Fast Blind Rotation for Bootstrapping FHEs

Binwu Xiang*[1,2,3], Jiang Zhang[2(✉)], Yi Deng[1,3], Yiran Dai[1,2,3], and Dengguo Feng[2]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China.
{xiangbinwu,deng,daiyiran}@iie.ac.cn
[2] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China.
zhangj@sklc.org,fengdg@263.net
[3] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China.

Abstract. Blind rotation is one of the key techniques to construct fully homomorphic encryptions with the best known bootstrapping algorithms running in less than one second. Currently, the two main approaches, namely, AP and GINX, for realizing blind rotation are first introduced by Alperin-Sheriff and Peikert (CRYPTO 2014) and Gama, Izabachene, Nguyen and Xie (EUROCRYPT 2016), respectively.

In this paper, we propose a new blind rotation algorithm based on a GSW-like encryption from the NTRU assumption. Our algorithm has performance asymptotically independent from the key distributions, and outperforms AP and GINX in both the evaluation key size and the computational efficiency (especially for large key distributions). By using our blind rotation algorithm as a building block, we present new bootstrapping algorithms for both LWE and RLWE ciphertexts.

We implement our bootstrapping algorithm for LWE ciphertexts, and compare the actual performance with two bootstrapping algorithms, namely, FHEW/AP by Ducas and Micciancio (EUROCRYPT 2015) and TFHE/GINX by Chillotti, Gama, Georgieva and Izabachène (Journal of Cryptology 2020), that were implemented in the OpenFHE library. For parameters with ternary key distribution at 128-bit security, our bootstrapping only needs to store evaluation key of size 18.65MB for blind rotation, which is about 89.8 times smaller than FHEW/AP and 2.9 times smaller than TFHE/GINX. Moreover, our bootstrapping can be done in 112ms on a laptop, which is about 3.2 times faster than FHEW/AP and 2.1 times faster than TFHE/GINX. More improvements are available for large key distributions such as Gaussian distributions.

Keywords: Lattices, Fully Homomorphic Encryption, Bootstrapping, Blind Rotations

* This work was done while I was a visiting student in the group of Dr. Jiang Zhang during 2021-2023 at State Key Laboratory of Cryptology, Beijing, China.

1 Introduction

Fully homomorphic encryption (FHE) allows to perform computations on encrypted data without decryption and is one of the main cryptographic tools for privacy computation. Currently, almost all known FHE constructions follow the same paradigm introduced by Gentry [28]: 1) construct a somewhat homomorphic encryption (SHE) using noise-based encryptions (e.g., Regev encryption [42]) which typically can only support a limited number of homomorphic operations due to the increase of noise size; 2) design a bootstrapping algorithm that essentially homomorphically computes the decryption of SHE to reduce the noise size of ciphertexts. The initial attempts [21,11,20,43,45,9,10,25] mainly focus on the construction of SHEs supporting the evaluation of their own (augmented) decryption circuits, and then use the bootstrapping theorem [28] to convert SHEs into FHEs. As bootstrapping is the central component for almost all existing FHEs and is much more complex than other elementary operations, it has become the main efficiency bottleneck in practical implementations.

