IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2024, No. 1, pp. 563–591. DOI:10.46586/tches.v2024.i1.563-591

Faster Bootstrapping via Modulus Raising and Composite NTT

Zhihao Li[1,2], Ying Liu[1,2], Xianhui Lu[1,2](✉), Ruida Wang[1,2], Benqiang Wei[1,2], Chunling Chen[1,2] and Kunpeng Wang[1,2]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{lizhihao,luxianhui}@iie.ac.cn

Abstract. FHEW-like schemes utilize exact gadget decomposition to reduce error growth and ensure that the bootstrapping incurs only polynomial error growth. However, the exact gadget decomposition method requires higher computation complexity and larger memory storage. In this paper, we improve the efficiency of the FHEW-like schemes by utilizing the composite NTT that performs the Number Theoretic Transform (NTT) with a composite modulus. Specifically, based on the composite NTT, we integrate modulus raising and gadget decomposition in the external product, which reduces the number of NTTs required in the blind rotation from $2(d_g+1)n$ to $2(\lceil d_g/2\rceil+1)n$. Furthermore, we develop a modulus packing technique that uses the Chinese Remainder Theorem (CRT) and composite NTT to bootstrap multiple LWE ciphertexts through one blind rotation process.
We implement the bootstrapping algorithms and evaluate the performance on various benchmark computations using both binary and ternary secret keys. Our results of the single bootstrapping process indicate that the proposed approach achieves speedups of up to $1.7\times$, and reduces the size of the blind rotation key by 50% under specific parameters. Finally, we instantiate two ciphertexts in the packing procedure, and the experimental results show that our technique is around $1.5\times$ faster than the two bootstrapping processes under the 127-bit security level.

Keywords: Gadget decomposition · Composite NTT · External Product · Modulus Raising · Packing

1 Introduction

Homomorphic encryption (HE) is a prospective cryptographic primitive that performs arbitrary computation on ciphertexts without access to the secret key. Due to its confidentiality, HE schemes have emerged as a core technology for applications such as privacy-preserving cloud computations [MSM+22]. The first fully homomorphic encryption scheme was proposed by Gentry [Gen09] in 2009, and since then, the field has seen significant progress. The common FHE schemes are typically divided into three classes based on the data types: BGV and BFV schemes for modular arithmetic over finite fields, which are usually used for small integer computations [BGV14, Bra12, FV12]; CKKS scheme for approximate computations over real and complex numbers [CKKS17, CHK+18]; FHEW and TFHE schemes for evaluating boolean circuits, which are well-suited for comparisons and decision diagram computations [DM15, CGGI16].

Nowadays, these homomorphic encryption schemes are based on the (Ring) Learning With Errors assumptions, where a small amount of error (noise) is introduced to the encrypted message in the encryption process. However, the error can accumulate during

Received: 2023-07-15 Accepted: 2023-09-15 Published: 2023-12-04

circuit evaluations and even corrupt the plaintext if it exceeds a certain threshold. As a result, managing noise effectively has become a central concern in the design of FHE schemes. At present, there exist two popular methods for reducing error in FHE schemes. The first method is gadget decomposition, including digit decomposition and RNS decomposition, in which an element is broken down into smaller digits. Although this approach can reduce noise, it suffers from efficiency bottlenecks due to its quadratic growth. Modulus raising, introduced by Gentry et al. [GHS12], offers a more efficient alternative compared to gadget decomposition. However, a larger ciphertext modulus is required in the HE cryptosystem, which may reduce the security level of the scheme. Thus, some works [KPZ21a, CCH$^+$22] explore a hybrid method that combines the advantages of both gadget decomposition and modulus switching to achieve a balance between noise control and security level.

Furthermore, choosing an appropriate ciphertext modulus is also crucial in FHE schemes since it determines the upper bound on the noise level that the scheme can tolerate during the computation. A sufficiently large ciphertext modulus must be chosen in advance to account for noise growth during calculation, or a bootstrapping procedure must be used to reset the noise and keeps the modulus within a reasonable range. FHEW [DM15, MP21] and TFHE [CGGI16, CGGI20] schemes typically use the latter strategy and are known for efficient bootstrapping. Their efficiency is mainly due to the small ciphertext modulus, which allows for the use of CPU native types to represent ciphertexts in both the coefficient representation and Discrete Fourier Transform (DFT) representation. Specifically, the two schemes follow the same bootstrapping framework, which involves the homomorphic decryption of an LWE ciphertext through a blind rotation procedure. However, there are some differences in terms of their underlying algebraic structures and implementation details.

The TFHE scheme typically relies on the real torus $\mathbb{T}$, which is the set of real numbers modulo 1, represented as the interval $[0, 1)$. In practice, torus elements are not represented with an infinite number of digits but instead approximated to a finite precision. At the level of implementation of the algorithm, the TFHE scheme usually utilizes 32 or 64 bits data to represent the ciphertext modulus in TFHE-lib [CGGI20] and TFHE-rs [BSJJ22], which offers the advantage of performing modulo operations for free based on the data type. However, this set can only utilize the Fast Fourier Transform (FFT) to accelerate polynomial multiplication. On the other hand, the choice of ciphertext modulus in the FHEW-like schemes is more flexible. Typically, the modulus can be set to a prime number, which enables it to perform more homomorphic operations than powers-of-two. For instance, the trace operation [CDKS21] requires the polynomial dimension and the modulus to be coprime. In scenarios like this case, the NTT outperforms the FFT in terms of efficiency.

These original FHEW and TFHE schemes focus on bootstrapping single LWE ciphertext. Micciancio et al. [MS18] proposed a novel refreshing procedure that can simultaneously refresh multiple LWE ciphertexts, which makes it more suitable for practical applications. Building upon this, subsequent works, such as [LW23a] and [LW23b], have significantly improved the asymptotic cost per gate bootstrap to homomorphic multiplications. It is worth noting that these schemes heavily rely on certain algebraic structures, and therefore, these works are all designed based on FHEW-like schemes.

1.1 Contributions and Techniques

In this paper, we focus on the FHEW-like bootstrapping and use composite NTT to optimize and improve the blind rotation procedure.

Composite NTT. The methods for performing NTT using composite moduli can be categorized into different strategies. we analyze these mathematical principles for these methods and investigate the computational environments and tasks associated with HE in which they are best suited. We provide a proof for the approach [HP22] that constructs the root for the composite NTT, and extend this method to encompass multiple moduli, enhancing its versatility and applicability.

Reduce the number of NTTs in blind rotation. Based on the composite NTT, we present two more flexible and improved variants of the external product. In the first method, the RGSW ciphertext is represented as $\mathsf{RLWE}_{PQ}(Psm)$ and $\mathsf{RLWE}_{PQ}(Pm)$ ciphertexts, where s is the secret key, m is the massage, and P is a temporary modulus. The new external product between RLWE and RGSW ciphertexts is

$$\left\lfloor \frac{a \cdot \mathsf{RLWE}_{PQ}(Psm) + b \cdot \mathsf{RLWE}_{PQ}(Pm)}{P} \right\rceil ,$$

where the division is performed to reduce error growth. We also integrate the modulus raising, digit decomposition, and RNS decomposition in the hybrid method, which can further reduce noise by the decomposition. Compared to the $2(d_g+1)n$ of NTTs required for gadget decomposition, our methods only involve $2(\lceil d_g/2\rceil+1)n$ in GINX-based [CGGI16] blind rotation. Furthermore, we show that the proposed techniques can accelerate the blind rotation based on the AP [DM15], and LMK [LMK+23] methods in FHEW-like schemes.

Composite NTT-based packing bootstrapping. We introduce a novel packing bootstrapping algorithm for FHEW-like schemes. In particular, we can use CRT to pack some independent accumulators into one large composite modulus. By performing the composite NTT, these accumulators only need to perform one external product operation with a large modulus in each CMux gate. We remark that this technique gains from the application of composite NTT and does not apply the FFT-based TFHE scheme. Our approach offers adaptable deployment capabilities for platforms that implement various machine word lengths, especially hardware-accelerated architectures (i.e., the state-of-the-art ASIC accelerator SHARP [KKC+23]). Finally, we implement the above methods and provide some comparisons and analyses.

Table 1: Comparison of GINX blind rotation with different external products, where GD is the gadget decomposition, MR is the modulus raising, and HY is the hybrid method. In operation counts, NTT is the Number Theoretic Transforms, HM is the Hadamard Multiplication, and GDP and DRP are the gadget decomposition and division and rounding for polynomials, respectively, where $d_g, d'_g (d_g > d'_g)$ are the length of the gadget decomposition.

Methods		# Operations
GD	Binary	$2(d_g+1)n$ NTTs + $2n$ GDPs + $(4d_g+2)n$ HMs
	Ternary	$2(d_g+1)n$ NTTs + $2n$ GDPs + $(8d_g+4)n$ HMs
MR	Binary	$4n$ NTTs + $2n$ DRPs + $6n$ HMs
	Ternary	$4n$ NTTs + $2n$ DRPs + $12n$ HMs
HY	Binary	$2(d'_g+1)n$ NTTs + $2n$ GDPs + $2n$ DRPs + $(4d'_g+2)n$ HMs
	Ternary	$2(d'_g+1)n$ NTTs + $2n$ GDPs + $2n$ DRPs + $(8d'_g+4)n$ HMs

Performance Comparison and Implementation.

- We comprehensively analyze the proposed bootstrapping algorithm under different parameters, including the variance of noise growth, computational complexity, and decryption failure rate. We summarize the number of operations required in the blind rotation process for different methods in Table 1.

- Compared to exact decomposition in FHEW-likes schemes, our bootstrapping algorithm reduces the key sizes by 50% for blind rotation and achieves a speedup of up to 1.7 $\times$ compared to gadget decomposition.
- We implement the packing bootstrapping algorithm that bootstraps two LWE ciphertexts within 64 bits of CPU machine word lengths. The result shows that the proposed method is 1.5 $\times$ faster than the two bootstrapping processes.

1.2 Related Work

Number Theoretic Transform (NTT), Fast Fourier Transform (FFT), and Toom–Cook multiplication can be used to efficiently perform polynomial multiplication. The NTT algorithm needs to satisfy that $Q \equiv 1 \pmod{2N}$, which guarantees the existence of the $2N$-th primitive root of unity. Regarding other NTT-unfriendly rings, Chung et al. [CHK$^+$21] propose that one can lift the polynomial ring to a larger NTT-friendly ring that covers all results without modular reduction. Compared to Toom–Cook multiplication, this method can improve the efficiency for unprotected Saber implementations on the Cortex-M4. After that, Abdulrahman et al. [ACC$^+$21] note that the implementation for [CHK$^+$21] has a large memory footprint. The method [ACC$^+$21] utilized multi-moduli NTTs to enable a very stack-efficient implementation competitive in memory usage. In terms of composite NTT, Heinz et al. [HP22] also propose a method to construct the root for the composite NTT. However, their work lacks conclusive proof and does not consider the scenario of multiple moduli.

Despite the advantages of NTT and FFT, it is incompatible with the noise control methodology since modulus switching is involved in gadget decomposition and modulus raising. Therefore, frequent switching of data form between the coefficient representation and DFT representation is required during the homomorphic encryption algorithm, which also leads to the consumption of a large number of computational resources [JLK$^+$21]. Recently, Kim et al. [KLSS23] accelerated the key-switching in the Full-RNS setting for the CKKS scheme, which introduces a second gadget decomposition to reduce NTT computations. This work focuses on the multi-precision modulus, and the benefits do not appear to be evident in the RNS-based FHEW-like schemes.

The blind rotation procedure in FHEW and TFHE bootstrapping can homomorphically compute the RLWE ciphertext of $X^{\sum_{i=0}^{n} a_i s_i}$, which needs massive NTTs or FFTs, and Hadamard multiplications. Currently, there are three strategies for performing blind rotation: the AP method [ASP14, DM15] using a_i as a selector to pick all the evaluation keys that encrypt $E(a \cdot s_i)$, and these are accumulated by the external product; and the GINX method [CGGI16] that homomorphically performs CMux gate, which is more effective for binary and ternary secret key distributions; and the LMK method [LMK$^+$23] that uses the ring automorphisms and RLWE-based key switching technique to support the arbitrary distribution of secret keys, which is subsequently used by [DMKMS23] to support packaging bootstrapping.

1.3 Paper Organization

The rest of the paper is organized as follows. We provide the necessary background knowledge and some general tools in FHE schemes in Section 2. In Section 3, we present some methods and comparisons for performing polynomial multiplication with composite modulus. In Section 4, we show the new bootstrapping algorithm with the composite NTT technique. In Section 5, we suggest some analysis of noise growth and details of the algorithm execution and experimental results. Finally, we conclude the paper in Section 6.

