $R = \mathbb{Z}[X]/(X^n + 1)$ where n is a power of 2. The plaintext space is $R_t = R/tR$, and the secret key $\boldsymbol{s}$ is an element of $R_q = R/qR$. In practice, each coefficient of $\boldsymbol{s}$ is usually sampled from the ternary distribution χ_{key}. A BGV ciphertext is a pair $(\boldsymbol{a}, \boldsymbol{b})$ of elements in R_q.

Specifically, the BGV scheme is essentially parameterized by a sequence of decreasing moduli $q_L \gg q_{L-1} \gg \cdots \gg q_0$. For $0 \leq \ell \leq L$, a level-ℓ ciphertext $(\boldsymbol{a}^{(\ell)}, \boldsymbol{b}^{(\ell)})$ of a message $\boldsymbol{m} \in R_t$ under the key $\boldsymbol{s}^{(\ell)} \in R$ satisfies

$$\mathbf{Dec}(\boldsymbol{a}^{(\ell)}, \boldsymbol{s}^{(\ell)}) = \boldsymbol{m} + t \cdot \boldsymbol{e}^{(\ell)} \bmod (q_\ell, \Phi(X))$$

where $\boldsymbol{e}^{(\ell)}$ is the *noise* of the ciphertext $(\boldsymbol{a}^{(\ell)}, \boldsymbol{b}^{(\ell)})$ and $t \cdot \|\boldsymbol{e}^{(\ell)}\|_\infty \ll q_\ell$. After each homomorphic operation, modulus q_ℓ of level ℓ needs to be switched to $q_{\ell-1}$ of level $\ell - 1$ by *Modulus Switching*, and the corresponding key is also switched by *Key Switching*. When the level comes to 0, we cannot perform any more homomorphic operations and thus require the bootstrapping procedure, i.e. we have to refresh the ciphertext to obtain a new one with a level-L secret key.

In this paper, we denote a BGV ciphertext of $\boldsymbol{m} \in R_t$ under the secret key $\boldsymbol{s}$ as $\mathbf{BGV}^t_{\boldsymbol{s},q}(\boldsymbol{m}, \boldsymbol{e}) \in R_q^2$, where $\boldsymbol{e}$ is an error. Notice that $\mathbf{BGV}^t_{\boldsymbol{s},q}(\boldsymbol{m}, \boldsymbol{e})$ is in fact an RLWE ciphertext $\mathbf{RLWE}_{\boldsymbol{s},q}(\boldsymbol{m}, t \cdot \boldsymbol{e})$. Therefore, the properties of RLWE ciphertexts introduced in Section 2.2 also work for BGV ciphertexts.

2.4 A Ring Variant of the GSW Scheme

In this section, we adapt the definitions of $\mathbf{RLWE}'$ and $\mathbf{RGSW}$ from [30, 27] to describe a ring variant of the GSW HE scheme [24, 2] over $\mathcal{R}$.

Let $\overrightarrow{g} = (g_0, g_1, \ldots, g_{d-1}) \in \mathbb{Z}^d$ be a gadget vector. Let $\boldsymbol{h} \in \mathcal{R}_Q$ be a polynomial and $g^{-1}(\boldsymbol{h}) = (\boldsymbol{h}_0, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_{d-1}) \in \mathcal{R}^d$ be a gadget decomposition of $\boldsymbol{h}$ such that $\boldsymbol{h} = \sum_{i=0}^{d-1} g_i \cdot \boldsymbol{h}_i$. For a power of two modulus Q, we will use a power gadget vector $(1, B, B^2, \ldots, B^{d-1})$ with a power of two B.

For a gadget vector $\overrightarrow{g} \in \mathbb{Z}^d$, we define

$$\mathbf{RLWE}'_{\boldsymbol{s},Q}(\boldsymbol{m}) := (\mathbf{RLWE}_{\boldsymbol{s},Q}(g_0 \cdot \boldsymbol{m}, \boldsymbol{e}_0), \ldots, \mathbf{RLWE}_{\boldsymbol{s},Q}(g_{d-1} \cdot \boldsymbol{m}, \boldsymbol{e}_{d-1})) \in \mathcal{R}_Q^{2\times d}.$$

In order to explain the homomorphic scalar multiplication, we firstly introduce the multiplication between a scalar and an RLWE ciphertext. Let $\boldsymbol{c} \in \mathcal{R}_Q$ be a scalar and $\mathbf{RLWE}_{\boldsymbol{s},Q}(\boldsymbol{m}, \boldsymbol{e}) := (\boldsymbol{a}, \boldsymbol{b})$. The multiplication between $\boldsymbol{c}$ and $\mathbf{RLWE}_{\boldsymbol{s},Q}(\boldsymbol{m}, \boldsymbol{e})$ is defined as

$$\boldsymbol{c} \cdot \mathbf{RLWE}_{\boldsymbol{s},Q}(\boldsymbol{m}, \boldsymbol{e}) := (\boldsymbol{c} \cdot \boldsymbol{a}, \boldsymbol{c} \cdot \boldsymbol{b}) = \mathbf{RLWE}_{\boldsymbol{s},Q}(\boldsymbol{c} \cdot \boldsymbol{m}, \boldsymbol{c} \cdot \boldsymbol{e}).$$

In the rest of this paper, we regard an RLWE ciphertext as a single element in the procedure of the scalar multiplication, so we can further generalize the multiplication between a scalar and an RLWE ciphertext to "inner product" between a vector of scalars and a vector of RLWE ciphertexts.

On the basis of the above discussion, the homomorphic scalar multiplication between $\mathbf{RLWE}'_{\boldsymbol{s},Q}(\boldsymbol{m})$ and a polynomial $\boldsymbol{h} \in \mathcal{R}_Q$ is defined as

$$\begin{aligned}
&\boldsymbol{h} \odot \mathbf{RLWE}'_{\boldsymbol{s},Q}(\boldsymbol{m}) \\
&= \langle g^{-1}(\boldsymbol{h}), (\mathbf{RLWE}_{\boldsymbol{s},Q}(g_0 \cdot \boldsymbol{m}, \boldsymbol{e}_0), \cdots, \mathbf{RLWE}_{\boldsymbol{s},Q}(g_{d-1} \cdot \boldsymbol{m}, \boldsymbol{e}_{d-1}))\rangle \\
&= \sum_{i=0}^{d-1} \boldsymbol{h}_i \cdot \mathbf{RLWE}_{\boldsymbol{s},Q}(g_i \cdot \boldsymbol{m}, \boldsymbol{e}_i) = \mathbf{RLWE}_{\boldsymbol{s},Q}(\sum_{i=0}^{d-1} g_i \cdot \boldsymbol{h}_i \cdot \boldsymbol{m}, \sum_{i=0}^{d-1} \boldsymbol{h}_i \cdot \boldsymbol{e}_i) \\
&= \mathbf{RLWE}_{\boldsymbol{s},Q}(\boldsymbol{h} \cdot \boldsymbol{m}, \boldsymbol{e}') \in \mathcal{R}_Q^2
\end{aligned}$$

The procedure of the homomorphic scalar multiplication $\boldsymbol{h} \odot \mathbf{RLWE}'_{\boldsymbol{s},Q}(\boldsymbol{m})$ produces an RLWE ciphertext $\mathbf{RLWE}_{\boldsymbol{s},Q}(\boldsymbol{h} \cdot \boldsymbol{m}, \boldsymbol{e}')$ where $\boldsymbol{e}' = \sum_{i=0}^{d-1} \boldsymbol{h}_i \cdot \boldsymbol{e}_i$. The following lemma states the computational complexity of the homomorphic scalar multiplication and the upper bound of the error of the resulting ciphertext.