Essentially, the decryption algorithm for most existing FHEs is to compute a function $g(\langle \mathbf{c}, \mathbf{s} \rangle \mod q) = g(\sum_i c_i s_i \mod q)$, where $\mathbf{c} = (c_0, \ldots, c_{n-1}) \in \mathbb{Z}_q^n$ is the ciphertext, $\mathbf{s} = (s_0, \ldots, s_{n-1}) \in \mathbb{Z}_q^n$ is the secret key, and g is a simple decoding algorithm that may vary depending on specific schemes. The most costly operation for bootstrapping is to homomorphically compute the modular function $f(\mathbf{c}, \mathbf{s}) = \sum_i c_i s_i \mod q$ for some integer modulus q. There are mainly three approaches to do this in practical implementations. The first approach focuses on BGV ciphertext [10] with decryption function $g(\langle \mathbf{c}, \mathbf{s} \rangle \mod q) = (\sum_i c_i s_i \mod q) \mod p$ for some integer p (e.g., $p = 2$). The central idea of this approach is to convert $f(\mathbf{c}, \mathbf{s}) = \sum_i c_i s_i \mod q$ with a general q to $f'(\mathbf{c}, \mathbf{s}) = \sum_i c_i s_i \mod q'$ with a special modulus $q' = p^r$ so that $g(\langle \mathbf{c}, \mathbf{s} \rangle \mod q)$ can be computed by extracting the bit decompositions of $\langle \mathbf{c}, \mathbf{s} \rangle$ in base p [12,9,10,25,29,32,14]. The second approach targets CKKS ciphertext [16] with decryption function $g(\langle \mathbf{c}, \mathbf{s} \rangle \mod q) = \sum_i c_i s_i \mod q$ for approximate numbers. The idea behind this approach is to use another math function, e.g., $\frac{q}{2\pi} \sin\left(\frac{2\pi}{q} \langle \mathbf{c}, \mathbf{s} \rangle\right)$, to approximate the modular function $\langle \mathbf{c}, \mathbf{s} \rangle \bmod q$. Since this approach will inherently introduce extra noises to the resulting ciphertext, the choice of the approximate math functions significantly influences the computation performance and the quality of the result ciphertext [16,15,13,36,31,8]. To perform the bootstrapping, both approaches above require to homomorphcially compute a polynomial (of the secret key $\mathbf{s}$) with relatively large degree, which not only need hundreds of megabytes to several gigabytes to store the bootstrapping keys, but also require tens/hundreds of seconds to bootstrap a single ciphertext.

The third approach is called blind rotation [5,22], which allows to fast bootstrap a single ciphertext in less than one second. Basically, this approach uses two layers of encryptions: the first layer is a simple Regev-like encryption [42] which has very limited homomorphic capacity, and the second layer is a ring-based GSW-like encryption [30] which is specially designed to homomorphically compute the first-layer decryption algorithm. In more detail, let $R = \mathbb{Z}[X]/(X^N + 1)$

Table 1. Asymptotic comparison of different blind rotations in the number $\#R_Q$ of R_Q elements in the evaluation key and the number #mul of multiplications in R_Q for computation (where B is an integer in bit-decomposing the first-layer ciphertexts for AP, and U is a public set in bit-decomposing the first-layer secret keys for GINX).

Method	$\#R_Q$	#mul
AP [5,22,41]	$4d\lceil \log_B q \rceil (B-1)n$	$4d\lceil \log_B q \rceil n$
GINX [26,19]	$4d\|U\|n$	$4d\|U\|n^{\dagger}$
Lee et al. [37]	$2d(2n+q+1)$	$2d(3n+2)$
Ours	$d(n+q)$	$d(2n+1)$

† Recently, Bonte et al. [7] reduced the computational cost of GINX for the special case of ternary keys (i.e., $|U|=2$) by a half, but still keep the evaluation key size unchanged. See Section 1.3 for more discussions.

with $q = 2N$ be the cyclotomic ring used in the second layer, then the modular function $f(\mathbf{c}, \mathbf{s}) = \sum_i c_i s_i \mod q$ in the first-layer decryption can be easily done in the exponent of the ring element $X \in R$ because the order of X is exactly q in R and $X^{f(\mathbf{c},\mathbf{s})} = X^{\sum_i c_i s_i \mod q} = X^{\sum_i c_i s_i}$. By multiplying a rotation polynomial $r(X) = \sum_i iX^{-i}$, one can easily decode $\sum_i c_i s_i \mod q$ from the exponent of $X^{\sum_i c_i s_i}$ to the constant term of the polynomial $r(X) \cdot X^{\sum_i c_i s_i}$. The blind rotation actually refers to the procedure of computing $r(X) \cdot X^{\sum_i c_i s_i}$ from a given ciphertext vector $\mathbf{c} = (c_0, \ldots, c_{n-1})$ and some encryptions of the secret key $\mathbf{s} = (s_0, \ldots, s_{n-1})$. There are mainly two ways for realizing blind rotation, namely, AP [5,22,41] and GINX [26,19]. The first one [5,22,41] has performance asymptotically independent from the first-layer key distributions, but it has to store very large evaluation keys (e.g., more than 1.6GB at 128-bit security [39]). The second one [26,19,18] has performance linear in the bit-decomposition size of the first-layer secret key, which only needs to store very small evaluation keys for small key distributions (e.g., 54MB at 128-bit security for ternary key distribution [39]), but its evaluation keys will quickly increase to hundreds of megabytes when the secret key $\mathbf{s}$ is chosen from a distribution with large bit-decomposition size (e.g., Gaussian distributions as recommended in [1]). Recently, Lee et al. [37] presented a new blind rotation algorithm which has performance asymptotically independent from the first-layer key distributions as AP and outperforms GINX for large first-layer key distributions.

