Carousel: Fully Homomorphic Encryption from Slot Blind Rotation Technique

Seonhong Min and Yongsoo Song

Seoul National University, Seoul, Republic of Korea
{minsh,y.song}@snu.ac.kr

Abstract. Fully Homomorphic Encryption (FHE) enables secure computation of functions on ciphertexts without requiring decryption. Specifically, AP-like HE schemes exploit an intrinsic bootstrapping method called *blind rotation*. In blind rotation, a look-up table is homomorphically evaluated on the input ciphertext through the iterative multiplication of monomials. However, the algebraic structure of the multiplicative group of monomials imposes certain limitations on the input and output plaintext space: 1. only a fraction of the input plaintext space can be bootstrapped, 2. the output plaintext space is restricted to subsets of real numbers.

In this paper, we design a novel bootstrapping method called *slot blind rotation*. The key idea of our approach is to utilize the automorphism group instead of monomials. More specifically, the look-up table is encoded into a single polynomial using SIMD (Single Instruction Multiple Data) packing and is rotated via a series of homomorphic multiplications and automorphisms. This method achieves two significant advantages: 1. the entire input plaintext space can be bootstrapped, 2. a more broad output plaintext space, such as complex numbers or finite field/rings can be supported.

Finally, we present a new HE scheme leveraging the slot blind rotation technique and provide a proof-of-concept implementation. We also demonstrate the the benchmark results and provide recommended parameter sets.

1 Introduction

Fully Homomorphic Encryption (FHE) is a cryptographic primitive that enables an unlimited number of arithmetic operations on encrypted data without decryption. Thanks to this functionality, FHE has been employed as a universal tool for constructing various cryptographic protocols. Since Gentry's pioneering construction [14], which introduced the ingenious bootstrapping technique, numerous HE schemes have been proposed, including BGV [5], BFV [4,11], AP/FHEW [1,10], TFHE [7] and CKKS [6].

Among these schemes, the AP-like cryptosystems (e.g., FHEW [10], LMKCDEY [21], TFHE [7]) are widely used due to their low latency and intrinsic functionality known as *functional bootstrapping*, which enables the evaluation of arbitrary univariate functions. This functionality stems from their unique bootstrapping method called *blind rotation*. In blind rotation, a look-up table is homomorphically rotated by the 'phase' $b + \langle \mathbf{a}, \mathbf{s} \rangle \pmod q$ for the input LWE ciphertext $(b, \mathbf{a}) \in \mathbb{Z}_q^{n+1}$ under secret $\mathbf{s} \in \mathbb{Z}^n$. Then, with pre-assigned values in the look-up table, the function evaluation of the input message corresponds to the first index of the output look-up table. In the bootstrapping process of existing AP-like cryptosystems, blind rotation is instantiated by sequentially multiplying monomials to a polynomial whose coefficients are the look-up table values. Since homomorphic monomial multiplication introduces only additive noise, the bootstrapping can be performed with small noise growth, proportional to the length of the input ciphertext.

However, despite the efficiency of AP-like blind rotation method, it has some limitations concerning the input and output message domain. Firstly, AP-like blind rotation methods can only bootstrap ciphertexts whose input message resides within a $\phi(M)/M$ fraction of the message space, where the scheme's base ring is $R = \mathbb{Z}[X]/\Phi_M(X)$. To address this issue, many works [8,19,30,9,26,29] focused on resolving the limitation over the negacyclic ring $R = \mathbb{Z}[X]/(X^N + 1)$ for some power-of-two N. However, all the methods require at least two bootstrappings. Secondly, the output message domain is restricted to subsets of real numbers, as the look-up table values are encoded into the polynomial coefficients.

1.1 Our Results

In this paper, we address the limitations of AP-like blind rotation methods by introducing a new framework called *slot blind rotation*. In the BFV cryptosystem, finite field or ring elements can be encoded into the 'slots' of a plaintext using the isomorphism between $\mathbb{Z}_\mathfrak{p}[X]/\Phi_M(X)$ and $\prod_{i=1}^N \mathbb{Z}_\mathfrak{p}[X]/F_i(X)$, where the factorization of $\Phi_M(X)$ modulo $\mathfrak{p}$ is given by $\prod_{i=1}^N F_i(X)$ for some polynomials F_i. This encoding strategy allows the homomorphically permutataion of the encoded messages in slots, without consuming any multiplicative levels, using the homomorphic evaluation of automorphisms.

Our main observation is that when the permutation structure forms a cyclic group, blind rotation can be instantiated over the slots. Suppose the slot structure is isomorphic to a cyclic group $\mathbb{Z}_N$, and a cyclic rotation by index i is instantiated with an automorphism Ψ_i. Then, we can homomorphically compute $\Psi_{[b+\langle \mathbf{a},\mathbf{s}\rangle]_N}(\mathcal{F})$ for the input ciphertext $(b, \mathbf{a}) \in \mathbb{Z}_N^{n+1}$ under a secret key $\mathbf{s} \in \mathbb{Z}^n$ and an encoded look-up table $\mathcal{F}$. If the function value associated with the phase $[b + \langle \mathbf{a}, \mathbf{s} \rangle]_N$ is assigned to the $[b + \langle \mathbf{a}, \mathbf{s} \rangle]_N$-th entry of the look-up table $\mathcal{F}$, the blind rotation can be completed by extracting the first entry of the rotated look-up table $\Psi_{[b+\langle \mathbf{a},\mathbf{s}\rangle]_N}(\mathcal{F})$.

However, homomorphically rotating the look-up table by index $b + \langle \mathbf{a}, \mathbf{s} \rangle$ is quite challenging without knowing the secret information $\mathbf{s}$. We achieve this by adopting the homomorphic MUX gate from GINX/CGGI bootstrapping [13,7]. The homomorphic MUX gate selects between two values m_0 and m_1, with respect to the binary value b, with an RGSW encryption of b. More precisely, the MUX gate computes $m_0 + (m_1 - m_0) \cdot b$ to select m_b. Using this relation, rotation by index $b + \langle \mathbf{a}, \mathbf{s} \rangle$ can be instantiated by rotating the look-up table $\mathcal{F}$ by index $a_i \cdot s_i$ at i-th iteration, where a_i and s_i are i-th entry of the input ciphertext and the secret key, respectively. Assuming that the secret keys are sampled from a binary distribution, rotation by index $a_i \cdot s_i$ is equivalent to selecting between $\mathcal{F}$ and $\Psi_{a_i}(\mathcal{F})$ based on the value s_i. Consequently, the look-up table can be successfully rotated in the presence of RGSW encryptions of each component s_i $(0 \le i < n)$ of the secret $\mathbf{s} = (s_0, \dots s_{n-1})$. As a result, we can bootstrap an LWE ciphertext of dimension n under a binary secret using n RGSW ciphertexts and N rotation keys.

Building upon this novel blind rotation technique, we design a new homomorphic encryption scheme that leverages the subring structure proposed by Arita and Handa [2,3]. To enhance the efficiency of our scheme, we incorporate optimization techniques from [20]. We also provide a prototype implementation to validate the correctness of the proposed method and present the benchmark results.

1.2 Related Work

In [27], the authors introduced a blind rotation technique for BFV scheme to achieve an amortized bootstrapping. Their method involves evaluating automorphisms over a generating element in the finite field and subsequently extracting the rotated index using the homomorphic norm function. On the other hand, Liu et al. [22,23] proposed a different blind rotation technique which also utilizes automorphisms to construct an amortized bootstrapping of LWE ciphertexts. Their approach leverages the special structure of the base ring.

2 Preliminary

2.1 Notation

For a positive integer m, $\Phi_M(X)$ denotes the m-th cyclotomic polynomial. The euler phi function is denoted by $\phi(\cdot)$. The l_2 norm and infinity norm of a vector $\mathbf{x} \in \mathbb{R}^n$ are denoted by $\|\mathbf{x}\|_2$ and $\|\mathbf{x}\|_\infty$. We denote an inner product of two vectors $\mathbf{a}$ and $\mathbf{b}$ by $\langle \mathbf{a}, \mathbf{b} \rangle$. The uniform distribution from a finite set $\mathcal{S}$ is denoted by $\mathcal{U}(\mathcal{S})$. A size of a finite set $\mathcal{S}$ is denoted by $|S|$. $\delta_a(x)$ denotes the delta function, which outputs 1 when $x = a$, and zero otherwise. For a polynomial $p(X)$, the constant term of p is denoted by $(p)_0$. $[x]_n$ denotes the remainder of x modulo n, *i.e.,* $[x]_n = x \pmod n$.

2.2 LWE assumption and Variants

In the following, we describe the decisional LWE [28] and its ring variants [25,2,18].

Definition 1 (Decisional LWE). *Let $n, Q > 0$ be positive integers, and ψ an error distribution over $\mathbb{Z}$. Then, for $\mathbf{s} \in \mathbb{Z}^n$, we define the LWE distribution $\mathcal{D}^{\mathsf{LWE}}_{\mathbf{s},\psi}(\mathbf{s})$ over $\mathbb{Z}_q^{n+1}$ as follows: sample $\mathbf{a} \leftarrow \mathcal{U}(\mathbb{Z}_Q^n)$ and $e \leftarrow \psi$, and output $(b, \mathbf{a})$ where $b = -\langle \mathbf{a}, \mathbf{s} \rangle + e \pmod{Q}$. The decisional LWE problem calls for distinguishing $\mathcal{D}^{\mathsf{LWE}}_{\mathbf{s},\psi}$ from $\mathcal{U}(\mathbb{Z}_Q^{n+1})$.*

Ring-LWE (RLWE) is LWE problem instantiated over the ring of integers of some cyclotomic number field K.

Definition 2 (Decisional RLWE). *Let $m, q > 0$ be positive integers, and ψ an error distribution over $R := \mathbb{Z}[X]/\Phi_M(X)$. Then, for $t \in R$, we define the RLWE distribution $\mathcal{D}^{\mathsf{RLWE}}_{t,\psi}(t)$ over $R_Q^2 = (R/QR)^2$ as follows: sample $a \leftarrow \mathcal{U}(R_Q)$ and $e \leftarrow \psi$, and output (b, a) where $b = -at + e \pmod{Q}$. The decisional RLWE problem calls for distinguishing $\mathcal{D}^{\mathsf{RLWE}}_{t,\psi}$ from $\mathcal{U}(R_Q^2)$.*

The RLWE problem can be generalized over a subring of the ring of integers of a cyclotomic field, which is invariant to a specific set of automorphisms.

Definition 3 (Decisional RLWE Over Decomposition Rings). *Let R be a decomposition ring, Q be a positive integer,and χ a key distribution over R. Then, for $t \in R$, we define the RLWE distribution $\mathcal{D}^{\mathsf{RLWE}}_{\chi,\psi}(t)$ over $R_Q^2 = (R/QR)^2$ as follows: sample $a \leftarrow \mathcal{U}(R_Q)$ and $e \leftarrow \psi$, and output (b, a) where $b = -at + e \pmod{Q}$. The decisional RLWE problem calls for distinguishing $\mathcal{D}^{\mathsf{RLWE}}_{t,\psi}$ from $\mathcal{U}(R_Q^2)$.*

2.3 Basic Lattice-based Cryptosystems

In this subsection, we describe the baseline lattice-based cryptosystems, namely LWE, Gadget Encryption, GSW and their ring variants.

Based on the hardness of (R)LWE problem, we can construct (R)LWE cryptosystems. LWE ciphertexts and RLWE ciphertexts have the forms $(b, \mathbf{a}) \in \mathbb{Z}_Q^{n+1}$ and $(b, a) \in R_Q^2$ respectively, where n is the LWE dimension and $R_q = R/QR$ for some decomposition field R.

Now, we describe the gadget encryption. First, let us define the notion of *gadget decomposition*, which is a widely-used technique to reduce the noise growth during homomorphic computations. Suppose that the ciphertext modulus Q is a product of two integers $P, q > 0$, *i.e.,* $Q = Pq$. Then, if there exists a map $h : \mathbb{Z}_q \to \mathbb{Z}^d$ for an integer vector $\mathbf{g} \in \mathbb{Z}^d$ which satisfies the following conditions for any $a \in \mathbb{Z}_q$ and some real $\mathcal{B} > 0$, we call $h, \mathbf{g}$ and P a *decomposition function*, a *gadget vector* and a *special modulus*, respectively.

$$\langle h(a), \mathbf{g} \rangle = a \text{ and } \|h(a)\|_\infty \leq \mathcal{B}.$$

In the ring setting, the decomposition function $h : R_q \to R^d$ corresponding to the gadget vector $\mathbf{g} \in \mathbb{Z}^d$ can be easily obtained by applying the gadget decomposition to each coefficients of the ring element. We now present the description of gadget encryption and RGSW scheme below.

• $\underline{\mathsf{GadEnc.Enc}(t;\mu)}$: Given the RLWE secret $t \in R$ and a message $\mu \in R$, sample $\mathbf{a} \leftarrow \mathcal{U}(R_Q)^d$ and $\mathbf{e} \leftarrow \psi^d$. Output $\mathbf{C} = [\mathbf{b} \mid \mathbf{a}] \in R_Q^{d\times 2}$ where $\mathbf{b} = -\mathbf{a}\cdot t + \mu \cdot P\mathbf{g} + \mathbf{e} \pmod{Q}$.

• $\underline{\mathsf{RGSW.Enc}(t;\mu)}$: Given the RLWE secret $t \in R$ and a message $\mu \in R$, sample $\mathbf{a} \leftarrow \mathcal{U}(R_Q)^{2d}$ and $\mathbf{e} \leftarrow \psi^{2d}$. Output $\overline{\mathbf{C}} = [\mathbf{b} \mid \mathbf{a}] \in R_Q^{2d\times 2}$ where $\mathbf{b} = -\mathbf{a}\cdot t + \mu \cdot \begin{bmatrix} P\mathbf{g} & \mathbf{0} \\ \mathbf{0} & P\mathbf{g} \end{bmatrix} + \mathbf{e} \pmod{Q}$.