2 Background

2.1 Notation

We denote as $\mathbb{Z}$ the set of integers, $\mathbb{R}$ as the set of reals. We use lower-case bold letters for vectors and upper-case bold letters for matrices. $\langle \mathbf{a}, \mathbf{b} \rangle$ is the inner product between two vectors. We denote $\mathbb{Z}_Q$ the ring $\mathbb{Z}/Q\mathbb{Z}$, and the scope of $\mathbb{Z}_Q$ is $[-Q/2, Q/2) \cap \mathbb{Z}$. We denote $\mathbb{Z}_Q^*$ as the residue ring modulo Q, and the centered remainder of x modulo Q as $[x]_Q$. For a real number r, we write the floor, ceiling, and round functions as $\lfloor r \rfloor$ $\lceil r \rceil$ $\lfloor r \rceil$, respectively. For a set of k co-prime moduli $Q_1, ..., Q_k$, we denote $Q_i^* = Q/Q_i \in \mathbb{Z}$ and $\widetilde{Q}_i = Q_i^{*-1} \pmod{Q_i} \in \mathbb{Z}_{Q_i}$.

Furthermore, we denote the $2N$-th cyclotomic ring by $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ and the quotient ring by $\mathcal{R}_Q = \mathbb{Z}_Q[X]/(X^N + 1)$ with coefficients in $\mathbb{Z}_Q$, where N is a power of 2. For a polynomial $m(X) = m_0 + m_1 X + \cdots + m_{N-1} X^{N-1} \in \mathcal{R}$, its coefficients are represented by the vector $\mathsf{Coefs}(m) = (m_0, m_1, ..., m_{N-1})$. In our notation, sometimes a polynomial is denoted by $a(X)$, and sometimes it is denoted by a. The multiplication operations are indicated by the $\cdot$ and $\odot$ symbols, where the former is used for number and polynomial multiplication, while the latter is used for Hadamard multiplication. We use $x \leftarrow D$ to denote the sampling of x according to distribution D. We denote $\mathsf{Var}(\mathsf{err}(\mathsf{ct}))$ as the variance of error for the ciphertext ct. Fianlly, we denote $\|\mathbf{a}\|_p = \left(\sum_{i=1}^{n} |a_i|^p\right)^{1/p}$ the p -norm of a vector $\mathbf{a} \in \mathbb{Z}^n$ and compute the p-norm with a polynomial by taking its coefficient vector.

2.2 Gadget Decomposition

Gadget Decomposition. Gadget decomposition includes digit decomposition and RNS decomposition. The digit decomposition can break a number down into individual digits using a radix base. Let $a \in \mathcal{R}_Q$ and $d_g = \left\lfloor \log_{B_g} Q \right\rceil + 1$. The decomposition function $\mathbf{g}_d^{-1}$ and the expansion function $\mathbf{g}$ with the radix base B_g are:

$$\mathbf{g}_d^{-1}(a) = \left([a]_{B_g}, \left[\left\lfloor \frac{a}{B_g} \right\rceil \right]_{B_g}, \cdots, \left[\left\lfloor \frac{a}{B_g^{d_g - 1}} \right\rceil \right]_{B_g} \right) \in \mathcal{R}_{B_g}^{d_g},$$

$$\mathbf{g}_d = \left(B_g^0, B_g, \cdots, B_g^{d_g - 1} \right) \in \mathcal{R}_Q^{d_g}.$$

Thus, we can get $\langle \mathbf{g}_d^{-1}(a), \mathbf{g} \rangle \equiv a \mod Q$. Furthermore, Residual Number System (RNS) decomposition is another gadget decomposition technique, and the details will be described in the following section.

2.3 Gaussian Distribution

The Gaussian function is defined as a distribution over $\mathbb{Z}$ and each element in $\mathbb{Z}$ is sampled with probability proportional to its probability mass function value under a Gaussian distribution over $\mathbb{R}$. The Gaussian function is

$$\rho_{\sigma,c}(x) = \exp\left(-\frac{|x - c|^2}{2 \cdot \sigma^2} \right),$$

where $\sigma, c \in \mathbb{R} \geq 0$ and then

$$\rho_{\sigma,c}(\mathbb{Z}) = \sum_{i=-\infty}^{\infty} \rho_{\sigma,c}(i).$$

The discrete Gaussian distribution with standard deviation σ and mean c is a distribution on $\mathbb{Z}$ with the probability of $x \in \mathbb{Z}$ given by $\mathcal{D}_{\delta,c} = \rho_{\sigma,c}(x)/\rho_{\sigma,c}(\mathbb{Z})$. If $c = 0$, we denote this distribution by χ_δ.

2.4 Learning With Errors

We recall the learning with errors (LWE) assumption[Reg09] and Ring learning with errors (RLWE) assumption [LPR13]as follows.

- **LWE Sample.** A valid LWE sample is a vector $(\mathbf{a}, b) \in \mathbb{Z}_q^{n+1}$ that satisfies $b = \langle \mathbf{a}, \mathbf{s} \rangle + e \bmod q$, where $\mathbf{s}$ is the secret key for LWE sample, $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ is a uniformly random vector, and error $e \leftarrow \chi_\delta$ is chosen from an error distribution. Then, $(\mathbf{a}, b)$ is a fresh ciphertext of 0.

- **RLWE Sample.** A valid RLWE sample is a pair $(a, b) \in \mathcal{R}_Q^2$ that satisfies $b = a \cdot s + e \bmod Q$, where s is the secret key for RLWE sample, a is uniformly random in $\mathcal{R}_Q$, and the error $e \leftarrow \chi_\delta^N$ is chosen from the error distribution. Then, (a, b) is a fresh ciphertext of 0. Thus, we can define the LWE and RLWE ciphertexts as

  $$\mathsf{LWE}_{\mathbf{s},q}^n(m) = (\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + \left\lfloor \frac{q}{t} \cdot m \right\rceil + e) \in \mathbb{Z}_q^{n+1},$$
  $$\mathsf{RLWE}_{s,Q}^N(m) = (a, b = a \cdot s + \left\lfloor \frac{Q}{t} \cdot m \right\rceil + e) \in \mathcal{R}_Q^2.$$

  Sometimes, the dimension N and secret key s may be omitted for the sake of simplicity. The message m can be recovered if satisfying $e < \frac{q}{2t}$ by the (R)LWE decryption process

$$\mathsf{LWE}_{\mathbf{s},q}^{-1}(\mathbf{a}, b) = \left\lfloor \frac{t}{q} \cdot [b - \langle \mathbf{a}, \mathbf{s} \rangle]_q \right\rceil_t, \text{ and } \mathsf{RLWE}_{s,Q}^{-1}(a, b) = \left\lfloor \frac{t}{Q} \cdot [b - a \cdot s]_Q \right\rceil_t.$$

Arithmetic The structure of ciphertexts in LWE and RLWE allows for homomorphic addition and scalar multiplication operations. For instance, given the LWE samples $\mathsf{ct}_1 = (\mathbf{a_1}, b_1)$ and $\mathsf{ct}_2 = (\mathbf{a_2}, b_2)$, their terms can be added together to obtain: $\mathsf{ct}_1 + \mathsf{ct}_2 = (\mathbf{a_1} + \mathbf{a_2}, b_1 + b_2)$. Moreover, the multiplication between a ciphertext $\mathsf{ct}_1 = (\mathbf{a_1}, b_1)$ and a scalar cleartext z can be obtained directly from the addition operation: $z \cdot \mathsf{ct}_1 = (z \cdot \mathbf{a_1}, z \cdot b_1)$.

2.5 Original RGSW Ciphertext and External Product

The original RGSW cryptosystem [GSW13, DM15] involves some RLWE samples and gadget matrix $\mathbf{G}$, which can be denoted by $I_2 \otimes \mathbf{g}^T$, where $\mathbf{g} = (B_g^0, B_g, \cdots, B_g^{d_g - 1})$. To construct the RGSW ciphertext, we sample $2d_g$ RLWE ciphertexts $\{\mathsf{ct}_0, \cdots, \mathsf{ct}_{2d_g - 1}\} \in \mathsf{RLWE}_{s,Q}^N(0)$. Then the ciphertext CT of m is of the form:

$$\mathsf{RGSW}_{s,Q}(m) = \begin{pmatrix} \mathsf{ct}_0 \\ \vdots \\ \mathsf{ct}_{2d_g-1} \end{pmatrix} + m \cdot \mathbf{G} \in \mathcal{R}_Q^{2d_g \times 2}.$$

External Product We define the original external product as $\boxdot_D$ that involve the digit decomposition. Given the RLWE and RGSW ciphertexts, the external product outputs a new RLWE ciphertext as

$$\begin{aligned} \mathsf{ct}' &= \mathsf{RLWE}_{s,Q}(\mu) \boxdot_D \mathsf{RGSW}_{s,Q}(m) \\ &= \mathbf{g}_d^{-1}(\mathsf{RLWE}_{s,Q}(\mu)) \cdot \mathsf{RGSW}_{s,Q}(0) + m \cdot \mathsf{RLWE}_{s,Q}(\mu) \\ &= \mathsf{RLWE}_{s,Q}(0) + \mathsf{RLWE}_{s,Q}(m \cdot \mu) \\ &= \mathsf{RLWE}_{s,Q}(m \cdot \mu). \end{aligned}$$

Error analysis. The noise in ct' is given by $e' = \sum_{i=0}^{d_g-1}(a_i \cdot e_i + b_i \cdot e_{i+l}) + m \cdot e$, where $e_0, ..., e_{2d_g-1}$ are the noise terms of the RGSW ciphertext and e is the noise term of the RLWE ciphertext. In bootstrapping, $m \in \pm X^k$ is used as messages in the RGSW ciphertext. Thus, we can get the variance of e' is

$$\mathsf{Var}(\mathsf{err}(\mathsf{ct}')) \leq \frac{N d_g B_g^2}{6} \cdot \mathsf{Var}(\mathsf{err}(\mathsf{CT})) + \mathsf{Var}(\mathsf{err}(\mathsf{ct})) \tag{1}$$

Note that we define two additional forms of RGSW ciphertext in Section 4, and distinguish them based on their modulus.

2.6 NTT-based Multiplication

The Number Theoretic Transform (NTT) is a variation of the Discrete Fourier Transform (DFT) over the finite field. The NTT algorithm can convert a polynomial from its coefficient representation to the NTT representation, enabling Hadamard multiplication and significantly reducing the computation complexity from $O(N^2)$ to $O(N \log N)$.

To achieve NTT-based multiplication, the polynomial ring $\mathbb{Z}_Q[X]/(f(X)g(X))$ is mapped into several rings of the lower order, i.e, $\mathbb{Z}_Q[X]/(f(X)) \times \mathbb{Z}_Q[X]/(g(X))$, where $f(X)$ and $g(X)$ are coprime. If the prime modulus satisfies the condition $Q \equiv 1 \pmod{2N}$, then there exists a $2N$-th primitive root of unity ζ in $\mathbb{Z}_Q$ that satisfies $X^N + 1 = X^N - \zeta^N \pmod{Q}$. Thus, we can get:

$$\mathbb{Z}_Q[X]/(X^N + 1) \to \mathbb{Z}_Q[X]/(X^{\frac{N}{2}} + \zeta^{\frac{N}{2}}) \times \mathbb{Z}_Q[X]/(X^{\frac{N}{2}} - \zeta^{\frac{N}{2}}).$$

Due to the symmetric property of ζ, the polynomial can be further decomposed into N polynomials of 1 degree, i.e.,

$$\mathbb{Z}_Q[X]/(X^N + 1) \to \mathbb{Z}_Q[X]/(X - \zeta) \times \mathbb{Z}_Q[X]/(X - \zeta^3) \times \cdots \times \mathbb{Z}_Q[X]/(X - \zeta^{2N-1}).$$

Thus, for a polynomial $a(X) \in \mathcal{R}_Q$, we can obtain the length-N vector using the CRT

$$(a(X) \bmod (X - \zeta), a(X) \bmod (X - \zeta^3), \cdots, a(X) \bmod (X - \zeta^{2N-1})).$$

Depending on this decomposition, we can define the NTT representation as $\mathrm{NTT}(a) = (A_0, \cdots, A_{N-1})$, where

$$A_i = \sum_{j=0}^{N-1} \mathsf{Coefs}(a)_j \cdot \zeta^{j(2i+1)} (\bmod\ Q) \text{ for } i = 0, 1, \cdots, N-1.$$

The iNTT process is symmetric and omitted. Then, for the multiplication of two polynomials $c(X) = a(X) \cdot b(X) \in \mathcal{R}_Q$, we can compute the process as follows

$$\mathsf{Coefs}(c) = \mathrm{iNTT}(\mathrm{NTT}(a) \odot \mathrm{NTT}(b)).$$

The detailed NTT and iNTT algorithms are described in Appendix A.

