Carousel: Fully Homomorphic Encryption with Bootstrapping over Automorphism Group

Intak Hwang, Seonhong Min, and Yongsoo Song

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

Abstract. Homomorphic Encryption (HE) enables the secure computation of functions on ciphertexts without requiring decryption. Specifically, AP-like HE schemes exploit an intrinsic bootstrapping method called blind rotation. In existing blind rotation methods, a look-up table is homomorphically evaluated on the input ciphertext through iterative multiplication of monomials. However, the algebraic structure of the multiplicative group of monomials imposes certain limitations on the input plaintext space, as it can bootstrap only a fraction of the input plaintext space.

In this work, we introduce a new HE scheme, Carousel, that solves this problem. 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 that can be rotated via a series of homomorphic multiplications and automorphisms. We instantiate Carousel with subring encoding proposed by Arita and Handa (ICISC '17) and provide a proof-of-concept implementation. Our benchmark result shows that Carousel can bootstrap 4-bit integers in under 30ms.

Keywords: Fully Homomorphic Encryption, Bootstrapping, Blind Rotation

1 Introduction

Fully Homomorphic Encryption (FHE) is a cryptographic primitive that allows unlimited arithmetic operations to be performed on encrypted data without the need to decrypt it. Among the different FHE schemes, the (Ring) Learning with Errors (LWE)-based ones, such as BGV [BGV14], BFV [Bra12,FV12], AP/FHEW [ASP14,DM15], TFHE [CGGI20], and CKKS [CKKS17] demonstrate the highest performance. The ciphertexts in these schemes contain some noise for security purposes, and this noise is amplified during homomorphic operations such as additions and multiplications. This results in a restriction on the circuit's width and depth, thereby limiting the original scheme to be only somewhat homomorphic. To extend these schemes to fully homomorphic ones, an intrinsic technique known as *bootstrapping* [Gen09] can be employed. In essence, bootstrapping accepts a noisy ciphertext as input and homomorphically evaluates the decryption circuit on it, consequently producing a new ciphertext that

encrypts the same message as the input. Thus, as long as the bootstrapping circuit is manageable within the maximum noise level, we can perform the desired circuit evaluation by applying bootstrapping whenever the intermediate ciphertexts reach the noise threshold.

Unfortunately, bootstrapping is one of the most challenging and time-consuming operations in homomorphic computation with FHE. As a result, significant progress on this subject has been made over time, particularly in the area of functional bootstrapping, which has recently gained traction as a research topic of interest. Functional bootstrapping is a special type of bootstrapping, which allows us to evaluate some univariate function on an encrypted message while reducing the noise. Although this topic was investigated for a variety of schemes [LMS24,LW25,AKP24,BKSS25], AP-like cryptosystems (e.g., FHEW [DM15], LMKCDEY [LMK+23], TFHE [CGGI20]) are more highlighted due to their flexibility and low latency. AP-like cryptosystems can bootstrap a Regev encryption [Reg09] (often referred to as B/FV scheme [Bra12,FV12] when fully homomorphic), which has the form of $(b, \mathbf{a}) \in \mathbb{Z}_Q^{n+1}$, such that $b + \langle \mathbf{a}, \mathbf{s} \rangle = \lfloor \frac{Q}{p} m \rceil + e \pmod{Q}$ with $Q, p \in \mathbb{Z}$, $\mathbf{s} \in \mathbb{Z}^n$, $m \in \mathbb{Z}_p$, and $e \in \mathbb{Z}$ being the ciphertext and message space modulus, the secret key, the message, and a small noise, respectively. In their bootstrapping procedure, the ciphertext modulus is firstly switched to some small positive integer $M > 0$ to reduce the cost of bootstrapping, by multiplying the constant $\frac{M}{Q}$ to each element of the ciphertext and then rounding. In other words, we compute $(\hat{b}, \hat{\mathbf{a}}) = \lfloor (b, \mathbf{a}) \rceil$. Assuming that the new ciphertext modulus M is large enough, it holds that $\hat{b} + \langle \hat{\mathbf{a}}, \mathbf{s} \rangle = \lfloor \frac{M}{p} m \rceil + \frac{M}{Q} e + e_{rnd} \pmod{M}$ for the rounding error e_{rnd}, of which the size is bounded by $\|\mathbf{s}\|_1/2$. Then, the decryption circuit can be represented as an evaluation of a lookup table of size M, which outputs the corresponding message m for all possible values for the plaintext $\hat{b} + \langle \hat{\mathbf{a}}, \mathbf{s} \rangle \pmod{M}$[1]. In AP-style bootstrapping, this lookup table evaluation is instantiated with the rotation of the lookup table in a homomorphic manner. More specifically, the lookup table (LUT) is cyclically rotated by the phase of the ciphertext $(\hat{b}, \hat{\mathbf{a}}) \in \mathbb{Z}_M^{n+1}$ given the encryptions of s_i's, where s_i denotes i-th index of the secret $\mathbf{s}$. Then, by taking the first entry of the rotated LUT, the decryption circuit is successfully evaluated. Moreover, an arbitrary function can be evaluated by simply replacing the values of the lookup table from m to $f(m)$.

This intrinsic bootstrapping method is often referred to as *blind rotation*. In AP-like cryptosystems, blind rotation is implemented with multiplication by monomials. To be precise, they use the fact that the set of monomials $\{X^i \mid 0 \leq i < M\}$ forms a multiplicative subgroup over the polynomial ring $R_Q := \mathbb{Z}_Q[X]/\Phi_M(X)$ where $\Phi_M(X)$ is the M-th cyclotomic polynomial. With this structure, we can easily simulate the modulo M operation for free. Then, a single lookup table can be encoded as some polynomial $\mathcal{F} \in R_Q$ called the *test*

[1] Essentially, this is equivalent to computing a rounding function $\lfloor \frac{p}{M} x \rceil$ for the input $x \in \mathbb{Z}_M$.

vector, so that the constant term of $\mathcal{F} \cdot X^{\alpha}$ is α-th entry of the lookup table. [2] From this observation, the blind rotation is done by multiplying the monomial $X^{\hat{b}+\langle \hat{\mathbf{a}}, \mathbf{s} \rangle}$ to the test vector $\mathcal{F}$. While this AP-style blind rotation gains efficiency, there is a limitation on the evaluated lookup tables. Note that all the polynomials in the base ring R_Q are represented with $\phi(M)$ elements, where $\phi(\cdot)$ is the Euler phi function. Since $\phi(M) < M$, they do not have enough degree of freedom to encode a length M LUT. To circumvent this problem, many existing implementations [Zam22] only utilize $\phi(M)/M$ fraction of the plaintext space. Nevertheless, this approach is not compatible with the basic arithmetic operations such as addition/multiplication, as the underlying plaintext may not fall into the $\phi(M)/M$ fraction, leading to an inefficient circuit design. Hence, naturally, we can ask a question:

Can a blind rotation algorithm
naturally cover the entire input plaintext space?

The current existing solutions [CLOT21a,KS23,YXS+21,CBSZ23,MHW+24,WWL+24] focus on optimizing cases where the cyclotomic degree M of the base ring $R = \mathbb{Z}[X]/\Phi_M(X)$ is a power of two. These methods utilize AP-style blind rotation in a black-box manner, instead of modifying the blind rotation algorithm itself. Consequently, more than two AP-style blind rotations need to be executed to achieve the ideal functionality of the blind rotation.

Our Contribution. We provide a new blind rotation framework, which can bootstrap without any limitations on the input plaintext space with a single execution. It leverages the Galois group structure for the lookup table evaluation, instead of the multiplicative group of monomials. Also, we construct a new FHE scheme, called 'Carousel', based on this new framework. Carousel has two modes, namely Carousel-C and Carousel-S, depending on the encoding structure of the lookup table. Carousel-C shows low bootstrapping latency due to the small noise growth, and Carousel-S has better compatibility with the BFV scheme, supporting a natural multiplication.

Table 1. Summary of the functional bootstrapping performance of Carousel scheme.

p	Carousel-C	Carousel-S
2^2	19.0ms	28.4ms
2^3	18.9ms	28.8ms
2^4	19.6ms	29.6ms

We implement a proof-of-concept implementation of Carousel in Go to prove its efficiency. To accelerate performance, we use the binary block key structure proposed by Lee, Min, Seo, and Song [LMSS23]. Table 1 describes the performance of two modes of Carousel, Carousel-C and Carousel-S. All experiments were

[2] Such a polynomial $\mathcal{F}$ always exists for any M. See [JW22] for more information.

performed on a machine equipped with 11th Gen Intel(R) Core(TM) i9-11900, with 32 GB of RAM.

Technical Overview. The core idea of our work is to perform the blind rotation over the automorphism group instead of the multiplicative subgroup of monomials. The automorphism operations are supported by any RLWE-based schemes, in addition to the basic arithmetic operations, namely addition, subtraction, and multiplication. The ring automorphism operation plays an essential role in extending the functionality of encoding, rather than just componentwise operations, but also linear algebra [HS14,CHM+25] and ciphertext conversion [CDKS21,BCK+23]. To provide a better parameterization for these applications, base rings with a cyclic automorphism group structure are generally preferred. Note that the Galois group of the cyclotomic extension is cyclic when the cyclotomic degree is a prime power [3], and a subring of some prime power cyclotomic ring can be used as the base ring for various HE schemes. For example, a decomposition ring, which is a subring composed of elements that are invariant under a certain set of automorphisms, of a prime power cyclotomic ring is used [AH17,AH20,KS19].