1.1 Our results

In this work, we follow the two-layer framework to construct FHEs with fast bootstrapping algorithms. First, we present a second-layer GSW-like encryption based on the NTRU assumption, which supports very fast key-switching for ring automorphisms. Then, we propose a new blind rotation algorithm by crucially using ring automorphisms and its associated key-switchings. Our algorithm has performance asymptotically independent from the first-layer key distribution and

Table 2. Experimental comparison of different blind rotations using parameters at 128-bit security (where the column "Key distrib." denotes the key distributions used for the first-layer encryption; the column "EVK" denotes the evaluation key size for blind rotation; the last column "Timing" denotes the timing for running the whole bootstrapping algorithm).

Algorithms	Parameters $(n, q, d, N, \log_2 Q, B, \lvert U\rvert)$	Key distrib.	EVK (MB)	Timing (ms)
FHEW/AP [22,6]	STD128 [39] (512,1024,4,1024,27,32,-)	Ternary	1674	359
TFHE/GINX [19,6]	STD128 [39] (512,1024,4,1024,27,-,2)	Ternary	54	234
Ours	P128T (512,1024,5,1024,19.9,-,-)	Ternary	18.65	112
	P128G (465,1024,5,1024,19.9,-,-)	Gaussian	17.90	100

outperforms AP and GINX in both the evaluation key size and the computation efficiency. By using our blind rotation algorithm as a building block, we construct new bootstrapping algorithms for both LWE-based and RLWE-based first-layer ciphertexts. Finally, we implement our bootstrapping algorithm for LWE-based first-layer ciphertexts, and the experiment shows that it can be efficiently done in 112ms on a laptop for parameters at 128-bit security.[4]

In Table 1, we give a theoretical comparison of different blind rotations, where n, q are the dimension and modulus for the first-layer encryption; N, Q are the dimension and modulus of the ring $R_Q = \mathbb{Z}_Q[X]/(X^N + 1)$ for the second-layer GSW-like encryption; d is the bit-decomposition dimension for multiplying a second-layer GSW-like ciphertext; $|U|$ is the number of bits needed to represent the full range of secret key samples, e.g. $|U| = 2$ for ternary secrets. Since q is usually less than $2n$ for typically choices of parameters [39], one can see from Table 1 that our blind rotation has asymptotically the best performance in both evaluation key size and computational efficiency. We note that Lee et al.'s blind rotation [37] also has similar asymptotically performance as ours, but unlike us, they rely on an RLWE-based second-layer GSW-like encryption. Our NTRU-based GSW-like encryption allows us to obtain improvements over Lee et al. [37] by a factor about 2.7 times in the evaluation key size, and about 3 times in the computational efficiency.

In Table 2, we give an experimental comparison of different blind rotations. All the figures are obtained by running the corresponding algorithms on the

[4] We note that the timing of 13ms for a single GINX-like bootstrapping reported in [19] is obtained using implementation with binary secret keys (i.e., $|U| = 1$) and AVX2 instructions. The timing of their implementation using portable C language on our laptop is actually 227ms for 128-bit security parameter, which is roughly 2 times slower than our algorithm, close to our theoretical estimation in Table 1.