Lemma 1. *Let E be an upper bound of error in $\boldsymbol{RLWE}'_{\boldsymbol{s},Q}(\boldsymbol{m})$. The homomorphic scalar multiplication between $\boldsymbol{RLWE}'_{\boldsymbol{s},Q}(\boldsymbol{m})$ and an element $\boldsymbol{h} \in \mathcal{R}_Q$ can be computed in time $\tilde{O}(dN)$, and the error of the resulting RLWE ciphertext is bounded by $2d\sqrt{N}BE$.*

Proof. Note that computing $\boldsymbol{h}_i \cdot \mathbf{RLWE}_{\boldsymbol{s},Q}(g_i \cdot \boldsymbol{m}, \boldsymbol{e}_i)$ requires 2 multiplications of two elements in $\mathcal{R}$, so the whole procedure requires $2d$ multiplications. In addition, the complexity of multiplications for ring elements of $\mathcal{R}$ by FFT (Fast Fourier Transform) is $O(N \log N)$. Generally, the time complexity of the homomorphic scalar multiplication is no more than $\tilde{O}(dN)$.

The error of the output RLWE ciphertext is $\boldsymbol{e}' = \sum_{i=0}^{d-1} \boldsymbol{h}_i \cdot \boldsymbol{e}_i$, and $\|\boldsymbol{h}_i\|_\infty \leq B, \|\boldsymbol{e}_i\|_\infty \leq E$. Therefore, the error of the output ciphertext is bounded by $2d\sqrt{N}BE$.

The form of ciphertexts of our RGSW scheme is defined as follows.

Definition 6 (RGSW ciphertexts). *Let Q be a positive integer and N be a power of 2. Let $\overrightarrow{g} \in \mathbb{Z}^d$ be a gadget vector. An RGSW encryption $\boldsymbol{m} \in \mathcal{R}$ under the secret key $\boldsymbol{s} \in \mathcal{R}$ is defined as*

$$\boldsymbol{RGSW}_{\boldsymbol{s},Q}(\boldsymbol{m}) := (\boldsymbol{RLWE}'_{\boldsymbol{s},Q}(\boldsymbol{s}\cdot\boldsymbol{m}), \boldsymbol{RLWE}'_{\boldsymbol{s},Q}(\boldsymbol{m})) \in \mathcal{R}_Q^{4\times d}$$

We can define an external product between an RLWE ciphertext and an RGSW ciphertext, similar to [9, 10].

Lemma 2 (External Product). *Let $\boldsymbol{RLWE}_{\boldsymbol{s},Q}(\boldsymbol{m}_1, \boldsymbol{e}) = (\boldsymbol{a}, \boldsymbol{b})$ be an RLWE ciphertext and $g^{-1}(\boldsymbol{a}) = (\boldsymbol{a}_i)_{0\le i\le d-1}, g^{-1}(\boldsymbol{b}) = (\boldsymbol{b}_i)_{0\le i\le d-1}$. Let*

$$\begin{aligned}
&\boldsymbol{RGSW}_{\boldsymbol{s},Q}(\boldsymbol{m}_2)\\
=&(\boldsymbol{RLWE}'_{\boldsymbol{s},Q}(\boldsymbol{s}\cdot\boldsymbol{m}_2), \boldsymbol{RLWE}'_{\boldsymbol{s},Q}(\boldsymbol{m}_2))\\
=&((\boldsymbol{RLWE}_{\boldsymbol{s},Q}(g_i\cdot\boldsymbol{s}\cdot\boldsymbol{m}_2, \boldsymbol{e}_{1,i}))_{0\le i\le d-1}, (\boldsymbol{RLWE}_{\boldsymbol{s},Q}(g_i\cdot\boldsymbol{m}_2, \boldsymbol{e}_{2,i}))_{0\le i\le d-1})
\end{aligned}$$

be an RGSW ciphertext. The external product between these two ciphertexts is computed by

$$\boldsymbol{RLWE}_{\boldsymbol{s},Q}(\boldsymbol{m}_1, \boldsymbol{e})\times\boldsymbol{RGSW}_{\boldsymbol{s},Q}(\boldsymbol{m}_2) = \boldsymbol{a}\odot\boldsymbol{RLWE}'_{\boldsymbol{s},Q}(\boldsymbol{s}\cdot\boldsymbol{m}_2)+\boldsymbol{b}\odot\boldsymbol{RLWE}'_{\boldsymbol{s},Q}(\boldsymbol{m}_2).$$

in time $\tilde{O}(dN)$ and produces an RLWE ciphertext $\boldsymbol{RLWE}_{\boldsymbol{s},Q}(\boldsymbol{m}_1\cdot\boldsymbol{m}_2, \boldsymbol{e}')$ where $\boldsymbol{e}' = \boldsymbol{m}_2\cdot\boldsymbol{e}+\sum_{i=0}^{d-1}\boldsymbol{a}_i\boldsymbol{e}_{1,i}+\boldsymbol{b}_i\boldsymbol{e}_{2,i}$. If we assume that $\boldsymbol{m}_2 = X^v$ with some integer v, $\|\boldsymbol{e}\|_\infty \le E_1$ and $\|\boldsymbol{e}_{1,i}\|_\infty, \|\boldsymbol{e}_{2,i}\|_\infty \le E_2$, then $\|\boldsymbol{e}'\|_\infty \le E_1 + 4\sqrt{N}dBE_2$.

Proof. The external product is computed as

$$\begin{aligned}
&\boldsymbol{a}\odot\mathbf{RLWE}'_{\boldsymbol{s},Q}(\boldsymbol{s}\cdot\boldsymbol{m}_2) + \boldsymbol{b}\odot\mathbf{RLWE}'_{\boldsymbol{s},Q}(\boldsymbol{m}_2)\\
=&\ \mathbf{RLWE}_{\boldsymbol{s},Q}(\boldsymbol{a}\cdot\boldsymbol{s}\cdot\boldsymbol{m}_2, \sum_{i=0}^{d-1}\boldsymbol{a}_i\cdot\boldsymbol{e}_{1,i}) + \mathbf{RLWE}_{\boldsymbol{s},Q}(\boldsymbol{b}\cdot\boldsymbol{m}_2, \sum_{i=0}^{d-1}\boldsymbol{b}_i\cdot\boldsymbol{e}_{2,i})\\
=&\ \mathbf{RLWE}_{\boldsymbol{s},Q}((\boldsymbol{a}\cdot\boldsymbol{s}+\boldsymbol{b})\cdot\boldsymbol{m}_2, \sum_{i=0}^{d-1}\boldsymbol{a}_i\cdot\boldsymbol{e}_{1,i} + \boldsymbol{b}_i\cdot\boldsymbol{e}_{2,i})\\
=&\ \mathbf{RLWE}_{\boldsymbol{s},Q}(\boldsymbol{m}_1\cdot\boldsymbol{m}_2 + \boldsymbol{e}\cdot\boldsymbol{m}_2, \sum_{i=0}^{d-1}\boldsymbol{a}_i\cdot\boldsymbol{e}_{1,i} + \boldsymbol{b}_i\cdot\boldsymbol{e}_{2,i})\\
=&\ \mathbf{RLWE}_{\boldsymbol{s},Q}(\boldsymbol{m}_1\cdot\boldsymbol{m}_2, \boldsymbol{m}_2\cdot\boldsymbol{e} + \sum_{i=0}^{d-1}\boldsymbol{a}_i\cdot\boldsymbol{e}_{1,i} + \boldsymbol{b}_i\cdot\boldsymbol{e}_{2,i}) \in \mathcal{R}_Q^2
\end{aligned}$$

In fact, the external product is computed by two homomorphic scalar multiplications. Hence, according to Lemma 1, we know the time complexity of the external product procedure is within $\tilde{O}(dN)$.