Our blind rotation framework also assumes a base ring R with a cyclic Galois group, where some automorphism Ψ generates the entire automorphism group [4]. Let N be the order of Ψ, p the plaintext modulus, and Q the ciphertext modulus. We write R_Q to denote the quotient ring R/QR. Now, assume that there exist encoding and decoding algorithms $\mathsf{Ecd} : \mathbb{Z}_p^N \to R_Q$ and $\mathsf{Dcd} : R_Q \to \mathbb{Z}_p$, such that $\mathsf{Dcd}\left(\Psi^i\left(\mathsf{Ecd}(\mathbf{v})\right)\right) = v_i$ for any $i \in \mathbb{Z}_N$ and vector $\mathbf{v} = (v_0, \ldots, v_{N-1}) \in \mathbb{Z}_p^N$. We discuss the existence and construction of the encoding and decoding algorithms later. Then, given a Regev encryption $(b, \mathbf{a}) \in \mathbb{Z}_Q^{n+1}$ of message $m \in \mathbb{Z}_p$ under the secret $\mathbf{s} \in \mathbb{Z}^n$, similar to AP-like cryptosystems, we first change the ciphertext modulus to N to minimize the lookup table size. In other words, we compute $\hat{b} = \lfloor \frac{N}{Q} b \rceil \in \mathbb{Z}_N$ and $\hat{\mathbf{a}} = \lfloor \frac{N}{Q} \mathbf{a} \rceil \in \mathbb{Z}_N^n$, so that

$$\hat{b} + \langle \hat{\mathbf{a}}, \mathbf{s} \rangle = \lfloor \frac{N}{p} m \rceil + e \pmod{N}$$

for some small noise $e \in \mathbb{Z}$ with $|e| < \frac{N}{2p}$. Then, given a function $f : \mathbb{Z}_p \to \mathbb{Z}_p$, our new blind rotation first generates a lookup table $\mathbf{v} = (v_i)_{0 \le i < N} \in \mathbb{Z}_p^N$ such that $v_i = f(\lfloor \frac{p}{N} i \rceil)$, then returns $\mathsf{Dcd}\left(\Psi^{\hat{b} + \langle \hat{\mathbf{a}}, \mathbf{s} \rangle}(\mathsf{Ecd}(\mathbf{v}))\right)$. By definition, the output value is

$$v_{[b + \langle \hat{\mathbf{a}}, \mathbf{s} \rangle]_N} = f\left(\lfloor \frac{p}{N} (\lfloor \frac{N}{p} m \rceil + e) \rceil\right) = f(m)$$

regardless of the input value, hence achieving the ideal blind rotation functionality.

[3] We remark that this is not true for power of two. For a power-of-two $M \ge 8$, the Galois group of $\mathbb{Q}(\zeta_M)$ is isomorphic to $\mathbb{Z}_2 \oplus \mathbb{Z}_{M/4}$. However, it is also widely used due to the fast Discrete Fourier Transformation (DFT) algorithm.

[4] The Galois group of base ring does not necessarily need to be cyclic, but we focus on the cyclic case in this paper.

To realize this framework, the automorphism $\Psi^{\hat{b}+\langle \hat{\mathbf{a}}, \mathbf{s} \rangle}$ needs to be homomorphically evaluated on the encoding $\mathcal{F}$ of the LUT, with secret information $\mathbf{s}$. We compute this by sequentially applying automorphism $\Psi^{\hat{a}_i s_i}$ to $\Psi^{\hat{b}}(\mathcal{F})$ for $0 \leq i < n$, where $\hat{\mathbf{a}} = (a_0, \ldots, a_{n-1})$. To achieve this, we adopt the homomorphic MUX gate-based approach from GINX/CGGI bootstrapping [GINX16,CGGI20]. Assuming a binary key $\mathbf{s} = (s_0, \ldots, s_{n-1}) \in \mathbb{Z}^n$, an automorphism $\Psi^{\hat{a}_i s_i}$ is equivalent to $\Psi^{\hat{a}_i}$ if $s_i = 1$, and identity if $s_i = 0$. Therefore, we can devise the following linear relation:

$$\Psi^{\hat{a}_i s_i} = s_i \cdot \Psi^{\hat{a}_i} + (1 - s_i) \cdot id = id + s_i \cdot (\Psi^{\hat{a}_i} - id) \tag{1}$$

where id denotes the identity automorphism. To evaluate this homomorphically, we provide the encryption of s_i, and key-switching keys for all non-trivial automorphisms $(\Psi, \ldots, \Psi^{N-1})$ as the bootstrapping key. By composing Eq. (1) for all $0 \leq i < n$, we obtain the closed form of our blind rotation circuit:

$$\left(id + s_{n-1} \cdot (\Psi^{\hat{a}_{n-1}} - id)\right) \circ \cdots \circ \left(id + s_0 \cdot (\Psi^{\hat{a}_0} - id)\right) \left(\Psi^{\hat{b}}(\mathcal{F})\right). \tag{2}$$

with input (modulus-switched) ciphertext $(\hat{b}, \hat{\mathbf{a}}) \in \mathbb{Z}_N^{n+1}$ under the secret $\mathbf{s} \in \mathbb{Z}^n$, and the encoding $\mathcal{F}$.

Now we discuss the encoding and decoding of the lookup table. Despite the abundance of many potential candidates, we are mostly interested in encoding and decoding functions that can be efficiently computed and are robust to noise growth. Specifically, we want the encoding and decoding algorithms to be quasi-linear $\tilde{O}(N)$ to the length of the lookup table N, as we use large $N \geq 2^{10}$ under practical and secure parameters. On the other hand, as homomorphic operations introduce noise, we would like the encoding and decoding to be noise-tolerant. In this work, we investigate two possible choices for the encoding and the corresponding decoding, namely the coefficient and slot encodings. These two encodings are commonly used encoding structures in various FHE schemes, including BGV, B/FV, and CKKS.

In conventional FHE schemes, the coefficient encoding directly utilizes the message vector as the coefficient vector of the message polynomial under a certain choice of the basis, mainly the monomial basis. It generally offers a decoding function with low noise growth and versatile parameter selection. To be specific, one can extract each entry of the lookup table by computing some linear combination of a few numbers of polynomial coefficients, where the linear combination has small coefficients that are unrelated to the plaintext space. Especially in FHEW/TFHE cryptosystems, the coefficient encoding allows us to extract only the constant term after the blind rotation. In this paper, we mainly focus on cases where the desired value can be obtained by extracting a single coefficient after the blind rotation, because the action of automorphisms on the coefficient vector is more than just a permutation in general cases.

On the other hand, the slot encoding exploits the Discrete Fourier Transformation (DFT) over the ring R_p. Specifically, given the lookup table, it firstly

finds a polynomial $q \in R_p$, whose evaluations at the roots of unity are equal to each entry, and uses $\lfloor \frac{Q}{p} \cdot q \rceil \in R_Q$ as an encoding. Implementing our blind rotation with the slot encoding is easier than the coefficient encoding, as the automorphism directly acts as a rotation of the encoded message vector [5]. However, under the slot encoding, decoding consumes a depth, meaning that the noise increases by a magnitude of p, the plaintext modulus.

Carousel. Based on our new blind rotation framework, we provide a new FHE scheme, called 'Carousel'. Carousel scheme supports integer arithmetic (e.g., addition, multiplication) modulo some prime power $p = \mathfrak{p}^r$ and arbitrary function evaluation. It operates over a subring R of a prime cyclotomic ring $\mathbb{Z}[X]/\Phi_M(X)$, which is invariant to the automorphism $X \mapsto X^{\mathfrak{p}}$, for some prime M. Then, for $N = \frac{\phi(M)}{d} = \frac{M-1}{d}$ where d is the order of p over $\mathbb{Z}_M^\times$, all the elements in R can be represented with only N coefficients. In this work, we investigate two bases, namely η-basis $\{\eta_i\}_{0 \le i < N}$ and ξ-basis $\{\xi_i\}_{0 \le i < N}$ that correspond to the monomial basis and the DFT basis, respectively.

The bootstrapping procedure of Carousel is composed of three algorithms, namely SampleExtract, KeySwitch, and BlindRotate. Its input and output are both RLWE ciphertexts under the ring key $t = \sum_{i=0}^{N-1} t_i \eta_i \in R$, of which the encoding may vary. During SampleExtract, the decoding function is homomorphically computed to generate an LWE ciphertext under the vectorized ring key $\mathbf{t} = (t_0, \ldots, t_{N-1}) \in \mathbb{Z}$. The following KeySwitch algorithm switches the secret of the input LWE ciphertext $\mathbf{t}$ to $\mathbf{s} = (s_0, \ldots, s_{n-1}) \in \mathbb{Z}^n$ for $n < N$, making the decryption circuit as compact as possible. Then the BlindRotate algorithm runs our new blind rotation framework homomorphically. It can switch between two modes: Carousel-C and Carousel-S, which encode the lookup table using the coefficient and slot encodings, respectively. In this paper, we mainly focus on how to implement SampleExtract and BlindRotate since KeySwitch is an already well-studied topic that does not need to be modified for Carousel.

We provide an efficient SampleExtract algorithm for η-basis, by directly writing the relation

$$\mu \approx b + a \cdot t = \sum_{i=0}^{N-1} b_i \eta_i + \left(\sum_{i=0}^{N-1} a_i \eta_i \right) \cdot \left(\sum_{i=0}^{N-1} t_i \eta_i \right) \pmod{Q}$$

for the input RLWE ciphertext $(b, a) \in R_Q$. On the other hand, the core formula Eq. (1) for BlindRotate is instantiated with a gadget-based approach. With the gadget-based approach, we can perform a homomorphic multiplication by s_i and automorphism by $\hat{a}_i$ with only an additive noise growth. Homomorphic evaluation of automorphism Ψ^i is instantiated with a RLEV ciphertext [6], which

[5] For the general case where the automorphism group does not form a cyclic group, the automorphisms act as a dimensional rotation of a hypercube. For more information, we refer to [HS14].

[6] RLEV is a cryptosystem that is frequently called as gadget encryption or key-switching key. In this work, we follow [CLOT21a] and refer it to RLEV, meaning Ring-leveled ciphertext.

encrypts $\Psi^i(t)$ under secret t. To allow the multiplicative operation, we utilize an RGSW [GSW13] ciphertext, which is essentially a stack of encryptions of s_i and $t \cdot s_i$ under secret t. As a result, we can obtain a compact parameter, leading to a small latency.

Finally, we implement Carousel in Go to prove the practicality of Carousel. We use Bergland's variant of real FFT [Ber68] to accelerate polynomial arithmetic. Also, we leverage the key-reusing technique and fast blind rotation techniques from [LMSS23].

Related Work. There have been few attempts to instantiate blind rotation with automorphisms. In [OPP23], the FHEW decryption circuit was evaluated in a SIMD manner by replacing the monomial with a generating element of a finite field, then the algebraic norm was computed to evaluate the lookup table. On the other hand, Liu et al. [LW23a,LW23b] proposed a different blind rotation technique that operates over a tensor ring, computing the decryption circuit over the exponents of the roots of unity. These works focused on the theoretical possibility for the amortized bootstrapping of LWE ciphertexts, rather than the practicality we target. On the other hand, Lee et al. [LMK+23] proposed another variant of the AP-like cryptosystem, which exploits homomorphic automorphism operations. Their method aims to multiply monomials, unlike ours, which directly leverages the automorphism group to perform blind rotation.

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 [Reg09] and its ring variants [LPR13,AH17,KS19].