Now we present the gadget product and the external product, which are multiplicative operations between a ring element and a gadget encryption, and between an RLWE ciphertext and an RGSW ciphertext, respectively.

Definition 4 (Gadget Product). *Let $\mathbf{C} \in R^{d\times 2}$ be a gadget encryption and $a \in R_Q$ be a ring element. The gadget product, denoted by $a \odot \mathbf{C}$ is defined as follows:*

$$(a, \mathbf{C}) \mapsto a \odot \mathbf{C} := h(\lfloor \frac{a}{P} \rceil)^\top \cdot \mathbf{C} \pmod{Q}.$$

Definition 5 (External Product). *Let $\overline{\mathbf{C}} \in R^{2d\times 2}$ be an RGSW ciphertext and $c = (b, a) \in R_Q^2$ be an RLWE ciphertext. The external product, denoted by $a \boxdot \mathbf{C}$ is defined as follows:*

$$(c, \mathbf{C}) \mapsto c \boxdot \mathbf{C} := \begin{bmatrix} h(\lfloor \frac{b}{P} \rceil)^\top & h(\lfloor \frac{a}{P} \rceil)^\top \end{bmatrix} \cdot \mathbf{C} \pmod{Q}.$$

Below, we briefly show the correctness of the gadget product and external product. First, we introduce the notion of *phase*, which is an affine function corresponding to the (R)LWE secret. The phase for LWE ciphertext, $\varphi_{\mathbf{s}} : \mathbb{Z}_Q^{n+1} \mapsto \mathbb{Z}_Q$, is defined by $\varphi_{\mathbf{s}}(b, \mathbf{a}) = b + \langle \mathbf{a}, \mathbf{s} \rangle \pmod{Q}$ and the phase for RLWE ciphertext, $\varphi_t : R_Q^2 \mapsto R_Q$, is defined by $\varphi_t(b, a) = b + a \cdot t \pmod{Q}$.

Now, for a gadget encryption $\mathbf{C}$ of μ under secret t, the phase of the output of the gadget product is

$$\varphi_t(a \odot \mathbf{C}) = \langle h(\lfloor \frac{a}{P} \rceil), \mathbf{C} \cdot \begin{bmatrix} 1 \\ t \end{bmatrix} \rangle \approx \langle h(\lfloor \frac{a}{P} \rceil), \mu \cdot P\mathbf{g} \rangle = P \lfloor \frac{a}{P} \rceil \cdot \mu \approx a \cdot \mu \pmod{Q}.$$

Similarly, for an RGSW ciphertext $\overline{\mathbf{C}}$ of μ under secret t, the phase of the output of the external product is

$$\begin{aligned} \varphi_t((b, a) \boxdot \overline{\mathbf{C}}) &= \langle \begin{bmatrix} h(\lfloor \frac{b}{P} \rceil) & h(\lfloor \frac{a}{P} \rceil) \end{bmatrix}, \overline{\mathbf{C}} \cdot \begin{bmatrix} 1 \\ t \end{bmatrix} \rangle \\ &\approx \langle \begin{bmatrix} h(\lfloor \frac{b}{P} \rceil) & h(\lfloor \frac{a}{P} \rceil) \end{bmatrix}, [\mu \cdot P\mathbf{g} \ \ \mu \cdot t \cdot P\mathbf{g}] \rangle \\ &= \mu \cdot (P \lfloor \frac{b}{P} \rceil + P \lfloor \frac{a}{P} \rceil \cdot t) \approx \mu \cdot \varphi_t(b, a) \pmod{Q}. \end{aligned}$$

2.4 BFV cryptosystem

The BFV [11] is a fully homomorphic encryption scheme that supports arithmetic operations between finite field/ring elements. In the BFV scheme, the message and the noise are separated by storing the message in the highest bits and the noise in the lowest bits (LSB). Let Q and $\mathfrak{p}$ be the ciphertext and plaintext moduli, respectively. Then, the phase of an LWE-based BFV encryption of a message $m \in \mathbb{Z}_{\mathfrak{p}}$ is given by $\frac{Q}{\mathfrak{p}} \cdot m + e$, for some small error $e \in \mathbb{Z}$. Similarly, for the base ring R, the phase of an RLWE-based BFV encryption of a message $\mu \in R_{\mathfrak{p}} = R/\mathfrak{p}R$ is $\frac{Q}{\mathfrak{p}} \cdot \mu + e$ for some noise $e \in R$ with small $\|e\|_\infty$.

RLWE-based BFV scheme supports the SIMD (Single-Instruction-Multi-Data) arithmetic. Suppose that the plaintext modulus $\mathfrak{p}$ is a prime, and the base ring is given by $R = \mathbb{Z}[X]/\Phi_M(X)$. Then, $\Phi_M(X)$ can be factored into k irreducible F_i of degree d, where d is the multiplicative order of $\mathfrak{p}$ modulo M and $k = \phi(M)/d$. Consequently, $R_{\mathfrak{p}}$ is isomorphic to a direct product of finite fields $GF(\mathfrak{p}^d)^k$, referred to as the *slots*. Based on this isomorphism, we can pack a vector of finite field elements into a single polynomial over $R_{\mathfrak{p}}$. This result can be even further generalized. Suppose that the plaintext modulus $\mathfrak{p} = p^r$ for some prime p and an integer $r > 1$. Then, one can apply Hensel lifting on the factorization of Φ_M over $\mathbb{Z}_p$ to compute the factorization of Φ_M modulo p^r. This leads to an isomorphism between $R_{\mathfrak{p}}$ and a direct product of polynomial rings modulo p^r. On the other hand, Arita and Handa [2] proposed using a subring of $\mathbb{Z}[X]/\Phi_M(X)$ to pack an integer vector $\mathbb{Z}_{p^r}^k$ into a polynomial with only k coefficients. Let the base ring R be a subring of $\mathbb{Z}[X]/\Phi_M(X)$ that is invariant under the automorphism $X \mapsto X^p$. Then, the elements of the base ring R can be represented with only k coefficients and can encode a vector over $\mathbb{Z}_{\mathfrak{p}}^k$, hence achieve the optimal packing density 1.

With packing, the point-wise addition and multiplication of vectors can be instantiated via the addition and multiplication of polynomials. Besides these operations, rotation is another operations that can be performed homomorphically. In essence, rotation shifts the encoded vector by a certain index using ring automorphisms. Depending on the choice of the base ring, the rotation may involve multiplications with mask vectors. In this paper, we will focus on the case where an arbitrary rotation can be instantiated with a single automorphism.

The RLWE-based BFV cryptosystem is described below.

- $\mathsf{BFV.Setup}(1^\lambda)$: Given the security parameter λ, set the base ring R, the ciphertext, plaintext and speical moduli Q, p^r and P, the key and noise distributions χ and ψ over R, the gadget vector and decomposition, $\mathbf{g} \in \mathbb{Z}^d$ and h. Return $\mathsf{pp} = (Q, p^r, P, \chi, \psi, \mathbf{g}, h)$

In the following descriptions, N is the dimension of the base ring R, Ψ_i denotes the automorphism that left shifts the encoded vector by index i over the base ring R, and the isomorphism between the plaintext space R_{p^r} and the message space $\mathfrak{M}$ is denoted by σ.

- $\mathsf{BFV.KeyGen}(\mathsf{pp})$:

– Sample $t \leftarrow \chi$.
– Set $\mathsf{rtk} = \{\mathsf{rtk}_i\}_{0 \leq i < N}$ for $\mathsf{rtk}_i \leftarrow \mathsf{GadEnc.Enc}(\Psi_{-i}(t); t)$.
– Set $\mathsf{rlk} \leftarrow \mathsf{GadEnc.Enc}(t^2; t)$.

• $\mathsf{BFV.Enc}(t; \mu)$: Sample $a \leftarrow \mathcal{U}(R_Q)$ and $e \leftarrow \psi$. Return $(b, a) \in R_Q^2$ where $b = -a \cdot t + \lfloor \frac{Q}{p^r} \cdot \mu \rceil + e \pmod Q$

• $\mathsf{BFV.Dec}(t; \mathsf{ct})$: Given $\mathsf{ct} \in R_Q^2$, return $\lfloor \frac{p^r}{Q} \cdot \varphi_t(\mathsf{ct}) \rceil \pmod{p^r}$.

• $\mathsf{BFV.Ecd}(\mathbf{m})$: Given the vector $\mathbf{m} \in \mathfrak{S}$, compute and return $\mu = \sigma^{-1}(\mathbf{m}) \in R_p$.

• $\mathsf{BFV.Dcd}(\mu)$: Given the polynomial $\mu \in R_{p^r}$, compute and return $\mathbf{m} = \sigma(\mu) \in \mathfrak{S}$.

• $\mathsf{BFV.Rot}(\mathsf{rtk}_i; \mathsf{ct})$ Given the ciphertext $\mathsf{ct} = (b, a) \in R_Q^2$, compute and return

$$\mathsf{ct}' = (\Psi_i(b), 0) + \Psi_i (a \odot \mathsf{rtk}_i) \pmod Q$$

for the index i of the given rotation key rtk_i. For simplicity, we will write $\mathsf{Rot}(\mathsf{rtk}_i; \mathsf{ct})$ in the later sections.

3 Prior Works

At the heart of AP-like cryptosystems, an intrinsic bootstrapping method called *blind rotation* is leveraged to support arbitrary function evaluation over encrypted messages. The underlying idea of blind rotation is to embed the ciphertext modulus into a cyclic multiplicative subgroup $\{X^i \mid 0 \leq i < M\} \cong \mathbb{Z}_M$ of the polynomial ring $R_Q := \mathbb{Z}_Q[X]/\Phi_M(X)$.

Let us explain in detail. For simplicity, we assume that the plaintext modulus p divides the ciphertext modulus Q. Given a function $f : \mathbb{Z}_p \to \mathbb{Z}_p$ that we want to evaluate over the input ciphertext, we define a polynomial $\mathcal{F}(X) \in \mathbb{Z}_Q[X]$ called the *test vector*, such that the constant term of $\mathcal{F}(X) \cdot X^i$ equals $f(\lfloor \frac{p}{M} \cdot i \rceil)$ for $0 \leq i < \phi(M)$. Note that such a polynomial $\mathcal{F}$ always exists over the ring R_Q. See [17] for more information.

Then, the constant term of $\mathcal{F}(X) \cdot X^{\lfloor \frac{M}{p} \cdot m \rceil + e}$ is $\frac{Q}{p} \cdot f(m)$ for any $m \in \mathbb{Z}_p$ and $e \in \mathbb{Z} \cap [-\frac{M}{2p}, \frac{M}{2p}]$ such that $0 \leq \lfloor \frac{M}{p} m \rceil + e < \phi(M)$. Now, let $(b, \mathbf{a}) = (b, a_0, \ldots, a_{n-1}) \in \mathbb{Z}_M^{n+1}$ be the input BFV encryption of message $m \in \mathbb{Z}_p$ under secret $\mathbf{s} \in \mathbb{Z}^n$. Then, $b + \langle \mathbf{a}, \mathbf{s} \rangle = \lfloor \frac{M}{p} \cdot m \rceil + e \pmod M$ for some noise $e \in \mathbb{Z}$. Therefore

$$\left(\mathcal{F} \cdot X^{b + \sum_{i=0}^{n-1} a_i s_i}\right)_0 = \left(\mathcal{F} \cdot X^{\lfloor \frac{M}{p} \cdot m \rceil + e}\right)_0 = \frac{Q}{p} \cdot f(m).$$

From this observation, any function f can be evaluated by multiplying the monomials $X^{a_i s_i}$ to $\mathcal{F} \cdot X^b$ and then extracting the constant term. To realize this with homomorphic encryption, we first set the ‘accumulator’ as a trivial encryption

$(0, \mathcal{F} \cdot X^b)$. Then, the monomials $X^{a_i s_i}$ are iteratively multiplied to the accumulator using the RGSW/Gadget Encryption cryptosystems.

There are two known ways of multiplying $X^{a_i s_i}$, namely AP/FHEW [1,10] and GINX/TFHE [13,7]. In AP/FHEW bootstrapping, the monomial $X^{a_i s_i}$ is directly multiplied to the accumulator, while GINX/TFHE bootstrapping utilizes homomorphic MUX gates to multiply the monomials. We will explain these methods in detail below.

3.1 AP/FHEW bootstrapping

In AP/FHEW bootstrapping, the monomial $X^{a_i s_i}$ is computed using RGSW encryptions of X^{s_i}. A naïve way to compute $X^{a_i s_i}$ is to simply provide encryptions of every possible $X^{a_i s_i}$. However, this approach requires an overwhelming space complexity of $O(M \cdot n)$. Ducas and Micciancio [10] showed that it is sufficient to provide RGSW encryptions of $X^{2^j s_i}$ for $0 \leq i < n$ and $0 \leq j < \lceil \log M \rceil$. To multiply $X^{a_i s_i}$ to the accumulator, we first compute a binary representation of a_i, *i.e.*, $a_i = \sum_{j=0}^{\lceil \log M \rceil - 1} a_{i,j} \cdot 2^j$. For each $0 \leq j < \lceil \log M \rceil$, we multiply the RGSW encryption of $X^{2^j s_i}$ to the accumulator if $a_{i,j} = 1$. By repeating this procedure for every $0 \leq i < n$, we can complete the bootstrapping.We remark that there is a time-space tradeoff from the use of B-digit representation instead of binary representation.