Since the error of the output ciphertext can be written as $\boldsymbol{e}' = \boldsymbol{m}_2\cdot\boldsymbol{e} + \sum_{i=0}^{d-1}\boldsymbol{a}_i\cdot\boldsymbol{e}_{1,i} + \boldsymbol{b}_i\cdot\boldsymbol{e}_{2,i}$, then we have $\|\boldsymbol{e}'\|_\infty \le E_1 + 4\sqrt{N}dBE_2$.

3 New Homomorphic Modular Reduction Algorithm

3.1 The Basic Idea

The basic idea of our homomorphic modular reduction is inspired by the "blind rotation" technique used in [17, 9, 10].

Notice that $\langle X \rangle = \{1, X, \ldots, X^{N-1}, -1, \ldots, -X^{N-1}\}$ forms a cyclic group in $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$. If we set $q = 2N$, then $\mathbb{Z}_q \cong \langle X \rangle$. This means that we can map an integer $v \in \mathbb{Z}_q$ to an element X^v of the group $\langle X \rangle$. Note that the degree of a test polynomial is at most N in $\mathcal{R}$, but there are $q = 2N$ possible values for $v \in \mathbb{Z}_q$, and we have to verify every possible value of v to obtain $[v]_t$. Therefore, we cannot use the test polynomials similar to the one proposed in [9].

We observe that, for $v \in [0, q-1]$ and $i \in [0, N-1]$,

$$\text{the constant term of } X^{-v} \cdot X^i = \begin{cases} 1, & i = v, \\ -1, & i = v - q/2, \\ 0, & \text{otherwise.} \end{cases}$$

holds. This result can be used to find v from the range $\{0, 1, \ldots, q-1\}$. However, this fact is not sufficient yet, since there are two possible non-zero values of the constant term of $X^{-v} \cdot X^i$ (i.e., 1 or -1). What we want is that the constant term of $X^{-v} \cdot X^i$ is equal to 1 whenever $i = v$ or $i = v - q/2$. Fortunately, we find that, for $0 \le i < N$ and $v \in \{0, 1, \ldots, q-1\}$,

$$\text{the constant term of } \left\lfloor \frac{1}{2} X^i \cdot X^{-v} + \frac{1}{4} \right\rceil = \begin{cases} \left\lfloor \frac{3}{4} \right\rceil = 1, & i = v, \\ \left\lfloor -\frac{1}{4} \right\rceil = 0, & i = v - q/2, \\ \left\lfloor \frac{1}{4} \right\rceil = 0, & \text{otherwise.} \end{cases}$$

$$\text{the constant term of } \left\lfloor -\frac{1}{2} X^i \cdot X^{-v} + \frac{1}{4} \right\rceil = \begin{cases} \left\lfloor -\frac{1}{4} \right\rceil = 0, & i = v, \\ \left\lfloor \frac{3}{4} \right\rceil = 1, & i = v - q/2, \\ \left\lfloor \frac{1}{4} \right\rceil = 0, & \text{otherwise.} \end{cases}$$

holds. Let α_i be the constant term of $\left\lfloor \frac{1}{2} X^i \cdot X^{-v} + \frac{1}{4} \right\rceil$ and β_i be the constant term of $\left\lfloor -\frac{1}{2} X^i \cdot X^{-v} + \frac{1}{4} \right\rceil$ for $0 \le i < N$. Utilizing the above equations, we can design the following method to obtain $[v]_t$ for $v \in \{0, 1, \ldots, q-1\}$:

$$\sum_{i=0}^{N-1} \left(\alpha_i \cdot [i]_t + \beta_i \cdot [i+N]_t \right) = [v]_t$$

As a toy example, let $q = 2N = 8$ and $v = 5$. We can obtain that $\alpha_0 = \lfloor \frac{1}{4} \rceil = 0$, $\alpha_1 = \lfloor -\frac{1}{4} \rceil = 0$, $\alpha_2 = \lfloor \frac{1}{4} \rceil = 0$, $\alpha_1 = \lfloor \frac{1}{4} \rceil = 0$, and that $\beta_0 = \lfloor \frac{1}{4} \rceil = 0$, $\beta_1 = \lfloor \frac{3}{4} \rceil = 1$, $\beta_2 = \lfloor \frac{1}{4} \rceil = 0$, $\beta_3 = \lfloor \frac{1}{4} \rceil = 0$. Hence, we have that

$$\sum_{i=0}^{3} (\alpha_i \cdot [i]_t + \beta_i \cdot [i+N]_t) = \beta_1 \cdot [1+4]_t = [5]_t.$$

3.2 Homomorphic Modular Reduction

In this subsection, we elaborate our new homomorphic modular reduction algorithm. The entire **HomModRed** procedure is formalized in Algorithm 1. Lemma 3 states the correctness of the homomorphic modular reduction and the noise performance of this procedure.

Algorithm 1 The **HomModRed** Algorithm

Input: An RLWE ciphertext $\mathbf{RLWE}_{2\boldsymbol{s},Q}(X^{-v}, 2t\boldsymbol{e}) = (\boldsymbol{a}, \boldsymbol{b}) \in \mathcal{R}_{2N}^2$
Output: An LWE ciphertext $\mathbf{LWE}_{\vec{s},Q}([v]_t, e')$ // $\vec{s}$ is the coefficient vector of $\boldsymbol{s}$

```
(a⃗, b) ← (0⃗, 0);
for i = 1 to N − 1 do
    (a, b) ← X · (a, b);
    (c, d) ← (a, ⌊½ b + ¼⌉),   (c′, d′) ← (−a, ⌊−½ b + ¼⌉);
    (c⃗, d) ← Extract((c, d)),   (c⃗′, d′) ← Extract((c′, d′));
    (a⃗, b) ← (a⃗, b) + [i]_t · (c⃗, d) + [i + N]_t · (c⃗′, d′);
end for
return (a⃗, b);
```

Lemma 3. *Let* $\boldsymbol{RLWE}_{2\boldsymbol{s},Q}(X^{-v}, 2t\boldsymbol{e}) = (\boldsymbol{a}, \boldsymbol{b})$ *where* $0 \le v \le 2N-1$ *and* $\|\boldsymbol{e}\|_\infty \le E$. *There exists an algorithm* ***HomModRed*** *that on input* $(\boldsymbol{a}, \boldsymbol{b})$, *outputs an LWE ciphertext* $\boldsymbol{LWE}_{\vec{s},Q}([v]_t, te')$ *in time* $O(N^2)$, *where* $|e'| \le tNE/2$.