same laptop using parameters providing at least 128-bit security. [5] The parameter STD128 for FHEW/AP [22] and TFHE/GINX [19] is the default parameter provided in the OpenFHE 0.9.1 library [6] and recommended in [39]. For comparison, we choose two sets of parameters P128T and P128G: P128T is set to have the same parameters (n, q) and ternary key distribution with STD128 for the first-layer LWE encryption; while P128G is set to use a Gaussian key distribution with variance 4/3. From Table 2, one can see that we only have to store an evaluation key of size 18.65MB for blind rotation using parameter P128T, which is about 89.8 times smaller than FHEW/AP and 2.9 times smaller than TFHE/GINX. Moreover, our bootstrapping algorithm can be done in 112ms, which is about 3.2 times faster than FHEW/AP and 2.1 times faster than TFHE/GINX. When using P128G with Gaussian key distribution, we can obtain an extra 10% improvements over P128T (while the performance of TFHE/GINX will become worse by roughly a factor of 1.5 when using our Gaussian key distribution).

1.2 Our Techniques

We now explain the techniques behind our results. Without loss of generality, a first-layer ciphertext is assumed to have a form of $(\mathbf{a}, b = \sum_{i=0}^{n-1} a_i s_i - \mathsf{noised}(m))$, where $\mathbf{a} = (a_0, \ldots, a_{n-1}), \mathbf{s} = (s_0, \ldots, s_{n-1}) \in \mathbb{Z}_q^n$ are respectively the randomness and the secret key used to encrypt some plaintext $m \in \mathbb{Z}_q$. Our goal is to homomorphically compute $\mathsf{noised}(m) = \sum_{i=0}^{n-1} a_i s_i - b \mod q$. At a high level, we want to remove the modulo q operation by choosing a ring $R_Q = \mathbb{Z}_Q[X]/(X^N+1)$ with N being a power of 2 and $q = 2N$ such that the order of X in R_Q is exactly q and the modulo q operation can be done for free in the exponent of X:

$$X^{\mathsf{noised}(m)} = X^{\sum_{i=0}^{n-1} a_i s_i - b \mod q} = X^{-b} X^{\sum_{i=0}^{n-1} a_i s_i}.$$

Then, by multiplying a rotation polynomial $r(X) = \sum_{i=0}^{q/2-1} i X^{-i} \in R_Q$ we can extract the value $\mathsf{noised}(m)$ from $X^{\mathsf{noised}(m)}$ to the constant term of the polynomial $r(X) \cdot X^{-b} X^{\sum_{i=0}^{n-1} a_i s_i}$,[6] which is equal to

$$r(X) \cdot X^{\mathsf{noised}(m)} = \mathsf{noised}(m) + \sum_{i \neq \mathsf{noised}(m)} i X^{\mathsf{noised}(m) - i}.$$

The above procedure is called blind rotation [5,22,26,19,41], which can be extended to any (q, N) with $q | 2N$ by multiplying a constant $2N/q$ in the exponent:

$$r(X^{\frac{2N}{q}}) \cdot X^{\frac{2N}{q}\mathsf{noised}(m)} = \mathsf{noised}(m) + \sum_{i \neq \mathsf{noised}(m)} i X^{\frac{2N}{q}(\mathsf{noised}(m) - i)}.$$

[5] Table 2 does not include the bootstrapping algorithms in [37,7] because the code for [37] is not publicly available, and [7] did not use the recommended parameters (See Section 1.3 for more discussions).

[6] Notice that $\mathsf{noised}(m) \in [0, q/2)$ needs to be satisfied. It is simply for the convenience of readers to understand the definition of the rotation polynomial here. For more details in setting up the rotation polynomial, one can refer to [5,22,26,19,41]

Note that $\mathbf{s} = (s_0, \ldots, s_{n-1})$ is the secret key, we can only compute

$$r(X^{\frac{2N}{q}}) \cdot X^{\frac{2N}{q}\mathsf{noised}(m)} = r(X^{\frac{2N}{q}}) \cdot X^{-\frac{2N}{q}b} X^{\frac{2N}{q}\sum_{i=0}^{n-1} a_i s_i}$$