2.7 Useful Algorithms

2.7.1 Sample Extraction

We show that the sample extraction technique [CGGI16] can extract the LWE ciphertext for the constant term of the polynomial. Given an RLWE cipheretext $\mathsf{ct} = (a, b) \in \mathsf{RLWE}^N_{s,Q}(m)$, it returns an LWE sample as

$$\mathsf{SampleExtraction}(\mathsf{ct}) = (a_0, -a_{N-1}, -a_{N-2}, ..., -a_1, b_0) \in \mathsf{LWE}^N_{\mathsf{Coefs}(s),Q}(m_0).$$

2.7.2 Key Switching

Key switching procedure [DM15] is an important technique in FHE schemes, which can change the LWE dimension without changing the message. The procedure is described as follows.

• The key switching key generation algorithm takes secret keys $z \in \mathbb{Z}^N$, $s \in \mathbb{Z}^n$ and a base B_k as input, outputs $\mathsf{ksk}_{i,j,v} \in \mathsf{LWE}^n_{\mathbf{s},Q_k}\left(vz_iB_k^j\right)$, where $v \in \{0,\dots,B_k-1\}$, for all $0 \le i \le N-1$, $0 \le j \le d_k-1$, and let $d_k = \lceil \log_{B_k} Q_k \rceil$.

• Given the key switching key $\mathsf{ksk}_{i,j}$ and a ciphertext $\mathsf{ct} = (\mathbf{a}, b) \in \mathsf{LWE}^N_{\mathbf{z},Q_k}(m)$, the key switching procedure computes the base B_k expansion of each coefficient $a_i = \sum_j a_{i,j}B_k^j$, and outputs

$$\begin{aligned}\mathsf{ct}' &= \mathsf{KeySwitch}(\mathsf{ct})\\ &= (\mathbf{0}, b) - \sum_{i,j} \mathsf{ksk}_{i,j,a_{i,j}} \bmod Q_k\\ &= (\mathbf{a}', b') \in \mathsf{LWE}^n_{\mathbf{s},Q_k}(m).\end{aligned}$$

Correctness. Let $\mathsf{ksk}_{i,j,v} = (\mathbf{a}'_{i,j,v}, \mathbf{a}'_{i,j,v}\cdot\mathbf{s} + vz_iB_{ks}^j + e_{i,j,v})$ for some $\mathbf{a}'_{i,j,v} \in \mathbb{Z}_q^n$ and $e_{i,j,v} \in \chi_\delta$. We can obtain that $\mathbf{a}' = -\sum_{i,j}\mathbf{a}'_{i,j,a_{i,j}}$ and $b' = b - \mathbf{a}\cdot\mathbf{z} + \mathbf{a}'\cdot\mathbf{s} - \sum_{i,j} e_{i,j,a_{i,j}}$. which is a new LWE ciphertext under the secret key $\mathbf{s}$. And the variance of the noise satisfies $\mathsf{Var}(\mathsf{err}(\mathsf{ct}')) \le Nd_k \cdot \mathsf{Var}(\mathsf{err}(\mathsf{ksk})) + \mathsf{Var}(\mathsf{err}(\mathsf{ct}))$.

2.7.3 Modulus Switching

The modulus switching technique can change the modulus of the ciphertext [BGV14, DM15]. Take as input a ciphertext $\mathsf{ct} = (\mathbf{a}, b) \in \mathsf{LWE}^n_{\mathbf{s},Q}(m)$, the modulus switching algorithm outputs a ciphertext as

$$\begin{aligned}\mathsf{ct}' &= \mathsf{ModSwitch}(\mathsf{ct})\\ &= (\lfloor \frac{q}{Q}\cdot\mathbf{a}\rceil, \lfloor \frac{q}{Q}\cdot b\rceil) \in \mathsf{LWE}^n_{\mathbf{s},q}(m).\end{aligned}$$

According to [DM15], the variance of noise satisfies $\mathsf{Var}(\mathsf{err}(\mathsf{ct}')) \le (\frac{q}{Q})^2 \cdot \mathsf{Var}(\mathsf{err}(\mathsf{ct})) + \frac{||\mathbf{s}||_2^2+1}{12}$. The correctness of the algorithm is given in Appendix B.

2.8 GINX Blind Rotation with Gadget Decomposition

Algorithm 1 GINX Blind Rotation with Gadget Decomposition.

```
Input:
    An LWE sample $\mathsf{ct} = (\mathbf{a}, b) \in \mathsf{LWE}^n_{\mathbf{s},q}(m)$, where $q|2N$.
    A bootstrapping key $\mathsf{bsk}_{s'}(s_i) : \{\mathsf{RGSW}_{s',Q}(s_i)\}$, for $0 \le i \le n-1$.
Output:
    An RLWE ciphertext $\mathsf{acc} \in \mathsf{RLWE}_{s',Q}(X^{-b+\sum_{i=0}^{n-1} a_is_i})$ .
1: Set $\mathsf{acc} = (0, X^{-b}) \in \mathcal{R}_Q^2$
2: for $i = 0$ to $n-1$ do
3:         $\mathsf{acc}_i = \mathbf{g}^{-1}(\mathsf{acc}_i)$
4:         $\mathsf{acc}'_i = \mathrm{NTT}(\mathsf{acc}_i)$
5:         $\mathsf{acc}'_i = ((X^{a_i}-1)\odot\mathsf{acc}'_i)\odot\mathsf{bsk}_i + \mathsf{acc}'_i \in \mathcal{R}_Q^2$
6:         $\mathsf{acc}_i = \mathrm{iNTT}(\mathsf{acc}_i)$
7: end for
8: return $\mathsf{acc}_{n-1}$.
```

We first present the GINX blind rotation algorithm with gadget decomposition. Given an LWE ciphertext $(\mathbf{a}, b)$ and n RGSW ciphertexts encrypting $(s_0, ..., s_{n-1})$, the blind rotation outputs a ciphertext $\mathsf{ct} \in \mathsf{RLWE}(X^{-b+\sum_{i=0}^{n-1} a_i s_i})$ as shown in Algorithm 1. In detail, the loop from lines 3-6 performs a CMux gate. It is easy to see that if $s_i = 0$, the first term of the supplement is disregarded since it encrypts 0. On the other hand, if $s_i = 1$, then $(\mathsf{acc} \cdot X^{a_i}) \boxdot \mathsf{RGSW}(1)$ equals the current accumulator value. Thus, the accumulator is replaced with the ciphertext of $X^{a_i s_i} \cdot \mathsf{acc}$. Furthermore, the CMux gate can be updated to the ternary CMux gate as shown in Section 4.

3 NTT Multiplication with Composite Modulus

In this section, we present an overview of the existing approaches employed in performing Number Theoretic Transform (NTT) with composite numbers, which include our construction as well. Furthermore, we undertake an extensive investigation into the theoretical principles behind these techniques and their suitability in diverse Homomorphic Encryption (HE) scenarios.

3.1 RNS Decomposition

A well-known technique is the Residue Number System (RNS), which uses the Chinese Remainder Theorem (CRT) to decompose multi-precision integers into vectors of NTT-friendly integers. It enables efficient operations using native (64-bit) integer types and reduces both the theoretical and practical computational overhead. More formally, for some distinct NTT-friendly moduli $Q_1, ..., Q_k$, the CRT yields an isomorphism

$$\mathbb{Z}_Q[X]/(X^N + 1) \to \mathbb{Z}_{Q_1}[X]/(X^N + 1) \times \cdots \times \mathbb{Z}_{Q_k}[X]/(X^N + 1),$$

where $Q = Q_1 \times \cdots \times Q_k$. The RNS representation of an element $a \in \mathcal{R}_Q$ relative to the RNS basis $Q_1, ..., Q_k$ is

$$\mathsf{CRT}_{\{Q_1,...,Q_k\}} = ([a]_{Q_1}, ..., [a]_{Q_k}) \in \mathcal{R}_{Q_i}^k.$$

Then one can perform the NTTs on the $[a]_{Q_i}$ with coefficient-wise over the cyclotomic rings. Similar to the digit decomposition, the corresponding RNS vector is

$$\mathbf{g}_r = \left([\widetilde{Q}_1 \cdot Q_1^*]_Q, \cdots, [\widetilde{Q}_k \cdot Q_k^*]_Q\right) \in \mathcal{R}_Q^k.$$

Similarly to the digit decomposition, we can obtain that $\langle \mathsf{CRT}(a), \mathbf{g}_r \rangle \equiv a \mod Q$. Furthermore, we denote the iCRT as

$$\mathsf{iCRT}_{\{Q_1,...,Q_k\}}([a]_{Q_1}, ..., [a]_{Q_k}) = \sum_{i=1}^{k} [a]_{Q_i} \cdot \widetilde{Q}_i \cdot Q_i^* \pmod{Q},$$

Note that some homomorphic operations, such as homomorphic multiplication in the CKKS scheme, need to switch this RNS basis to another RNS basis $P = P_1 \times \cdots \times P_l$ in the so-called fast basis extension technique that involves the iCRT process. Please refer to [CHK+19, KPZ21b] for more details.

3.2 NTT-unfriendly Rings

NTT-unfriendly rings mean that the parameters do not meet the requirements of section 2.6. For example, the NIST PQC finalist Saber [DKRV19] utilizes the power-of-two modulus, which is inherently incompatible with the NTT algorithm. Chung et al. [CHK+21] present a technique to implement NTT on these rings, yielding better performance than the original schemes.

Specifically, the main idea entails elevating the polynomial ring to a larger one, where the modulus can cover the intermediate results of polynomial multiplication. Then, the NTT algorithm can be performed correctly on this ring directly. Before lifting, one should think about the maximum value of the product. When considering a modulus Q, the magnitude of the coefficients resulting from the multiplication within the ring $\mathcal{R}_Q$ should not surpass $\frac{NQ^2}{4}$. Thus, one can choose an NTT-friendly prime modulus $Q' > \frac{NQ^2}{2}$ or multiple coprime NTT-friendly prime moduli p_i that satisfy $\prod p_i > \frac{NQ^2}{2}$. When using these moduli, the coefficients of the product will not be reduced during the polynomial multiplication, which guarantees the correctness of NTT-based multiplication in the larger ring. The subsequent processing can be summarized in the following three steps:

1. Lift polynomial coefficients on NTT-unfriendly Ring to one or multiple NTT-friendly Rings.
2. Perform NTT-based polynomial multiplications on the new NTT-friendly Rings.
3. Map the results back to the original NTT-unfriendly Ring by using either a modulo operation or inverse Chinese Remainder Theorem (iCRT).

Moreover, the methods of mixed-radix decomposition and Good's permutation are proposed by [CHK+21] to deal with the case that polynomial dimension N does not satisfy the parameter requirements for NTT evaluation. Since we focus on the modulus, these details have been omitted.

3.3 NTT with Composite Modulus

In contrast to the aforementioned approaches, we delve into the mathematical essence of the NTT algorithm and explore the construction of a $2N$-th primitive root of unity for the composite number $Q = Q_1 \times \cdots \times Q_k$, where Q_i are the distinct NTT-friendly numbers.

The NTT requirement that $Q \equiv 1 \pmod{2N}$ ensures the presence of the $2N$-th primitive root of unity in $\mathbb{Z}_Q$. Consequently, the set $r^1, r^2, ..., r^{Q-1}$ forms a cyclic group denoted as $\mathbb{Z}_Q^*$, where r serves as the generator of this cyclic group with modulus Q. As a result, $\zeta = r^{\frac{Q-1}{2N}} \in \mathbb{Z}_Q$ has periodicity of $2N$. That is $\zeta^{2N} = 1(\text{mod } Q)$ and $\zeta^N = -1(\text{mod } Q)$, where ζ is the $2N$-th primitive root of unity in $\mathcal{R}_Q$. So, this property makes it suitable for decomposing the modulo polynomial $X^N + 1$, as explained in Section 2.6.