Proof. At the i-th loop, we have that

$$\begin{aligned}(\boldsymbol{c}, \boldsymbol{d}) &= (\boldsymbol{a} \cdot X^i, \left\lfloor \frac{1}{2}(-\boldsymbol{a} \cdot X^i \cdot 2\boldsymbol{s} + 2t\boldsymbol{e} \cdot X^i + X^{-v} \cdot X^i) + \frac{1}{4} \right\rceil) \\ &= (\boldsymbol{a} \cdot X^i, -\boldsymbol{a} \cdot X^i \cdot \boldsymbol{s} + t\boldsymbol{e} \cdot X^i + \left\lfloor \frac{1}{2} X^{-v} \cdot X^i + \frac{1}{4} \right\rceil),\end{aligned}$$

$$\begin{aligned}(\boldsymbol{c}', \boldsymbol{d}') &= (-a \cdot X^i, \left\lfloor -\frac{1}{2}(-\boldsymbol{a} \cdot X^i \cdot 2\boldsymbol{s} + 2t\boldsymbol{e} \cdot X^i + X^{-v} \cdot X^i) + \frac{1}{4} \right\rceil) \\ &= (-\boldsymbol{a} \cdot X^i, \boldsymbol{a} \cdot X^i \cdot \boldsymbol{s} - t\boldsymbol{e} \cdot X^i + \left\lfloor -\frac{1}{2} X^{-v} \cdot X^i + \frac{1}{4} \right\rceil).\end{aligned}$$

Hence, we have that $(\overrightarrow{c}, d)$ and $(\overrightarrow{c}', d')$ satisfy

$$(\overrightarrow{c}, d) = (\overrightarrow{c}, -\langle \overrightarrow{c}, \overrightarrow{s} \rangle + te + \alpha),$$
$$(\overrightarrow{c}', d') = (\overrightarrow{c}', -\langle \overrightarrow{c}', \overrightarrow{s} \rangle - te + \beta),$$

where $\overrightarrow{s}$ is the coefficient vector of $\boldsymbol{s}$, e is the constant term of $\boldsymbol{e} \cdot X^i$, α is the constant term of $\lfloor \frac{1}{2} X^{-v} \cdot X^i + \frac{1}{4} \rceil$ and β is the constant term of $\lfloor -\frac{1}{2} X^{-v} \cdot X^i + \frac{1}{4} \rceil$.

Therefore, after $N-1$ iterations, $(\overrightarrow{a}, b)$ can be represented as

$$\overrightarrow{a} = \sum_{i=1}^{N-1} [i]_t \overrightarrow{c} + [i+N]_t \overrightarrow{c}',$$

$$b = -\langle \overrightarrow{a}, \overrightarrow{s} \rangle + t \sum_{i=1}^{N-1} ([i]_t - [i+N]_t) e_i + \sum_{i=1}^{N-1} ([i]_t \alpha_i + [i+N]_t \beta_i).$$

According to the discussion in Section 3.1, we have that

$$b = -\langle \overrightarrow{a}, \overrightarrow{s} \rangle + te' + [v]_t$$

where $e' = \sum_{i=1}^{N-1} ([i]_t - [i+N]_t) e_{i,0}$, therefore $(\overrightarrow{a}, b)$ is an LWE ciphertext of $[v]_t$ under the key $\overrightarrow{s}$.

In one loop, Step 3 in Algorithm 1 is just to perform cyclic rotation with the cycled entries negated on coefficients of $\boldsymbol{a}, \boldsymbol{b}$, and Step 5 is just to extract and shuffle several coefficients from polynomials. We think these two steps hardly contribute to the computational complexity of the whole algorithm. Step 4 in Algorithm 1 is to perform scalar multiplications in $\mathcal{R}$, and the computational complexity of scalar multiplications in $\mathcal{R}$ is $O(N)$. Step 6 requires two multiplications between a scalar and a vector of integers and an addition of two vectors of integers, hence the complexity of this step is $O(N)$. Therefore, the whole algorithm runs in time $O(N^2)$.

Now we start to analyze the noise performance. Based on the properties of power-of-two cyclotomic ring described in Section 2.1, we can get that $\|e \cdot X^i\|_\infty = \|e\|_\infty \leq E$, so $|e_i| \leq \|e \cdot X^i\|_\infty \leq E$. In addition we know $|[i]_t - [i+N]_t| \leq t/2$. Therefore, $|e'| \leq tNE/2$.

4 Improved Bootstrapping for BGV Scheme

Let the BGV ciphertext to be bootstrapped is $\mathbf{BGV}^t_{\boldsymbol{s},q_L}(\boldsymbol{m}, \boldsymbol{e}) \in R^2_{q_L}$. The blueprint of our bootstrapping procedure is described below and illustrated in Figure 1.

Modulus Switching. One fixes a power of two N and compute a new ciphertext $\mathbf{BGV}^t_{\boldsymbol{s},2N}(\boldsymbol{m}, \boldsymbol{e}')$ which encrypts the same plaintext $\boldsymbol{m}$ but has smaller size. In addition, it needs to be stressed that $\|\boldsymbol{m} + t\boldsymbol{e}'\|_\infty \leq N$.

Extraction. $\mathbf{BGV}^t_{\boldsymbol{s},2N}(\boldsymbol{m},\boldsymbol{e}')$ is in fact an RLWE ciphertext $\mathbf{RLWE}_{\boldsymbol{s},2N}(\boldsymbol{m},t\boldsymbol{e}')$. Here we use **Extract** to obtain n LWE ciphertexts $\{\mathbf{LWE}_{\overrightarrow{s},2N}(m_i,te'_i)\}_{0\le i<n}$ from $\mathbf{RLWE}_{\boldsymbol{s},2N}(\boldsymbol{m},t\boldsymbol{e}')$, where $\overrightarrow{s}$ is the coefficient vector of $\boldsymbol{s}$.

Blind Rotation. For an LWE ciphertext $\mathbf{LWE}_{\overrightarrow{s},2N}(m,te') = (\overrightarrow{a},b)$, we use a blind rotation procedure to get an RLWE ciphertext $\mathbf{RLWE}_{2\boldsymbol{z},Q}(X^{-\tilde{m}},2t\boldsymbol{e}'')$, where $\tilde{m} = b + \sum_{j=0}^{n-1} a_j \cdot s_j$. The output of this step is n RLWE ciphertexts of the form $\mathbf{RLWE}_{2\boldsymbol{z},Q}(X^{-\tilde{m}},t\boldsymbol{e}'')$.

Homomorphic Modular Reduction. When the above steps are done, we obtain n RLWE ciphertexts. Then, we will apply Algorithm 1 to each of RLWE ciphertexts, resulting in n LWE ciphertexts $\{\mathbf{LWE}_{\overrightarrow{z},Q}([\tilde{m}_i]_t,t\tilde{e}_i)\}_{0\le i<n}$.

Repacking. Finally, we repack the LWE ciphertexts output by the previous step into one single ciphertext which encrypts $\boldsymbol{m}$.