by using ciphertexts that encrypt $\mathbf{s}$. Previously, this is done by either using the AP method [5,22,41] which relies on the decompositions of $a_i = \sum_j a_{i,j} B^j$ in some base B and encrypts all possible values of $a_{i,j} s_i \in \mathbb{Z}_q$ in the evaluation key, or using the the GINX method [26,19] which relies on the bit decompositions of $s_i = \sum_{u \in U} s_{i,u} u$ for some public set U and encrypts all $s_{i,u} \in \{0, 1\}$ in the evaluation key. Unlike AP and GINX, we use the idea of Lee et al. [37] to perform blind rotation by using ring automorphisms in R_Q. Basically, given a ciphertext $c(X) \in R_Q = \mathbb{Z}[X]$ that encrypts X^{s_i} using secret key $f(X)$, we can easily obtain a ciphertext $c(X^{a_i})$ that encrypts $X^{a_i s_i}$ by applying the automorphism $X \to X^{a_i}$ to $c(X)$ if $a_i \in \mathbb{Z}_q$ is coprime to $2N$. One problem is that $c(X^{a_i})$ is not a ciphertext under the original secret key $f(X)$ but a related secret key $f(X^{a_i})$. We need to perform a key-switching to convert $c(X^{a_i})$ back to a ciphertext that encrypts $X^{a_i s_i}$ under the secret key $f(X)$ to allow further homomorphic computations (recall that we have to homomorphically compute $r(X^{\frac{2N}{q}}) \cdot X^{-\frac{2N}{q}b} X^{\frac{2N}{q}(\sum_{i=0}^{n-1} a_i s_i)}$). The other problem is that each $a_i \in \mathbb{Z}_q$ in a given ciphertext $(\mathbf{a}, b)$ can be an even integer (with probability $1/2$) and thus is not coprime to $2N$. This means that $c(X^{a_i})$ may become an invalid ciphertext if we directly replace X with X^{a_i} in $c(X)$ for an even integer a_i. Fortunately, for most parameters used in practice [39] the requirement $q|N$ is satisfied, we can instead compute $X^{\frac{2N}{q} a_i s_i} = X^{(\frac{2N}{q} a_i + 1) s_i - s_i} = X^{w_i s_i} X^{-s_i}$, where $w_i = \frac{2N}{q} a_i + 1 < 2N$ is always an odd integer and thus is coprime to $2N$ for all $a_i \in \mathbb{Z}_q$ (recall that N is a power of 2). Thus, we can compute

$$r(X^{\frac{2N}{q}}) \cdot X^{-\frac{2N}{q}b} X^{\frac{2N}{q}(\sum_{i=0}^{n-1} a_i s_i)} = r(X^{\frac{2N}{q}}) \cdot X^{-\frac{2N}{q}b} X^{\sum_{i=0}^{n-1} w_i s_i} X^{-\sum_{i=0}^{n-1} s_i}$$

by using ring automorphisms almost without introducing extra restrictions on the choice of N. In the following, we first give an NTRU-based GSW-like encryption for the second layer such that the key-switching for ring automorphisms can be done as efficient as a single GSW-like homomorphic multiplication.

NTRU-based GSW-like encryption. Let $f, f' \in R_Q$ be two polynomials with small coefficients that are invertible in R_Q. Let (τ, Δ) be two integer parameters depending on the encoding of m in $\mathsf{noised}(m)$ of the first-layer encryption. We define a scalar NTRU ciphertext that "encrypts" plaintext $u \in R_Q$ under secret key f as follows:

$$\mathrm{NTRU}_{Q,f,\tau,\Delta}(u) = \tau \cdot g/f + \Delta \cdot u/f \in R_Q,$$

where $g \in R_Q$ is a noise polynomial with small coefficients. Noted that both the noise term and the message are divided by f. This essentially requires by a circular-secure/KDM-secure assumption on NTRU, which is also used in [7,35].

Let B be an integer and $d = \lceil \log_B Q \rceil$, we also define a vector NTRU ciphertext that "encrypts" plaintext $v \in R_Q$ under secret key f' as follows:

$$\mathrm{NTRU}'_{Q,f',\tau}(v) := (\tau \cdot g_0/f' + v, \tau \cdot g_1/f' + B \cdot v, \cdots, \tau \cdot g_{d-1}/f' + B^{d-1} \cdot v) \in R_Q^d,$$