Recently, Lee et al. [21] introduced the automorphism operation to compute $X^{a_i s_i}$. Provided the RGSW encryptions of X^{s_i} and the automorphism keys, one can evaluate the map $X \mapsto X^{a_i}$ to the encryption of X^{s_i} to obtain $X^{a_i s_i}$. Note that this map is a well-defined automorphism if and only if $\gcd(a_i, M) = 1$, so this may require additional multiplications of X^{s_i}. In practice, their method includes an optimization technique to reduce the time and space complexity. For simplicity, let us assume that the automorphism group $\mathbb{Z}_M^\times$ is cyclic and each entry a_i of ciphertext resides in $\mathbb{Z}_M^\times$. Then, each a_i can be written as a power of the generator g of $\mathbb{Z}_M^\times$, *i.e.*, $a_i = g^j$. Now, Let us define sets $I_i := \{j \mid a_j = g^i\}$ for $0 \leq i < \phi(M)$. Then, we can deduce the following relation:

$$\begin{aligned}\sum_{i=0}^{n-1} a_i s_i &= \sum_{i=0}^{\phi(M)-1} g^i \cdot \sum_{j \in I_i} s_j \\ &= \sum_{j \in I_0} s_j + g \left(\sum_{j \in I_1} s_j + g \left(\sum_{j \in I_2} s_j + \cdots + g \sum_{j \in I_{\phi(M)-1}} s_j \right) \right) \pmod{M}.\end{aligned}$$

From this relation, the bootstrapping can be performed as follows: 1. Set the accumulator $\mathcal{F}(X^g) \cdot (X^g)^b$. 2. Every i-th iteration, multiply the RGSW encryptions of X^{s_j} to the accumulator, where $j \in I_{\phi(M)-i}$. 3. Evaluate the Frobenius map $X \mapsto X^g$ to the accumulator. We remark that it is easy to generalize this idea to cases where the base ring has a non-cyclic automorphism group.

3.2 GINX/TFHE bootstrapping

During GINX/TFHE blind rotation, the accumulator is updated through iterative evaluations of homomorphic MUX gate. If the key $\mathbf{s}$ has binary coefficients, multiplying $X^{a_i s_i}$ to the accumulator ACC is essentially selecting between ACC and $X^{a_i} \cdot \mathsf{ACC}$ based on the value of s_i. The arithmetic representation of the MUX gate between ACC and X^{a_i} is given as follows:

$$X^{a_i s_i} \cdot \mathsf{ACC} = (1 - s_i) \cdot \mathsf{ACC} + s_i \cdot X^{a_i} \cdot \mathsf{ACC} = \mathsf{ACC} + s_i \cdot (X^{a_i} - 1) \cdot \mathsf{ACC}$$

Using this representation, the blind rotation is instantiated with repetitive external products, given the RGSW encryptions of s_i. This idea can be further generalized to handle arbitrary key distributions [16]. Let $\{u_j\}_{j \in \mathcal{J}_i}$ be the set of possible values of s_i for an index set $\mathcal{J}_i$. Then, we can derive the following equation for $X^{a_i s_i}$ and the accumulator.

$$X^{a_i s_i} \cdot \mathsf{ACC} = \sum_{j \in \mathcal{J}_i} X^{a_i u_j} \cdot \delta_{u_j}(s_i) \cdot \mathsf{ACC}$$

From this equation, we can compute $X^{a_i s_i}$ efficiently with only a single gadget decomposition [20]. More precisely, the gadget decomposition $h(\mathsf{ACC})$ is reused for the external products between the RGSW encryptions of $\delta_{u_j}(s_j)$ and ACC. Then, the sum of each external product is computed. We remark that it is sufficient to provide only $|\mathcal{J}_i| - 1$ keys, since the remaining key can be homomorphically computed from the fact that their sum is always one.

4 Slot Blind Rotation

In this section, we present a new framework for blind rotation, called *slot blind rotation*. Our framework exploits the structure of the automorphism group of the base ring, in contrast to AP-like cryptosystems, which rely on the multiplicative group of monomials. Specifically, we modify the GINX bootstrapping so that the accumulator is homomorphically rotated using the homomorphic MUX gate. As a result, we naturally achieve full-domain bootstrapping with a single bootstrapping. Moreover, the range of the function evaluated is extended from the commutative ring $\mathbb{Z}_p$ to complex numbers or a polynomial ring.

4.1 Homomorphic MUX gate

In the construction of slot blind rotation, the homomorphic MUX gate from the GINX/TFHE bootstrapping is employed. Briefly speaking, the homomorphic MUX gate selects between two (or more) input RLWE ciphertexts with respect to some binary value b, encrypted with an RGSW encryption. Specifically, given an RGSW encryption $\overline{\mathbf{C}} \in R_Q^{2d \times 2}$ of a binary value $b \in \mathbb{B}$ under secret t and RLWE ciphertexts $\mathsf{ct}_0, \mathsf{ct}_1 \in R_Q^2$, the homomorphic MUX gate computes the following.

$$\mathsf{ct}' := \mathsf{ct}_0 + (\mathsf{ct}_1 - \mathsf{ct}_0) \boxdot \overline{\mathbf{C}} \pmod{Q}$$

Then, the output ciphertext ct' satisfies the equation $\varphi_t(\mathsf{ct}') \approx \varphi_t(\mathsf{ct}_b) \pmod Q$, since

$$\varphi_t(\mathsf{ct}') \approx \varphi_t(\mathsf{ct}_0) + b \cdot \varphi_t(\mathsf{ct}_1 - \mathsf{ct}_0) \approx (1-b) \cdot \varphi_t(\mathsf{ct}_0) + b \cdot \varphi_t(\mathsf{ct}_1) \pmod Q.$$

Note that $(1-b) \cdot \varphi_t(\mathsf{ct}_0) + b \cdot \varphi_t(\mathsf{ct}_1)$ is $\varphi_t(\mathsf{ct}_0)$ if $b = 0$, and $\varphi_t(\mathsf{ct}_1)$ if $b = 1$, the equation $\varphi_t(\mathsf{ct}') \approx \varphi_t(\mathsf{ct}_b)$ holds indeed.

4.2 Our Framework

Let n be the input LWE dimension, N be the number of the slots of the baseline scheme, $\mathfrak{M}$ be the message space, p and t be the input and output plaintext moduli and R be the base ring. Assume that the slot structure of the baseline scheme forms a cyclic group $\mathbb{Z}_N$, our framework aims to rotate the test vector $\mathcal{F}$ by index $b + \langle \mathbf{a}, \mathbf{s} \rangle$ for an input LWE ciphertext $(b, \mathbf{a}) \in \mathbb{Z}_N^{n+1}$ under secret $\mathbf{s} \in \mathbb{Z}^n$. In other words, $\Psi_{b+\langle \mathbf{a},\mathbf{s}\rangle}(\mathcal{F})$ is homomorphically computed. To be precise, Given the evaluated function $f : \mathbb{Z}_p \to \mathfrak{M}$, we generate a polynomial $\mu(X) \in R_t$ which is an encoding of a vector

$$\left(f(\lfloor \frac{p}{N} \cdot 0 \rceil), f(\lfloor \frac{p}{N} \cdot 1 \rceil), \ldots, f(\lfloor \frac{p}{N} \cdot (N-1) \rceil)\right) \in \mathfrak{M}^N.$$

It is worth noting that there is a redundancy in the look-up table. In other words, it is composed of p blocks of N/p consecutive function values. Precisely, if the index i is confined in the interval $\left[\frac{N}{p} \cdot m - \frac{N}{2p}, \frac{N}{p} \cdot m + \frac{N}{2p}\right]$, it follows $f(\lfloor \frac{p}{N} \cdot i \rceil) = f(m)$. We utilize this fact to bootstrap a BFV ciphertext. If $(b, \mathbf{a})$ is a BFV encryption of message $m \in \mathbb{Z}_p$ under secret $\mathbf{s}$, it holds that $b + \langle \mathbf{a}, \mathbf{s} \rangle = \lfloor \frac{N}{p} m \rceil + e \pmod N$ for some small $e \in \mathbb{Z} \in [-\frac{N}{2p}, \frac{N}{2p}]$. Then, it follows that the first entry of $\sigma(\Psi_{b+\langle \mathbf{a},\mathbf{s}\rangle}(\mu))$ is $f\left(\lfloor \frac{p}{N} \cdot (\lfloor \frac{N}{p} m \rceil + e) \rceil\right) = f(m)$. From this observation, for the test vector $\mathcal{F} = \frac{Q}{p} \cdot \mu$, we can perform functional bootstrapping by homomorphically computing $\Psi_{b+\langle \mathbf{a},\mathbf{s}\rangle}(\mathcal{F})$ by rotating $\Psi_b(\mathcal{F})$ by the index $a_i \cdot s_i$ for $0 \le i < n$, and extract the value contained in the first slot.

To achieve this, we want to rotate the accumulator ACC by index $a_i s_i$ for $0 \le i < n$. Suppose that each entry s_i of the secret is drawn from a binary distribution, rotating ACC by $a_i s_i$ is essentially to rotate ACC by the index a_i if s_i, and do nothing otherwise. From this observation, this can be realized with an iterative evaluation of homomorphic MUX gate. More precisely, we generate the rotation key $\mathsf{rtk} = \{\mathsf{rtk}_i\}_{0 \le i < N}$, and the blind rotation key $\mathsf{brk} = \{\mathsf{brk}_i\}_{0 \le i < n}$ where brk_i is an RGSW encryption of s_i. Then, for every i-th iteration,

$$\mathsf{ACC} \leftarrow \mathsf{ACC} + (\mathsf{Rot}(\mathsf{rtk}_{a_i}; \mathsf{ACC}) - \mathsf{ACC}) \boxdot \mathsf{brk}_i$$

is computed. The exact algorithm for some test vector $\mathcal{F}$ is provided in Alg. 1.

Algorithm 1 Slot Blind Rotation

Input: The blind rotation key brk, the rotation key rtk, LWE ciphertext $\mathbf{c} = (b, a_0, \ldots, a_{n-1}) \in \mathbb{Z}_N^{n+1}$ and the test vector $\mathcal{F} \in R_Q$.

1: $\mathsf{ACC} \leftarrow (\Psi_b(\mathcal{F}), 0) \in R_Q^2$
2: **for** $i = 0 : n-1$ **do**
3: $\quad \mathsf{ACC} \leftarrow \mathsf{ACC} + (\mathsf{Rot}(\mathsf{rtk}_{a_i}; \mathsf{ACC}) - \mathsf{ACC}) \boxdot \mathsf{brk}_i$
4: **end for**

Output: Return ACC

4.3 Comparison

We analyze the computational complexity and key size of our method, and compare it to the existing AP-like bootstrapping methods. As the computational costs of our framework and previous bootstrapping methods are dominated by homomorphic multiplications, namely external and gadget products, we estimate the computational complexity by counting the number of these operations.

Our framework requires a single external and gadget products in each iteration, leading to a total of n external and gadget products. On the other hand, we measure the key size by counting the number of RGSW and gadget encryptions. To encrypt each key component s_i $(0 \leq i < n)$, n RGSW encryptions are required. Additionally, we require $N-1$ rotation keys for every rotation Ψ_i $(1 \leq i < N)$, which are essentially $N-1$ gadget encryptions.

It is worth noting that there exists a space and time trade-off for the rotation keys. Specifically, we can provide only $\log N$ rotation keys which corresponds to $1, 2, \ldots, 2^{\lceil \log N \rceil - 1}$-th rotations. Then, the i-th rotation can be instantiated by a composition of these rotations using the binary representation of i. To be precise, given that the binary representation of i is $i = \sum_{j=0}^{\lceil \log N \rceil - 1} i_j \cdot 2^j$, the accumulator can be rotated by index 2^j if i_j is nonzero. By using this trade-off, number of the used gadget encryptions is reduced to $O(\log N)$, while the time complexity is increased by a factor of $\lceil \log N \rceil$. We also remark that this can be generalized to b-ary representation, with $b \log_b N$ keys.

Method	# RGSW	# Gadget Encryption	# External products	# Gadget Product
AP/FHEW [1,10]	$n \cdot \log N$	0	$n \cdot \log N$	0
LMKCDEY (Alg.4) [21]	$n+1$	$w+1$	$\sim \frac{3}{2}n + 1$	$\sim n + \frac{N-n}{w}$
GINX/CGGI [13,7]	n	0	n	0
Ours	n	$N-1$	n	n

Table 1. Comparison between our framework and AP-like methods

In Table 1, the number of RGSW and gadget encryptions, and the number of the external and gadget products required to perform blind rotation are given, for previous AP-like bootstrapping algorithms and ours. In the table, n denotes the length of input LWE ciphertexts, and N denotes the ring dimension. For LMKCDEY scheme, w denotes the windowing parameter, which is used to reduce the time complexity for bootstrapping.

Assuming that we fix the parameters n and N for all the schemes. Then, our scheme requires a smaller key size than AP/FHEW, and it needs a larger key size than the LMKCDEY and GINX/CGGI due to the extra rotation keys. Note that this can be mitigated by the aforementioned time and space trade-off.

Recall that the external product is essentially two gadget products, the time complexity of AP/FHEW, LMKCDEY, GINX/CGGI and our scheme is given as $2n \log N$, $\sim 4n + (N - n)/w + 1$, $2n$ and $3n$, respectively, in terms of the number of gadget products. Therefore, our scheme is expected to be $\sim 0.66 \log N$ times faster than FHEW, more than 1.33 times faster than LMKCDEY at most, and 1.5 times slower than GINX for a single bootstrapping.

Now we describe the advantages our new framework has, over the existing AP-like bootstrapping methods. Firstly, it naturally supports the full-domain functional bootstrapping (FDFB), without any additional operations other than a single bootstrapping. Also, our framework supports a broader set of the plaintext space for the output ciphertext. Finally, the multiplication can be naturally performed, without any additional operations. We describe them in detail below.

Full-Domain Functional Bootstrapping (FDFB) In AP-like cryptosystems, there are some limitations on the input ciphertexts, or the evaluated function. For the cyclotomic degree M of the base ring, AP-like cryptosystems can cover only a $\phi(M)/M$ fraction of the plaintext space due to the structure of the multiplicative group generated by the monomials. Especially when using the power-of-two cyclotomic ring of integers, the bootstrappable plaintext space is only half of the original plaintext space. While it is possible to evaluate negacyclic functions over the whole plaintext space using the negacyclic property of the power-of-two cyclotomic ring, it is not commonly used in the applications.