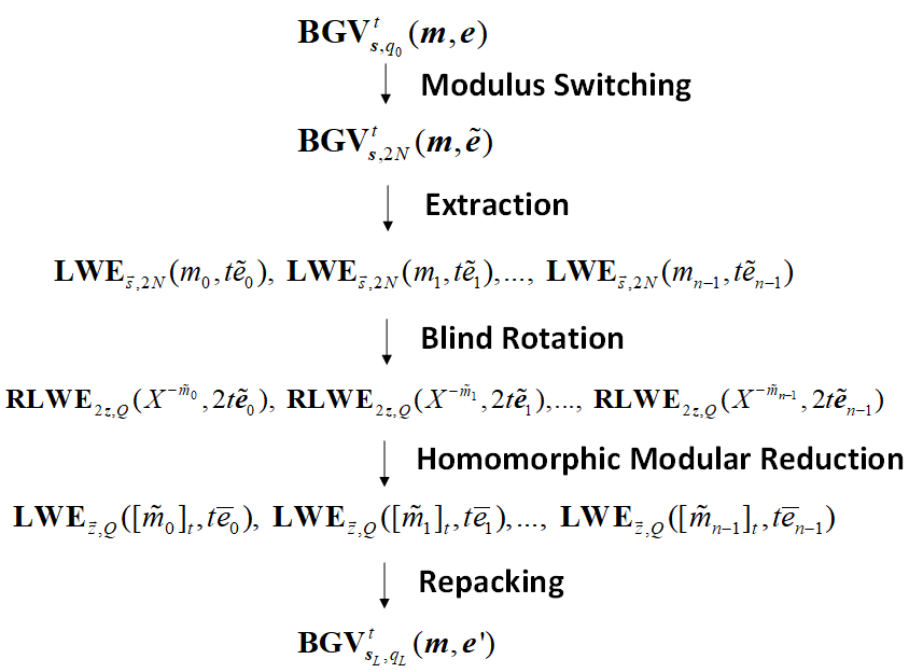

Fig. 1. Bootstrapping procedure

In the following, we elaborate two steps of our bootstrapping procedure—blind rotation and repacking.

4.1 Blind Rotation

After the **Extraction** procedure, we obtain n LWE ciphertexts. Each of these ciphertexts can be represented as $\mathbf{LWE}_{\overrightarrow{s},2N}(m, te) = (\overrightarrow{a}, b)$ satisfying

$$b + \langle \overrightarrow{a}, \overrightarrow{s} \rangle = b + \sum_{i=0}^{n-1} a_i \cdot s_i = m + te.$$

For an LWE ciphertext $\mathbf{LWE}_{\overrightarrow{s},2N}(m, te) = (\overrightarrow{a}, b)$, we start the blind rotation with $\mathbf{ACC} \leftarrow \mathbf{RLWE}_{2\boldsymbol{z},Q}(X^{-b}, 0)$ (For simplicity, we set $\mathbf{RLWE}_{2\boldsymbol{z},Q}(X^{-b}, 0) = (\mathbf{0}, X^{-b})$. To homomorphically compute $X^{-b-\sum_{i=0}^{n-1} a_i \cdot s_i}$, we need encryptions of s_i, which form the alleged *bootstrapping key*). The bootstrapping key is $\mathbf{bk} = \{\mathbf{RGSW}_{2\boldsymbol{z},Q}(s_i^+), \mathbf{RGSW}_{2\boldsymbol{z},Q}(s_i^-)\}$, where

$$\begin{cases} s_i^+ = 1, & s_i = 1 \\ s_i^+ = 0, & \text{otherwise} \end{cases}, \quad \begin{cases} s_i^- = 1, & s_i = -1 \\ s_i^- = 0, & \text{otherwise} \end{cases} \quad \text{for } i \in [0, N-1]$$

We iteratively compute

$$\mathbf{RGSW}_{2\boldsymbol{z},Q}(X^{-a_i \cdot s_i}) = \mathbf{I}_2 \otimes \overrightarrow{g} + (X^{-a_i} - 1)\mathbf{RGSW}_{2\boldsymbol{z},Q}(s_i^+) + (X^{a_i} - 1)\mathbf{RGSW}_{2\boldsymbol{z},Q}(s_i^-)$$

where $\mathbf{I}_2$ is a 2×2 identity matrix and $\otimes$ means tensor product. (In fact, $\mathbf{I}_2 \otimes \overrightarrow{g}$ is a trivial RGSW encryption of 1 under any key.) The above equation is correct since $s_i \in \{-1, 0, 1\}$ and at least one of s_i^+ and s_i^- is zero. Then we update $\mathbf{ACC} \leftarrow \mathbf{ACC} \odot \mathbf{RGSW}_{2\boldsymbol{z},Q}(X^{-a_i s_i})$.

After $\mathbf{ACC}$ is updated iteratively, the result is

$$\mathbf{RLWE}_{2\boldsymbol{z},Q}(X^{-b-a_0 s_0 - \cdots - a_{n-1} s_{n-1}}, 2t\tilde{\boldsymbol{e}}) = \mathbf{RLWE}_{2\boldsymbol{z},Q}(X^{-(m+te)}, 2t\tilde{\boldsymbol{e}})$$

The **BlindRotate** algorithm is described in Algorithm 2.

Algorithm 2 The **BlindRotate** Algorithm

Input: An LWE ciphertext $\mathbf{LWE}_{\overrightarrow{s},2N}(m, te) = (\overrightarrow{a}, b) \in \mathbb{Z}_{2N}^{n+1}$;
A bootstrapping key $\mathbf{bk} = \{\mathbf{RGSW}_{2\boldsymbol{z},Q}(s_i^+), \mathbf{RGSW}_{2\boldsymbol{z},Q}(s_i^-)\}$;
Output: An RLWE ciphertext $\mathbf{RLWE}_{2\boldsymbol{z},Q}(X^{-(m+te)}, 2t\tilde{\boldsymbol{e}})$;

```
ACC ← (0, X^{-b});
for i = 0 to n − 1 do
    ACC ← ACC ⊙ (I_2 ⊗ g⃗ + (X^{-a_i} − 1)RGSW_{2z,Q}(s_i^+) + (X^{a_i} − 1)RGSW_{2z,Q}(s_i^-));
end for
return ACC;
```

The following lemma concludes the noise growth and computational complexity of Algorithm 2.

Lemma 4. *Let* $\boldsymbol{LWE}_{\vec{s},2N}(m, te) = (\vec{a}, b) \in \mathbb{Z}_{2N}^{n+1}$ *be an LWE ciphertext and* $\mathbf{bk} = \{\boldsymbol{RGSW}_{2\boldsymbol{z},Q}(s_i^+), \boldsymbol{RGSW}_{2\boldsymbol{z},Q}(s_i^-)\}$ *be a bootstrapping key. There exists a* ***BlindRotate*** *algorithm that on input* $\boldsymbol{LWE}_{\vec{s},2N}(m, te) = (\vec{a}, b) \in \mathbb{Z}_{2N}^{n+1}$ *and* $\mathbf{bk}$*, outputs an RLWE ciphertext* $\boldsymbol{RLWE}_{2\boldsymbol{z},Q}(X^{-(m+te)}, 2t\tilde{\boldsymbol{e}})$ *in time* $\tilde{O}(dN^2)$*, and* $\|\tilde{\boldsymbol{e}}\|_\infty$ *is bounded by* $4n\sqrt{N}dBE_{bk}$ *if error of the bootstrapping key is bounded by* E_{bk}*.*

Proof. The correctness of this algorithm is stated above. Here we mainly prove the computational complexity and the noise performance. The cost of this algorithm is dominated by Step 3 of Algorithm 2, and the operation is essentially an external product. By Lemma 2, we know the cost of Step 3 is $\tilde{O}(dN)$ and error $\boldsymbol{e}_i$ of $\mathbf{ACC}_i$ at i-th loop satisfies $\|\boldsymbol{e}_i\|_\infty \leq \|\boldsymbol{e}_{i-1}\|_\infty + 4\sqrt{N}dBE_{bk}$. Therefore, the computational cost of the whole algorithm is no more than $\tilde{O}(dnN)$, and the error of the output ciphertext is bounded by $4n\sqrt{N}dBE_{bk}$.