However, if the modulus Q is a composite number, the generator r exists only if Q takes the form of 4, p^k, or $2p^k$, where p is a prime and k is an integer. It should be noted that using the primitive root directly to generate a $2N$-th root that satisfies the NTT requirements is not possible. To solve this problem, Heinz et al. [HP22] propose a method to construct the root for the composite NTT, while applying it to the attack and defense of the measurement channel. Given two NTT-friendly numbers Q_1, Q_2, let ζ_{Q_1} and ζ_{Q_2} be the $2N$-th primitive root of unity for the polynomials $\mathcal{R}_{Q_1}$ and $\mathcal{R}_{Q_2}$, respectively. The method that construct the $2N$-th primitive root of unity ζ_Q in [HP22] as

$$\zeta_Q = \zeta_{Q_1} \cdot Q_2 \cdot (Q_2^{-1} \bmod Q_1) + \zeta_{Q_2} \cdot Q_1 \cdot (Q_1^{-1} \bmod Q_2). \tag{2}$$

In practice, the Equation 2 is often replaced with the slightly more efficient method as

$$\zeta_Q = \zeta_{Q_1} + Q_2 \cdot [(\zeta_{Q_2} - \zeta_{Q_1}) \cdot (Q_2^{-1} \bmod Q_1) \bmod Q_1].$$

It is easy to verify that these two methods of constructing roots are equivalent. Then, we provide complete proof of their approach.

Lemma 1. *Let $Q = Q_1 \times Q_2$ and $Q_i = 1 \pmod{2N}$ for $i = 1, 2$, Equation 2 outputs the $2N$-th primitive root ζ with the properties $\zeta^{2N} = 1 \pmod{Q}$ and $\zeta^N = -1 \pmod{Q}$.*

Proof. The equation 2 uses the inverse Chinese Residue Theorem to integrate two roots ζ_{Q_1} and ζ_{Q_2}, we have

$$\begin{cases} \zeta_Q = \zeta_{Q_1} (\text{mod } Q_1) \\ \zeta_Q = \zeta_{Q_2} (\text{mod } Q_2). \end{cases}$$

Since the ζ_{Q_1} and ζ_{Q_2} are the primitive roots of unity of ring $\mathcal{R}_{Q_1}$ and $\mathcal{R}_{Q_2}$, respectively, the following equation holds

$$\begin{cases} \zeta_Q^{2N} = 1 (\text{mod } Q_1) \\ \zeta_Q^{2N} = 1 (\text{mod } Q_2), \end{cases} \quad \begin{cases} \zeta_Q^{N} = -1 (\text{mod } Q_1) \\ \zeta_Q^{N} = -1 (\text{mod } Q_2). \end{cases}$$

Therefore, we can derive that $\zeta_Q^{2N} = 1(\text{mod } Q_1 \cdot Q_2)$ and $\zeta_Q^N = -1(\text{mod } Q_1 \cdot Q_2)$, which means that ζ_Q^N satisfies the property of being periodic and symmetric. □

We extend this method to composite modulus consisting of multiple distinct NTT-friendly moduli, as illustrated in Algorithm 2. In this approach, we employ a binary tree technique to minimize the number of multiplications, instead of directly utilizing $k - 1$ iCRT operations. This optimization allows for more efficient computations and improved performance.

Algorithm 2 Construction of the $2N$-th Root for Modulus $Q = Q_1 \times \cdots \times Q_k$

Input:
k distinct NTT-friendly moduli $Q_1, \cdots, Q_k$ and the corresponding $2N$-th primitive roots $\zeta_{Q_1}, \cdots, \zeta_{Q_k}$.

Output:
Run the CrootGen$(1, k)$ function to obtain the $2N$-th primitive root of unity ζ for modulus Q.

```
1: function CrootGen($w, v$)
2:     if $w = v$, return $\{\zeta_{Q_w}, Q_v\}$
3:     $mid = w + \lfloor (v - w)/2 \rfloor$
4:     $\{Q_j, \zeta_{Q_j}\} \leftarrow$ CrootGen($w, mid$)
5:     $\{Q_k, \zeta_{Q_k}\} \leftarrow$ CrootGen($mid + 1, v$)
6:     Set $Q_m = Q_j \cdot Q_k$ and $\zeta_{Q_m} = \zeta_{Q_j} \cdot Q_k \cdot (Q_k^{-1} \bmod Q_j) + \zeta_{Q_k} \cdot Q_j \cdot (Q_j^{-1} \bmod Q_k)$
7:     return $\{Q_m, \zeta_{Q_m}\}$
8: end function
```

It is worth noting that the NTT and iNTT algorithms with composite modulus differ from traditional algorithms solely in their input. This implies that we can input the root ζ and its inverse into Algorithms 6 and 7 respectively, to perform the composite NTT and iNTT operations, collectively referred to as Com-NTT. This approach allows for the seamless integration of composite modulus into the NTT and iNTT computations, enhancing the flexibility and compatibility of the NTT algorithm.

3.4 Applications

These methods for polynomial multiplication with composite modulus are aimed at different application scenarios for HE schemes. The RNS technique is commonly used in BGV, BFV, and CKKS schemes. By incorporating the RNS technique, these schemes can effectively deal with larger moduli while improving computational efficiency. RNS variants have emerged as the preferred choice in practical implementations, featured in software libraries like SEAL [SEA22] and OpenFHE [BBB+22].

Compared to RNS decomposition, the composite NTT approach provides a more flexible strategy. On the one hand, it is compatible with the RNS decomposition. In addition, it can directly perform the Number Theoretic Transform (NTT) on the composite modulus. In general, the latter option is more suitable for HE operations within a 64-bit ciphertext modulus. For instance, the external product is improved in FHEW-like schemes, as demonstrated in Section 4.

The method described in [CHK+21] applies to predetermined ring parameters, including the dimension and modulus of the polynomial. For instance, the original CKKS scheme [CKKS17] uses the ciphertext modulus $q = p^l$ to reduce the error resulting from the rescaling operation. The implementation in the HEAAN library adopts the strategy outlined in [CHK+21], which lifts the ciphertext modulus to utilize the RNS-based NTT implementation. In summary, depending on various scenarios and application requirements in the homomorphic encryption scheme, it is possible to identify suitable parameter settings and NTT strategies that effectively accelerate the underlying polynomial operations.

4 Faster FHEW-like Bootstrapping with Modulus Raising

The external product and blind rotation in the FHEW-like scheme use exact gadget decomposition to reduce noise, which means that only digit decomposition is utilized to reduce the error growth. We introduce the modulus raising technique into the FHEW-like schemes and then propose a hybrid method for external products by integrating digit decomposition, RNS decomposition, and modulus raising. These methods improve the efficiency of blind rotation and bootstrapping. Finally, we show the new technique to bootstrap two LWE ciphertexts in one blind rotation.

4.1 External product with modulus raising

Instead of using a single ciphertext modulus in RGSW ciphertext, our method involves the composite number consisting of multiple NTT-friendly moduli. More precisely, the ciphertext modulus of RGSW is set to the composite number PQ, where $PQ \approx Q_g$. Sample two RLWE ciphertexts $\{\mathsf{ct}_0, \mathsf{ct}_1\} \in \mathsf{RLWE}_{s,PQ}(0)$ and the RGSW ciphertext takes the following form:

$$\mathsf{RGSW}_{s,PQ}(m) = \begin{pmatrix} \mathsf{ct}_0 \\ \mathsf{ct}_1 \end{pmatrix} + m \cdot \begin{pmatrix} P & 0 \\ 0 & P \end{pmatrix} \in \mathcal{R}_{PQ}^{2\times 2}.$$

Definition 1. (Variant External Product). We define variant external product as $\boxdot_M$, which is performed between $\mathsf{ct} = (a, b) \in \mathsf{RLWE}_{s,Q}(\mu)$ and $\mathsf{CT} = \mathsf{RGSW}_{s,PQ}(m)$ ciphertexts as

$$\begin{aligned} \mathsf{ct}' &= \mathsf{RLWE}_{s,Q}(\mu) \boxdot_M \mathsf{RGSW}_{s,PQ}(m) \\ &= \lfloor \frac{[\mathsf{RLWE}_{s,Q}(\mu) \cdot \mathsf{RGSW}_{s,PQ}(0) + mP \cdot \mathsf{RLWE}_{s,Q}(\mu)]_{PQ}}{P} \rceil \\ &= \frac{\mathsf{RLWE}_{s,PQ}(mP \cdot \mu)}{P} \\ &= \mathsf{RLWE}_{s,Q}(m \cdot \mu). \end{aligned}$$

The error generated in $\boxdot_M$ is $e' = \frac{a \cdot e_0 + b \cdot e_1}{P} + e + e_{round}$, where e_0 and e_1 are the error terms of the RGSW ciphertext, e is the error term of the RLWE ciphertext, and e_{round} is the error caused by the rounding operation. And the variance of e' is

$$\begin{aligned} \mathsf{Var}(\mathsf{err}(\mathsf{ct}')) &\le \frac{2NQ^2}{12P^2} \cdot \mathsf{Var}(\mathsf{err}(\mathsf{CT})) + \mathsf{Var}(\mathsf{err}(\mathsf{ct})) + \frac{||s||_2^2 + 1}{12} \\ &\le \frac{NQ^2}{6P^2} \cdot \mathsf{Var}(\mathsf{err}(\mathsf{CT})) + \mathsf{Var}(\mathsf{err}(\mathsf{ct})) + \frac{N+2}{24}. \end{aligned} \quad (3)$$

where the factor $\frac{1}{12}$ corresponds to the variance of discrete uniform distribution in the range $[-1/2, 1/2]$ and the ternary secret key distribution ensures that $||s||_2 \le \sqrt{N/2}$.

4.1.1 Bootstrapping Procedure with Modulus Raising and Composite NTT

Then, we show the FHEW-like bootstrapping procedure based on the external product with modulus raising. For a ternary LWE secret key $\mathbf{s} \in \{-1, 0, 1\}^n$, our bootstrapping key generation process [BIP+22] is described as follows.

$$\mathsf{bsk} = \begin{cases} \mathsf{bsk}_{i,0} = \mathsf{RGSW}_{s',PQ}(0), \mathsf{bsk}_{i,1} = \mathsf{RGSW}_{s',PQ}(1), \text{if } (s_i = -1); \\ \mathsf{bsk}_{i,0} = \mathsf{RGSW}_{s',PQ}(0), \mathsf{bsk}_{i,1} = \mathsf{RGSW}_{s',PQ}(0), \text{if } (s_i = 0); \\ \mathsf{bsk}_{i,0} = \mathsf{RGSW}_{s',PQ}(1), \mathsf{bsk}_{i,1} = \mathsf{RGSW}_{s',PQ}(0), \text{if } (s_i = 1). \end{cases} \quad (4)$$

Note that the bootstrapping keys are precomputed and stored in the NTT representation, and can be reused in the bootstrapping process.