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

Let ζ_M be the M-th root of unity, and $K := \mathbb{Q}(\zeta_M)$ be the cyclotomic number field. Then, the LWE problem can be instantiated over a subring of the ring of integers which is invariant to a certain set of automorphisms, which is referred to as a *decomposition ring*. We note that this is an extension of the original RLWE problem [LPR13,SSTX09].

Definition 2 (Decisional RLWE). *Let R be a decomposition ring, Q be a positive integer, and ψ an error distribution over R. Then, for $t \in R$, we define the RLWE distribution $\mathcal{D}^{\mathsf{RLWE}}_{R,Q,\psi}(t)$ 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 $b = -at + e \pmod{Q}$. For the key distribution χ over R, the decisional RLWE problem calls for distinguishing $\mathcal{D}^{\mathsf{RLWE}}_{R,Q,\psi}(t)$ for a secret $t \leftarrow \chi$ 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 the (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 coefficient of the ring element. We now present the description of gadget encryption and the RGSW scheme below. In the following, ψ denotes the noise distribution.

• $\underline{\mathsf{GadEnc.Enc}(t; \mu)}$: 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 3 (Gadget Product). *Let $\mathbf{C} \in R_Q^{d\times 2}$ be a gadget encryption and $a \in R_Q$ be a ring element. The gadget product, denoted by $\odot : R_Q \times R_Q^{d\times 2} \to R_Q^2$ 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 4 (External Product). *Let $\overline{\mathbf{C}} \in R_Q^{2d\times 2}$ be an RGSW ciphertext and $c = (b, a) \in R_Q^2$ be an RLWE ciphertext. The external product, denoted by $\boxdot : R_Q^2 \times R_Q^{2d\times 2} \to R_Q^2$ is defined as follows:*

$$(c, \mathbf{C}) \mapsto c \boxdot \mathbf{C} := \left[h(\lfloor \frac{b}{P} \rceil)^\top \; h(\lfloor \frac{a}{P} \rceil)^\top\right] \cdot \mathbf{C} \pmod{Q}.$$

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

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

2.4 Homomorphic Evaluation of Automorphisms

To homomorphically evaluate ring automorphisms on encrypted data, the gadget product is leveraged. Let Ψ be an automorphism defined over the base decomposition ring R, and $\mathsf{ct} = (b, a) \in R_Q^2$ is an RLWE ciphertext under the secret $t \in R$. Then, given the automorphism key $\mathsf{atk} = \mathsf{GadEnc.Enc}(t; \Psi(t))$ associated with the automorphism Ψ, the homomorphic automorphism algorithm $\mathsf{Aut}(\mathsf{atk}; \mathsf{ct})$ computes and outputs the following:

$$\mathsf{ct}' = (\Psi(b), 0) + \Psi(a) \odot \mathsf{atk} \pmod{Q}.$$

The algorithm above successfully evaluates the automorphism on the plaintext of the ciphertext, as the phase of the output of $\mathsf{ct}' = (b', a') \leftarrow \mathsf{Aut}(\mathsf{ct}, \mathsf{atk})$ is

$$\varphi_t(\mathsf{ct}') \approx \Psi(b) + \Psi(a) \cdot \Psi(t) = \Psi(b + at) = \Psi(\varphi_t(\mathsf{ct})) \pmod{Q}.$$

2.5 Decomposition Rings

Let us review the technical details on the decomposition rings, which will be used in the construction of Carousel. Below we choose the base ring R to be a subring of $\mathbb{Z}[X]/\Phi_M(X)$ for some prime M, which is invariant to the automorphism $X \mapsto X^{g^N}$, where g is a generator of $\mathbb{Z}_M^\times$ and N is some integer such that $N \mid M-1$. We write $o = (M-1)/N$, and $\Psi = X \mapsto X^g$. Also, we assume that o is even, so that the conjugate automorphism $X \mapsto X^{-1}$ is generated by $X \mapsto X^{g^N}$. Finally, the η-basis over R is given as $\{\eta_i\}_{0 \le i < N}$ for $\eta_i = \sum_{j=0}^{o} X^{g^{i+jN}}$.

ξ-basis. 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 the M-th root of unity over $\mathbb{C}$. 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 automorphism $X \mapsto X^N$. Moreover, as a is invariant under the conjugate automorphism as well, $a(\zeta_M^{g^i})$ 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 $\xi_0, \ldots, \xi_{N-1} \in \mathbb{R}[X]/\Phi_M(X)$, such that

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

for $0 \le i, j < N$. Now, let us write $a = \sum_{i=0}^{N-1} a_i \xi_i \in R$ for some $a_i \in \mathbb{R}$. Then, we have

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

This implies that the automorphism Ψ^i acts as a cyclic right shift of the coefficient vector in the ξ-basis by the index i. It is worth noting that the pointwise multiplication of coefficient vectors in ξ-basis directly translates into the multiplication over R, allowing us to perform fast ring arithmetic over R.

Conversion between η- and ξ-bases. Arita and Handa [AH17] proposed a fast conversion between η- and ξ-bases with a quasilinear complexity of $O(N \log N)$ to the input length. As Bernstein [Ber23] pointed out, their method is a folded version of Rader's DFT algorithm [Rad68].

Let us commence by investigating the conversion from η-basis to ξ-basis. Given a ring element a, let $\mathbf{a} = (a_0, \ldots, a_{N-1})$ and $\mathbf{b} = (b_0, \ldots, b_{N-1})$ be the

coefficient 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 $0 \leq i < N$. Evaluating this 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 (DFT) and inverse Fast Fourier Transform (iDFT), 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 [AH17].

Encoding and Decoding. Assume that the base ring R is a decomposition ring of which the number of messages is maximized, with respect to the plaintext modulus $p = \mathfrak{p}^r$. Then, the resolutions of unity $\{\tau_i\}_{0 \leq i < N}$ can be defined, as described in Sec. 3.2. We observe a similarity in the ξ-basis and the resolutions of unity, their only difference being the choice of space for the root of unity ζ_M. This allows us to perform the slot encoding and decoding with convolution, similar to the conversions between η- and ξ-bases. Specifically, the coefficient vector of the encoded polynomial of an LUT $\mathbf{v} = (v_i)_{0 \leq i < N} \in \mathbb{Z}_p^N$ can be obtained by computing a cyclic convolution between the coefficient vector of τ_0 and $\mathbf{v}$. Similarly, an encoding $\mu = \sum_{i=0}^{N-1} \mu_i \cdot \eta_i \in R_p$ can be decoded with a convolution between the coefficient vector $(\mu_0, \mu_{N-1}, \ldots, \mu_1) \in \mathbb{Z}_p^N$ and the coefficient vector of $M \cdot \tau_0 + o$ [7].

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 norms over R.

Proposition 1 (Norm Bound [AH17]). *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 commonly used distributions is presented below.

[7] We remark that this is only true when the base ring R is conjugate-invariant. Otherwise, we need to use the coefficient vector of $\Psi^{N/2}(M \cdot \tau_0 + o)$. In this work, we only consider the conjugate-invariant case.

Lemma 1 (Canonical Variance [AH20]).

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

Finally, we also remark that it holds that $\mathrm{Var}_\xi(ab) \le \|a\|_\xi^2 \cdot \mathrm{Var}_\xi(b)$ 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.

3 Our Framework

In this section, we first describe our new blind rotation algorithm, then discuss the potential encoding structures for this algorithm. Finally, we compare our algorithm to the prior works.

3.1 Our Blind Rotation

Let R be the base ring with cyclic automorphism group generated by Ψ, Q the ciphertext modulus, n the dimension of the input LWE ciphertext, and N the order of Ψ. We assume an encoding function $\mathsf{Ecd} : \mathbb{Z}_p^N \to R_Q$ and corresponding decoding function $\mathsf{Dcd} : R_Q \to \mathbb{Z}_p$, such that $\mathsf{Dcd}\left(\Psi^i\left(\mathsf{Ecd}(\mathbf{v})\right)\right) = v_i$ for any $\mathbf{v} = (v_0, \ldots, v_{n-1}) \in \mathbb{Z}_p^N$ and $0 \le i < N$. We investigate the encoding and decoding functions in the following subsection. For now, we use them as black-box functionality.

Algorithm 1 Our Blind Rotation

Input: LWE ciphertext $\mathbf{c} = (b, a_0, \ldots, a_{n-1}) \in \mathbb{Z}_Q^{n+1}$ under a binary secret $\mathbf{s} = (s_0, \ldots, s_{n-1}) \in \mathbb{Z}^n$, and a lookup table $\mathbf{v} \in \mathbb{Z}_p^N$.

1: $\mathcal{F} \leftarrow \mathsf{Ecd}(\mathbf{v})$
2: $\hat{b} \leftarrow \lfloor \frac{N}{Q} \cdot b \rceil$ and $\hat{a}_i \leftarrow \lfloor \frac{N}{Q} \cdot a_i \rceil$ for all $0 \le i < n$.
3: $\mathsf{ACC} \leftarrow \Psi^{\hat{b}}(\mathcal{F}) \in R_Q$
4: **for** $i = 0 : n - 1$ **do**
5: $\quad \mathsf{ACC} \leftarrow \mathsf{ACC} + \left(\Psi^{\hat{a}_i}(\mathsf{ACC}) - \mathsf{ACC}\right) \cdot s_i \pmod{Q}$
6: **end for**

Output: Return $\mathsf{Dcd}(\mathsf{ACC})$

We describe our blind rotation algorithm in Algorithm 1. It takes an LWE ciphertext $\mathbf{c} = (b, a_0, \ldots, a_{n-1}) \in \mathbb{Z}_Q^{n+1}$ under a binary secret $\mathbf{s} = (s_0, \ldots, s_{n-1}) \in \mathbb{Z}^n$, and a lookup table $\mathbf{v}$ as input, and outputs $v_{[\hat{b} + \langle \hat{\mathbf{a}}, \mathbf{s} \rangle]_N}$ where $\hat{b} = \lfloor \frac{N}{Q} \cdot b \rceil$ and $\hat{\mathbf{a}} = \lfloor \frac{N}{Q} \cdot \mathbf{a} \rceil$. In each iteration of the for loop in the algorithm (lines 4-6), the automorphism $\Psi^{\hat{a}_i s_i}$ is applied to the accumulator ACC. This is due to the

homomorphic MUX gate from the GINX/TFHE bootstrapping. Precisely, we write the selection with a linear combination of inputs c_0, c_1 and the selector $b \in \{0,1\}$, *i.e.,* $c := (1-b)c_0 + b \cdot c_1 = c_0 + (c_1 - c_0) \cdot b$. Hence we conclude that c is c_0 if $b = 0$, and c_1 if $b = 1$, $c = c_b$ essentially. As a result, assuming that s_i is binary, line 5 of Algorithm 1 outputs $\Psi^{\hat{a}_i}(\mathsf{ACC})$ if $s_i = 1$, and outputs $\mathsf{ACC} = \Psi^0(\mathsf{ACC})$ if $s_i = 0$. As $\hat{a}_i s_i$ is $\hat{a}_i$ if $s_i = 1$, and 0 if $s_i = 0$, it computes $\Psi^{\hat{a}_i s_i}(\mathsf{ACC})$ indeed. So, with ACC initialized with $\Psi^{\hat{b}}(\mathcal{F})$, the output of the for loop is $\Psi^{\hat{a}_{n-1}s_{n-1}}\left(\dots\left(\Psi^{\hat{a}_0 s_0}(\Psi^{\hat{b}}(\mathcal{F}))\right)\right) = \Psi^{\hat{b}+\langle\hat{\mathbf{a}},\mathbf{s}\rangle}(\mathcal{F})$. To combine this with the first and last line of the algorithm, we conclude that the algorithm output is given by $\mathsf{Dcd}(\Psi^{\hat{b}+\langle\hat{\mathbf{a}},\mathbf{s}\rangle}(\mathsf{Ecd}(\mathbf{v})))$. From the fact that Ψ^N is an identity automorphism and the relations of the encoding and decoding functions, the final output can be seen as $v_{[\hat{b}+\langle\hat{\mathbf{a}},\mathbf{s}\rangle]_N}$.