where $\{g_i\}_{0 \le i \le d-1}$ are also noise polynomials with small coefficients. Given a polynomial $c \in R_Q$, let $\mathrm{BitDecom}_B(c)$ be the deterministic function that converts $c = \sum_{i=0}^{d-1} c_i \cdot B^i$ into a d-dimensional vector $(c_0, c_1, \ldots, c_{d-1}) \in R_B^d$. Now, we define the external product $\odot$ between a polynomial $c \in R_Q$ and a vector NTRU ciphertext $\mathbf{c}' = \mathrm{NTRU}'_{Q,f',\tau}(v)$ as follows:

$$c \odot \mathbf{c}' = \sum_{i=0}^{d-1} c_i c'_i = \tau \cdot \left(\sum_{i=0}^{d-1} g_i c_i \right) / f' + cv,$$

where $(c_0, \ldots, c_{d-1}) = \mathrm{BitDecom}_B(c) \in R_B^d$ and $\mathbf{c}' = (c'_0, \ldots, c'_{d-1}) \in R_Q^d$.

Note that if $c = \mathrm{NTRU}_{Q,f,\tau,\Delta}(u)$ is a scalar NTRU ciphertext with $f' = f$ and $v \in R_Q$ has small coefficients, we have that $c \odot \mathbf{c}' = \sum_{i=0}^{d-1} c_i c'_i = \tau \cdot g'/f + \Delta \cdot uv/f$ is a valid ciphertext $\mathrm{NTRU}_{Q,f,\tau,\Delta}(uv)$ that encrypts $uv \in R_Q$ with $g' = \sum_{i=0}^{d-1} g_i c_i + gv$. Moreover, in order to convert $c = \mathrm{NTRU}_{Q,f,\tau,\Delta}(u)$ into a ciphertext $\hat{c} = \mathrm{NTRU}_{Q,f',\tau,\Delta}(u)$ that encrypts the same plaintext u under another secret key f', it is sufficient to set $v = f/f'$ and $\hat{c} = c \odot \mathbf{c}' = \sum_{i=0}^{d-1} c_i c'_i = \tau \cdot \hat{g}/f' + \Delta \cdot u/f'$, where $\hat{g} = \sum_{i=0}^{d-1} g_i c_i + g$. This means that both homomorphic multiplication and key-switching can be efficiently done by using a single external product, which in turn only needs d multiplications in R_Q.

Fast blind rotation with small key size. Recall that the goal of blind rotation is to homomorphically compute

$$r(X^{\frac{2N}{q}}) \cdot X^{-\frac{2N}{q}b} X^{\frac{2N}{q}(\sum_{i=0}^{n-1} a_i s_i)} = r(X^{\frac{2N}{q}}) \cdot X^{-\frac{2N}{q}b} X^{\sum_{i=0}^{n-1} w_i s_i} X^{-\sum_{i=0}^{n-1} s_i},$$

where both $r(X^{\frac{2N}{q}}) \cdot X^{-\frac{2N}{q}b} \in R_Q$ and $w_i = \frac{2N}{q} a_i + 1 < 2N$ can be publicly computed from the ciphertext $(\mathbf{a}, b = \sum_{i=0}^{n-1} a_i s_i - \mathsf{noised}(m))$ that needs to be bootstrapped. A key feature required for previous blind rotations [5,22,26,19,41,7] is that $r(X^{\frac{2N}{q}}) \cdot X^{-\frac{2N}{q}b} \in R_Q$ can be naturally treated as a scalar "ciphertext" (with zero noise), so that one can directly multiply it with a vector ciphertext that encrypts $X^{\frac{2N}{q}(\sum_{i=0}^{n-1} a_i s_i)}$ by a single external product to obtain a (scalar) ciphertext that encrypts $r(X^{\frac{2N}{q}}) \cdot X^{-\frac{2N}{q}b} X^{\frac{2N}{q}(\sum_{i=0}^{n-1} a_i s_i)}$. However, this feature is not satisfied for our NTRU-based GSW-like encryption, because a scalar NTRU ciphertext that encrypts $r(X^{\frac{2N}{q}}) \cdot X^{-\frac{2N}{q}b}$ in our above scheme has the form of $\tau \cdot g/f + \Delta \cdot (r(X^{\frac{2N}{q}}) \cdot X^{-\frac{2N}{q}b})/f \in R_Q$, which cannot be publicly created even using zero noise $g = 0$.