Algorithm 3 GINX Bootstrapping with Modulus Raising and Composite NTT

```
Input:
    An LWE sample $\mathsf{ct} = (\mathbf{a}, b) \in \mathsf{LWE}^n_{\mathbf{s},q}(m)$, where $q|2N$.
    A bootstrapping key $\mathsf{bsk}$.
    A key switching key $\mathsf{ksk}_{\mathbf{s}}(s'_i)$ as shown in Section 2.7.
    A test polynomial $tv$ embedding a look-up table $f$.
Output:
    An LWE sample $\mathsf{ct}' \in \mathsf{LWE}^n_{\mathbf{s},q}(f(m))$.
1:  Set $\mathsf{acc} = (0, X^{-b} \cdot tv) \in \mathcal{R}^2_Q$
2:  for $i = 0$ to $n-1$ do
3:          $\mathsf{acc}'_i = \text{Com-NTT}(\mathsf{acc}_i)$
4:          $\mathsf{acc}'_i = \mathsf{acc}'_i \odot (\mathbf{1} + (X^{a_i} - 1) \odot \mathsf{bsk}_{i,0} + (X^{-a_i} - 1) \odot \mathsf{bsk}_{i,1}) \in \mathcal{R}^2_{PQ}$
5:          $\mathsf{acc}_i = \text{Com-iNTT}(\mathsf{acc}_i)$
6:          $\mathsf{acc}_i = \lfloor \frac{\mathsf{acc}_i}{P} \rceil \in \mathcal{R}^2_Q$
7:  end for
8:  $\mathsf{ct}' = \mathsf{SampleExtract}(\mathsf{acc}_{n-1})$
9:  $\mathsf{ct}' = \mathsf{KeySwitch}(\mathsf{ct}')$
10: $\mathsf{ct}' = \mathsf{ModSwitch}(\mathsf{ct}')$
11: return $\mathsf{ct}'$.
```

Algorithm 3 presents the improved bootstrapping algorithm for the GINX method by using modulus raising. The procedure begins with an LWE ciphertext as usual. In line 4 of the algorithm, let

$$\mathsf{CT}_{Mux,i} = (\mathbf{1} + (X^{a_i} - 1) \odot \mathsf{bsk}_{i,0} + (X^{-a_i} - 1) \odot \mathsf{bsk}_{i,1})$$

is a ternary CMux gate, and it is easy to verify that $\mathsf{acc}_i \boxdot_M \mathsf{CT}_{Mux,i}$ yields the ciphertext $\mathsf{acc}'_i = \mathsf{RLWE}(X^{a_i \cdot s_i})$, where $\mathbf{1}$ is a noiseless RGSW ciphertext. In more detail, the accumulator acc is viewed as two polynomials with modulus PQ, and performs the composite NTTs. To reduce the number of NTT transformations and rounding operations, we generate beforehand a table containing all NTT representations of $X^i - 1$ with modulus PQ, where $0 \le i \le 2N - 1$. Subsequently, we utilize the ciphertext a_i to retrieve the corresponding NTT representation for $X^{a_i} - 1$ and $X^{-a_i} - 1$, enabling direct Hadamard multiplication with bootstrapping keys. However, the division operation needs to be performed in the coefficient representation, which involves two iNTT operations. By utilizing n external products, we can get the ciphertext as

$$\begin{aligned}\mathsf{acc} &= \mathsf{RLWE}_{s,Q}(tv \cdot X^{-b+\sum_{i=0}^{n-1} a_i s_i}) \\ &= \mathsf{RLWE}_{s,Q}(tv \cdot X^{-(\lfloor \frac{q}{t} \cdot m \rceil + e)}).\end{aligned}$$

Note that the test vector tv serves two purposes. It not only refreshes the noise but also embeds a lookup table $f : \mathbb{Z}_t \to \mathbb{Z}_t$ by defining the coefficients of the polynomial as follows:

$$tv(X) = \sum_{i=0}^{N-1} \frac{Q}{t} \cdot f(\lfloor \frac{t}{q} \cdot i \rceil) \cdot X^i.$$

This method, known as functional bootstrapping, is described in [CJP21, KS21] schemes. Then, we can get an LWE ciphertext $\mathsf{LWE}^N_{\mathsf{Coefs}(s),Q}(f(m))$ through the sample extraction operation. Finally, the key switching and modulus switching operations are performed to obtain the ciphertext $\mathsf{ct}' = \mathsf{LWE}^n_{\mathbf{s},q}(f(m)) \in \mathbb{Z}_q^{n+1}$, which completes the entire bootstrapping procedure.

Remark 4.1. *Note that the RNS decomposition method can also be utilized for polynomial multiplication in the external product. However, this approach requires additional NTT operations, as explained in Appendix C. The computational complexity of blind rotation for the Algorithms 1, 3 and 8 is outlined in Table 2.*

Table 2: Comparison of blind rotation for different algorithms under the ternary secret key, where the length is set to $d_g = 2$ for gadget decomposition in Algorithm 3.

	# NTTs	# Modular multiplication	# Rounding	# Decomposition
Algorithm 1	$6n$	$20nN$	$\times$	$2nN$
Algorithm 3	$4n$	$14nN$	$2nN$	$\times$
Algorithm 8	$8n$	$26nN$	$\times$	$\times$

4.1.2 Error analysis

In this subsection, we analyze the variance of error for Algorithm 3. Firstly, the error growth in blind rotation is caused by a sequence of n external products i.e., $\mathsf{acc}\boxdot_{i=0}^{n-1}\mathsf{CT}_{Mux}$ $= (...((\mathsf{acc}_0 \boxdot \mathsf{CT}_{Mux,0}) \boxdot \mathsf{CT}_{Mux,1})... \boxdot \mathsf{CT}_{Mux,n-1})$. Thus, we can obtain variance by using n times Equation 3 as

$$\begin{aligned}\mathsf{Var}(\mathsf{err}(\mathsf{acc})) &\leq \frac{NQ^2}{6P^2} \cdot \mathsf{Var}(\mathsf{err}(\mathsf{CT}_{Mux,n-1})) + \mathsf{Var}(\mathsf{err}(\mathsf{acc}_{n-2})) + \frac{N+2}{24} \\ &\leq \frac{NQ^2}{6P^2} \cdot \mathsf{Var}(\mathsf{err}(\mathsf{CT}_{Mux,n-1})) + \frac{NQ^2}{6P^2} \cdot \mathsf{Var}(\mathsf{err}(\mathsf{CT}_{Mux,n-2})) \\ &\quad + \mathsf{Var}(\mathsf{err}(\mathsf{acc}_{n-3})) + 2 \cdot \frac{N+2}{24} \\ &\quad \vdots \\ &\leq \frac{NQ^2}{6P^2} \cdot \sum_{i=0}^{n-1} \mathsf{Var}(\mathsf{err}(\mathsf{CT}_{Mux,i})) + \mathsf{Var}(\mathsf{err}(\mathsf{acc}_0)) + n \cdot \frac{N+2}{24}\end{aligned}$$

Due to the fact that $\mathsf{Var}(\mathsf{err}(\mathsf{CT}_{Mux,i})) = (||X^{a_i} - 1||_2^2 + ||X^{-a_i} - 1||_2^2) \cdot \mathsf{Var}(\mathsf{err}(\mathsf{bsk})) \leq 4 \cdot \mathsf{Var}(\mathsf{err}(\mathsf{bsk}))$ and the initial RLWE is noise-free, i.e., $\mathsf{Var}(\mathsf{err}(\mathsf{acc}_0)) = 0$. We can obtain the variance from the blind rotation as

$$\mathsf{Var}(\mathsf{err}(\mathsf{acc})) \leq \frac{2nNQ^2}{3P^2} \cdot \mathsf{Var}(\mathsf{err}(\mathsf{bsk})) + \frac{n \cdot (N+2)}{24}.$$

Then, we can get the variance of the error from the key switching operation as

$$\mathsf{Var}(\mathsf{err}(\mathsf{ct}')) \leq Nd_k \cdot \mathsf{Var}(\mathsf{err}(\mathsf{ksk})) + \mathsf{Var}(\mathsf{err}(\mathsf{ct})).$$

Finally, after modulus switching, we can conclude that the variance of the error generated by the bootstrapping process is

$$\mathsf{Var}(\mathsf{err}(\mathsf{ct}')) \leq \frac{q^2}{Q^2} \cdot [\frac{2nNQ^2}{3P^2} \cdot \mathsf{Var}(\mathsf{err}(\mathsf{bsk})) + \frac{n \cdot (N+2)}{24} + Nd_k \cdot \mathsf{Var}(\mathsf{err}(\mathsf{ksk}))] + \frac{2+n}{24}.$$

Compared to the gadget decomposition, our method introduces an additional error that is derived from the rounding operation. In Section 5, we demonstrate that the error is negligible for the decryption failure rate.

4.2 Hybrid External Product and Blind Rotation

In Definition 1, we only use the modulus switch to reduce noise growth. Following that, we present a hybrid external product operation in Definition 2. Firstly, we show a hybrid approach based on digit decomposition and modulus raising. Given a modulus Q and the base B'_g, we can denote the gadget matrix $\mathbf{G}$ as $I_2 \otimes (P \cdot \mathbf{g}'_d)^T \in \mathbb{Z}_{PQ}^{2d'_g \times 2}$, where $d'_g = \left\lfloor \log_{B'_g} Q \right\rceil + 1$. Sample $2d'_g$ RLWE ciphertexts $\{\mathsf{ct}_0, \cdots, \mathsf{ct}_{2d'_g-1}\} \in \mathsf{RLWE}^N_{s,PQ}(0)$, we have the RGSW ciphertext of m as

$$\mathsf{RGSW}_{s,PQ}(m) = \begin{pmatrix} \mathsf{ct}_0 \\ \vdots \\ \mathsf{ct}_{2d'_g-1} \end{pmatrix} + m \cdot \mathbf{G} \in \mathcal{R}_{PQ}^{2d'_g \times 2}.$$

Definition 2. (Hybrid External Product). We define the hybrid external product as $\boxdot_H$,

$$\begin{aligned} \mathsf{ct}' &= \mathsf{RLWE}_{s,Q}(\mu) \boxdot_H \mathsf{RGSW}_{s,PQ}(m) \\ &= \lfloor \frac{\mathsf{RLWE}_{s,Q}(\mu) \boxdot_D \mathsf{RGSW}_{s,PQ}(m)}{P} \rceil \\ &= \lfloor \frac{\mathsf{RLWE}_{s,PQ}(Pm \cdot \mu)}{P} \rceil \\ &= \mathsf{RLWE}_{s,Q}(m \cdot \mu). \end{aligned} \tag{5}$$

The error generated by $\boxdot_H$ is $e' = \sum_{i=0}^{d'_g-1}(a_i \cdot e_i + b_i \cdot e_{i+d'_g})/P + e + e_{round}$, and its variance is

$$\mathsf{Var}(\mathsf{err}(\mathsf{ct}')) \leq \frac{Nd'_g B'^2_g}{6P^2} \cdot \mathsf{Var}(\mathsf{err}(\mathsf{CT})) + \mathsf{Var}(\mathsf{err}(\mathsf{ct})) + \frac{N+2}{24}.$$

Afterward, Algorithm 4 shows the second improved GINX blind rotation algorithm using the hybrid external product. The gadget decomposition and modulus switching are utilized to reduce error growth in lines 3 and 7, respectively. The correctness of the algorithm can be directly derived from the new external product. Compared to the case of $d_g > 2$ for Algorithm 1 with the ternary secret key, the new blind rotation algorithm involves a lesser number of NTT operations.

Algorithm 4 GINX Blind Rotation with Hybrid External Product.

```
Input:
    An LWE sample $\mathsf{ct} = (\mathbf{a}, b) \in \mathsf{LWE}^n_{\mathbf{s},q}(m)$, where $q|2N$.
    A blind rotation key $\mathsf{bsk}_{\mathbf{s}}(s_i)$ as shown in Equation 4 using hybrid RGSW.
Output:
    An RLWE ciphertext $\mathsf{acc} \in \mathsf{RLWE}_{s,Q}(X^{-b+\sum_{i=0}^{n-1} a_i s_i})$ .
1: Set $\mathsf{acc} = (0, X^{-b}) \in \mathcal{R}^2_Q$
2: for $i = 0$ to $n-1$ do
3:         $\mathsf{acc}'_i = \mathbf{g}'^{-1}(\mathsf{acc}_i) \in \mathcal{R}^{d'_g}_{B_g}$
4:         $\mathsf{acc}'_i = \text{Com-NTT}(\mathsf{acc}'_i)$
5:         $\mathsf{acc}'_i = ((X^{a_i} - 1) \odot \mathsf{acc}'_i) \odot \mathsf{bsk}_{i,0} + ((X^{-a_i} - 1) \odot \mathsf{acc}'_i) \odot \mathsf{bsk}_{i,1} + \mathsf{acc}'_i \in \mathcal{R}^2_{PQ}$
6:         $\mathsf{acc}_i = \text{Com-iNTT}(\mathsf{acc}'_i)$
7:         $\mathsf{acc}_i = \lfloor \frac{\mathsf{acc}_i}{P} \rceil \in \mathcal{R}^2_Q$
8: end for
9: return acc.
```

Remark 4.2. *We remark that the digit decomposition in Equation 5 can be replaced by RNS decomposition with the same number of NTTs and Hadamard multiplications. To be specific, we can select d'_g NTT-friendly numbers Q_i, and let $Q = Q_1 \times \cdots \times Q_{d'_g}$. In this way, the gadget matrix $\mathbf{G}$ associated with the RNS decomposition can be expressed as $I_2 \otimes (P \cdot \mathbf{g}'_r)^T \in \mathbb{Z}_{PQ}^{2d'_g \times 2}$. Thus, the external product can perform the RNS decomposition against the accumulator acc', and then use the composite NTTs to compute the subsequent Hadamard multiplications. Table 3 exhibits the computation complexity of blind rotation for Algorithms 1 and 4.*

Table 3: Comparison of blind rotation for Algorithms 1 and 4 with the ternary secret key, where the d_g and d'_g are the length for gadget decomposition and hybrid method, respectively, where $d_g > 2$ and $d'_g = \lceil d_g/2 \rceil$.