To circumvent this problem, several works have focused on bootstrapping methods that can cover the entire plaintext space, known as *full-domain functional bootstrapping* (FDFB) [8,19,30,24,9,26]. These FDFB methods achieve the full-domain functionality by performing more than two bootstrappings. Their approach typically involves handling the top bit which makes the input message overflow the half fraction of plaintext space. However, since their methods require multiple bootstrappings, they can be computationally expensive. In contrast, our method can perform FDFB with only a single bootstrapping inherently, as it embeds the ciphertext modulus into the cyclic automorphism group. This embedding allows us to naturally cover the entire plaintext space without the need for multiple bootstrapping steps, making our approach more efficient.

General Plaintext Space Our slot blind rotation framework can leverage a broader set of output plaintext spaces, including finite field/ring elements and complex numbers. In previous AP-like cryptosystems, the message space was always required to be a subset of the real numbers $\mathbb{R}$ since it needed to be encoded in the coefficients. On the other hand, our scheme can accommodate any message space as the range, as long as it is supported by some SIMD HE scheme. For example, we can perform blind rotation over finite field elements using the BFV as a baseline scheme, and even handle complex numbers via the CKKS scheme [6].

Natural Multiplication In previous AP-like methods, multiplication between the ciphertexts was not naturally supported. In prior works, LWE ciphertexts were converted into RLWE ciphertexts for efficient multiplication, exploiting key-switching from LWE to RLWE [19], or homomorphic trace evaluation [29,15]. On the other hand, our slot blind rotation framework can perform ciphertext multiplication naturally, without any modification to the input ciphertexts. Recall that the message is stored in the first slot. Since the slot structure supports SIMD arithmetic, the message remains in the first slot even after multiplication. Therefore, the ciphertext form remains unchanged without the need for any additional algorithms.

5 Fully Homomorphic Encryption From Slot Blind Rotation

In this section, we propose a new fully homomorphic encryption scheme for small integers, based on the slot blind rotation technique. Our scheme is built upon the decomposition ring of a ring of integers of some prime cyclotomic field, as proposed by Arita and Handa [2].

5.1 Subring of Prime Cyclotomic Rings

As specified in the beginning of this section, our new FHE scheme targets the small integers. Recall that our framework embeds the ciphertext modulus into the slot structure for blind rotation. To optimize the bootstrapping performance, we need to maximize the number of the slots packed in a single ciphertext. In other words, we aim to have the packing density as close to one as possible, which is given by

$$\#\text{ of the slots } / \text{ } \#\text{ of the coefficients.}$$

However, standard SIMD packing strategy does not offer a dense packing for small integers since the cyclotomic polynomials do not fully split modulo small integers. Instead, the SIMD packing structure is given by a direct product of finite field. To circumvent this problem, we utilize the BFV variant over the decomposition ring [2,3]. Their idea is built upon the fact that if we pack integers as constant finite field elements in each slot, the resulting polynomial resides in a subring of the cyclotomic ring that is invariant to the Frobenius automorphism. Moreover, any element in such a subring can be represented with the same number of the coefficients as the number of slots. By using a subring that is invariant to the Frobenius map as the base ring instead of the entire cyclotomic ring, we can achieve an optimal packing with a packing density of one.

The main challenge in utilizing the subring is to find efficient algorithms for encoding/decoding and fast multiplication. Naïvely, these algorithms can be conducted in the original ring with many coefficients, but it may result in performance deterioration in homomorphic computation. Therefore, it is necessary to construct efficient algorithms for basic operations over the subring. Below, we review the basics and the algorithms for the decomposition ring, which were presented in [2].

For two distinct primes p and M, the base ring R is chosen as a subring of the cyclotomic ring $\mathbb{Z}[X]/\Phi_M(X)$, so that it is invariant to the automorphism $X \mapsto X^p$. Let $N = \phi(M)/o = (M-1)/o$, where o is the multiplicative order of p modulo M. A ring element of R_{p^r} can be represented with N coefficients, and also can pack N elements from the commutative ring $\mathbb{Z}_{p^r}$. In the following sections, we assume that $-1 \in \langle p \rangle$ as an element of the multiplicative group $\mathbb{Z}_M^\times$.

η- and ξ-bases We describe the η-basis and ξ-basis, which are the equivalents of the coefficient and evaluation forms over the subring. In other words, addition, subtraction and multiplication over the ring R with respect to the η-basis translate to point-wise computations of the coefficients with respect to the ξ-basis.

Let us begin by describing the η-basis. First, observe that R is essentially a set of stable elements under the group action of automorphism subgroup $G_p := \langle X \mapsto X^p \rangle$. Since the automorphism group of $\mathbb{Z}[X]/\Phi_M(X)$ is a cyclic group, there exists some $g \in \mathbb{Z}_M^\times$ such that the automorphism group of $\mathbb{Z}[X]/\Phi_M(X)$ is generated by the map $X \mapsto X^g$. It is easy to show that $\langle X \mapsto X^p \rangle = \langle X \mapsto X^{g^N} \rangle$ because the multiplicative order of p is o and $N \cdot o = \phi(M) = M-1$. Under the action of G_p, the monomial basis $\{X, \ldots, X^{M-1}\}$ of $\mathbb{Z}[X]/\Phi_M(X)$ is decomposed into N orbits $\mathcal{O}_i = \{X^{g^{i+jN}} \mid 0 \le j < o\}$ for $0 \le i < N$. Since any ring element $a \in R$ is stable under the group action G_p, its coefficients in each orbit $\mathcal{O}_i$ remain the same. Hence, a can be written as a linear combination of $\eta_i := \sum_{j=0}^{o-1} X^{g^{i+jN}}$ for $0 \le i < N$.

Now, we describe the ξ-basis. The evaluation form of a ring element $a \in R$ is given by $a(\zeta_M^{g^0}), \ldots, a(\zeta_M^{g^{M-2}}) \in \mathbb{C}$, where ζ_M is M-th root of unity. Then, for all $0 \le i < N$,

$$a(\zeta_M^{g^i}) = a(\zeta_M^{g^{i+N}}) = \cdots = a(\zeta_M^{g^{i+(o-1)N}})$$

as a is stable under the group action G_p. Moreover, since the conjugate automorphism $X \mapsto X^{-1}$ is contained in G_p, $a(\zeta_M^{g^i})$ is always real. Hence, it only requires N evaluation points $a(\zeta_M^{g^0}), \ldots, a(\zeta_M^{g^{N-1}}) \in \mathbb{R}$ to represent the ring element a. In other words, a can be written as a linear combination

$$a = \sum_{i=0}^{N-1} a(\zeta_M^{g^i}) \cdot \xi_i$$

for some polynomials $\xi_0, \dots, \xi_{N-1} \in \mathbb{R}[X]/\Phi_M(X)$, such that

$$\xi_i(\zeta_M^{g^j}) = \begin{cases} 1 & \text{if } \ i = j \\ 0 & \text{otherwise} \end{cases}$$

for $0 \le i, j < N$.

Automorphism Now we discuss how the automorphism acts on each η- and ξ- bases. In the following sections, we set the i-th rotation $\Psi_i(X) = X \mapsto X^{g^i}$. Given a ring element $a = \sum_{j=0}^{N-1} a_j \eta_j = \sum_{j=0}^{N-1} b_j \xi_j \in R$, we have

$$\Psi_i(a) = \sum_{j=0}^{N-1} a_{[-i+j]_N} \eta_j = \sum_{j=0}^{N-1} b_{[i+j]_N} \xi_j,$$

since

$$\begin{aligned} \Psi_i(a) &= \sum_{j=0}^{N-1} a_j \cdot \Psi_i(\eta_j) = \sum_{j=0}^{N-1} a_j \cdot \sum_{k=0}^{o-1} (X^{g^i})^{g^{j+kN}} \\ &= \sum_{j=0}^{N-1} a_j \cdot \sum_{k=0}^{o-1} X^{[i+j]_N + kN} = \sum_{j=0}^{N-1} a_j \cdot \eta_{[i+j]_N} = \sum_{j=0}^{N-1} a_{[-i+j]_N} \eta_j \end{aligned}$$

and

$$\begin{aligned} \Psi_i(a) &= \sum_{j=0}^{N-1} \Psi_i\left(a(X)\right)(\zeta^{g^j}) \cdot \xi_j = \sum_{j=0}^{N-1} a(X^{g^i})(\zeta^{g^j}) \cdot \xi_j \\ &= \sum_{j=0}^{N-1} a(\zeta^{g^{i+j}}) \cdot \xi_j = \sum_{j=0}^{N-1} a(\zeta^{g^{[i+j]_N}}) \cdot \xi_j = \sum_{j=0}^{N-1} b_{[i+j]_N} \cdot \xi_j. \end{aligned}$$

In other words, the rotation by i acts as a cyclic left shift of the coefficient vector in the η-basis, and a cyclic right shift of the coefficient vector in the ξ-basis by the index i.

Conversion between $\boldsymbol{\eta}$- and $\boldsymbol{\xi}$-bases The ring multiplication over the subring R can be performed with $O(N \log N)$ time complexity using fast basis conversion between the η- and ξ-bases. Let us commence by investigating the conversion from η-basis to ξ-basis. Given a ring element a, let $\mathbf{a} = (a_0, \dots, a_{N-1})$ and

$\mathbf{b} = (b_0, \ldots, b_{N-1})$ be the cofficient vectors with respect to η- and ξ-bases, respectively. Then, the i-th coefficient b_i in ξ-basis can be computed as

$$b_i = a(\zeta_M^{g^i}) = \sum_{j=0}^{N-1} a_j \cdot \eta_j(\zeta_M^{g^i}) = \sum_{j=0}^{N-1} a_j \cdot \Psi_i\left(\eta_j(\zeta_M)\right) = \sum_{j=0}^{N-1} a_j \cdot \eta_{[i+j]_N}(\zeta_M)$$

by definition, for any $0 \leq i < N$. Therefore, the basis change can be written as a linear transformation

$$\begin{bmatrix} b_{N-1} \\ \vdots \\ b_1 \\ b_0 \end{bmatrix} = \begin{bmatrix} \eta_{N-2}(\zeta_M) & \ldots & \eta_0(\zeta_M) & \eta_{N-1}(\zeta_M) \\ \vdots & \ddots & & \vdots \\ \eta_0(\zeta_M) & \ldots & \eta_2(\zeta_M) & \eta_1(\zeta_M) \\ \eta_{N-1}(\zeta_M) & \ldots & \eta_1(\zeta_M) & \eta_0(\zeta_M) \end{bmatrix} \cdot \begin{bmatrix} a_{N-1} \\ \vdots \\ a_1 \\ a_0 \end{bmatrix}.$$

Note that evaluating this linear transformation is essentially computing a cyclic convolution of the vectors $(a_0, a_{N-1}, \ldots, a_1)$ and $(\eta_i(\zeta_M))_{0 \leq i < N}$. Therefore, the basis conversion can be efficiently performed with a single Fast Fourier Transform (FFT) and inverse Fast Fourier Transform (iFFT), hence with $O(N \log N)$ time complexity. Similarly, the conversion from ξ-basis to η-basis can be realized with a cyclic convolution of the vectors $(b_0, b_{N-1}, \ldots, b_1)$ and $\left(\frac{1}{M}\left(\eta_i(\overline{\zeta_M}) - o\right)\right)_{0 \leq i < N}$. For further details, refer to Sec. 3 of [2]. It is worth noting that since $-1 \in \langle p \rangle$, all the coefficient vectors $\mathbf{a}$ and $\mathbf{b}$ and the convolved vectors $(\eta_i(\zeta_M))_{0 \leq i < N}$ and $\left(\frac{1}{M}(\eta_i(\overline{\zeta_M}) - o)\right)_{0 \leq i < N}$ are real. Hence, the base convolution can be executed with approximately half the complexity, using the real FFT trick.

Packing The 'packing' over R_{p^r} can be realized as a conversion from the ξ-basis to η-basis, but modulo p^r. In R_{p^r}, the ξ-basis corresponds to a set of polynomials $\{\tau_i\}_{0 \leq i < N}$, referred to as the *resolution of unities*. Let the factorization of the M-th cyclotomic polynomial $\Phi_M(X)$ be given by $\Phi_M(X) = \prod_{i=0}^{N-1} F_i(X)$, where F_i are degree o polynomials. We define a polynomial $G_0 = \prod_{i \neq 0} F_i(X)$, which serves as the 'complement' of the polynomial F_0. Using the Chinese Remainder Theorem of ring ideals, the i-th resolution of unity τ_i can be determined. Specifically, we let $\tau_0 = \left[G_0^{-1}\right]_{F_0} \cdot G_0$, and $\tau_i = \Psi_i(\tau_0)$ for $0 < i < N$. This construction ensures that

$$\tau_i = \begin{cases} 1 & \text{if } \ i = j \\ 0 & \text{otherwise} \end{cases} \pmod{F_j}$$

for all $0 \leq i, j < N$, and forms the ξ-basis.

Given the resolution of unities $\{\tau_i\}_{0 \leq i \leq N}$, a vector $\mathbf{a} = (a_0, \ldots, a_{N-1}) \in \mathbb{Z}_{p^r}^N$ can be packed into a ring element $a = \sum_{i=0}^{N-1} a_i \cdot \tau_i$. The ring arithmetic on a directly translates to point-wise arithmetic over the packed vector $\mathbf{a}$. The packing and unpacking operations can also be efficiently computed using convolution, similar to the conversions in R. Specifically, a vector $\mathbf{m} = (m_0, \ldots, m_{N-1}) \in \mathbb{Z}_{p^r}^N$ can be packed into a ring element in R_{p^r} by computing a cyclic convolution between the coefficient vector of τ_0 and the vector $(m_0, m_{N-1}, \ldots, m_1)$. Similarly, a ring element $\mu = \sum_{i=0}^{N-1} \mu_i \cdot X^i \in R_{p^r}$ can be unpacked by computing a convolution between the vectors $(\mu_0, \mu_{N-1}, \ldots, \mu_1) \in \mathbb{Z}_{p^r}^N$ and the coefficient vector of $M \cdot \tau_0 - o$ if $-1 \in \langle p \rangle$.