Discussion. In the described algorithm, we assumed a binary secret. However, some may assume ternary or Gaussian keys for security reasons, or a more compact parameter set. It is worth noting that our work can also cover the non-binary cases, by adopting a more generalized version of the homomorphic MUX gate [JP22]. Briefly speaking, we can homomorphically select between all possible values of $\Psi^{\hat{a}_i s_i}$, with respect to the value of s_i, given selectors that indicate if s_i is a certain value. In this work, we only focus on the binary case, which guarantees optimized performance.

3.2 Encoding and Decoding

We provide two potential encoding structures that are robust to the noise introduced by homomorphic computations. It can be obtained by adopting a similar strategy to Regev encryption, storing a message to the most significant bit (MSB) so that the noise fills up the least significant bits (LSB) without harming the message. Let us assume $R = \mathbb{Z}[X]/\Phi_M(X)$ for the cyclotomic degree $M \in \mathbb{Z}$, which is a common choice in lattice-based cryptography. In this paper, we only consider prime M for simplicity. Then the automorphism group of R is isomorphic to the multiplicative group $\mathbb{Z}_M^\times \cong \mathbb{Z}_{M-1}$ and is composed of automorphisms $X \mapsto X^i$ for all $0 < i < M$.

Coefficient Encoding. The most straightforward way of encoding values into a polynomial is storing them into the coefficients of the polynomial, with respect to a certain basis [8]. The easiest way would be, given a vector $\mathbf{v} = (v_0, \dots, v_{M-2}) \in \mathbb{Z}_p^{M-1}$, generating a polynomial $\mathcal{F} = \sum_{i=0}^{M-2} \lfloor \frac{Q}{p} v_i \rceil X^i \in R_Q$, as in AP-like cryptosystems. However, this naïve method cannot be used as an encoding function directly as some elements can be fixed under a certain set of automorphisms. For example, if M is a power-of-two, the constant term of the encoding is always v_0. Therefore, we devise a more suitable basis for encoding in coefficient form.

[8] In field algebra, such basis is called the *normal integral basis*. We do not discuss much about it in this paper, however, we believe that this could be an interesting mathematical tool to instantiate Carousel in arbitrary rings.

Let us begin with the basic case, where M is an odd prime. Let g be the generator of the multiplicative group $\mathbb{Z}_M^\times$. Then, all the automorphisms can be written as $X \mapsto X^{g^i}$ for all $0 \le i < M-1$. Therefore, the set of monomials $\{X, X^g, \ldots, X^{g^{M-2}}\}$ can be generated by applying automorphisms to X. Moreover, they form a basis of the base ring R [9]. Let us write $\Psi = X \mapsto X^g$. Then, under this basis, for any $a(X) = \sum_{i=0}^{M-2} a_i X^{g^i}$, we have

$$\Psi^j(a) = \sum_{i=0}^{M-2} a_i \Psi^j(X^{g^i}) = \sum_{i=0}^{M-2} a_i \left(X^{g^i}\right)^{g^j} = \sum_{i=0}^{M-2} a_i X^{g^{i+j}}$$

for any j. Observe that the coefficient vector of the resulting polynomial is $(a_{M-j-1}, \ldots, a_{M-1}, a_0, \ldots, a_{M-j-2})$, therefore, the automorphism Ψ^i simply right-rotates the coefficient vector by the index i. Based upon this observation, we can instead encode the lookup table $\mathbf{v} = (v_0, \ldots, v_{M-2})$ as $\mathcal{F} = \sum_{i=0}^{M-2} \lfloor \frac{Q}{p} v_i \rceil X^{g^{M-i-1}} \in R_Q$ and decode it by taking the coefficient of X term, then extracting the MSB by multiplying $\frac{p}{Q}$ and rounding.

We remark that this can be further generalized to a decomposition ring. Recall that a decomposition ring is essentially a subring of $\mathbb{Z}[X]/\Phi_M(X)$ which is invariant under a certain set of automorphisms. We assume that R only contains elements invariant under an automorphism $X \mapsto X^N$ for some $N \mid M-1$, without loss of generality. Let us write $o = (M-1)/N$. Then, the monomial basis collapses into the following basis set: $\eta_i := \sum_{j=0}^{o-1} X^{g^{i+jN}}$ for $0 \le i < N$. We call this an η-basis, following [AH17,AH20]. For the quotient group of this ring's automorphism group, we can choose the representatives of the coset as $X \mapsto X^{g^i}$ for $0 \le i < N$. Then, given a ring element $a = \sum_{i=0}^{N-1} a_i \eta_i \in R_Q$, we have

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

hence, Ψ^i also right-rotates the coefficient vector by the index i over the decomposition ring as well. Therefore, similar to the cyclotomic ring case, we can encode the lookup table $\mathbf{v} = (v_i)_{0 \le i < N}$ as $\mathcal{F} = \sum_{i=0}^{N-1} \lfloor \frac{Q}{p} v_i \rceil \eta_{N-i-1} \in R_Q$, and decode it by taking the coefficient of the η_0 term and extracting the MSB.

Slot Encoding. We also discuss another encoding structure, often referred to as 'slot encoding'. It is a commonly utilized encoding structure in the B/FV scheme [Bra12,FV12] and the BGV scheme [BGV14] to encode several Galois

[9] Note that the only monomial with wraparound by $\Phi_M(X) = X^{M-1} + \cdots + X + 1$ is X^{M-1}, resulting in $-X^{M-2} - \cdots - 1$. Since all the other monomials $X^{M-2}, \ldots, X$ are generated with automorphism, and $-X^{M-2} - \cdots - 1$ has 1, R is spanned by this set.

ring elements into a single polynomial. Assume a prime number M, and some prime power $p = \mathfrak{p}^r$ for a prime $\mathfrak{p}$ and an integer $r > 0$. Then, the cyclotomic polynomial $\Phi_M(X)$ is factored into N irreducible polynomials F_i of degree o, where o is the multiplicative order of $\mathfrak{p}$ modulo M and $N = \phi(M)/o$. Specifically, such F_i's can be obtained by $F_i = \prod_{j=0}^{o-1}(X - \zeta_M^{g^i\mathfrak{p}^j}) = \prod_{j=0}^{o-1}(X - \zeta_M^{g^{i+jN}})$ for a root ζ_M of Φ_M over $GR(p, o)$ [10]. Due to the Chinese Remainder Theorem (CRT), there exists a corresponding polynomial $\mu(X) \in \mathbb{Z}_p[X]/\Phi_M(X)$ such that $\mu = v_i \pmod{F_i}$ for all $0 \le i < N$, given an arbitrary $\mathbf{v} = (v_i)_{0\le i<N} \in \mathbb{Z}_p$. We remark that there exists a set of polynomials $\{\tau_i\}_{0\le i<N}$ such that $\mu = \sum_{i=0}^{N-1} v_i\tau_i$. This can be seen as a basis with respect to the slot encoding, often referred to as the *resolution of unities*. The i-th resolution of unity τ_i can be determined as $\tau_j = \left[\left(\prod_{j\neq i} F_j(X)\right)^{-1}\right]_{F_i} \prod_{j\neq i} F_j(X) \in R_p$. This construction ensures that

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

for all $0 \le i, j < N$. We remark that this implies that $\tau_i \cdot \tau_j = 0$, and $\tau_i \cdot \tau_i = \tau_i$ for all $0 \le i \neq j < N$, hence the ring arithmetic on μ directly translates to pointwise arithmetic over the packed messages $\mathbf{v}$. Also, $\Psi^j(\tau_i) = \tau_{[i-j]_N}$ for all $0 \le i, j < N$ where $\Psi = X \mapsto X^g$, from the definition of F_i's and τ_i's. This implies that the automorphism Ψ^i acts as a left rotation of the encoded message by the index i. Consequently, the encoding of a lookup table $\mathbf{v} = (v_i)_{0\le i<N} \in \mathbb{Z}_p^N$ can be done by $\mathcal{F} = \sum_{i=0}^{N-1} \lfloor \frac{Q}{p} \cdot v_i \rceil \tau_i \in R_Q$.

We further discuss the efficiency of the slot encoding. Note that the aforementioned encoding structure encodes only $N = (M-1)/o$ messages into a polynomial with $M - 1$ coefficients. This leads to d times larger polynomial size compared to the encrypted data, resulting in practical inefficiency. To optimize the blind rotation performance, we need to maximize the number of messages packed in a single polynomial. In other words, we aim to have the packing density as close to one as possible, which is given by

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

This issue can be resolved by taking a decomposition ring R of $\mathbb{Z}[X]/\Phi_M(X)$ that is invariant to the automorphism $X \mapsto X^N$, as the base ring. Observe that every element in R can be represented with only N coefficients, and all the integer encodings reside in this decomposition ring as they are invariant to the automorphism $X \mapsto X^N$. Hence, we obtain the optimal packing density 1 with such a choice of R.

Now we discuss the decoding. Based on the encoding, we need to extract the coefficient, with respect to the basis τ_0 of R_p. Traditionally, one can evaluate some linear transformation to convert to coefficient encoding via automorphisms [HS14], and then exploit the decoding function for coefficient encoding.