	Algorithm 1	Algorithm 4	
		Digit decomposition	RNS decomposition
# NTTs	$2n(d_g+1)$	$2n(d'_g+1)$	$2n(d'_g+1)$
# Modular multiplication	$(8d_g+4)nN$	$(8d'_g+4)nN$	$(8d'_g+4)nN$
# Rounding	$\times$	$2nN$	$2nN$
# Decomposition	$2nN$	$2nN$	$\times$
# Modular	$\times$	$\times$	$2nN$

4.3 Packing Bootstrapping with Composite NTT

We show that the proposed composite NTT technique can support packing bootstrapping procedures. To simplify our algorithm, we use binary keys in this section. Details of this subroutine are given in Algorithm 5. Specifically, when packing l bootstrapping procedures, we first need to choose l sets for parameters P_kQ_k, and set $PQ = \prod P_kQ_k$, where $k \in [1, l]$. Given l blind rotation keys $\mathsf{brk}_{P_kQ_k}$ that are generated in Section 4.2, we can precompute to generate the new blind rotation key using iCRT as

$$\mathsf{brk}_{PQ} = \mathsf{iCRT}_{\{P_1Q_1,\ldots,P_lQ_l\}}(\mathsf{brk}_{P_1Q_1}, \ldots, \mathsf{brk}_{P_lQ_l}) \qquad (6)$$

Algorithm 5 Packing Bootstrapping using CRT and composite NTT.

```
Input:
    $l$ LWE samples $\mathsf{ct}_k = (\mathbf{a}_k, b_k) \in \mathsf{LWE}^n_{\mathbf{s},q}(m_k)$, for $k \in [1, l]$.
    The blind rotation key $\mathsf{brk}_{PQ}$ as shown in Equation 6.
Output:
    $l$ RLWE ciphertexts $\mathsf{acc}_k \in \mathsf{RLWE}_{s,Q_k}(X^{-b_k + \sum_{i=0}^{n-1} a_{k,i} s_i})$.
1: Set $l$ accumulators $\mathsf{acc}_k = (0, X^{-b_k}) \in \mathcal{R}^2_{Q_k}$.
2: for $i = 0$ to $n-1$ do
3:     $\mathsf{acc}_{k,i} = \mathbf{g}'^{-1}_d(\mathsf{acc}_{k,i})$, for all $k \in [1, l]$
4:     $\mathsf{acc}_i = \mathsf{iCRT}_{\{P_1Q_1,...,P_lQ_l\}}(\mathsf{acc}_{1,i}, ..., \mathsf{acc}_{l,i}) \in \mathcal{R}^2_{PQ}$
5:     $\mathsf{acc}'_i = \text{Com-NTT}(\mathsf{acc}_i)$
6:     $\mathsf{acc}'_i = \mathsf{acc}'_i \odot \mathsf{brk}_i + \mathsf{acc}'_i \in \mathcal{R}^2_{PQ}$
7:     $\mathsf{acc}_i = \text{Com-iNTT}(\mathsf{acc}'_i)$
8:     $\{\mathsf{acc}_{1,i}, ..., \mathsf{acc}_{l,i}\} = \mathsf{CRT}_{\{P_1Q_1,...,P_lQ_l\}}(\mathsf{acc}_i) \in \mathcal{R}^4_{P_kQ_k}$
9:     $\mathsf{acc}_{k,i} = \lfloor \frac{\mathsf{acc}_{k,i}}{P_k} \rceil \in \mathcal{R}^2_{Q_k}$, for all $k \in [1, l]$
10:    $\mathsf{acc}_{k,i} = \mathsf{acc}_{k,i} \cdot (X^{a_{k,i}} - 1)$, for all $k \in [1, l]$ .
11: end for
12: return $\{\mathsf{acc}_k\}_{k \in [1,l]}$.
```

In order to improve the expensive external product operation in CMux gates for the l blind rotations processes. We utilize the CRT to merge the l accumulators acc_k into polynomial ring $\mathcal{R}_{PQ}$. Subsequently, we utilize the composite NTT to perform a single external product operation. Here, we need $2d'_g + 2$ NTTs and $4d'_g$ Hadamard multiplications in polynomial ring $\mathcal{R}_{PQ}$ in lines 5-7 of the algorithm. To achieve this purpose, we incur additional operations, i.e., the CRT and iCRT processes. Furthermore, we use gadget decomposition for each LWE ciphertext to reduce noise and advance to the next iteration. The detailed analyses of algorithm correctness and noise growth have been omitted, as they can be directly derived from the properties of the CRT.

It is worth noting that the number of packing bootstrapping of the proposed algorithm is directly related to the machine word length of the experimental platform. Typically, by taking advantage of the CPU's capacity to handle 64 bits, we can achieve maximum gain by packing two bootstrapping procedures together, resulting in improved overall performance.

4.4 Extensions for other FHE schemes

4.4.1 Improved Blind rotations for AP and LMK Methods

AP Blind Rotation. The AP blind rotation [DM15, ASP14] supports arbitrary types of secret key distributions. The idea is to decompose the LWE ciphertext and extract the associated blind rotation key. Then, the blind rotation keys are accumulated through some external products. Our technique can improve the AP blind rotation by utilizing the external product $\boxdot_M$ and $\boxdot_H$ Given an LWE secret key $\mathbf{s} \in \mathbb{Z}^n_q$, the AP blind rotation key is generated as

$$\mathsf{bsk} = \left\{ \mathsf{bsk}_{i,j,v} = \mathsf{RGSW}_{s',PQ}(Y^{vB_r^j s_i}) \right\} \tag{7}$$

where $i \in [0, n-1]$, $j \in [0, \log_{B_r} q - 1]$ and $v \in \mathbb{Z}_{B_r}$. The decomposition base $B_r \geq 2$ can offer a tradeoff between space and computational complexity. For an LWE cipheretxt, we can decompose each term a_i as $a_i = \sum_{j=0}^{\log_{B_r} q-1} a_{i,j}$ and accumulator acc is updated for all $a_{i,j}$ that

$$\mathsf{acc} = \mathsf{acc} \boxdot_M \mathsf{bsk}_{i,j,a_{i,j}}.$$

The detailed algorithm is described in Appendix D.

LMK Blind Rotation. The LMK blind rotation [LMK+23] improves the AP method for efficiently supporting arbitrary secret key distributions by utilizing ring automorphisms and RLWE-based key switching in the FHEW cryptosystem. The technical overview of the high-level structure of their solution is as follows.

Given a current accumulator $\mathsf{RLWE}(g_{i-1}(X))$, one can use automorphism $\psi_{a_i^{-1}} : X \to X^{a_i^{-1}}$ to get an encryption of $g_{i-1}(X^{a_i^{-1}})$. Then, with the blind rotation key $\mathsf{RGSW}(X^{s_i})$, the external product is performed to the ciphertext of $g_{i-1}(X^{a_i^{-1}}) \cdot X^{s_i}$. Finally, the ring automorphism $\psi_{a_i} : X \to X^{a_i}$ is utilized to get the accumulator that encrypt $g_{i-1}(X^{a_i s_i})$. After repeating this process n times, the accumulator can be calculated as $\mathsf{RLWE}(X^{-b+\sum_{i=0}^{n-1} a_i s_i})$. During this process, the automorphisms ψ_a exist only for odd values due to the power-of-two cyclotomic setting. Their work introduces some solutions and optimizes the algorithm by reordering the secret key.

In bootstrapping, key-switching is necessary to transform the ciphertext into encryption under the original key following the automorphism operation. The proposed external product technique can be utilized to accelerate both the external product and key-switching operations. We omit the algorithm, refer to [LMK+23] for the detailed process.

4.4.2 NTRU-based External product

Our technique can also apply to the NTRU-based external product and bootstrapping procedure. In particular, Bonte et al. [BIP+22] construct NTRU-based GSW-like ciphertext (NGS), where the NGS ciphertext is represented as a vector polynomial and performed external product by gadget decomposition. We can improve the NTRU-based external product by using techniques similar to section 4.2. These techniques serve to further improve the efficiency of NTRU-based bootstrapping. However, in practice, careful consideration must be given to parameter selection due to the dense sublattice attack, which restricts the modulus size of the NTRU problem below $n^{2.484+o(1)}$[DvW21].

5 Parameters and Performance

This section presents a comprehensive analysis of the proposed scheme, including parameter setting, error growth, and decryption failure rates. Additionally, we compare the bootstrapping experimental results of the proposed method with the gadget decomposition. This analysis will offer valuable insights into the respective strengths and weaknesses of these methods, allowing us to make choices about which method is more suitable for different parameters and scenarios.

5.1 Parameters and Noise Growth

The proposed algorithm in Section 4 works with the following parameters:

- λ, Security level ;
- t, Plaintext modulus for the LWE sample, $t = 4$ by default;
- n, Lattice dimension for the LWE sample;
- q, Ciphertext modulus for LWE sample;
- N, Ring dimension for RLWE/RGSW;
- σ, Standard deviation of Gaussian distribution;
- Q, Ciphertext modulus for the RLWE sample and key-switching;

Table 4: Bootstrapping parameters for the modulus raising

Parameters	Secret key	λ	n	N	σ	q	P	Q
B_1024_36	Binary	94	512	1024	3.19	512	$\approx 2^{20}$	$\approx 2^{16}$
T_1024_36	Ternary	96	512	1024	3.19	512	$\approx 2^{20}$	$\approx 2^{16}$
B_1024_27	Binary	127	512	1024	3.19	1024	$\approx 2^{11}$	$\approx 2^{16}$
T_1024_27	Ternary	131	512	1024	3.19	1024	$\approx 2^{11}$	$\approx 2^{16}$
B_2048_50	Binary	139	1024	2048	3.19	1024	$\approx 2^{28}$	$\approx 2^{22}$
T_2048_50	Ternary	141	1024	2048	3.19	1024	$\approx 2^{28}$	$\approx 2^{22}$

- P, Modulus used in the RGSW sample and external product operation.

Table 4 presents specific parameters used in implementation with the modulus raising and hybrid methods. The secret keys are selected from binary and ternary distributions, which are employed in the TFHE and FHEW schemes. We choose the numbers PQ that satisfy the requirements for the composite NTT as shown in Section 3.3.

Table 5 displays specific parameters for gadget decomposition and key-switching. The two columns labeled B_g' and d_g' are used in the hybrid external product, while corresponding to parameter sets **B_1024_27** and **T_1024_27** in table 4. The remaining parameters are used in the external product based on gadget decomposition and key-switching, which is provided by [MP21] scheme. Note that the bootstrapping with procedure with gadget decomposition entails an additional step of modulus switching to a smaller modulus Q, then performing the key switching operation.

- Q_g, RLWE ciphertext modulus for gadget decomposition;
- B_g, Gadget base for RGSW encryption, which breaks integer Q_g into d_g digits;
- B_k, Gadget base for key switching, which breaks integer Q_g into d_k digits;

Table 5: Bootstrapping parameters for the gadget decomposition and key-switching

Parameters	λ	Q_g	B_g	d_g	B_g'	d_g'	B_k	d_k
(**B**)**T_1024_36**	>94	$\approx 2^{36}$	2^{18}	2	$\times$	$\times$	2^9	2
(**B**)**T_1024_27**	>127	$\approx 2^{27}$	2^7	4	2^8	2	2^7	2
(**B**)**T_2048_50**	>139	$\approx 2^{50}$	2^{25}	2	$\times$	$\times$	2^5	5

The security level of HE schemes is determined by several factors, including the secret key distribution, the dimensions and modulus of the (R)LWE sample, and the standard deviations of the error according to the HE standard [ACC⁺18]. Then, we use the LWE estimator [APS15] to estimate the security level, which calculates the complexity of primal attacks via the shortest vector problem, decoding, and dual-lattice attacks. Table 6 provides the cost of specific attacks using the BKZ.sieve cost model.

Afterward, we analyze the error growth and decryption failure probability with these methods under different parameters. Specifically, the bootstrapping procedure results in a ciphertext with an error from a Gaussian distribution with standard deviation $\sigma = \sqrt{\frac{q^2}{Q^2}(\sigma_{ACC}^2 + \sigma_{KS}^2) + \sigma_{MS}^2}$, where the σ_{ACC}^2 plays a prominent role in determining the overall error magnitude. We compare the error growth of three different methods generated by blind rotation with the ternary secret key distribution, and the variances of errors are

$$\sigma_{ACC-MU}^2 = \frac{q^2}{Q^2} \cdot (\frac{2nN \cdot Q^2 \cdot 3.19^2}{3P^2} + \frac{n \cdot (N+2)}{24}).$$

Table 6: Security estimations for the parameter sets.