In the scheme description, the conversion from the η-basis to the τ-basis is denoted by $\sigma : R_{p^r} \to \mathbb{Z}_{p^r}^N$, and the inverse conversion is denoted by σ^{-1}.

Canonical Norm and Variance We define the infinity and canonical norms for ring elements in R. The infinity norm, denoted by $\|a\|_\infty$, is defined as $\|\mathbf{a}_\eta\|_\infty$ and the canonical norm, denoted by $\|a\|_\xi$, is defined as $\|\mathbf{a}_\xi\|_\infty$, where $\mathbf{a}_\eta$ and $\mathbf{a}_\xi$ denote the coefficient vectors of a in the η- and ξ-bases, respectively. We provide some useful facts about the canonical and infinity norm over R.

Proposition 1 (Norm Bound [2]). *For any $a, b \in R$, the following holds.*

1. $\|ab\|_\xi \leq \|a\|_\xi \cdot \|b\|_\xi$.
2. $\|a\|_\infty \leq \sqrt{\frac{N}{M}} \cdot \|a\|_\xi \approx \frac{1}{\sqrt{o}} \|a\|_\xi$.

In addition, we define the variance of the canonical embedding over R, denoted by Var_ξ. The canonical variance, $\mathrm{Var}_\xi(a)$, denotes the (maximum) variance of the coefficients of a in ξ-basis. The canonical variance for ring elements drawn from some widely-used distributions is presented below.

Lemma 1 (Canonical Variance [3]).

1. *For $a_i \leftarrow \mathcal{U}([-B, B] \cap \mathbb{Z})$ $(0 \leq i < N)$, $\mathrm{Var}_\xi(\sum_{i=0}^{N-1} a_i \eta_i) = \frac{1}{3} oNB^2$.*
2. *For $a_i \leftarrow \mathcal{D}_\sigma$ $(0 \leq i < N)$, $\mathrm{Var}_\xi(\sum_{i=0}^{N-1} a_i \eta_i) = \sigma^2 oN$.*
3. *For $a_i \leftarrow \mathcal{U}(\{0, \pm 1\})$ $(0 \leq i < N)$, $\mathrm{Var}_\xi(\sum_{i=0}^{N-1} a_i \eta_i) \leq oN$.*

Finally, we also remark that it holds that $\mathrm{Var}_\xi(ab) \leq \|a\|_\xi^2 \cdot \mathrm{Var}_\xi(b)$ for any ring element $a \in R$ and random variable $b \in R$. This inequality can be easily derived from the fact that the multiplication on the ξ-vector is an element-wise multiplication of the coefficient vectors.

5.2 Scheme Description

$\mathcal{D}_\alpha$ and $\mathcal{D}_\beta$ denote the discrete Gaussian distributions with standard deviations α and β, defined over $\mathbb{Z}$ and R in η-basis, respectively. More specifically, sampling from $\mathcal{D}_\beta$ is equivalent to sampling each coefficient from a discrete Gaussian distribution with standard deviation β, with respect to the η-basis. In our scheme, we choose the noise distribution $\psi = \mathcal{D}_\beta$.

• $\mathsf{Setup}(1^\lambda)$: Given the security parameter λ, set the LWE dimension n, the base ring R, the ciphertext, plantext and special moduli Q, p^r and P, P', the noise parameter α and β, and the gadget vector and decomposition $\mathbf{g} \in \mathbb{Z}^d$ and h with respect to the special modulus P for RLWE, and the gadget vector and decomposition $\mathbf{g}' \in \mathbb{Z}^{d'}$ and h' with respect to the special modulus P' for LWE.

• $\mathsf{KeyGen}(\mathsf{pp})$:

– Sample $\mathbf{s} \leftarrow \mathcal{U}(\{0,1\}^n)$ and $t_i \leftarrow \mathcal{U}(\{-1,0,1\})$ for $0 \le i < N$, and set $t = \sum_{i=0}^{N-1} t_i \cdot \eta_i$.
– Set $\mathsf{rtk} = \{\mathsf{rtk}_i\}_{0 \le i < N}$ for $\mathsf{rtk}_i \leftarrow \mathsf{GadEnc.Enc}(\Psi_{-i}(t); t)$ for $0 \le i < N$.
– Set $\mathsf{brk} = \{\mathsf{brk}_i\}_{0 \le i < n}$ for $\mathsf{brk}_i \leftarrow \mathsf{RGSW.Enc}(t; s_i)$ for $0 \le i < n$.
– Sample $\mathsf{ksk}_{i,1} \leftarrow \mathcal{U}(\mathbb{Z}_Q^{d' \times n})$ and $\mathbf{e}_i \leftarrow \mathcal{D}_\alpha^{d'}$, and set $\mathsf{ksk}_i = (\mathsf{ksk}_{i,0} | \mathsf{ksk}_{i,1}) \in \mathbb{Z}_Q^{d' \times (n+1)}$ where $\mathsf{ksk}_{i,0} = -\mathsf{ksk}_{i,1} \cdot \mathbf{s} + P' \mathbf{g}' \cdot t_i + \mathbf{e}_i \pmod{Q}$ for $0 \le i < N$. Set $\mathsf{ksk} = \{\mathsf{ksk}_i\}_{0 \le i < N}$.

• $\mathsf{Enc}(t; m)$: Given a message $m \in \mathbb{Z}_{p^r}$, sample $a \leftarrow \mathcal{U}(R_Q)$ and $e \leftarrow \mathcal{D}_\beta$. Return $(b, a) \in R_Q^2$ where $b = -a \cdot t + \frac{Q}{p^r} \cdot m + e \pmod{Q}$.

• $\mathsf{Dec}(t; \mathsf{ct})$: Given $\mathsf{ct} \in R_Q^2$, return the first entry of $\sigma(\lfloor \frac{p^r}{Q} \cdot \varphi_t(\mathsf{ct}) \rceil)$.

We note that the basic ciphertext form is RLWE in our scheme, and will only care about the first entry of the encoded vector in the decryption.

• $\mathsf{Add}(\mathsf{ct}_1, \mathsf{ct}_2)$: Given two encryptions $\mathsf{ct}_1, \mathsf{ct}_2 \in R_Q^2$, compute and output $\mathsf{ct}' = \mathsf{ct}_1 + \mathsf{ct}_2 \pmod{Q}$.

• $\mathsf{Sub}(\mathsf{ct}_1, \mathsf{ct}_2)$: Given two encryptions $\mathsf{ct}_1, \mathsf{ct}_2 \in R_Q^2$, compute and output $\mathsf{ct}' = \mathsf{ct}_1 - \mathsf{ct}_2 \pmod{Q}$

The bootstrapping of our scheme consists of three steps: 1. Sample Extraction, 2. Key Switching and 3. Blind Rotation. Our scheme is distinguished from TFHE by two algorithms: the sample extraction and the blind rotation. We describe those three algorithms in detail below.

Sample Extraction Unlike the previous AP-like cryptosystems, our sample extraction step requires generating an LWE ciphertext which encrypts the first entry of the slot. Naïvely, this can be achieved by computing a homomorphic matrix-vector multiplication between the FFT matrix and the encrypted vector, however, it is computationally expensive. Therefore, we propose another algorithm which is computationally much cheaper.

First, we remark that there should exist at least one coefficient of τ_0 that is coprime to the base prime p. Otherwise, τ is a multiple of p and therefore it is zero modulo p, which is not true. Therefore, without loss of generality, we assume that the constant term of τ_0 is some integer $\alpha \in \mathbb{Z}_{p^r}$ and $p \nmid \alpha$. (Since the slot has a cyclic structure, we can simply shift all the slots by some index so that the first slot corresponds to the ξ-basis of which the constant term is not multiple of p.) Now, suppose that we are given an encoding μ of the vector $(m_0, \ldots, m_{N-1}) \in \mathbb{Z}_{p^r}$, *i.e.*, $\mu = \sum_{i=0}^{N-1} m_i \tau_i$. As α is coprime to p^r, the inverse α^{-1} exists in the commutative ring $\mathbb{Z}_{p^r}$, and therefore $(\alpha^{-1}\tau_0) \cdot \mu = \sum_{i=0}^{N-1} (\alpha^{-1}\tau_0) \cdot m_i \tau_i = \alpha^{-1} m_0 \tau_0$ by the orthogonality of ξ-basis. Since the constant term of τ_0 in the η-basis is α, the constant term of $\alpha^{-1} m_0 \tau_0$ is m_0. Therefore, we can simply multiply $[\alpha^{-1}\tau_0]_{p^r}$ to the input ciphertext and extract the constant term in η-basis homomorphically. The algorithm is given below.

• $\mathsf{SampleExtract}(\mathsf{ct})$: Given the ciphertext $\mathsf{ct} = (b, a) \in R_Q^2$, output $\mathbf{c} = (b_0, \mathbf{a}) \in \mathbb{Z}_Q^{N+1}$ where b_0 is the constant term of $[\alpha^{-1}\tau_0]_{p^r} \cdot b \pmod Q$ in η-basis, and $\mathbf{a}$ is the first row of the coefficient matrix of $[\alpha^{-1}\tau_0]_{p^r} \cdot a \pmod Q$ in η-basis.

Key Switching The key-switching step is essentially the same to the previous AP-like cryptosystems.

• $\mathsf{KeySwitch}(\mathsf{ksk}; \mathbf{c})$: Given the LWE ciphertext $\mathbf{c} = (b, a_0, \ldots, a_{N-1}) \in \mathbb{Z}_Q^{N+1}$, compute and output $\mathbf{c}' = (b, \mathbf{0}) + \sum_{i=0}^{N-1} h'(a_i)^\top \cdot \mathsf{ksk}_i \pmod Q$.

Blind Rotation The blind rotation algorithm directly follows the slot blind rotation framework we proposed in Sec. 4.

• $\mathsf{BlindRotate}(\mathsf{brk}, \mathsf{rtk}; \mathbf{c}, f)$: Run and output of the result of Alg. 2.

Algorithm 2 Blind Rotation

Input: The blind rotation key brk, rotation key rtk, LWE ciphertext $\mathbf{c} = (b, a_0, \ldots, a_{n-1}) \in \mathbb{Z}_Q^{n+1}$ and the evaluated function $f : \mathbb{Z}_p \to \mathbb{Z}_p$.

1: $\tilde{b} \leftarrow \lfloor \frac{N}{Q} \cdot b \rceil \in \mathbb{Z}_N$ and $\tilde{a}_i = \lfloor \frac{N}{Q} \cdot a_i \rceil \in \mathbb{Z}_N$ for $0 \le i < n$.
2: $\mathcal{F} \leftarrow \frac{Q}{p^r} \cdot \sum_{i=0}^{N-1} f(\lfloor \frac{p^r}{N} \cdot i \rceil) \cdot \tau_i \in R_Q$
3: $\mathsf{ACC} \leftarrow (\Psi_{\tilde{b}}(\mathcal{F}), 0) \in R_Q^2$
4: **for** $i = 0 : n - 1$ **do**
5: $\quad \mathsf{ACC} \leftarrow \mathsf{ACC} + (\mathsf{Rot}(\mathsf{ACC}; \mathsf{rtk}_{\tilde{a}_i}) - \mathsf{ACC}) \boxdot \mathsf{brk}_i$
6: **end for**

Output: Return ACC

Finally, the bootstrapping algorithm of our scheme can be written as follows.

• $\mathsf{Bootstrap}(\mathsf{brk}, \mathsf{rtk}, \mathsf{ksk}; \mathsf{ct}, f)$: Given the ciphertext $\mathsf{ct} \in R_Q^2$, function $f : \mathbb{Z}_p \to \mathbb{Z}_p$, the blind rotation key brk, the rotation key rtk and the key-switching key ksk, run Alg. 3.

Algorithm 3 Functional Bootstrapping

Input: RLWE ciphertext $\mathsf{ct} \in R_Q^2$, function $f : \mathbb{Z}_p \to \mathbb{Z}_p$, the blind rotation key brk, rotation key rtk, and the key-switching key ksk.

1: $\mathbf{c} \leftarrow \mathsf{SampleExtract}(\mathsf{ct})$
2: $\mathbf{c}' \leftarrow \mathsf{KeySwitch}(\mathbf{c}, \mathsf{ksk})$
3: $\mathsf{ct}' \leftarrow \mathsf{BlindRotate}(\mathbf{c}', f, \mathsf{rtk}, \mathsf{brk})$

Output: Return ct'

Security The security of our scheme relies on the LWE assumption over the decomposition ring [2,3]. On the other hand, note that the blind rotation key brk encrypts the LWE secret $\mathbf{s}$ under ring secret t, the rotation key rtk encrypts the 'rotated' ring secret t under t, and the key-switching key ksk encrypts the ring secret t under the LWE secret $\mathbf{s}$. Therefore, we require a circular security assumption for (R)LWE to have the proposed scheme semantically secure.