[10] Note that the multiplicative order of the Galois Ring $GR(p, o)$ is given by $\mathfrak{p}^{o(r-1)}(\mathfrak{p}^o - 1)$. By the definition of order, $\mathfrak{p}^o - 1 = 0 \pmod{M}$ and therefore $M \mid \mathfrak{p}^{o(r-1)}(\mathfrak{p}^o - 1)$, implying that there exists a nontrivial ζ_M such that $\zeta_M^M = 1$.

However, this can be computationally inefficient as it requires $O(\sqrt{N})$ automorphisms. Instead, we devise a computationally efficient method that does not require any automorphism operation. We begin by noting that there should exist at least one coefficient of τ_0 that is coprime to the base prime $\mathfrak{p}$. Without loss of generality, assume that the first coefficient of τ_0 (which corresponds to η_0) is some integer $\gamma \in \mathbb{Z}_p$ and $gcd(\gamma, p) = 1$ [11]. Now, suppose that we are given an encoding $\mu \in R_Q$ of the lookup table $\mathbf{v} = (v_i)_{0 \le i < N} \in \mathbb{Z}_p$, *i.e.*, $\mu = \sum_{i=0}^{N-1} v_i \tau_i$. As γ is coprime to p, the inverse γ^{-1} exists in the commutative ring $\mathbb{Z}_p$, and thus $(\gamma^{-1}\tau_0) \cdot \mu = \sum_{i=0}^{N-1} (\gamma^{-1}\tau_0) \cdot v_i \tau_i = \gamma^{-1} v_0 \tau_0$. Since the first term of τ_0 in the η-basis is γ, the first term of $\gamma^{-1} v_0 \tau_0$ is v_0. Therefore, we can simply multiply $[\gamma^{-1}\tau_0]_p$ to the input ciphertext and extract the first term in η-basis.

Comparison. There are two main differences between the coefficient and slot encodings, namely the level consumption and the compatibility with SIMD FHE schemes. The coefficient encoding is somewhat similar to the encoding strategy of the existing AP-like cryptosystems. Its decoding can be instantiated level-free, but it does not support the multiplicative operation naturally. To enable multiplicative operations, it needs to be converted to slot encoding via automorphisms. On the other hand, the slot encoding enables us to perform additive and multiplicative operations without any additional afterprocessing, sacrificing (half) a multiplicative level from decoding.

It is an interesting compromise, as AP-like schemes generally aim to set the tightest parameter to enable low-latency functional bootstrapping. When following such a tradition with our framework, it would be necessary to leverage the coefficient encoding. Nevertheless, a number of recent works on AP-like cryptosystems incorporate large parameters which can tolerate multiplicative operations to enhance the overall performance, including [CLOT21b,MHW+24,HMS25]. Under these circumstances, we expect that the slot encoding may come in handy.

4 Carousel: Fully Homomorphic Encryption

In this section, we propose Carousel, a new fully homomorphic encryption scheme that is based on our new blind rotation algorithm.

4.1 Scheme Description

Now, we describe Carousel scheme. It has two modes, namely Carousel-C and Carousel-S. These two modes only differ in the encoding of the message, and can be freely converted to each other via the (functional) bootstrapping. A detailed description will be given below.

It supports integer arithmetic for the commutative ring $\mathbb{Z}_p$, where $p = \mathfrak{p}^r$ for some prime $\mathfrak{p}$ and integer $r > 0$. Below, $\mathcal{D}_\alpha$ and $\mathcal{D}_\beta$ denote the discrete

[11] Since Ψ operates as a left shift of the coefficient vector, the coefficient vector of τ_i is essentially a left shift of the coefficient vector of τ_0 by index i. Therefore, we can simply choose any τ_i of which the first coefficient is a coprime with p to encode v_0.

Gaussian distributions with standard deviations α and β, defined over $\mathbb{Z}$ and R in η-basis, respectively. We note that 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.

• $\mathsf{Setup}(1^\lambda)$: Given the security parameter λ, set the LWE dimension n, the cyclotomic degree M, the ciphertext and plaintext moduli Q, 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. Output $\mathsf{pp} = (n, M, Q, p, \alpha, \beta, \mathbf{g}, h, \mathbf{g}', h')$.

In the following, we use the following terminology given the parameters:

- R denotes the base decomposition that is a subring of $\mathbb{Z}[X]/\Phi_M(X)$ invariant under the automorphism $X \mapsto X^{\mathfrak{p}}$.
- N and o denote degree of the decomposition ring N, and the multiplicative order of $\mathfrak{p}$ modulo M. Note that $N \cdot o = M - 1$.
- $\{\eta_i\}_{0 \le i < N}$ and $\{\xi_i\}_{0 \le i < N}$ denotes the η- and ξ-bases of R, respectively. $\{\tau_i\}_{0 \le i < N}$ indicates the resolutions of unity over R_p. Here, we choose τ_0 to have first term which is coprime to p.
- For g, a generator of the multiplicative group $\mathbb{Z}_M^\times$, Ψ denotes the automorphism $X \mapsto X^g$.
- $\sigma : R_p \to \mathbb{Z}_p^N$ denotes the isomorphism between R_p and $\mathbb{Z}_p^N$, as discussed in Sec. 3.2.

• $\mathsf{KeyGen}(\mathsf{pp})$:

- Sample $\mathbf{s} \leftarrow \mathcal{U}(\{0,1\}^n)$ and $t_i \leftarrow \mathcal{U}(\{0,1\})$ for $0 \le i < N$, and set $t = \sum_{i=0}^{N-1} t_i \cdot \eta_i$.
- Set $\mathsf{atk} = \{\mathsf{atk}_i\}_{0 \le i < N}$ for $\mathsf{atk}_i \leftarrow \mathsf{GadEnc.Enc}(t; \Psi^i(t))$ 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 $\mathsf{ksk} = \{\mathsf{ksk}_i\}_{0 \le i < N}$.

Now we describe the encryption and decryption algorithm. Note that the decryption algorithm differs with respect to the encoding mode of the input ciphertext due to the difference in the decoding function. We note that the basic ciphertext form is RLWE in our scheme, and we will only care about the first entry of the encoded vector in the decryption.

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

• $\mathsf{Dec}(t; \mathsf{ct})$: Given $\mathsf{ct} \in R_Q^2$, compute $\mu = \lfloor \frac{p}{Q} \cdot \varphi_t(\mathsf{ct}) \rceil$. Then, evaluate the decoding function on μ and output the result.

Note that the arithmetic operations, such as addition/subtraction/multiplication, are essentially the same as BFV encryption. We refer the readers to [AH17,AH20] for further details. We stress that the multiplication can be applied to only slot-encoded ciphertexts.

Now we describe the functional bootstrapping of Carousel. It consists of three steps: 1. Sample Extraction, 2. Key Switching, and 3. Blind Rotation. These three algorithms are described in the following.

Sample Extraction. Our sample extraction algorithm homomorphically computes the decoding function, with respect to the encoding structure of the input ciphertext. Regardless of the encoding, the decoding function includes the extraction of the coefficient of η_0 term. Assume that we are given $\mathsf{ct} = (b, a) \in R_Q^2$ such that $a = \sum_{i=0}^{N-1} a_i \eta_i$ and $b = \sum_{i=0}^{N-1} b_i \eta_i$, under the key t. Observe that $\eta_0 = X + X^{g^N} + \cdots + X^{g^{(o-1)N}}$, we aim to directly compute the coefficient of X term of $a \cdot t$. With a direct computation, we obtain

$$
\begin{aligned}
a \cdot t &= \Big(\sum_{i=0}^{N-1} a_i \sum_{j=0}^{o-1} X^{g^{i+jN}}\Big)\Big(\sum_{i=0}^{N-1} t_i \sum_{j=0}^{o-1} X^{g^{i+jN}}\Big) \\
&= \sum_{i=0}^{N-1} t_i \sum_{\beta=0}^{o-1} a_{[\log_g(-g^{i+\beta N})]_N} + X \cdot \sum_{i=0}^{N-1} t_i \sum_{\beta=0}^{o-1} a_{[\log_g(1-g^{i+\beta N})]_N} + \ldots \pmod{X^M - 1} \\
&= \eta_0 \cdot \sum_{i=0}^{N-1} t_i (-o \cdot a_i + \sum_{\beta=0}^{o-1} a_{[\log_g(1-g^{i+\beta N})]_N}) + \ldots \pmod{\Phi_M(X)}.
\end{aligned}
$$

Based on this observation and the discussion in Sec. 3.2, we devise the following sample extraction algorithm.

- $\mathsf{SampleExtract}(\mathsf{ct})$: Given the ciphertext $\mathsf{ct} = (b, a) \in R_Q^2$, do the following.
  - Compute $\mathsf{ct}' \leftarrow [\gamma^{-1}\tau_0]_p \cdot \mathsf{ct}$ if ct is slot-encoded, and $\mathsf{ct}' \leftarrow \mathsf{ct}$ otherwise.
  - Output $(b', \mathbf{a}') \in \mathbb{Z}_Q^{N+1}$ where $b' = b_0$, $a'_i = -o \cdot a_i + \sum_{\beta=0}^{o-1} a_{[\log_g(1-g^{i+\beta N})]_N} \pmod{Q}$ for all $0 \le i < N$, and $\mathbf{a}' = (a'_0, \ldots, a'_{N-1})$.

Key Switching. The key-switching algorithm homomorphically switches the key of the input LWE ciphertext from the vectorized ring secret $\mathbf{t} \in \mathbb{Z}^N$ to a smaller one, $\mathbf{s} \in \mathbb{Z}^n$.

- $\mathsf{KeySwitch}(\mathsf{ksk}; \mathbf{c})$: 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. Our blind rotation algorithm is described in Alg. 2. In our algorithm, the homomorphic MUX gate is implemented with the external product. This choice allows us to perform blind rotation with only additive noise growth. It is also worth noting that the encoding function Ecd differs with respect to the encoding mode of the output ciphertext.