Parameters	Gadget decomposition		
	uSVP	dec	dual
B_1024_36	94.3	110.7	97.5
T_1024_36	96.3	110.7	101.1
B_1024_27	127.6	160.0	133.4
T_1024_27	131.6	160.7	138.7
B_2048_50	139.1	158.7	143.2
T_2048_50	141.1	158.7	145.3

$$\sigma^2_{ACC-GD} = \frac{q^2}{Q_g^2} \cdot (\frac{2nNd_gB_g^2 \cdot 3.19^2}{3}).$$

$$\sigma^2_{ACC-HY} = \frac{q^2}{Q^2} \cdot (\frac{2nNd'_gB'^2_g \cdot 3.19^2}{3P^2} + \frac{n \cdot (N+2)}{24}).$$

We note that the modulus raising and hybrid methods introduce additional noise terms due to the rounding operation. However, we have a smaller value for the length of decomposition that satisfies $d'_g = \lceil d_g/2 \rceil$, where $d'_g = 1$ is used in the modulus raising method. Typically, for the sets of parameters of the ternary key **T_1024_36**, **T_1024_27**, and **T_2048_50**. the results in Tables 4 and 5 can be obtained as follows:

$$\sigma^2_{ACC-GD} = 1.028 \cdot \sigma^2_{ACC-MU},$$

$$\sigma^2_{ACC-HY} = 1.274 \cdot \sigma^2_{ACC-GD}.$$

We remark that these comparison results give an intuition that the ratio of noise is so small as to be negligible, which can also be verified in the decryption failure rate. Specifically, in experiments, we evaluate one homomorphic addition for the NAND gate in bootstrapping, where $t = 4$, and the probability of decryption failure can be calculated using the following formula:

$$1 - \mathsf{erf}(\frac{q/8}{2\sigma}),$$

erf is the Gaussian function. Table 7 presents the decryption failure rate for parameter sets **T_1024_36**, **T_1024_27**, and **T_2048_50** with the ternary secret key distribution. From the table, we can see that the error gap has almost no effect on the decryption failure rate. Finally, we also performed a large number of experiments on the LWE samples with various key distributions, and the results of the experiments were consistent with the theoretical analysis.

Table 7: Decryption failure rates for Bootstrapping.

Parameters	Fail. prob.		
	GD	MU	HY
T_1024_36	2^{-40}	2^{-40}	$\times$
T_1024_27	2^{-53}	$\times$	2^{-53}
T_2048_50	2^{-100}	2^{-100}	$\times$

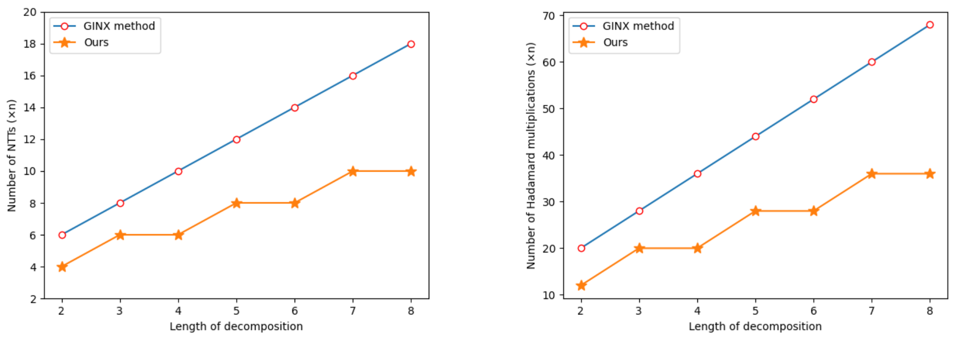

Figure 1: Comparison of blind rotations with different lengths of decomposition

5.2 Key Sizes

In this subsection, we analyze the key sizes for the aforementioned methods. Table 8 illustrates the sizes of blind rotation keys for both gadget decomposition and modulus raising. It is worth noting that the sizes of the key switching have been omitted from the table since they remain consistent for all the methods.

Table 8: Sizes of blind rotation keys.

Parameters	Gadget decomposition		Modulus raising	
B_1024_36	$4nNd_g \log_2 Q_g$	18 MB	$4nN \log_2 PQ$	9 MB
T_1024_36	$8nNd_g \log_2 Q_g$	36 MB	$8nN \log_2 PQ$	18 MB
B_1024_27	$4nNd_g \log_2 Q_g$	27 MB	$4nNd'_g \log_2 PQ$	13.5 MB
T_1024_27	$8nNd_g \log_2 Q_g$	54 MB	$8nNd'_g \log_2 PQ$	27 MB
B_2048_50	$4nNd_g \log_2 Q_g$	100 MB	$4nN \log_2 PQ$	50 MB
T_2048_50	$8nNd_g \log_2 Q_g$	200 MB	$8nN \log_2 PQ$	100 MB

Across all parameter sets, the modulus raising and composite NTT techniques achieve a remarkable reduction in key size, amounting to 50% compared to the gadget decomposition. This notable outcome holds great promise, particularly for hardware acceleration, as the binary key distribution necessitates a mere 9 MB of key size.

5.3 Implementation and Experiment performance

Firstly, we compare the number of NTTs and Hadamard multiplications in bootstrapping using different decomposition lengths d_g in Figure 1. In addition, we implement Algorithms 3 and 4. The evaluation environment was a commodity desktop computer system with an Intel(R) Core(TM) i5-12500 CPU @ 3.00GHz and 64 GB of RAM, running Ubuntu 22.04.2 LTS. The compiler was g++ 11.3.0. The experiments of identity bootstrapping evaluation are presented in Table 9, and each result is an average of 5000 executions.

The proposed method demonstrates a significant improvement in runtime compared to the gadget decomposition technique. In particular, we achieve speedups of around 1.5 × and 1.7 × under the specific sets of parameters, respectively, which is consistent with the expected results described in Section 4 and Figure 5.3. After that, we show the effect of gain when using our method in AP-based and LMK-based blind rotation under 127-bit security levels as shown in Table 10.

Table 9: Single-threaded timing results for bootstrapping

Parameters	Gadget decomposition	Modulus raising
B_1024_36	37.63 ms	25.61 ms
T_1024_36	51.94 ms	34.38 ms
B_1024_27	68.58 ms	37.04 ms
T_1024_27	93.46 ms	55.67 ms
B_2048_50	159.26 ms	106.15 ms
T_2048_50	205.98 ms	137.44 ms

Table 10: Comparison AP-based and LMK-based blind rotations in 127-bit security level with the ternary secret key, where GD is the gadget decomposition, the hybrid method is used in our technique, and $d_g = 3$ and $d_g = 4$ are the lengths of decomposition in AP and LMK method, respectively.

Methods		# Operations
AP-based	GD	$50n$ NTTs + $80n$ HMs
	Ours	$30n$ NTTs + $40n$ HMs
LMK-based	GD	$(8n + 412)$ NTTs + $(12n + 612)$ HMs
	Ours	$(6n + 309)$ NTTs + $(8n + 408)$ HMs

Moreover, Table 11 presents the experiment results for the packing method as shown in Algorithm 5. Under the **B_1024_27** parameter setting, the packing approach achieves efficiency gains of approximately 2.6 $\times$ and 1.5 $\times$ when compared to Algorithms 1 and 4 for two bootstrappings, respectively. This substantial improvement holds practical significance for the practical application of FHEW-like schemes.

Table 11: Bootstrapping comparisons for two LWE ciphertexts with the parameter **B_1024_27**, where the corresponding times for Algorithms 1 and 4 are simply doubled.

	Times	Fail. prob.
Algorithm 1	137.16 ms	2^{-53}
Algorithm 4	74.08 ms	2^{-53}
Algorithm 5	51.85 ms	2^{-53}

Finally, we note that the proposed algorithms can not apply the FFT-based TFHE scheme. In terms of time performance, the TFHE scheme incorporates optimizations and accelerations for AVX instructions, as seen in libraries such as TFHE-lib [CGGI20] and TFHE-rs [BSJJ22] libraries. Generally, the AVX-512 instructions can potentially deliver a substantial 3 $\sim$ 4 times speedup compared to the baseline performance in C++. One of our future works is to utilize AVX-512 instructions to accelerate the proposed algorithms. This enhancement will allow us to further optimize the performance of our scheme and make it compatible with advanced AVX instructions.

6 Conclusion

In this paper, we propose a faster bootstrapping procedure for FHEW-like schemes. By introducing a composite NTT technique, we integrate the gadget decomposition and modulus raising into the external product operation, we can reduce the number of NTTs required in the blind rotation process. Furthermore, we introduce a packing method that can bootstrap two LWE ciphertexts using a blind rotation process based on composite NTT. The results of the implementation of the proposed algorithm show gains in both efficiency and size. Our methods have the potential to improve the FHEW-like schemes and can be applied in practical scenarios.

Acknowledgments

We are grateful for the helpful comments from the anonymous reviewers. This work was supported by the Huawei Technologies Co., Ltd and CAS Project for Young Scientists in Basic Research (Grant No. YSBR-035).

References

[ACC$^+$18] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, et al. Homomorphic encryption security standard. *HomomorphicEncryption. org, Toronto, Canada, Tech. Rep*, 11, 2018.

[ACC$^+$21] Amin Abdulrahman, Jiun-Peng Chen, Yu-Jia Chen, Vincent Hwang, Matthias J. Kannwischer, and Bo-Yin Yang. Multi-moduli ntts for saber on cortex-m3 and cortex-m4. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(1):127–151, Nov. 2021.

[APS15] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[ASP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I 34*, pages 297–314. Springer, 2014.

[BBB$^+$22] Ahmad Al Badawi, Jack Bates, Flavio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, Zeyu Liu, Daniele Micciancio, Ian Quah, Yuriy Polyakov, Saraswathy R.V., Kurt Rohloff, Jonathan Saylor, Dmitriy Suponitsky, Matthew Triplett, Vinod Vaikuntanathan, and Vincent Zucca. Openfhe: Open-source fully homomorphic encryption library. Cryptology ePrint Archive, Paper 2022/915, 2022. https://eprint.iacr.org/2022/915.

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

[BIP$^+$22] Charlotte Bonte, Ilia Iliashenko, Jeongeun Park, Hilder VL Pereira, and Nigel P Smart. Final: Faster fhe instantiated with ntru and lwe. *Cryptology ePrint Archive*, 2022.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual Cryptology Conference*, pages 868–886. Springer, 2012.

[BSJJ22] Lars Brenna, Isak Sunde Singh, Håvard Dagenborg Johansen, and Dag Johansen. Tfhe-rs: A library for safe and secure remote computing using fully homomorphic encryption and trusted execution environments. *Array*, 13:100118, 2022.

[CCH$^+$22] Anamaria Costache, Benjamin R Curtis, Erin Hales, Sean Murphy, Tabitha Ogilvie, and Rachel Player. On the precision loss in approximate homomorphic encryption. *Cryptology ePrint Archive*, 2022.

[CDKS21] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient homomorphic conversion between (ring) lwe ciphertexts. In *International Conference on Applied Cryptography and Network Security*, pages 460–479. Springer, 2021.

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachene. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *international conference on the theory and application of cryptology and information security*, pages 3–33. Springer, 2016.

[CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Tfhe: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020.

[CHK$^+$18] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 360–384. Springer, 2018.

[CHK$^+$19] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. A full rns variant of approximate homomorphic encryption. In *Selected Areas in Cryptography–SAC 2018: 25th International Conference, Calgary, AB, Canada, August 15–17, 2018, Revised Selected Papers 25*, pages 347–368. Springer, 2019.

[CHK$^+$21] Chi-Ming Marvin Chung, Vincent Hwang, Matthias J Kannwischer, Gregor Seiler, Cheng-Jhih Shih, and Bo-Yin Yang. Ntt multiplication for ntt-unfriendly rings: New speed records for saber and ntru on cortex-m4 and avx2. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 159–188, 2021.

[CJP21] Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In *International Symposium on Cyber Security Cryptography and Machine Learning*, pages 1–19. Springer, 2021.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International conference on the theory and application of cryptology and information security*, pages 409–437. Springer, 2017.

[DKRV19] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber. submission to the nist post-quantum cryptography standardization project. [NIS], 2019.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 617–640. Springer, 2015.

[DMKMS23] Gabrielle De Micheli, Duhyeong Kim, Daniele Micciancio, and Adam Suhl. Faster amortized fhew bootstrapping using ring automorphisms. *Cryptology ePrint Archive*, 2023.