5.3 Correctness

We provide the noise analysis of our scheme below. In the following, $\mathcal{B}$ and $\mathcal{B}'$ denote the bounds of the result of the gadget decomposition h and h', corresponding to the gadget vectors $\mathbf{g}$ and $\mathbf{g}'$, respectively. Additionally, the variance of a random variable $x \in \mathbb{R}$ is denoted by $\mathrm{Var}(x)$ and the (maximum) variance of each entry of the coefficient vector of $x \in R$ in ξ-basis by $\mathrm{Var}_\xi(x)$. We make some assumptions on the gadget decomposition. If a is drawn from a uniform distribution $\mathcal{U}(\mathbb{Z}_Q)$, we suppose that the decomposition result $h(a)$ and the remainder $a - h(\lfloor \frac{a}{P} \rceil)^\top \cdot P\mathbf{g}$ follow the uniform distributions $\mathcal{U}(\mathbb{Z} \cap [-\mathcal{B}, \mathcal{B}])$ and $\mathcal{U}(\mathbb{Z} \cap [-\frac{P}{2}, \frac{P}{2}])$, respectively. Note that these two uniform distributions have variances of $\frac{1}{3}B^2$ and $\frac{1}{12}P^2$, respectively. Also, we approximate the variance of the distributions $\mathcal{D}_\alpha$ and $\mathcal{D}_\beta$ with α^2 and β^2, respectively, as α and β are chosen as sufficiently large real numbers in practice. Now, we provide the noise analysis for the basic operations for our scheme.

Lemma 2 (Gadget Product). *Let $\mathbf{C} \in R^{d\times 2}$ be a fresh gadget encryption of $\mu \in R$, and $a \leftarrow \mathcal{U}(R_Q)$. Then,*

$$\mathrm{Var}_\xi(\mu \cdot a - \varphi_t(a \odot \mathbf{C})) \leq \frac{1}{12} oNP^2 \|\mu\|_\xi^2 + \frac{1}{3} do^2N^2\mathcal{B}^2\beta^2.$$

Proof. Let $\mathbf{e} \in R^d$ be the error vector of the gadget encryption $\mathbf{C}$ under secret t. Then, we have

$$\begin{aligned} \mu \cdot a - \varphi_t(a \odot \mathbf{C}) &= \mu \cdot a - h(\lfloor \frac{a}{P} \rceil)^\top \cdot (\mu \cdot P\mathbf{g} + \mathbf{e}) \\ &= \mu \cdot \left(a - h(\lfloor \frac{a}{P} \rceil)^\top \cdot P\mathbf{g}\right) - h(\lfloor \frac{a}{P} \rceil)^\top \cdot \mathbf{e}. \end{aligned}$$

Since $a - h(\lfloor \frac{a}{P} \rceil)^\top \cdot P\mathbf{g}$ follows a uniform distribution, $\mathrm{Var}_\xi \left(a - h(\lfloor \frac{a}{P} \rceil)^\top \cdot P\mathbf{g}\right) = \frac{1}{12} oNP^2$ by Lemma 1. Therefore,

$$\mathrm{Var}_\xi \left(\mu \cdot (a - h\lfloor \frac{a}{P} \rceil)^\top \cdot P\mathbf{g}\right) \leq \frac{1}{12} oNP^2 \|\mu\|_\xi^2$$

. On the other hand, the canonical variance of $h(\lfloor \frac{a}{P} \rceil)$ is $\frac{1}{3} oN\mathcal{B}^2$ and the canonical variance of $\mathbf{e}$ is $\beta^2 oN$ by Lemma 1. Therefore, $\mathrm{Var}_\xi \left(h(\lfloor \frac{a}{P} \rceil)^\top \cdot \mathbf{e}\right) = \frac{1}{3} do^2N^2\mathcal{B}^2\beta^2$. It concludes the proof. □

The noise growth from rotation operation is given as follows.

Corollary 1 (Rotation). *Let* $\mathsf{ct} \in R_Q^2$ *be an RLWE ciphertext, and* rtk_i *be the* i*-th rotation key for secret* t*. Then,*

$$\mathrm{Var}_\xi\left(\Psi_i\left(\varphi_t(\mathsf{ct})\right) - \varphi_t\left(\mathsf{Rot}(\mathsf{rtk}_i;\mathsf{ct})\right)\right) \le \frac{1}{12}o^2N^2P^2 + \frac{1}{3}do^2N^2\mathcal{B}^2\beta^2.$$

Proof. Let $\mathsf{ct} = (b, a) \in R_Q^2$. Then,

$$\begin{aligned}\Psi_i(\varphi_t(\mathsf{ct})) - \varphi_t(\mathsf{Rot}(\mathsf{rtk}_i;\mathsf{ct})) &= \Psi_i(b + at) - \varphi_t\left((\Psi_i(b), 0) + \Psi_i\left(a \odot \mathsf{rtk}_i\right)\right) \\ &= \Psi_i(b) + \Psi_i(a \cdot t) - \Psi_i(b) - \Psi_i\left(\varphi_{\Psi_{-i}(t)}(a \odot \mathsf{rtk}_i)\right) \\ &= \Psi_i\left(a \cdot t - \varphi_{\Psi_{-i}(t)}(a \odot \mathsf{rtk}_i)\right).\end{aligned}$$

By the proof of Lemma. 2, $\mathrm{Var}_\xi\left(a \cdot t - \varphi_{\Psi_{-i}(t)}(a \odot \mathsf{rtk}_i)\right) \le \frac{1}{12}oNP^2\mathrm{Var}_\xi t + \frac{1}{3}do^2N^2\mathcal{B}^2\beta^2$. Then, by Lemma. 1, $\mathrm{Var}_\xi t \le oN$. By substituting this bound, we can obtain the error bound $\frac{1}{12}o^2N^2P^2 + \frac{1}{3}do^2N^2\mathcal{B}^2\beta^2$ for $a \cdot t - \varphi_{\Psi_{-i}(t)}(a \odot \mathsf{rtk}_i)$. Since the automorphism only rotates the ξ-basis,

$$\begin{aligned}\mathrm{Var}_\xi\left(\Psi_i\left(a \cdot t - \varphi_{\Psi_{-i}(t)}(a \odot \mathsf{rtk}_i)\right)\right) &= \mathrm{Var}_\xi\left(a \cdot t - \varphi_{\Psi_{-i}(t)}(a \odot \mathsf{rtk}_i)\right) \\ &\le \frac{1}{12}o^2N^2P^2 + \frac{1}{3}do^2N^2\mathcal{B}^2\beta^2.\end{aligned}$$

□

Lemma 3 (External Product). *Let* $\overline{\mathbf{C}} \in R^{2d\times 2}$ *be a fresh RGSW encryption of* $\mu \in R$ *under secret* t*, and* $\mathsf{ct} \in R_Q^2$ *be an RLWE ciphertext. Then,*

$$\mathrm{Var}_\xi(\mu \cdot \varphi_t(\mathsf{ct}) - \varphi_t(\mathsf{ct} \boxdot \mathbf{C})) \le \frac{1}{12}oNP^2(1 + oN)\|\mu\|_\xi^2 + \frac{2}{3}do^2N^2\mathcal{B}^2\beta^2$$

Proof. Note that the external product between the ciphertext $\mathsf{ct} = (b, a) \in R_Q^2$ and a fresh RGSW encryption of $\mu \in R$ is essentially computing $b \odot \mathbf{C}_b + a \odot \mathbf{C}_a$ where $\mathbf{C}_b$ and $\mathbf{C}_a$ are fresh gadget encryptions of μ and $\mu \cdot t$, respectively, the final variance of the external product is bounded by

$$\begin{aligned}&\frac{1}{12}oNP^2\|\mu\|_\xi^2 + \frac{1}{3}do^2N^2\mathcal{B}^2\beta^2 + \frac{1}{12}oNP^2\|\mu\|_\xi^2 \cdot \mathrm{Var}_\xi(t) + \frac{1}{3}do^2N^2\mathcal{B}^2\beta^2 \\ &\le \frac{1}{12}oNP^2\|\mu\|_\xi^2(1 + oN) + \frac{2}{3}do^2N^2\mathcal{B}^2\beta^2.\end{aligned}$$

□

Using this lemma, we can obtain the following error bound of the homomorphic MUX gate.

Corollary 2 (Homomorphic MUX Gate). *Let* $\overline{\mathbf{C}} \in R_Q^{2d\times 2}$ *be an RGSW encryption of some binary value* $b \in \{0, 1\}$ *under secret* t*, and* $\mathsf{ct}_0, \mathsf{ct}_1 \in R_Q^2$ *be RLWE ciphertexts. Then, for* $\mathsf{ct}' = \mathsf{ct}_0 + (\mathsf{ct}_1 - \mathsf{ct}_0) \boxdot \overline{\mathbf{C}} \pmod{Q}$*,*

$$\mathrm{Var}_\xi(\varphi_t(\mathsf{ct}_b) - \varphi_t(\mathsf{ct}')) \le \frac{1}{12}oNP^2(1 + oN) + \frac{2}{3}do^2N^2\mathcal{B}^2\beta^2.$$

Proof. Thanks to the linearity of the phase function $\varphi_t(\cdot)$, we have

$$\begin{aligned}\mathrm{Var}_\xi\left(\varphi_t(\mathsf{ct}_b) - \varphi_t(\mathsf{ct}')\right) &= \mathrm{Var}_\xi\left(\varphi_t(\mathsf{ct}_b) - \varphi_t(\mathsf{ct}_0) - \varphi_t\left((\mathsf{ct}_1 - \mathsf{ct}_0) \boxdot \overline{\mathbf{C}}\right)\right) \\ &= \mathrm{Var}_\xi\left(\varphi_t(\mathsf{ct}_b - \mathsf{ct}_0) - \varphi_t\left((\mathsf{ct}_1 - \mathsf{ct}_0) \boxdot \overline{\mathbf{C}}\right)\right).\end{aligned}$$

We remark that $\varphi_t(\mathsf{ct}_b - \mathsf{ct}_0) = b \cdot \varphi_t(\mathsf{ct}_1 - \mathsf{ct}_0)$, since if $b = 0$, $\varphi_t(\mathsf{ct}_b - \mathsf{ct}_0) = 0 = 0 \cdot \varphi_t(\mathsf{ct}_1 - \mathsf{ct}_0)$ and if $b = 1$, $\varphi_t(\mathsf{ct}_b - \mathsf{ct}_0) = \varphi_t(\mathsf{ct}_1 - \mathsf{ct}_0) = 1 \cdot \varphi_t(\mathsf{ct}_1 - \mathsf{ct}_0)$. Therefore, we can apply Lem. 3 by setting $\mu = b$ and $\mathsf{ct} = \mathsf{ct}_1 - \mathsf{ct}_0$ to obtain the bound $\frac{1}{12} oNP^2(1 + oN) + \frac{2}{3} do^2N^2\mathcal{B}^2\beta^2$, since $\|b\|_\xi^2 = b \le 1$. □

Now, we prove the noise growth from our blind rotation algorithm.

Theorem 1 (Blind Rotation). *Let* $\mathbf{c} = (b, \mathbf{a}) \in \mathbb{Z}_Q^{n+1}$ *be an LWE ciphertext. Then, for* $\mathsf{ct} \leftarrow \mathsf{BlindRotate}(\mathsf{brk}, \mathsf{rtk}; \mathbf{c}, f)$*, it holds that*

$$\mathrm{Var}_\xi(\varphi_t(\mathsf{ct}) - \Psi_{\tilde{b} + \langle \tilde{\mathbf{a}}, \mathbf{s}\rangle}(\mathcal{F})) \le \frac{1}{12} noNP^2(1 + 2oN) + ndo^2N^2\mathcal{B}^2\beta^2$$

where $\tilde{b} = \lfloor \frac{N}{Q} \cdot b \rceil$, $\tilde{\mathbf{a}} = \lfloor \frac{N}{Q} \cdot \mathbf{a} \rceil$ *and* $\mathcal{F} \leftarrow \frac{Q}{p^r} \cdot \sum_{i=0}^{N-1} f(\lfloor \frac{p^r}{N} \cdot i \rceil) \cdot \tau_i$.

Proof. We will prove by induction. Let ACC_i be the accumulator ACC before the i-th iteration of the for loop in Alg. 2. Then, $\varphi_t(\mathsf{ACC}_0) - \Psi_{\tilde{b}}(\mathcal{F}) = 0$. Now, we have $\mathsf{ACC}_{i+1} = \mathsf{ACC}_i + (\mathsf{Rot}(\mathsf{rtk}_{\tilde{a}_i}; \mathsf{ACC}_i) - \mathsf{ACC}_i) \boxdot \mathsf{brk}_i$. Then,

$$\mathrm{Var}_\xi\left(\varphi_t\left(\mathsf{Rot}(\mathsf{rtk}_{\tilde{a}_i}; \mathsf{ACC}_i)\right) - \Psi_{\tilde{a}_i}(\varphi_t(\mathsf{ACC}_i))\right) \le \frac{1}{12} o^2N^2P^2 + \frac{1}{3} do^2N^2\mathcal{B}^2\beta^2$$

by Cor. 1. Consequently, we have

$$\begin{aligned}&\mathrm{Var}_\xi\left(\varphi_t\left(\mathsf{ACC}_{i+1}\right) - \Psi_{\tilde{a}_i s_i}\left(\varphi_t(\mathsf{ACC}_i)\right)\right) \\ &\le \mathrm{Var}_\xi\left(\varphi_t\left(\mathsf{Rot}(\mathsf{rtk}_{\tilde{a}_i}; \mathsf{ACC}_i)\right) - \Psi_{\tilde{a}_i}(\varphi_t(\mathsf{ACC}_i))\right) + \frac{1}{12} oNP^2(1 + oN) + \frac{2}{3} do^2N^2\mathcal{B}^2\beta^2 \\ &\le \frac{1}{12} o^2N^2P^2 + \frac{1}{3} do^2N^2\mathcal{B}^2\beta^2 + \frac{1}{12} oNP^2(1 + oN) + \frac{2}{3} do^2N^2\mathcal{B}^2\beta^2 \\ &= \frac{1}{12} oNP^2(1 + 2oN) + do^2N^2\mathcal{B}^2\beta^2.\end{aligned}$$

By iteratively applying this inequality, we obtain

$$\begin{aligned}&\mathrm{Var}_\xi\left(\varphi_t(\mathsf{ct}) - \Psi_{\tilde{b} + \langle \tilde{\mathbf{a}}, \mathbf{s}\rangle}(\mathcal{F})\right) = \mathrm{Var}_\xi\left(\varphi_t(\mathsf{ACC}_n) - \Psi_{\tilde{b} + \langle \tilde{\mathbf{a}}, \mathbf{s}\rangle}(\mathcal{F})\right) \\ &\le \mathrm{Var}_\xi\left(\Psi_{\tilde{a}_{n-1} s_{n-1}}\left(\varphi_t(\mathsf{ACC}_{n-1}) - \Psi_{\tilde{b} + \sum_{i=0}^{n-2} a_i s_i}(\mathcal{F})\right)\right) + \frac{1}{12} oNP^2(1 + 2oN) + do^2N^2\mathcal{B}^2\beta^2 \\ &\le \cdots \le \mathrm{Var}_\xi\left(\Psi_{\sum_{i=0}^{n-1} \tilde{a}_i s_i}\left(\varphi_t(\mathsf{ACC}_0) - \Psi_{\tilde{b}}(\mathcal{F})\right)\right) + \frac{1}{12} noNP^2(1 + 2oN) + ndo^2N^2\mathcal{B}^2\beta^2 \\ &\le \frac{1}{12} noNP^2(1 + 2oN) + ndo^2N^2\mathcal{B}^2\beta^2.\end{aligned}$$

□

We prove the noise growth from the sample extraction of our algorithm.