4.2 Repacking

As described at the beginning of Section 4, we need a procedure to repack several LWE ciphertexts into a BGV ciphertext, so we resort to the repacking technique proposed in [31]. The **Repack** algorithm is formalized in Algorithm 3, and Lemma 5 illustrates the correctness of the repacking algorithm and the error growth during this procedure.

Algorithm 3 The **Repack** Algorithm

Input: $\{\mathbf{LWE}_{\vec{z},Q}(m_i, te_i) = (\vec{a}_i, b_i)\}_{0\leq i<n}$;
$\mathbf{rpk} = \{\mathbf{BGV}^t_{\boldsymbol{s},q}(z_j \cdot g_k, \boldsymbol{e}_{j,k})\}_{0\leq j<N, 0\leq k<d}$;
Output: $\mathbf{BGV}^t_{\boldsymbol{s},q}(\boldsymbol{m}, \boldsymbol{e})$; $//m(X) = \sum_{i=0}^{n-1} m_i X^i$

```
b = Σ_{i=0}^{n-1} b_i X^i;
for j = 0 to n − 1 do
    a_j = Σ_{i=0}^{n-1} a_{i,j} X^i;
    (a_{j,0}, ..., a_{j,d-1}) ← g^{-1}(a_j)
end for
(ã, b̃) ← (0, b) + Σ_{j,k} a_{j,k} · BGV^t_{s,q}(z_j · g^k, e_{j,k});
return (ã, b̃);
```

Lemma 5. *Algorithm 3 is an algorithm that on input* $\boldsymbol{LWE}_{\vec{z},Q}(m_i, te_i) = (\vec{a}_i, b_i)$ *for* $0 \leq i < n$ *with error* E*, and a repacking key*

$$\mathbf{rpk} = \{\boldsymbol{BGV}^t_{\boldsymbol{s},q}(z_j \cdot g_k, \boldsymbol{e}_{j,k})\}_{0\leq j<N, 0\leq k<d} = \{(\boldsymbol{c}_{j,k}, \boldsymbol{d}_{j,k})\}_{0\leq j<N, 0\leq k<d}$$

with error E_{repack}*, outputs a BGV ciphertext* $\boldsymbol{BGV}^t_{\boldsymbol{s},q}(\boldsymbol{m}, \boldsymbol{e})$ *encrypting* $\boldsymbol{m} = \sum_{i=0}^{n-1} m_i X^i$ *in time* $\tilde{O}(dnN)$*, with error at most* $E + 2\sqrt{n}dNE_{\text{repack}}$*.*

Proof. For an LWE ciphertext $\mathbf{LWE}_{\overrightarrow{z},Q}(m_i, te_i) = (\overrightarrow{a}_i, b_i)$, we have that $b_i + \langle \overrightarrow{a}_i, \overrightarrow{z} \rangle = m_i + te_i$, i.e., $b_i + \sum_{j=0}^{N-1} a_{i,j} \cdot z_j = m_i + te_i$. We also have that $\sum_{i=0}^{n-1} a_{i,j} X^i = \sum_{k=0}^{d-1} \boldsymbol{a}_{j,k} \cdot g^k$, and

$$\mathbf{BGV}^t_{\boldsymbol{s},q}(z_j \cdot g_k, \boldsymbol{e}_{j,k}) = (\boldsymbol{c}_{j,k}, \boldsymbol{d}_{j,k}) = (\boldsymbol{c}_{j,k}, -\boldsymbol{c}_{j,k}\boldsymbol{s} + t\boldsymbol{e}_{j,k} + z_j \cdot g^k)$$

Hence, it holds that

$$\begin{aligned}
(\tilde{\boldsymbol{a}}, \tilde{\boldsymbol{b}}) &= (\boldsymbol{0}, \boldsymbol{b}) + \sum_{j,k} \boldsymbol{a}_{j,k} \cdot (\boldsymbol{c}_{j,k}, \boldsymbol{d}_{j,k}) \\
&= (\boldsymbol{0}, \boldsymbol{b}) + (\sum_{j,k} \boldsymbol{a}_{j,k}\boldsymbol{c}_{j,k}, -\sum_{j,k} \boldsymbol{a}_{j,k}\boldsymbol{s} + t\sum_{j,k} \boldsymbol{a}_{j,k}\boldsymbol{e}_{j,k} + \sum_{j,k} z_j \cdot g^k \cdot \boldsymbol{a}_{j,k}) \\
&= (\boldsymbol{0}, \boldsymbol{b}) + (\sum_{j,k} \boldsymbol{a}_{j,k}\boldsymbol{c}_{j,k}, -\sum_{j,k} \boldsymbol{a}_{j,k}\boldsymbol{s} + t\sum_{j,k} \boldsymbol{a}_{j,k}\boldsymbol{e}_{j,k} + \sum_{j=0}^{N-1}\sum_{i=0}^{n-1} a_{i,j} X^i \cdot z_j) \\
&= (\sum_{j,k} \boldsymbol{a}_{j,k}\boldsymbol{c}_{j,k}, -\sum_{j,k} \boldsymbol{a}_{j,k}\boldsymbol{s} + t\sum_{j,k} \boldsymbol{a}_{j,k}\boldsymbol{e}_{j,k} + \sum_{i=0}^{n-1}(\sum_{j=0}^{N-1} a_{i,j} \cdot z_j) X^i + \sum_{i=0}^{n-1} b_i X^i) \\
&= (\sum_{j,k} \boldsymbol{a}_{j,k}\boldsymbol{c}_{j,k}, -\sum_{j,k} \boldsymbol{a}_{j,k}\boldsymbol{s} + t\sum_{j,k} \boldsymbol{a}_{j,k}\boldsymbol{e}_{j,k} + \sum_{i=0}^{n-1}(m_i + te_i) X^i) \\
&= (\sum_{j,k} \boldsymbol{a}_{j,k}\boldsymbol{c}_{j,k}, -\sum_{j,k} \boldsymbol{a}_{j,k}\boldsymbol{s} + t\sum_{j,k} \boldsymbol{a}_{j,k}\boldsymbol{e}_{j,k} + t\sum_{i=0}^{N-1} e_i X^i + \boldsymbol{m}) \\
&= \mathbf{BGV}^t_{\boldsymbol{s},q}(\boldsymbol{m}, \sum_i e_i X^i + \sum_{j,k} \boldsymbol{a}_{j,k}\boldsymbol{e}_{j,k})
\end{aligned}$$

Therefore, we obtain a BGV ciphertext which encrypts $m = \sum_i m_i X^i$.

Notice that error of the output ciphertext can be written as $\boldsymbol{e} = \sum_i e_i X^i + \sum_{j,k} \boldsymbol{a}_{j,k}\boldsymbol{e}_{j,k}$. Since $|e_i| \le E$ for $0 \le i < n$, we have $\| \sum_{i=0}^{n-1} e_i X^i \|_\infty \le E$. Using $\|e_{j,k}\|_\infty \le E_{\text{repack}}$, we get that $\| \sum_{j,k} \boldsymbol{a}_{j,k}\boldsymbol{e}_{j,k} \|_\infty \le 2\sqrt{n} dN E_{\text{repack}}$ Therefore, the error of the output is at most $E + 2\sqrt{n} dN E_{\text{repack}}$.

Step 6 of Algorithm 3 dominates the computational complexity of the entire algorithm. In this step, main operations consist of scaler multiplications and additions in $\mathbb{Z}[X]/(\Phi(X))$. Therefore, this algorithm runs in no more than $\tilde{O}(dnN)$.