[DvW21] Léo Ducas and Wessel van Woerden. Ntru fatigue: how stretched is overstretched? In *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part IV 27*, pages 3–32. Springer, 2021.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive*, 2012.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[GHS12] Craig Gentry, Shai Halevi, and Nigel P Smart. Homomorphic evaluation of the aes circuit. In *Advances in Cryptology–CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 850–867. Springer, 2012.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Annual Cryptology Conference*, pages 75–92. Springer, 2013.

[HP22] Daniel Heinz and Thomas Pöppelmann. Combined fault and dpa protection for lattice-based cryptography. *IEEE Transactions on Computers*, 2022.

[JLK$^+$21] Wonkyung Jung, Eojin Lee, Sangpyo Kim, Jongmin Kim, Namhoon Kim, Keewoo Lee, Chohong Min, Jung Hee Cheon, and Jung Ho Ahn. Accelerating fully homomorphic encryption through architecture-centric analysis and optimization. *IEEE Access*, 9:98772–98789, 2021.

[KKC$^+$23] Jongmin Kim, Sangpyo Kim, Jaewan Choi, Jaiyoung Park, Donghwan Kim, and Jung Ho Ahn. Sharp: A short-word hierarchical accelerator for robust and practical fully homomorphic encryption. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pages 1–15, 2023.

[KLSS23] Miran Kim, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. Accelerating he operations from key decomposition technique. *Cryptology ePrint Archive*, 2023.

[KPZ21a] Andrey Kim, Yuriy Polyakov, and Vincent Zucca. Revisiting homomorphic encryption schemes for finite fields. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 608–639. Springer, 2021.

[KPZ21b] Andrey Kim, Yuriy Polyakov, and Vincent Zucca. Revisiting homomorphic encryption schemes for finite fields. In *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part III 27*, pages 608–639. Springer, 2021.

[KS21] Kamil Kluczniak and Leonard Schild. Fdfb: Full domain functional bootstrapping towards practical fully homomorphic encryption. *arXiv preprint arXiv:2109.02731*, 2021.

[LMK$^+$23] Yongwoo Lee, Daniele Micciancio, Andrey Kim, Rakyong Choi, Maxim Deryabin, Jieun Eom, and Donghoon Yoo. Efficient fhew bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. In *Advances in Cryptology–EUROCRYPT 2023: 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part III*, pages 227–256. Springer, 2023.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)*, 60(6):1–35, 2013.

[LW23a] Feng-Hao Liu and Han Wang. Batch bootstrapping i: A new framework for simd bootstrapping in polynomial modulus. Springer-Verlag, 2023.

[LW23b] Feng-Hao Liu and Han Wang. Batch bootstrapping ii: Bootstrapping in polynomial modulus only requires $\tilde{o}(1)$ fhe multiplications in amortization. Springer-Verlag, 2023.

[MP21] Daniele Micciancio and Yuriy Polyakov. Bootstrapping in fhew-like cryptosystems. In *Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 17–28, 2021.

[MS18] Daniele Micciancio and Jessica Sorrell. Ring packing and amortized fhew bootstrapping. *Cryptology ePrint Archive*, 2018.

[MSM$^+$22] Chiara Marcolla, Victor Sucasas, Marc Manzano, Riccardo Bassoli, Frank HP Fitzek, and Najwa Aaraj. Survey on fully homomorphic encryption, theory, and applications. *Proceedings of the IEEE*, 110(10):1572–1609, 2022.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

[SEA22] Microsoft SEAL (release 4.0). https://github.com/Microsoft/SEAL, March 2022. Microsoft Research, Redmond, WA.

A NTT and iNTT Algorithm

The NTT and iNTT are shown in Algorithm 6 and Algorithm 7.

Algorithm 6 Algorithm for Number Theoretic Transform

Input:
A coefficient vector $\mathbf{a} = (a_0, a_1, ..., a_{N-1})$ for $a(X) \in \mathcal{R}_Q$.
A table ζ_{rev} computed by powers of ζ and stored in bit-reversed order, where $\zeta_{rev}[i] = \zeta^{\text{bit-reverse}(i)} \mod Q$.

Output:
An NTT vector of $\mathbf{a} \in \mathbb{Z}_Q^N$ in bit-reversed order.

```
1:  $t = N$
2:  for $(m = 1; m < 2N; m = 2m)$ do
3:      $t = t/2$
4:      for $(i = 0; i < m; i++)$ do
5:          $j_1 = 2 \cdot i \cdot t$
6:          $j_2 = j_1 + t - 1$
7:          for $(j = j_1; j \le j_2; j++)$ do
8:              $U = a_j$
9:              $V = a_{j+t} \cdot \zeta[m+i] \pmod Q$
10:             $a_j = U + V \pmod Q$
11:             $a_{j+t} = U - V \pmod Q$
12:         end for
13:     end for
14: end for
15: return NTT$(a)$.
```

Algorithm 7 Algorithm for Inverse Number Theoretic Transform

Input:
An NTT vector $\mathbf{a} \in \mathbb{Z}_Q^N$ in bit-reversed order.
A table ζ_{rev}^{-1} computed by powers of ζ^{-1} and stored in bit-reversed order, where $\zeta_{rev}^{-1}[i] = \zeta^{-\text{bit-reverse}(i)} \mod Q$.

Output:
A coefficient vector $\mathbf{a}$ of $a(X) \in \mathcal{R}_Q$ in normal order.

```
1:  $t = 1$
2:  for $(m = N; m > 1; m = m/2)$ do
3:      $j_1 = 0$
4:      $h = m/2$
5:      for $(i = 0; i < h; i++)$ do
6:          $j_2 = j_1 + t - 1$
7:          for $(j = j_1; j \le j_2; j++)$ do
8:              $U = a_j$
9:              $V = a_{j+t}$
10:             $a_j = U + V \pmod Q$
11:             $a_{j+t} = (U - V) \cdot \zeta_{rev}^{-1}[h+i] \pmod Q$
12:         end for
13:         $j_1 = j_1 + 2t$
14:     end for
15:     $t = 2t$
16: end for
17: for $(j = 0; j < N; j++)$ do
18:     $a_j = \frac{1}{N} \cdot a_j \pmod Q$
19: return $\mathbf{a}$ .
```

B Algorithm for modulus switching

We show the modulus switching algorithm as follows.

Lemma 2. *Input an LWE ciphertext* $\mathsf{ct} = (\mathbf{a}, b) \in \mathsf{LWE}^n_{\mathbf{s},Q}(m)$ *with the error variance* $\mathsf{Var}(\mathsf{err}(\mathsf{ct}))$*, the modulus switching operation outputs a new LWE ciphertext* ct' *with the error variance* $\mathsf{Var}(\mathsf{err}(\mathsf{ct}'))$.

Proof. Let the integers $Q > q > t$, the output ciphertext is

$$\begin{aligned}\mathsf{ct}' &= \mathsf{ModSwitch}(\mathsf{ct}) \\ &= (\lfloor \frac{q}{Q} \cdot \mathbf{a} \rceil, \lfloor \frac{q}{Q} \cdot b \rceil).\end{aligned}$$

By checking the decryption function, we can get $(\lfloor \frac{q}{Q} \cdot b \rceil - \left\langle \lfloor \frac{q}{Q} \cdot \mathbf{a} \rceil, \mathbf{s} \right\rangle) \bmod q = \frac{q}{Q} \cdot b - \left\langle \frac{q}{Q} \cdot \mathbf{a}, \mathbf{s} \right\rangle + \langle \mathbf{r}, \mathbf{s} \rangle + r = \frac{t}{q} \cdot m + \frac{q}{Q} \cdot e + \langle \mathbf{r}, \mathbf{s} \rangle + r$, where $r \in \mathbb{R}$ and $\mathbf{r} \in \mathbb{R}^n$ are in $[-1/2, 1/2]$. According to the central limit heuristic, the error is close to a gaussian distribution, and its variance is $\mathsf{Var}(\mathsf{err}(\mathsf{ct}')) \leq (\frac{q}{Q})^2 \cdot \mathsf{Var}(\mathsf{err}(\mathsf{ct})) + \frac{||\mathbf{s}||_2^2 + 1}{12}$, where the factor $\frac{1}{12}$ is the standard deviation of a uniform distribution in $[-1/2, 1/2]$.

□

C Bootstrapping with Modulus Raising and RNS

Algorithm 8 Blind Rotation with Modulus Raising and RNS

```
Input:
    An LWE sample $\mathsf{ct} = (\mathbf{a}, b) \in \mathsf{LWE}^n_{\mathbf{s},q}(m)$, where $q|2N$.
    A bootstrapping key $\mathsf{bsk}$ as shown in Equation 4.
Output:
    An RLWE ciphertext $\mathsf{acc} = \mathsf{RLWE}_{s,Q}(X^{-b+\sum_{i=0}^{n-1} a_i s_i})$ .
 1: Set $\mathsf{acc} = (0, X^{-b}) \in \mathcal{R}_Q^2$.
 2: for $i = 0$ to $n-1$ do
 3:         $\mathsf{acc}' = \mathsf{acc} \mod P \in \mathcal{R}_P^2$
 4:         $\mathsf{acc} = \mathrm{NTT}(\mathsf{acc})$ and $\mathsf{acc}' = \mathrm{NTT}(\mathsf{acc}')$
 5:         $\mathsf{acc} = \mathsf{acc} \odot [(\mathbf{1} + (X^{a_i} - 1) \odot \mathsf{bsk}_{i,0} + (X^{-a_i} - 1) \odot \mathsf{bsk}_{i,1})]_Q \in \mathcal{R}_Q^2$
 6:         $\mathsf{acc}' = \mathsf{acc}' \odot [(\mathbf{1} + (X^{a_i} - 1) \odot \mathsf{bsk}_{i,0} + (X^{-a_i} - 1) \odot \mathsf{bsk}_{i,1})]_P \in \mathcal{R}_P^2$
 7:         $\mathsf{acc} = \mathrm{iNTT}(\mathsf{acc})$ and $\mathsf{acc}' = \mathrm{iNTT}(\mathsf{acc}')$
 8:         $\mathsf{acc} = P^{-1} \cdot [\mathsf{acc} - (\mathsf{acc}' \mod Q)] \in \mathcal{R}_Q^2$
 9: end for
10: return $\mathsf{acc}$ .
```

Algorithm 8 shows the blind rotation by utilizing modulus raising and RNS techniques. To conduct polynomial multiplication between $\mathcal{R}_Q$ and $\mathcal{R}_{PQ}$, the conventional approach involves decomposing the bootstrapping key polynomials with modulus PQ into $\mathcal{R}_Q$ and $\mathcal{R}_P$ using the CRT, as demonstrated in Section 3.1. This decomposition process doubles the number of NTTs and Hadamard multiplications required. Moreover, similar to gadget decomposition, the division operation must be performed on the coefficient representation, which also requires the iNTT operations.

D AP Bootstrapping with Modulus Raising

We show the AP Bootstrapping with modulus raising in Algorithm 9. Moreover, similar to section 4.2, the hybrid external product method can also be used in this algorithm, where B_r is the decomposition base for LWE ciphertext.

Algorithm 9 AP Blind Rotation with Modulus Raising

```
Input:
    An LWE sample $\mathsf{ct} = (\mathbf{a}, b) \in \mathsf{LWE}^n_{\mathbf{s},q}(m)$, where $q|2N$.
    A bootstrapping key $\mathsf{bsk}$ as shown in Equation 7.
Output:
    An RLWE ciphertext $\mathsf{acc} = \mathsf{RLWE}_{s,Q}(X^{-b+\sum_{i=0}^{n-1} a_i s_i})$ .
 1: Set $\mathsf{acc} = (0, X^{-b}) \in \mathcal{R}_Q^2$.
 2: for $i = 0$ to $n - 1$ do
 3:     for $j = 0$ to $\log_{B_r}^{q} - 1$ do
 4:         $a_{i,j} = \lfloor a/B_r \rfloor \pmod{B_r}$
 5:         $\mathsf{acc} = \text{Com-NTT}(\mathsf{acc})$
 6:         $\mathsf{acc} \odot \mathsf{bsk}_{i,j,a_{i,j}}$
 7:         $\mathsf{acc} = \text{Com-iNTT}(\mathsf{acc})$
 8:         $\mathsf{acc} = \lfloor \frac{\mathsf{acc}_i}{P} \rceil \in \mathcal{R}_Q^2$
 9:     end for
10: end for
11: return $\mathsf{acc}$ .
```