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

Algorithm 2 $\mathsf{BlindRotate}(\mathsf{brk}, \mathsf{atk}; \mathbf{c}, f)$

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

1: $\hat{b} \leftarrow \lfloor \frac{N}{Q} \cdot b \rceil \in \mathbb{Z}_N$ and $\hat{a}_i = \lfloor \frac{N}{Q} \cdot a_i \rceil \in \mathbb{Z}_N$ for $0 \le i < n$.
2: $v_i \leftarrow f(\lfloor \frac{p}{N} i \rceil)$ for $0 \le i < N$.
3: $\mathcal{F} \leftarrow \mathsf{Ecd}((v_i)_{0 \le i < N})$.
4: $\mathsf{ACC} \leftarrow \left(\Psi^{\hat{b}}(\mathcal{F}), 0\right) \in R_Q^2$
5: **for** $i = 0 : n-1$ **do**
6: $\quad \mathsf{ACC} \leftarrow \mathsf{ACC} + (\mathsf{Aut}(\mathsf{atk}_{\hat{a}_i}; \mathsf{ACC}) - \mathsf{ACC}) \boxdot \mathsf{brk}_i$
7: **end for**

Output: Return ACC

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

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

Output: Return ct'

Security The security of our scheme relies on the LWE assumption and the RLWE assumption defined over the decomposition ring [AH17,AH20]. On the other hand, note that the blind rotation key brk encrypts the LWE secret $\mathbf{s}$ under ring secret t, the automorphism key atk encrypts the automorphisms of 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.

4.2 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, the variances of the distributions $\mathcal{D}_\alpha$ and $\mathcal{D}_\beta$ are α^2 and β^2, respectively. Now, we provide the noise analysis for the basic operations of 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,*

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

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,

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

. 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, $\mathrm{Var}_\xi\left(h(\lfloor \frac{a}{P} \rceil)^\top \cdot \mathbf{e}\right) = \frac{1}{3} do^2 N^2 \mathcal{B}^2 \beta^2$. It concludes the proof. □

The noise growth from the homomorphic automorphism is given as follows.

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

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

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

$$\begin{aligned}\Psi^i(\varphi_t(\mathsf{ct})) - \varphi_t(\mathsf{Aut}(\mathsf{atk}_i; \mathsf{ct})) &= \Psi^i(b + at) - \varphi_t\left((\Psi^i(b), 0) + \Psi^i(a) \odot \mathsf{atk}_i\right) \\ &= \Psi^i(b) + \Psi^i(a \cdot t) - \Psi^i(b) - \varphi_t(\Psi^i(a) \odot \mathsf{atk}_i) \\ &= \Psi^i(a) \cdot \Psi^i(t) - \varphi_t(\Psi^i(a) \odot \mathsf{atk}_i).\end{aligned}$$

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

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,*

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

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

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

□

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

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

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 $\|b\|_\xi^2 = b \leq 1$. □

Now, we prove the noise growth from our blind rotation algorithm. We follow the same notation from the algorithm.

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

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

where $\mathcal{F}$ is the lookup table encoding of the evaluated function f.

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, $\varphi_t(\mathsf{ACC}_0) - \Psi^{\hat{b}}(\mathcal{F}) = 0$. Now, we have $\mathsf{ACC}_{i+1} = \mathsf{ACC}_i + (\mathsf{Aut}(\mathsf{atk}_{\hat{a}_i}; \mathsf{ACC}_i) - \mathsf{ACC}_i) \boxdot \mathsf{brk}_i$. Then,

$$\mathrm{Var}_\xi\left(\varphi_t\left(\mathsf{Aut}(\mathsf{atk}_{\hat{a_i}}; \mathsf{ACC}_i)\right) - \Psi^{\hat{a}_i}(\varphi_t(\mathsf{ACC}_i))\right) \le \frac{1}{12}o^2N^2P^2 + \frac{1}{3}do^2N^2\mathcal{B}^2\beta^2$$

by Cor. 1. Consequently, we have

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

By iteratively applying this inequality, we obtain

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

□

We prove the noise growth from the sample extraction.

Theorem 2 (Sample Extraction). *Let* $\mathsf{ct} \in R_Q^2$ *be an RLWE ciphertext such that* $\varphi_t(\mathsf{ct}) = \frac{Q}{p}\cdot\mu + e$ *for some* $e \in R$ *close to spherical Gaussian. Then, for* $\mathbf{c} \leftarrow \mathsf{SampleExtract}(\mathsf{ct})$, $\mathrm{Var}\left(\varphi_{\mathbf{t}}(\mathbf{c}) - \frac{Q}{p}m_0\right) \le \mathrm{Var}_\xi(e)$ *if* μ *is coefficient encoded and* $\mu = \sum_{i=0}^{N-1} m_i\cdot\eta_i$, *and* $\mathrm{Var}\left(\varphi_{\mathbf{t}}(\mathbf{c}) - \frac{Q}{p}m_0\right) \le \frac{1}{12}p^2\mathrm{Var}_\xi(e)$ *if* μ *is slot encoded and* $\mu = \sum_{i=0}^{N-1} m_i\cdot\tau_i$.

Proof. When μ is coefficient encoded, it directly follows from the functionality of the sample extraction. Now we consider the case where μ is slot encoded. Heuristically, we assume that the coefficients of τ_0 are uniformly distributed over the set $[-p/2, p/2]\cap\mathbb{Z}$. Then, the coefficients of $\gamma^{-1}\tau_0$ are also uniformly distributed, and thus $\mathrm{Var}_\xi(\gamma^{-1}\tau_0) = \frac{1}{12}oN\cdot p^2$. Therefore, the noise term $\gamma^{-1}\tau_0\cdot e$ has the canonical variance $\mathrm{Var}_\xi(\gamma^{-1}\tau_0\cdot e) = \frac{1}{12}oN\cdot p^2\mathrm{Var}_\xi(e)$. We also use a heuristic that $\gamma^{-1}\tau_0\cdot e$ is close enough to some Gaussian distribution, specifically a multi-dimensional Gaussian with covariance matrix related to the matrix form of the polynomial $\gamma^{-1}\tau_0$. Then we extract the constant term in the η-basis of $\gamma^{-1}\tau_0\cdot e$ in the sample extraction step, resulting in this multi-dimensional Gaussian collapsing into a one-dimensional Gaussian distribution. Therefore, we can assume the variance for $\varphi_t(\mathbf{c}) - \frac{Q}{p}m_0$ to be $\frac{1}{12}p^2\mathrm{Var}_\xi(e)$. □

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

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

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

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* $\varphi_t(\mathsf{ct}) = \frac{Q}{p} \cdot \mu + e_{\mathsf{ct}}$ *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* $f : \mathbb{Z}_p \to \mathbb{Z}_p$, $\mathsf{ct}' \leftarrow \mathsf{Bootstrap}(\mathsf{brk}, \mathsf{atk}, \mathsf{ksk}; \mathsf{ct}, f)$ *satisfies that* $\varphi_t(\mathsf{ct}') = \frac{Q}{p} \cdot \mu' + e_{\mathsf{ct}'}$ *for some* m' *and* $e_{\mathsf{ct}'}$ *where*

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

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

when the input message is coefficient encoded and

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

when the input message is slot encoded, and

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

Proof. By Theorems 2 and 3, the LWE ciphertext after the key-switching step has the noise variance of $\mathrm{Var}_\xi(e_{\mathsf{ct}}) + \frac{1}{12}NP' + \frac{1}{3}d'NB'^2\alpha^2$ if the input message is coefficient encoded, and $\frac{1}{12}p^2\mathrm{Var}_\xi(e_{\mathsf{ct}}) + \frac{1}{12}NP' + \frac{1}{3}d'NB'^2\alpha^2$ otherwise. 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

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

if the input message is coefficient encoded, and

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

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

4.3 Comparison

Finally, we analyze the computational complexity and key size of Carousel, 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 in blind rotation, namely external and gadget products, we estimate the computational complexity by counting the number of these operations. On the other hand, as the size of the key-switching keys is essentially the same for Carousel and other schemes, we only focus on the size of the keys that are used for the blind rotation, which shows differences.

Let us fix the parameters n and N for all the schemes. The blind rotation of Carousel requires an external product and a gadget product in each iteration, leading to a total of n external and gadget products. On the other hand, Carousel requires the encryptions of each key component s_i $(0 \leq i < n)$, resulting in n RGSW ciphertexts. Additionally, it utilizes $N-1$ automorphism keys for every automorphism 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 automorphism keys. Specifically, we can provide only $\log N$ automorphism keys which correspond to Ψ, Ψ^2, …, $\Psi^{2^{\lceil \log N \rceil - 1}}$. Then, the automorphism Ψ^i can be instantiated with a composition of these automorphisms 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 automorphism Ψ^i can be homomorphically computed, by evaluating the automorphism Ψ^{2^j} if i_j is nonzero, for all $0 \leq j < \lceil \log N \rceil$. 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, requiring $b \log_b N$ keys.

Method	# RGSW	# Gadget Encryption	# External products	# Gadget Product
[ASP14,DM15]	$n \cdot \log N$	0	$n \cdot \log N$	0
[LMK+23] (Alg.4)	$n+1$	$w+1$	$\leq \frac{3}{2}n+1$	$\leq n + \frac{N-n}{w}$
[GINX16,CGGI20]	n	0	n	0
Carousel	n	$N-1$	n	n

Table 2. Comparison between our framework and AP-like methods. In the table, n and N denote the dimensions for LWE and RLWE, respectively, and w denotes the windowing parameter. Typically, $w \leq 16$ in practical parameter sets [LMK+23].

In Table 2, we provide the number of the RGSW and gadget encryptions, and the number of the external and gadget products required to perform blind rotation in the previous AP-like cryptosystem and in Carousel. For the LMKCDEY scheme, w denotes the windowing parameter, which is used to reduce the time complexity for bootstrapping. A typical choice of w is a small integer under 10 [LMK+23]. Carousel achieves a smaller key size than AP/FHEW, while requiring a larger key size than the LMKCDEY and GINX/CGGI. To compare the time complexity, we first note that the external product is essentially two gadget products. Also, it is worth noting that the time complexity of blind rotation of the LMKCDEY scheme is not deterministic, instead, it depends on the input ciphertext. Therefore, the number of gadget products performed in AP/FHEW, LMKCDEY, GINX/CGGI, and Carousel is given as $2n \log N$, $\sim 4n + (N-n)/w + 1$, $2n$, and $3n$, respectively. Solely based on this counting, we expect that Carousel 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 functional bootstrapping.

5 Implementation

In this section, we implement a proof-of-concept implementation of our scheme, and illustrate its performance. In the implementation, 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 $-D/2 < a_i \leq D/2$. 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.

5.1 Parameter Selection

When choosing the right parameter for Carousel, it is necessary to choose the ring degree M and the plaintext modulus $p = \mathfrak{p}^r$ adequately. Let us recall the conditions we require. Firstly, the ring degree is given by $N = (M-1)/o$, where o is the order of $\mathfrak{p}$ in Carousel. On the other hand, the ring degree N needs to be large enough to accommodate the lookup table and also to guarantee the right security. Additionally, we want to set M as small as possible since the noise variance for slot encoding is (almost) proportional to M^2, as analyzed in 4.2. On the implementation side, 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 have, the slower the latency for convolution can become. Therefore, for better efficiency, M should be factorized into many small primes. Therefore, in our implementation, we choose $M = 65537$ and $\mathfrak{p} = 2$, which guarantees a fast FFT with $N = 2048$.