4.3 Bootstrapping

Combining **HomModRed** described in Section 3.2 with **BlindRotate** and **Repack**, we can obtain an improved bootstrapping procedure for BGV scheme. Let a BGV ciphertext to be bootstrapped be parameterized by a plaintext modulus t, a modulus cyclotomic polynomial $X^n + 1$ with a power of two n, a modulus q_ℓ. The blueprint of our bootstrapping procedure is formalized by Algorithm 4.

Algorithm 4 The **Bootstrap** Algorithm

Input: A BGV ciphertext $\mathbf{BGV}^t_{\boldsymbol{s},q_\ell}(\boldsymbol{m},\boldsymbol{e}) = (\boldsymbol{a},\boldsymbol{b})$;
A bootstrapping key $\mathbf{bk} = \{\mathbf{RGSW}_{2\boldsymbol{z},q_L}(s_i^+), \mathbf{RGSW}_{2\boldsymbol{z},Q}(s_i^-)\}_{0\le i<n}$;
A repacking key $\mathbf{rpk} = \{\mathbf{BGV}^t_{\boldsymbol{s}^{(L)},q_L}(z_j \cdot g_k, \boldsymbol{e}_{j,k})\}_{0\le j<N, 0\le k<d}$;

Output: A level-L BGV-type ciphertext $\mathbf{BGV}^t_{\boldsymbol{s}^{(L)},q_L}(\boldsymbol{m},\tilde{\boldsymbol{e}})$;

1: $\mathbf{BGV}^t_{\boldsymbol{s}^{(\ell)},2N}(\boldsymbol{m},\boldsymbol{e}') \leftarrow \mathbf{ModSwitch}(\mathbf{BGV}^t_{\boldsymbol{s}^{(\ell)},q_\ell}(\boldsymbol{m},\boldsymbol{e}))$;
2: **for** $i = 0$ to $n-1$ **do**
3: $\quad\mathbf{LWE}_{\vec{s},2N}(m_i, te_i) \leftarrow \mathbf{Extract}_i(\mathbf{BGV}^t_{\boldsymbol{s}^{(\ell)},2N}(\boldsymbol{m},\boldsymbol{e}'))$;
4: $\quad\mathbf{RLWE}_{2\boldsymbol{z},q_L}(X^{-\tilde{m}_i}, \boldsymbol{e}_i) \leftarrow \mathbf{BlindRotate}(\mathbf{LWE}_{\vec{s},2N}(m_i, te_i), \mathbf{bk})$;
$\quad//\tilde{m}_i = m_i + te_i$;
5: $\quad\mathbf{LWE}_{\vec{z},q_L}(m_i, e_i') \leftarrow \mathbf{HomModRed}(\mathbf{RLWE}_{2\boldsymbol{z},q_L}(X^{-\tilde{m}_i}, \boldsymbol{e}_i))$;
6: **end for**
7: $(\tilde{\boldsymbol{a}},\tilde{\boldsymbol{b}}) \leftarrow \mathbf{Repack}(\{\mathbf{LWE}_{\vec{z},Q}(m_i, e_i'), \mathbf{rpk}\})$;
8: **return** $(\tilde{\boldsymbol{a}},\tilde{\boldsymbol{b}})$;

In the following, we state two theorems to show the noise growth and computational complexity of the entire bootstrapping procedure based on our new homomorphic modular reduction algorithm. For parameters used in our bootstrapping procedure, we set $t = O(1)$, $N = \Theta(n)$, $q_L = \mathrm{poly}(n)$, $B = O(1)$ and $d = O(\log n)$.

Noise analysis. The noise performance of the entire bootstrapping method is analyzed by the following theorem.

Theorem 2. *Given a bootstrapping key with initial noise E_{bk} and a repacking key with initial noise E_{rpk}, we can bootstrap BGV homomorphic encryption scheme within $\tilde{O}((n^{2.5}E_{\mathrm{bk}} + n^{1.5}E_{\mathrm{rpk}}))$.*

Proof. According to Lemma 4, the output ciphertext of **BlindRotate** has an error bounded by $4n\sqrt{N}dBE_{\mathrm{bk}}$. By Lemma 1, the error of the output ciphertext of **HomModRed** is bounded by $tN/2 \cdot E$, where E is an upper bound of the output of **BlindRotate**. Based on Lemma 5, we know that **Repack** outputs a ciphertext whose error is within $E' + 2\sqrt{n}dNE_{\mathrm{rpk}}$, where E' is an upper bound of the output of **HomModRed**. In conclusion, Algorithm 4 output a ciphertext whose error is within $2nN^{1.5}dBE_{\mathrm{bk}} + 2n^{0.5}dNE_{\mathrm{rpk}} = \tilde{O}(n^{2.5}E_{\mathrm{bk}} + n^{1.5}E_{\mathrm{rpk}})$.

Computational Complexity. Computational complexity of our bootstrapping method is stated by the following theorem.

Theorem 3. *The computation complexity of bootstrapping procedure described in Algorithm 4 is no more than $\tilde{O}(n^3)$.*

Proof. By Lemma 3, 4, 5, we can easily get that the computation complexity of our bootstrapping procedure is no more than $\tilde{O}(nN^2 + dnN^2 + dn^2N) = \tilde{O}(n^3)$.

Asymptotic parameters. In the worst case, to make decryption correct at level-L, we need to ensure that

$$\tilde{O}(n^{2.5}E_{\mathrm{bk}} + n^{1.5}E_{\mathrm{rpk}})^{M+1} \leq \frac{q_L}{2t}$$

Therefore $q_L = \tilde{O}(n^{2.5}E_{\mathrm{bk}} + n^{1.5}E_{\mathrm{rpk}})^{M+1}$, where M can be any positive constant integer that is chosen at first and independent of dimension n and security parameter λ. For instance, if we set $E_{\mathrm{bk}} = E_{\mathrm{rpk}} = \omega(\sqrt{n \log n})$ and $M = 1$, then $q_L = \tilde{O}(n^6)$ suffices. According to Theorem 3, the security of our scheme can be relied on the hardness of $\tilde{O}(n^{5.5})$-approximate SVP. The similar result in [14] is that the security of their scheme relies on $\tilde{O}(n^{6.5})$-approximate SVP. The following table illustrates the comparison results between Chen and Zhang's work and ours. The results show that the computational complexity of our bootstrapping procedure is almost the same with the CZ17 scheme in terms of algorithm complexity, but our scheme is more suitable for implementing than the CZ17 scheme. For noise performance, our bootstrapping scheme has lower noise accumulation than that of [14], hence it allows to use worst-case complexity assumption with lower approximation factors.

Table 1. Comparison with CZ17 scheme.

	Time Complexity	Noise Growth	Hard Problem
Our scheme	$\tilde{O}(n^3)$	$\tilde{O}(n^6)$	$\mathrm{GapSVP}_{\tilde{O}(n^{5.5})}$
CZ17	$\tilde{O}(n^3)$	$\tilde{O}(n^7)$	$\mathrm{GapSVP}_{\tilde{O}(n^{6.5})}$

5 Conclusion

Bootstrapping technique is a pivotal component to construct a fully homomorphic encryption scheme. The computational cost of bootstrapping procedure has a major influence on the performance of the entire FHE scheme. Meanwhile, it is vital for a bootstrapping scheme to incur polynomial noise since it requires worst-case lattice assumptions with polynomial approximation factors. Therefore, it is worthy to study a a better bootstrapping procedure with polynomial noise.