Theorem 2 (Sample Extraction). *Let* $\mathsf{ct} \in R_Q^2$ *be an RLWE ciphertext such that* $\varphi_t(\mathsf{ct}) = \frac{Q}{p^r} \cdot \mu + e$ *where* $\mu = \sum_{i=0}^{N-1} m_i \cdot \tau_i$ *and for some* $e \in R$ *close to Gaussian. Then, for* $\mathbf{c} \leftarrow \mathsf{SampleExtract}(\mathsf{ct})$*,*

$$\mathrm{Var}\left(\varphi_{\mathbf{t}}(\mathbf{c}) - \frac{Q}{p^r} m_0\right) \le \frac{1}{12} p^{2r} \mathrm{Var}_{\xi}(e)$$

where $\mathbf{t}$ *is the coefficient vector of the secret* t *in* η*-basis.*

Proof. We first use the heuristic that the coefficients of τ_0 are uniformly distributed over the set $[-p^r/2, p^r/2] \cap \mathbb{Z}$. Then, the coefficients of $\alpha^{-1}\tau_0$ are also uniformly distributed, and thus $\mathrm{Var}_{\xi}(\alpha^{-1}\tau_0) = \frac{1}{12} oN \cdot p^{2r}$. Therefore, the noise term $\alpha^{-1}\tau_0 \cdot e$ has the canonical variance $\mathrm{Var}_{\xi}(\alpha^{-1}\tau_0 \cdot e) = \frac{1}{12} oN \cdot p^{2r} \mathrm{Var}_{\xi}(e)$. We also use a heuristic that $\alpha^{-1}\tau_0 \cdot e$ is close enough to a Gaussian distribution, and then its variance on the η-basis will given by $\frac{1}{12} p^{2r} \mathrm{Var}_{\xi}(e)$. Since we extract the constant term in the η-basis of $\alpha^{-1}\tau_0 \cdot e$ in the sample extraction step, the variance for $\varphi_t(\mathbf{c}) - \frac{Q}{p^r} m_0$ will be $\frac{1}{12} p^{2r} \mathrm{Var}_{\xi}(e)$. □

Next, we prove the noise growth from the key-switching algorithm.

Theorem 3 (KeySwitch). *Let* $\mathbf{c} \in \mathbb{Z}_Q^{N+1}$ *be an LWE ciphertext. Then, for* $\mathbf{c}' \leftarrow \mathsf{KeySwitch}(\mathsf{ksk}; \mathbf{c})$*,*

$$\mathrm{Var}\left(\varphi_{\mathbf{t}}(\mathbf{c}) - \varphi_{\mathbf{s}}(\mathbf{c}')\right) \le \frac{1}{12} N P'^2 + \frac{1}{3} d' N \mathcal{B}'^2 \alpha^2$$

where $\mathbf{t}$ *is the coefficient vector of the secret* t *in* η*-basis.*

Proof. Let $\mathbf{c} = (b, a_0, \ldots, a_{N-1})$ and $\mathbf{e}_i$ be the error vector for the i-th key-switching key ksk_i. Then, we have

$$\begin{aligned} t_i \cdot a_i - \varphi_{\mathbf{s}}\left(h'(a_i)^\top \cdot \mathsf{ksk}_i\right) &= t_i \cdot a_i - h'(\lfloor \frac{a_i}{P'} \rceil)^\top \cdot (P'\mathbf{g}' \cdot t_i + \mathbf{e}_i') \\ &= t_i \cdot \left(a_i - h'(\lfloor \frac{a_i}{P'} \rceil)^\top \cdot P'\mathbf{g}'\right) - h'(\lfloor \frac{a_i}{P'} \rceil)^\top \cdot \mathbf{e}_i'. \end{aligned}$$

Then, the variance of $a_i - h'(\lfloor a_i \rceil P)^\top \cdot P' g'$ is $\frac{1}{12} P'^2$ and the variance of $h'(\lfloor \frac{a_i}{P'} \rceil)^\top \cdot \mathbf{e}_i'$ is $\frac{1}{3} d' \mathcal{B}'^2 \alpha^2$ because the variance of each entry of $\mathbf{e}'$ is α^2. Consequently, the variance of $t_i \cdot a_i - \varphi_{\mathbf{s}}(h'(a_i)^\top \cdot \mathsf{ksk}_i)$ is bounded by $\frac{1}{12} P'^2 + \frac{1}{3} d' \mathcal{B}'^2 \alpha^2$. Finally, since

$$\begin{aligned} \varphi_{\mathbf{t}}(\mathbf{c}) - \varphi_{\mathbf{s}}(\mathbf{c}') &= b + \sum_{i=0}^{N-1} a_i t_i - (b + \sum_{i=0}^{N-1} \varphi_{\mathbf{s}}(h'(a_i)^\top \cdot \mathsf{ksk}_i)) \\ &= \sum_{i=0}^{N-1} (a_i t_i - h'(a_i)^\top \cdot \mathsf{ksk}_i), \end{aligned}$$

the variance of $\varphi_{\mathbf{t}}(\mathbf{c}) - \varphi_{\mathbf{s}}(\mathbf{c}')$ is bounded by $\frac{1}{12} N P'^2 + \frac{1}{3} d' N \mathcal{B}'^2 \alpha^2$. □

Finally, we provide the main theorem for the bootstrapping.

Theorem 4 (Bootstrapping). *Let* $\mathsf{ct} \in R_Q^2$ *be an RLWE ciphertext such that* $\varphi_t(\mathsf{ct}) = \frac{Q}{p^r} \cdot \mu + e_{\mathsf{ct}}$ *where* $\mu = \sum_{i=0}^{N-1} m_i \cdot \tau_i$*, and* $e_{\mathsf{ct}} \in R$ *is close to Gaussian. Then, given the function* $f : \mathbb{Z}_p \to \mathbb{Z}_p$*,* $\mathsf{ct}' \leftarrow \mathsf{Bootstrap}(\mathsf{brk}, \mathsf{rtk}, \mathsf{ksk}; \mathsf{ct}, f)$ *satisfies that* $\varphi_t(\mathsf{ct}') = \frac{Q}{p^r} \cdot \mu' + e_{\mathsf{ct}'}$ *for some* m' *and* $e_{\mathsf{ct}'}$ *where*

$$\mu' = \Psi_{\frac{N}{p^r}m+e}\left(\sum_{i=0}^{N-1} f\left(\lfloor \frac{p^r}{N} \cdot i \rceil\right) \tau_i\right),$$

$$\mathrm{Var}(e) \leq \frac{N^2}{Q^2}\left(\frac{1}{12}p^{2r}\mathrm{Var}_\xi(e_{\mathsf{ct}}) + \frac{1}{12}NP'^2 + \frac{1}{3}d'N\mathcal{B}'^2\alpha^2\right) + \frac{n+1}{12}$$

and

$$\mathrm{Var}_\xi(e_{\mathsf{ct}'}) \leq \frac{1}{12}noNP^2(1+2oN) + ndo^2N^2\mathcal{B}^2\beta^2.$$

Proof. By Theorems 2 and 3, the LWE ciphertext after the key-switching step has the noise variance of

$$\frac{1}{12}p^{2r}\mathrm{Var}_\xi(e_{\mathsf{ct}}) + \frac{1}{12}NP' + \frac{1}{3}d'NB'^2\alpha^2.$$

The scaling by $\frac{N}{Q}$ and rounding this LWE ciphertext scales this error variance by $\frac{N^2}{Q^2}$ and introduces an additional rounding noise. Assuming that the entries of LWE ciphertexts follow the uniform distribution, the rounding noise can be regarded as a sum of $h+1$ uniform random variables over $[-0.5, 0.5]$ where h is the Hamming weight of the secret $\mathbf{s}$. Therefore, the rounding error is bounded by $\frac{1}{12}(1+n)$ as $n \geq h$, and therefore the variance of the noise term e of the scaled LWE ciphertext is bounded by

$$\mathrm{Var}(e) \leq \frac{N^2}{Q^2}\left(\frac{1}{12}p^{2r}\mathrm{Var}_\xi(e_{\mathsf{ct}}) + \frac{1}{12}NP'^2 + \frac{1}{3}d'N\mathcal{B}'^2\alpha^2\right) + \frac{n+1}{12}.$$

Finally, the correctness for μ' and error variance of $e_{\mathsf{ct}'}$ directly follows from Theorem 1. □

5.4 Optimizations

Block Binary Keys In [20], the authors suggested two optimization techniques for TFHE bootstrapping. We adapt their idea to enhance the performance of our scheme. Their first observation is that a sequential evaluation of ℓ two-to-one homomorphic Mux gates can be integerated into a single ℓ-to-one homomorphic Mux gate if the secret is drawn from a certain sparse key distribution, called the *block binary key distribution*. More specifically, for the LWE dimension $n = k \cdot \ell$, the block binary key distribution samples the secret key $\mathbf{s} = (s_0, \ldots, s_{n-1})$ such that there is at most one 1 among $s_{i\ell}, \ldots, s_{(i+1)\ell-1}$ for all $0 \leq i < k$. Then, at each iteration, it only remains to select between ACC, $\Psi_{\tilde{a}_{i\ell}}(\mathsf{ACC})$, ...,

$\Psi_{\tilde{a}_{(i+1)\ell-1}}(\mathsf{ACC})$ with respect to the value of $s_{i\ell}, \ldots, s_{(i+1)\ell-1}$. More precisely, we homomorphically compute

$$\begin{aligned}\Psi_{\sum_{j=0}^{\ell-1}\tilde{a}_{i\ell+j}s_{i\ell+j}}(\mathsf{ACC}) &= \mathsf{ACC} + \sum_{j=0}^{\ell-1}\left(\Psi_{\tilde{a}_{i\ell+j}}(\mathsf{ACC}) - \mathsf{ACC}\right)\cdot s_{i\ell+j} \\ &= \mathsf{ACC} + \sum_{j=0}^{\ell-1}\left(\Psi_{\tilde{a}_{i\ell+j}}(\mathsf{ACC}\cdot s_{i\ell+j}) - \mathsf{ACC}\cdot s_{i\ell+j}\right)\end{aligned}$$

for every i-th iteration. Note that the second equality is derived from the fact that s_i is invariant to any automorphism since it is a constant. Now, observe that we repeatedly compute the homomorphic multiplication between ACC and RGSW encryption of $s_{i\ell+j}$'s, the decompositions of ACC can be re-used for the external products. As a result, we can reduce the number of gadget decompositions from $3dn$ to $dn+2dk$, hence avoid repetitive computation of expensive FFT and iFFT. The complete slot blind rotation algorithm with block binary keys is given in Alg. 4.

Algorithm 4 Blind Rotation with Block Binary Keys

Input: The blind rotation key brk, rotation key rtk, LWE ciphertext $\mathbf{c} = (b, a_0, \ldots, a_{n-1}) \in \mathbb{Z}_Q^{n+1}$ and the evaluated function $f : \mathbb{Z}_p \to \mathbb{Z}_p$.

- $\tilde{b} \leftarrow \lfloor \frac{N}{Q}\cdot b\rceil \in \mathbb{Z}_N$ and $\tilde{a}_i = \lfloor \frac{N}{Q}\cdot a_i\rceil \in \mathbb{Z}_N$ for $0 \le i < n$.
- $\mathcal{F} \leftarrow \frac{Q}{p^r}\cdot\sum_{i=0}^{N-1} f(\lfloor \frac{p^r}{N}\cdot i\rceil)\cdot \tau_i \in R_Q$
- $\mathsf{ACC} \leftarrow (\Psi_{\tilde{b}}(\mathcal{F}), 0) \in R_Q^2$
- **for** $i = 0 : k-1$ **do**
  - $\mathsf{hACC} \leftarrow h(\mathsf{ACC})$
  - **for** $j = 0 : \ell - 1$ **do**
    - $\mathsf{tmp} \leftarrow \mathsf{hACC}^\top \cdot \mathsf{RGSW}(s_{\ell\cdot i+j})$
    - $\mathsf{ACC} \leftarrow \mathsf{ACC} + \mathsf{Rot}(\mathsf{rtk}_{\tilde{a}_{\ell\cdot i+j}}; \mathsf{tmp}) - \mathsf{tmp}$
  - **end for**
- **end for**

Output: Return ACC

In addition, they also proposed to reuse the LWE secret $\mathbf{s} \in \mathbb{Z}^n$ for the coefficinet of the RLWE secret $t \in R$. By doing so, the size of the key-switching is reduced by a factor $\frac{N-n}{N}$. Specifically, we set $t = \sum_{i=0}^{n-1} t_i\eta_i$ where $t_i = s_i$ for $0 \le i < n$ and $t_i \leftarrow \mathcal{U}(\{0, \pm 1\})$ for $n \le i < N$ where s_i is the i-th entry of the LWE secret $\mathbf{s} \in \mathbb{Z}^n$. Then, the key-switching algorithm can be modified as

$$\mathbf{c}' = (b, a_0, \ldots, a_{n-1}) + \sum_{i=n}^{N-1} h'(a_i)^\top \cdot \mathsf{ksk}_i \pmod{Q}$$

for the input ciphertext $\mathbf{c} = (b, a_0, \ldots, a_{N-1}) \in \mathbb{Z}_Q^{N+1}$.