In Table 3, the concrete parameter sets for our fully homomorphic encryption scheme are presented. We fix the ciphertext modulus Q to $Q = 2^{64}$ to leverage the natural modular arithmetic of 64-bit unsigned integers. In the table, we use n and α to indicate the LWE parameters, namely LWE dimension and standard deviation for the noise distribution $\mathcal{D}_\alpha$. β denotes the standard deviation for the noise distribution $\mathcal{D}_\beta$. Finally, D', d', D, and d are the gadget decomposition parameters for the key-switching and blind rotation. We set the parameters so that the bootstrapping failure probability is smaller than 2^{-128}, achieving at least 128-bit security. Note that we set the parameters for the coefficient mode and slot mode differently because the coefficient mode introduces smaller noise. This allows us to perform the blind rotation for the coefficient mode more lightly.

Mode	p	n	α	D'	d'	N	β	D	d
Carousel-C	2^2	687	$2^{49.07}$	2^2	6	2048	$2^{13.05}$	2^{20}	1
	2^3	788	$2^{46.60}$	2^4	3		$2^{13.09}$	2^{22}	1
	2^4	828	$2^{45.57}$	2^3	5		$2^{13.09}$	2^{23}	1
Carousel-S	2^2	687	$2^{49.07}$	2^2	6		$2^{13.05}$	2^{14}	2
	2^3	788	$2^{46.60}$	2^4	3		$2^{13.09}$	2^{16}	2
	2^4	828	$2^{45.57}$	2^3	5		$2^{13.09}$	2^{17}	2

Table 3. The parameters for Carousel scheme. Carousel-C and Carousel-S correspond to the coefficient and slot encodings, respectively.

5.2 Experimental Results

We describe the experimental results of our scheme in this subsection. The proposed scheme is implemented from scratch in Go, and all experiments were

performed on a machine equipped with 11th Gen Intel(R) Core(TM) i9-11900, with 32 GB of RAM. Our source code is available at `https://github.com/SNUCP/carousel`. We used optimization techniques proposed by Lee, Min, Seo, and Song [LMSS23] to accelerate Carousel. The Carousel scheme with these optimization techniques is described in Sec. A. Table 4 summarizes the running time of a single functional bootstrapping and the size of the evaluation keys, namely the blind rotation key brk, automorphism key atk, and the key-switching key ksk under the corresponding parameter set. Runtimes are averaged over ten executions. The benchmark results show a small latency of below 30ms for functional bootstrapping of $2, 3$ and 4 bit integers.

Table 4. The key sizes and latency for a single functional bootstrapping using Carousel.

Mode	p	brk	atk	ksk	Elapsed Time
Carousel-C	2^2	43.17MB	64.31MB	44.00MB	19.0ms
	2^3	49.51MB	64.31MB	23.36MB	18.9ms
	2^4	52.03MB	64.31MB	39.46MB	19.6ms
Carousel-S	2^2	86.26MB	128.52MB	44.00MB	28.4ms
	2^3	98.94MB	128.52MB	23.36MB	28.8ms
	2^4	103.97MB	128.52MB	39.46MB	29.6ms

Comparison. We briefly compare our scheme to other AP-like schemes, in terms of key size and performance. We used OpenFHE [BAB+22] library for the FHEW and LMKCDEY schemes, where the corresponding parameter sets are `STD128_AP` and `STD128_LMKCDEY`. Note that they only have parameter sets for binary data, which can be seen as a two-bit message space with a padding bit. For TFHE, we used the TFHE-rs library [Zam22], with parameters `V1_3_PARAM_MESSAGE_2_CARRY_0_COMPACT_PK_KS_PBS_GAUSSIAN_2M64`, `V1_3_PARAM_MESSAGE_2_CARRY_0_COMPACT_PK_KS_PBS_GAUSSIAN_2M64` and `V1_3_PARAM_MESSAGE_2_CARRY_0_COMPACT_PK_KS_PBS_GAUSSIAN_2M64`, for the message spaces 2^2, 2^3 and 2^4, respectively. All the experiments were run in the same environment, and the key sizes were estimated for a fair comparison. The experimental results are summarized in Table 5.

The result shows that a full-domain functional bootstrapping for Carousel is 5.8 and 6.3 times faster than a negacyclic functional bootstrapping of FHEW and LMKCDEY, respectively. On the other hand, a full-domain functional bootstrapping for Carousel is 1.27 to 1.72 times slower than a negacyclic functional bootstrapping for TFHE. We emphasize that the existing full-domain bootstrapping methods require more than two anti-circular bootstrappings with additional operations and parameter overheads, our proof-of-concept implementation of Carousel can be competitive even for the production-ready implementations such as TFHE-rs when the full-domain functionality is required.

Table 5. The latency and total key sizes for a single functional bootstrapping of FHEW [DM15], LMKCDEY [LMK+23], TFHE [Zam22] and Carousel-C.

Scheme	$p = 2^2$	$p = 2^3$	$p = 2^4$
FHEW	(114ms, 556.3MB)	-	-
LMKCDEY	(124ms, 55.4MB)	-	-
TFHE	(11.4ms, 118.4MB)	(12.5ms, 105.0MB)	(15.4ms, 122.4MB)
Carousel-C	(19.6ms, 151.5MB)	(19.4ms, 137.2MB)	(19.6ms, 155.8MB)

Acknowledgements. We thank Guillaume Hanrot for the fruitful discussion. This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. RS-2023-00211649) and the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. RS-2024-00399491, Development of Privacy-Preserving Multiparty Computation Techniques for Secure Multiparty Data Integration).

A Carousel with Block Binary Keys

In [LMSS23], the authors suggested two optimization techniques for TFHE bootstrapping. We adapt their idea to enhance the performance of Carousel. Their first observation is that a sequential evaluation of ℓ two-to-one homomorphic MUX gates can be integrated 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 $0 \le i < k$. Then, at each iteration, it only remains to select between ACC, $\Psi^{\hat{a}_{i\ell}}(\mathsf{ACC})$, $\ldots$, $\Psi^{\hat{a}_{(i+1)\ell-1}}(\mathsf{ACC})$ with respect to the value of $s_{i\ell}$, $\ldots$, $s_{(i+1)\ell-1}$. More precisely, we homomorphically compute

$$\begin{aligned}\Psi^{\sum_{j=0}^{\ell-1}\hat{a}_{i\ell+j}s_{i\ell+j}}(\mathsf{ACC}) &= \mathsf{ACC} + \sum_{j=0}^{\ell-1}\left(\Psi^{\hat{a}_{i\ell+j}}(\mathsf{ACC}) - \mathsf{ACC}\right)\cdot s_{i\ell+j} \\ &= \mathsf{ACC} + \sum_{j=0}^{\ell-1}\left(\Psi^{\hat{a}_{i\ell+j}}(\mathsf{ACC}\cdot s_{i\ell+j}) - \mathsf{ACC}\cdot s_{i\ell+j}\right) \qquad (3)\end{aligned}$$

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

$\hat{b} \leftarrow \lfloor \frac{N}{Q} \cdot b \rceil \in \mathbb{Z}_N$ and $\hat{a}_i = \lfloor \frac{N}{Q} \cdot a_i \rceil \in \mathbb{Z}_N$ for $0 \le i < n$.
$v_i \leftarrow f(\lfloor \frac{p}{N} i \rceil)$ for $0 \le i < N$.
$\mathcal{F} \leftarrow \mathsf{Ecd}((v_i)_{0 \le i < N})$.
for $i = 0 : k-1$ **do**
    $\mathsf{hACC} \leftarrow h(\mathsf{ACC})$
    for $j = 0 : \ell - 1$ **do**
        $\mathsf{tmp} \leftarrow \mathsf{hACC}^\top \cdot \mathsf{RGSW}(s_{\ell \cdot i + j})$
        $\mathsf{ACC} \leftarrow \mathsf{ACC} + \mathsf{Aut}(\mathsf{atk}_{\hat{a}_{\ell \cdot i + j}}; \mathsf{tmp}) - \mathsf{tmp}$
    end for
end for

Output: Return ACC

On the other hand, they also proposed to reuse the LWE secret $\mathbf{s} \in \mathbb{Z}^n$ for the coefficient 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 $\mathbf{c} = (b, a_0, \ldots, a_{N-1}) \in \mathbb{Z}_Q^{N+1}$.

Security and Parameters To set the parameters, we follow the heuristic from [LMSS23]. The authors considered several attacks to measure the security. Firstly, the recent meet-in-the-middle (MitM) attack by May [May21] can successfully attack the block key structure using its small entropy of the secret. Considering that it runs in $\mathcal{S}^{0.28}$ time for the size of key space $\mathcal{S}$, which is given by $(\ell+1)^k$ with the block binary distribution, we obtain the following inequality $128 \cdot 0.28 \le k \log_2(\ell+1)$ to meet the 128-bit security. On the other hand, we also consider the hybrid dual attack for sparse secrets. In the paper, we choose the block parameter accordingly, while maximizing the block length ℓ and satisfying the 128-bit security for both the MitM and hybrid dual attack.

A.1 Noise Analysis

Below, we provide the noise analysis for the blind rotation and key-switching algorithms of the block-key variant of the Carousel scheme. For simplicity, we follow the same notations to Sec. 4.2.

Theorem 5 (Blind Rotation with Block Key). *Let* $\mathbf{c} = (b, a) \in \mathbb{Z}_Q^{n+1}$ *be an LWE ciphertext. Then, for* $\mathsf{ct} \leftarrow \mathsf{BlindRotate}(\mathsf{brk}, \mathsf{atk}; \mathbf{c}, f)$*, it holds that*

$$\mathrm{Var}_\xi(\varphi_t(\mathsf{ct}) - \Psi^{\hat{b} + \langle \hat{\mathbf{a}}, \hat{\mathbf{s}} \rangle}(\mathcal{F})) \le \frac{1}{12} koNP^2(2 + (\ell+2)oN) + \frac{5}{3} ndo^2 N^2 \mathcal{B}^2 \beta^2$$

where $\mathcal{F}$ is the lookup table encoding of the evaluated function f.