In this paper, we investigate the homomorphic modular reduction algorithm in the procedure of bootstrapping BGV scheme. We use the idea of blind rotation to design a new homomorphic modular reduction algorithm and apply it to bootstrapping BGV scheme. We carry out an analysis of the noise accumulation and the computational cost. The results show that our bootstrapping scheme incurs polynomial noise accumulation and has lower noise growth, therefore our improvement decreases the approximate factor for the underlying worst-case lattice assumption from $\tilde{O}(n^{6.5})$ to $\tilde{O}(n^{5.5})$. Moreover, our new homomorphic modular reduction can be applied independently to other applications where one needs to perform modular reduction homomorphically.

References

1. Alperin-Sheriff J., Peikert C.: Practical bootstrapping in quasilinear time. Advances in Cryptology, Crypto 2013. Lecture Notes in Computer Science **8042**, 1-20, 2013.
2. Alperin-Sheriff J., Peikert C.: Faster bootstrapping with polynomial error. Advances in Cryptology, Crypto 2014. Lecture Notes in Computer Science **8616**, 297-314, 2014.
3. Brakerski Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. Advances in Cryptology, Crypto 2012. Lecture Notes in Computer Science **7417**, 868-886, 2012.
4. Bonnoron G., Ducas L., Fillinger M.: Large FHE gates from tensored homoorphic accumulator. Advances in Cryptology, Africacrypt 2018. Lecture Notes in Computer Science **10831**, 217-251, 2018.
5. Brakerski Z., Gentry C., Vaikuntanathan V.: Fully homomorphic encryption without bootstrapping. Innovations in Theoretical Computer Science, ITCS 2012, 309-325, ACM, 2012.
6. Brakerski Z., Vaikuntanathan V.: Efficient fully Homomorphic encryption from (standard) LWE. IEEE Foundations of Computer Science, FoCS 2011, 97-106, 2011.
7. Biasse J. F., Ruiz L.: FHEW with efficient multibit bootstrapping. Advances in Cryptology, Latincrypt 2015. Lecture Notes in Computer Science **9230**, 119-135, 2015.
8. Chen H., Chillotti I., Song Y.: Improved Bootstrapping for Approximate Homomorphic Encryption. Advances in Cryptology, Eurocrypt 2019. Lecture Notes in Computer Science **11477**, 34-54, 2019.
9. Chillotti I., Gama N., Georgieva M., Izabachène M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. Advances in Cryptology, Asiacrypto 2016. Lecture Notes in Computer Science **10031**, 3-33, 2016.
10. Chillotti I., Gama N., Georgieva M., Izabachène M.: TFHE: Fast fully homomorphic encryption over the torus. Journal of Cryptology, **33**(1), 34–91.
11. Chen H., Han K.: Homomorphic Lower Digits Removal and Improved FHE Bootstrapping. Advances in Cryptology, Eurocrypt 2018. Lecture Notes in Computer Science **10820**, 315-337, 2018.
12. Cheon H., Han K., Kim A., Kim M., Song Y.: Bootstrapping for Approximate Homomorphic Encryption. Advances in Cryptology, Eurocrypt 2018. Lecture Notes in Computer Science **10820**, 360-384, 2018.
13. Cheon J., Kim A., Kim M., Song Y.: Homomorphic encryption for arithmetic of approximate numbers. Advances in Cryptology, Asiacrypt 2017. Lecture Notes in Computer Science **10624**, 409-437, 2017.
14. Chen L., Zhang Z.: Bootstrapping fully homomorphic encryption with ring plaintexts within polynomial noise. Provable Security, ProvSec 2017, 285-304, 2017.
15. van Dijk M., Gentry C., Halevi S., Vaikuntanathan V.: Fully homomorphic encryption over the integers. Advances in Cryptology, Eurocrypt 2010. Lecture Notes in Computer Science **6110**, 24-42, 2010.
16. Doróz Y., Hu Y., Sunar B.: Homomorphic AES evluation using the modified LTV scheme. Des. Codes Crypt. **80**(2), 333-358, 2016.
17. Ducas L., Micciancio D.: FHEW: Bootstrapping homomorphic encryption in less than a second. Advances in Cryptology, Eurocrypt 2015. Lecture Notes in Computer Science **9056**, 617-640, 2015.
18. Gentry C.: A fully homomorphic encryption scheme[D]. Stanford University, 2009.

19. Gentry C.: Fully homomorphic encryption using ideal lattices. ACM Sypmosium on Theory of Computing, STOC 2009, 169-178, 2009.
20. Gentry C., Halevi S., Smart N.P.: Better bootstrapping in fully homomorphic encryption. Topics in Cryptology, PKC 2012. Lecture Notes in Computer Science **7293**, 1-16, 2012.
21. Gentry C., Halevi S., Smart N.P.: Fully homomorphic encryption with polylog overhead. Advances in Cryptology, Eurocrypt 2012. Lecture Notes in Computer Science **7237**, 465-482, 2012.
22. Gentry C., Halevi S., Peikert C., Smart N.P.: Ring switching in BGV-style homomorphic encryption. Security and Cryptography for Networks, SCN 2012. Lecture Notes in Computer Science **7485**, 19-37, 2012.
23. Gama N., Izabachène M., Nguyen P.Q., Xie X.: Structural lattice reduction: Generalized worst-case to average-case reductions and homomorphic cryptosystems. Advances in Cryptology, Eurocrypt 2016. Lecture Notes in Computer Science **9666**, 528-558, 2016.
24. Gentry C., Sahai A., Waters B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. Advances in Cryptology, Crypto 2013. Lecture Notes in Computer Science **8042**, 75-92, 2013.
25. Halevi S., Shoup V.: Algorithms in HElib. Advances in Cryptology, Crypto 2014. Lecture Notes in Computer Science **8616**, 554-571, 2014.
26. Halevi S., Shoup V.: Bootstrapping for HElib. Advances in Cryptology, Eurocrypt 2015. Lecture Notes in Computer Science **9056**, 641-670, 2015.
27. Kim A., Deryabin M., Eom J., et al.: General bootstrapping approach for RLWE-based homomorphic encryption. IACR Cryptol. ePrint Arch. 2021/691.
28. Lyubashevsky V., Peikert C., Regev O.: On ideal lattices and learning with errors over rings. Advances in Cryptology, Eurocrypt 2010. Lecture Notes in Computer Science **6110**, 1-23, 2010.
29. López-Alt A., Tromer E., Vaikuntanathan V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. ACM Symposium on Theory of Computing, STOC 2012, 1219-1234, 2012.
30. Micciancio, D., Polyakov, Y.:Bootstrapping in FHEW-like Cryptosystems. IACR Cryptol. ePrint Arch., 2020/86
31. Micciancio D., Sorrell J.: Ring packing and amortized FHEW bootstrapping. International Colloquium on Automata, Languages, and Programming, ICALP 2018, **107**, 100:1-100:14, 2018.
32. Orsini E., van de Pol J., Smart N.P.: Bootstrapping BGV ciphertexts with a Wider Choice of p and q. Topics in Cryptology, PKC 2013, 673-398, 2015.
33. Regev O.: On lattices, learning with errors, random linear codes, and cryptography. ACM Symposium on Theory of Computing, STOC 2005, 84-93, 2005.
34. Smart N.P., Vercauteren F.: Fully homomorphic SIMD operations. Des. Codes Cryptogr. **71**(1), 57-81, 2014.