Coefficient Packing We remark that our scheme is even compatible with the coefficient packing method, as in AP-like cryptosystems. Recall that the automorphisms right-shift the coefficient vector in ξ-basis, while they left-shift the coefficient vector in η-basis. Therefore, we can simply encode the messages in the coefficient vector, but in a reverse order. More precisely, given the function $f : \mathbb{Z}_p \to \mathbb{Z}_p$, the test vector can be set as

$$\mathcal{F} = \frac{Q}{p^r} \sum_{i=0}^{N-1} f(\lfloor \frac{p^r}{N} \cdot (N-i) \rceil) \cdot \eta_i$$

and run the blind rotation algorithm in a same manner. If the messages are packed in the coefficients, the sample extraction noise can be reduced, however, the multiplication might not be as natural as the slot packing.

6 Implementation

We provide the recommended parameters for our new fully homomorphic encryption scheme built upon our novel slot blind rotation framework. We also present benchmark results for our new fully homomorphic encryption scheme.

We utilize the base decomposition as the baseline gadget decomposition technique for our scheme. For some integers $d, D > 0$ such that $D^d \mid Q$, the base decomposition essentially decomposes the input $a \in \mathbb{Z}_{D^d}$ into a vector of its digits in base-D balanced representation. In other words, it outputs a vector $\mathbf{a} = (a_0, \ldots, a_{d-1})$ such that $a = \sum_{i=0}^{d-1} a_i \cdot D^i$ and $-D/2 < a_i \le D/2$. Then, naturally the special modulus P can be set as $P = Q/D^d$. In the following, we say that D and d are the gadget decomposition parameters if the gadget decomposition outputs a length d vector of each digit of the input value in base-D representation.

Parameter Selection When choosing the cyclotomic degree M for the baseline cyclotomic ring for the subring, we need to set M as small as possible, since the noise variance is (almost) proportional to M^2, as analyzed in 5.3. However, apart from the size of M, there are a few more conditions we need to consider for a better efficiency. Firstly, for the plaintext modulus p^r, we aim to choose M such that the multiplicative order of p modulo M is even. By doing so, the base subring is invariant to the conjugate automorphism $X \mapsto X^{-1}$, allowing us to use efficient real FFT during the basis conversion. Secondly, we choose M so that the ring degree N has as many small prime factors as possible. To perform the basis conversion, we perform a single (real) FFT and iFFT with length N to instantiate the convolution. Generally, the more large prime factors N has, the slower the latency for convolution can become. Therefore, for better efficiency, M should be factorized into many small primes.

In Table. 2, the concrete parameter sets for our fully homomorphic encryption scheme is presented. In the table, Q, p, r denote the ciphertext and the plaintext moduli Q and p^r. We fix the ciphertext modulus Q to $Q = 2^{64}$ to leverage the natural modular arithmetic of 64-bit unsigned integers. n and α denote the LWE paramters, namely LWE dimension and standard deviation for the noise distribution $\mathcal{D}_\alpha$, and M and β denote the base cyclotomic degree M for the subring, and the standard deviation for the noise distribution $\mathcal{D}_\beta$. Finally, D', d' and D and d are the gadget decomposition parameters for the key-switching and blind rotation. For the block length of block binary keys, we fix $\ell = 2$. We set the parameters so that the bootstrapping failure probability is smaller than 2^{-64}, achieving at least 128-bit security.

<table>
<tr><th>Set</th><th>Q</th><th>p</th><th>r</th><th>n</th><th>α</th><th>D'</th><th>d'</th><th>M</th><th>β</th><th>D</th><th>d</th></tr>
<tr><td>I</td><td rowspan="11">2^{64}</td><td rowspan="5">2</td><td rowspan="2">2</td><td rowspan="2">630</td><td rowspan="2">1.9×2^{17}</td><td rowspan="2">2^2</td><td rowspan="2">6</td><td>87211</td><td>1.021×2^{24}</td><td>2^8</td><td>4</td></tr>
<tr><td>II</td><td>65537</td><td>1.564×2^{12}</td><td>2^{10}</td><td>3</td></tr>
<tr><td>III</td><td rowspan="2">3</td><td rowspan="2">680</td><td rowspan="2">1.528×2^{16}</td><td rowspan="2">2^2</td><td rowspan="2">7</td><td>87211</td><td>1.021×2^{24}</td><td>2^8</td><td>4</td></tr>
<tr><td>IV</td><td>65537</td><td>1.564×2^{12}</td><td>2^{10}</td><td>3</td></tr>
<tr><td>V</td><td>4</td><td>750</td><td>1.707×2^{14}</td><td>2^2</td><td>7</td><td>174763</td><td>3.2</td><td>2^7</td><td>5</td></tr>
<tr><td>VI</td><td rowspan="2">3</td><td>1</td><td>600</td><td>1.642×2^{18}</td><td>2^2</td><td>6</td><td rowspan="2">176419</td><td rowspan="2">3.2</td><td>2^8</td><td>4</td></tr>
<tr><td>VII</td><td>2</td><td>700</td><td>1.061×2^{16}</td><td>2^2</td><td>7</td><td>2^8</td><td>4</td></tr>
<tr><td>VIII</td><td rowspan="2">5</td><td rowspan="2">1</td><td rowspan="2">650</td><td rowspan="2">1.320×2^{17}</td><td rowspan="2">2^2</td><td rowspan="2">6</td><td>38923</td><td>1.096×2^{27}</td><td>2^5</td><td>6</td></tr>
<tr><td>IX</td><td>221401</td><td>3.2</td><td>2^{10}</td><td>3</td></tr>
<tr><td>X</td><td>7</td><td>1</td><td>680</td><td>1.528×2^{16}</td><td>2^2</td><td>6</td><td>137089</td><td>1.348×2^{16}</td><td>2^8</td><td>4</td></tr>
<tr><td>XI</td><td>11</td><td>1</td><td>720</td><td>1.474×2^{15}</td><td>2^2</td><td>7</td><td>83791</td><td>1.431×2^8</td><td>2^8</td><td>4</td></tr>
</table>

Table 2. The recommended parameter set

Experimental Results We describe the experimental results of our scheme in this subsection. The proposed scheme is implemented from scratch in Julia, as no existing libraries support the subring FFT. For the base Fast Fourier Transformation (FFT) for convolution, we utilized the Julia wrapper for FFTW library [12]. All experiments were performed on a Mac mini with an Apple M2 processor and 24GB of Ram. Our source code is available at `https://github.com/SNUCP/carousel`.

Table 3 describes the running time of a single functional bootstrapping and the size of the evaluation keys, namely the blind rotation key brk, rotation key rtk and the key-switching key ksk under the corresponding parameter set.

Set	brk	rtk	ksk	Elapsed Time
I	125.229MB	160.338MB	14.789MB	138ms
II	118.923MB	193.171MB	21.291MB	70ms
III	135.168MB	160.388MB	17.616MB	154ms
IV	128.361MB	193.171MB	25.774MB	78ms
V	527.813MB	1.580GB	79.713MB	598ms
VI	196.756MB	438.069MB	29.702MB	207ms
VII	229.549MB	438.069MB	38.227MB	235ms
VIII	179.708MB	206.780MB	13.105MB	166ms
IX	161.498MB	335.254MB	31.718MB	131ms
X	159.157MB	222.674MB	19.779MB	153ms
XI	194.974MB	298.386MB	29.565MB	171ms

Table 3. The latency of functional bootstrapping for each parameter set I to XI.

Our benchmark results show a low latency of under 250 milliseconds for most of the parameter sets, except for the set V, which accommodates the largest plaintext modulus among all the parameter sets. Also, the evaluation key sizes for our scheme are under 1 Gigabytes (except for the set V), which is considerably small. Note that the size of the rotation keys are of a similar level to the blind rotation keys while we provide a larger number of rotation keys compared to the blind rotation keys, thanks to the small size of gadget encryption (a quarter of the RGSW ciphertext). We believe that there is still a room for further optimizations in the implementation, such as fast convolution or SIMD parallelization.

Discussion Unlike previous works [10,7,21], our scheme can leverage real FFT results thanks to the subring structure. In the presence of FFT accelerator, we expect that our scheme will have an advantage over the previous works. When implementing AP-like schemes, a substantial number of the point-wise arithmetic of FFT vectors are required. However, the point-wise multiplication between complex vectors of length $N/2$ requires at least $2N$ floating point multiplications and N floating point additions. In contrast, the point-wise multiplication between real vectors of length N only requires N floating point multiplications. Thus, the use of real FFT can reduce the floating point arithmetic by a factor of three in multiplication, leading to significant performance improvements.

References

1. Alperin-Sheriff, J., Peikert, C.: 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. pp. 297–314. Springer (2014)
2. Arita, S., Handa, S.: Subring homomorphic encryption. In: International Conference on Information Security and Cryptology. pp. 112–136. Springer (2017)
3. Arita, S., Handa, S.: Fully homomorphic encryption scheme based on decomposition ring. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **103**(1), 195–211 (2020)
4. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical gapsvp. In: Annual cryptology conference. pp. 868–886. Springer (2012)
5. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) **6**(3), 1–36 (2014)
6. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23. pp. 409–437. Springer (2017)
7. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Tfhe: fast fully homomorphic encryption over the torus. Journal of Cryptology **33**(1), 34–91 (2020)
8. Chillotti, I., Ligier, D., Orfila, J.B., Tap, S.: Improved programmable bootstrapping with larger precision and efficient arithmetic circuits for tfhe. 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. pp. 670–699. Springer (2021)
9. Clet, P.E., Boudguiga, A., Sirdey, R., Zuber, M.: Combo: A novel functional bootstrapping method for efficient evaluation of nonlinear functions in the encrypted domain. In: International Conference on Cryptology in Africa. pp. 317–343. Springer (2023)
10. Ducas, L., Micciancio, D.: Fhew: bootstrapping homomorphic encryption in less than a second. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 617–640. Springer (2015)
11. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Paper 2012/144 (2012), `https://eprint.iacr.org/2012/144`, `https://eprint.iacr.org/2012/144`
12. Frigo, M., Johnson, S.G.: The design and implementation of FFTW3. Proceedings of the IEEE **93**(2), 216–231 (2005). https://doi.org/10.1109/JPROC.2004.840301, special issue on "Program Generation, Optimization, and Platform Adaptation"
13. Gama, N., Izabachène, M., Nguyen, P.Q., Xie, X.: Structural lattice reduction: generalized worst-case to average-case reductions and homomorphic cryptosystems. In: Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35. pp. 528–558. Springer (2016)
14. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the forty-first annual ACM symposium on Theory of computing. pp. 169–178 (2009)

15. Ha, J., Lee, J.: Patching and extending the wwl+ circuit bootstrapping method to fft domains. Cryptology ePrint Archive (2024)
16. Joye, M., Paillier, P.: Blind rotation in fully homomorphic encryption with extended keys. In: International Symposium on Cyber Security, Cryptology, and Machine Learning. pp. 1–18. Springer (2022)
17. Joye, M., Walter, M.: Liberating tfhe: programmable bootstrapping with general quotient polynomials. In: Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography. pp. 1–11 (2022)
18. Kim, D., Song, Y.: Approximate homomorphic encryption over the conjugate-invariant ring. In: Information Security and Cryptology–ICISC 2018: 21st International Conference, Seoul, South Korea, November 28–30, 2018, Revised Selected Papers 21. pp. 85–102. Springer (2019)
19. Kluczniak, K., Schild, L.: Fdfb: full domain functional bootstrapping towards practical fully homomorphic encryption. IACR Transactions on Cryptographic Hardware and Embedded Systems (2023)
20. Lee, C., Min, S., Seo, J., Song, Y.: Faster tfhe bootstrapping with block binary keys. In: Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security. pp. 2–13 (2023)
21. Lee, Y., Micciancio, D., Kim, A., Choi, R., Deryabin, M., Eom, J., Yoo, D.: Efficient fhew bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 227–256. Springer (2023)
22. Liu, F.H., Wang, H.: Batch bootstrapping i: a new framework for simd bootstrapping in polynomial modulus. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 321–352. Springer (2023)
23. Liu, F.H., Wang, H.: Batch bootstrapping ii: bootstrapping in polynomial modulus only requires o(1) fhe multiplications in amortization. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 353–384. Springer (2023)
24. Liu, Z., Micciancio, D., Polyakov, Y.: Large-precision homomorphic sign evaluation using fhew/tfhe bootstrapping. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 130–160. Springer (2022)
25. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. Journal of the ACM (JACM) **60**(6), 1–35 (2013)
26. Ma, S., Huang, T., Wang, A., Zhou, Q., Wang, X.: Fast and accurate: efficient full-domain functional bootstrap and digit decomposition for homomorphic computation. IACR Transactions on Cryptographic Hardware and Embedded Systems (2024)
27. Okada, H., Player, R., Pohmann, S.: Homomorphic polynomial evaluation using galois structure and applications to bfv bootstrapping. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 69–100. Springer (2023)
28. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM) **56**(6), 1–40 (2009)
29. Wang, R., Wen, Y., Li, Z., Lu, X., Wei, B., Liu, K., Wang, K.: Circuit bootstrapping: faster and smaller. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 342–372. Springer (2024)
30. Yang, Z., Xie, X., Shen, H., Chen, S., Zhou, J.: Tota: fully homomorphic encryption with smaller parameters and stronger security. Cryptology ePrint Archive (2021)