Proof. Let ACC_i be the value of the accumulator before the i-th iteration of the outer for loop in Algorithm 4. Essentially, $\varphi_t(\mathsf{ACC}_0) - \Psi^{\hat{b}}(\mathcal{F}) = 0$. Let tmp_j^i be the value of the temporary variable tmp_j^i in the j-th iteration of the inner loop and i-th iteration of the outer loop. Note that tmp is a result of the external product with gadget decomposition reused, $\varphi_t(\mathsf{tmp}) = s_{i\ell+j} \cdot \varphi_t(\mathsf{ACC}_i) + e_{i,j}$ for some small $e_{i,j} \in R$, such that

$$\mathrm{Var}_\xi(e_{i,j}) \le \frac{1}{12} oNP^2 \|s_{i\ell+j}\|_\xi^2 (1+oN) + \frac{2}{3} do^2N^2\mathcal{B}^2\beta^2$$

by Lem. 3. Also,

$$\begin{aligned}\varphi_t(\mathsf{Aut}(\mathsf{atk}_{\hat{a}_{i\ell+j}}; \mathsf{tmp})) &= \Psi^{\hat{a}_{i\ell+j}}(\varphi_t(\mathsf{tmp})) + e'_{i,j} \\ &= s_{i\ell+j} \cdot \Psi^{\hat{a}_{i\ell+j}}(\varphi_t(\mathsf{ACC})) + \Psi^{\hat{a}_{i\ell+j}}(e_{i,j}) + e'_{i,j}\end{aligned}$$

for some small $e'_{i,j} \in R$, such that $\mathrm{Var}_\xi(e'_{i,j}) \le \frac{1}{12} o^2N^2P^2 + \frac{1}{3} do^2N^2\mathcal{B}^2\beta^2$ by Corollary 1. Now, by the definition, we have

$$\begin{aligned}\varphi_t(\mathsf{ACC}_{i+1}) &= \varphi_t(\mathsf{ACC}_i) + \sum_{j=0}^{\ell-1} \left(\varphi_t(\mathsf{Aut}(\mathsf{atk}_{\hat{a}_{i\ell+j}}; \mathsf{tmp})) - \varphi_t(\mathsf{tmp})\right) \\ &= \varphi_t(\mathsf{ACC}_i) + \sum_{j=0}^{\ell-1} \left(s_{i\ell+j} \cdot \Psi^{\hat{a}_{i\ell+j}}(\varphi_t(\mathsf{ACC})) + \Psi^{\hat{a}_{i\ell+j}}(e_{i,j}) + e'_{i,j} - s_{i\ell+j} \cdot \varphi_t(\mathsf{ACC}_i) - e_{i,j}\right) \\ &= \Psi^{\sum_{j=0}^{\ell-1} \hat{a}_{i\ell+j} s_{i\ell+j}}(\varphi_t(\mathsf{ACC}_i)) + \sum_{j=0}^{\ell-1} \left(\Psi^{\hat{a}_{i\ell+j}}(e_{i,j}) - e_{i,j} + e'_{i,j}\right).\end{aligned}$$

Note that the final equation directly follows from Eq. (3). Recall that there is at most one 1 and everything else is 0 among $s_{\ell \cdot i}, \ldots, s_{\ell(i+1)-1}$. Therefore, we have the bound on the noise growth for a single iteration of the outer loop as follows.

$$\mathrm{Var}_\xi\left(\sum_{j=0}^{\ell-1} (\Psi^{\hat{a}_{i\ell+j}}(e_{i,j}) - e_{i,j} + e'_{i,j})\right) \le \frac{1}{12} oNP^2(2 + (\ell+2)oN) + \frac{5}{3} \ell do^2N^2\mathcal{B}^2\beta^2$$

Since the noise only additively grows, we obtain the desired noise variance bound after k iterations. □

Theorem 6 (Key Switching with Block Key). *Let $\mathbf{c} \in \mathbb{Z}_Q^{N+1}$ be an LWE ciphertext. Then, for $\mathbf{c}' \leftarrow \mathsf{KeySwitch}(\mathsf{ksk}; \mathbf{c})$,*

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

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

Proof. Note that the gadget product is only performed $N-n$ times instead of N in the original scheme description, the noise growth is $\frac{N}{N-n}$ times smaller than the original key-switching algorithm. Therefore, we can obtain the claimed noise bound directly from Thm. 3. □

References

AH17. Seiko Arita and Sari Handa. Subring homomorphic encryption. In *International Conference on Information Security and Cryptology*, pages 112–136. Springer, 2017.

AH20. Seiko Arita and Sari Handa. Fully homomorphic encryption scheme based on decomposition ring. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 103(1):195–211, 2020.

AKP24. Andreea Alexandru, Andrey Kim, and Yuriy Polyakov. General functional bootstrapping using ckks. *Cryptology ePrint Archive*, 2024.

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

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

BCK+23. Youngjin Bae, Jung Hee Cheon, Jaehyung Kim, Jai Hyun Park, and Damien Stehlé. Hermes: efficient ring packing using mlwe ciphertexts and application to transciphering. In *Annual International Cryptology Conference*, pages 37–69. Springer, 2023.

Ber68. Glenn D. Bergland. Numerical analysis: A fast fourier transform algorithm for real-valued series. *Commun. ACM*, 11(10):703–710, October 1968.

Ber23. Daniel J Bernstein. Fast norm computation in smooth-degree abelian number fields. *Research in Number Theory*, 9(4):82, 2023.

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

BKSS25. Youngjin Bae, Jaehyung Kim, Damien Stehlé, and Elias Suvanto. Bootstrapping small integers with ckks. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 330–360. Springer, 2025.

Bra12. Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual cryptology conference*, pages 868–886. Springer, 2012.

CBSZ23. Pierre-Emmanuel Clet, Aymen Boudguiga, Renaud Sirdey, and Martin Zuber. Combo: A novel functional bootstrapping method for efficient evaluation of nonlinear functions in the encrypted domain. In *International Conference on Cryptology in Africa*, pages 317–343. Springer, 2023.

CDKS21. Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient homomorphic conversion between (ring) lwe ciphertexts. In *International conference on applied cryptography and network security*, pages 460–479. Springer, 2021.

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

CHM+25. Hyunho Cha, Intak Hwang, Seonhong Min, Jinyeong Seo, and Yongsoo Song. Matrigear: Accelerating authenticated matrix triple generation with scalable prime fields via optimized he packing. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 2249–2267. IEEE Computer Society, 2025.

CKKS17. Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 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*, pages 409–437. Springer, 2017.

CLOT21a. Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. 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*, pages 670–699. Springer, 2021.

CLOT21b. Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. Improved programmable bootstrapping with larger precision and efficient arithmetic circuits for tfhe. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 670–699. Springer, 2021.

DM15. Léo Ducas and Daniele Micciancio. Fhew: bootstrapping homomorphic encryption in less than a second. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 617–640. Springer, 2015.

FV12. Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Paper 2012/144, 2012. `https://eprint.iacr.org/2012/144`.

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

GINX16. Nicolas Gama, Malika Izabachène, Phong Q Nguyen, and Xiang Xie. Structural lattice reduction: generalized worst-case to average-case reductions and homomorphic cryptosystems. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*, pages 528–558. Springer, 2016.

GSW13. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 75–92. Springer, 2013.

HMS25. Intak Hwang, Seonhong Min, and Yongsoo Song. Practical circuit privacy/sanitization for tfhe. *Cryptology ePrint Archive*, 2025.

HS14. Shai Halevi and Victor Shoup. Algorithms in helib. In *Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I 34*, pages 554–571. Springer, 2014.

JP22. Marc Joye and Pascal Paillier. Blind rotation in fully homomorphic encryption with extended keys. In *International Symposium on Cyber Security, Cryptology, and Machine Learning*, pages 1–18. Springer, 2022.

JW22. Marc Joye and Michael Walter. Liberating tfhe: programmable bootstrapping with general quotient polynomials. In *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 1–11, 2022.

KS19. Duhyeong Kim and Yongsoo Song. 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*, pages 85–102. Springer, 2019.

KS23. Kamil Kluczniak and Leonard Schild. Fdfb: full domain functional bootstrapping towards practical fully homomorphic encryption. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023.

LMK+23. Yongwoo Lee, Daniele Micciancio, Andrey Kim, Rakyong Choi, Maxim Deryabin, Jieun Eom, and Donghoon Yoo. Efficient fhew bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 227–256. Springer, 2023.

LMS24. Dongwon Lee, Seonhong Min, and Yongsoo Song. Functional bootstrapping for packed ciphertexts via homomorphic lut evaluation. *Cryptology ePrint Archive*, 2024.

LMSS23. Changmin Lee, Seonhong Min, Jinyeong Seo, and Yongsoo Song. Faster tfhe bootstrapping with block binary keys. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, pages 2–13, 2023.

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

LW23a. Feng-Hao Liu and Han Wang. Batch bootstrapping i: a new framework for simd bootstrapping in polynomial modulus. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 321–352. Springer, 2023.

LW23b. Feng-Hao Liu and Han Wang. 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*, pages 353–384. Springer, 2023.

LW25. Zeyu Liu and Yunhao Wang. Relaxed functional bootstrapping: A new perspective on bgv/bfv bootstrapping. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 208–240. Springer, 2025.

May21. Alexander May. How to meet ternary lwe keys. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part II 41*, pages 701–731. Springer, 2021.

MHW+24. Shihe Ma, Tairong Huang, Anyu Wang, Qixian Zhou, and Xiaoyun Wang. Fast and accurate: efficient full-domain functional bootstrap and digit decomposition for homomorphic computation. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024.

OPP23. Hiroki Okada, Rachel Player, and Simon Pohmann. Homomorphic polynomial evaluation using galois structure and applications to bfv bootstrapping. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 69–100. Springer, 2023.

Rad68. Charles M Rader. Discrete fourier transforms when the number of data samples is prime. *Proceedings of the IEEE*, 56(6):1107–1108, 1968.

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

SSTX09. Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 617–635. Springer, 2009.

WWL+24. Ruida Wang, Yundi Wen, Zhihao Li, Xianhui Lu, Benqiang Wei, Kun Liu, and Kunpeng Wang. Circuit bootstrapping: faster and smaller. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 342–372. Springer, 2024.

YXS+21. Zhaomin Yang, Xiang Xie, Huajie Shen, Shiying Chen, and Jun Zhou. Tota: fully homomorphic encryption with smaller parameters and stronger security. *Cryptology ePrint Archive*, 2021.

Zam22. Zama. TFHE-rs: A Pure Rust Implementation of the TFHE Scheme for Boolean and Integer Arithmetics Over Encrypted Data, 2022. `https://github.com/zama-ai/tfhe-rs`.