Fortunately, we can solve this problem almost for free by carefully designing the evaluation key. Specifically, we create a set of ciphertexts that encrypts

$\mathbf{s} = (s_0, \ldots, s_{n-1})$ as follows:

$$\mathbf{evk}_0 = \mathrm{NTRU}'_{Q,f,\tau}(X^{s_0}/f), \qquad \mathbf{evk}_i = \mathrm{NTRU}'_{Q,f,\tau}(X^{s_i}) \text{ for } 1 \le i < n,$$
$$\mathbf{evk}_n = \mathrm{NTRU}'_{Q,f,\tau}(X^{-\sum_{i=0}^{n-1} s_i}).$$

We also create a set of ciphertexts for the key-switchings associated with all the needed ring automorphisms as follows:

$$\mathbf{ksk}_j = \mathrm{NTRU}'_{Q,f,\tau}\left(f(X^j)/f(X)\right) \text{ for all } j \in S = \left\{ \frac{2N}{q} i + 1 : 1 \le i \le q-1 \right\}.$$

The evaluation key is defined as $\mathbf{EVK}_{\tau,\Delta} = (\mathbf{evk}_0, \ldots, \mathbf{evk}_n, \{\mathbf{ksk}_j\}_{j \in S})$. Since each element in $\mathbf{EVK}_{\tau,\Delta}$ has d elements in R_Q, we only need to store $d(n+q)$ R_Q elements in total for the evaluation key.

Given $\mathbf{EVK}_{\tau,\Delta}$ and a ciphertext $(\mathbf{a}, b = \sum_{i=0}^{n-1} a_i s_i - \mathsf{noised}(m))$, we now describe our new blind rotation algorithm. First, for each a_i in $\mathbf{a} = (a_0, \ldots, a_{n-1})$ and $w_i = \frac{2N}{q} a_i + 1$, we compute $w'_i = w_i^{-1} \mod 2N$. We also set $w'_n = 1$ for ease of presentation. Then, we compute $c_0(X) = \left(\Delta \cdot r(X^{\frac{2N}{q} w'_0}) \cdot X^{-\frac{2N}{q} b w'_0}\right) \odot \mathbf{evk}_0$. Since $\mathbf{evk}_0 = \mathrm{NTRU}'_{Q,f,\tau}(X^{s_0}/f)$, one can easily check that

$$c_0(X) = \mathrm{NTRU}_{Q,f,\tau,\Delta}(r(X^{\frac{2N}{q} w'_0}) \cdot X^{-\frac{2N}{q} b w'_0} X^{s_0}).$$

By applying the automorphism $X \to X^{w_0 w'_1}$ to $c_0(X)$ (note that $w_0 w'_1$ is always coprime to $2N$), we can obtain a ciphertext

$$c'_0(X) = c_0(X^{w_0 w'_1}) = \mathrm{NTRU}_{Q,f(X^{w_0 w'_1}),\tau,\Delta}(r(X^{\frac{2N}{q} w'_1}) \cdot X^{-\frac{2N}{q} b w'_1} X^{w_0 w'_1 s_0})$$

under secret key $f(X^{w_0 w'_1})$. For our goal, we want to switch it back to a ciphertext that encrypts the same plaintext $r(X^{\frac{2N}{q} w'_1}) \cdot X^{-\frac{2N}{q} b w'_1} X^{w_0 w'_1 s_0}$ under secret key $f(X)$, which can be done by computing

$$\hat{c}_0(X) = \begin{cases} c'_0(X), & \text{if } w_0 w'_1 = 1; \\ c'_0(X) \odot \mathbf{ksk}_{w_0 w'_1}, & \text{otherwise.} \end{cases}$$

Note that $2N/q$ is a factor of $2N$ and $w_0, w_1 \in S \cup \{1\}$. This means that we always have that $w_0 w'_1 \in S \cup \{1\}$, and that $\mathbf{ksk}_{w_0 w'_1} = \mathrm{NTRU}'_{Q,f,\tau}\left(f(X^{w_0 w'_1})/f(X)\right)$ is included in the evaluation key $\mathbf{EVK}_{\tau,\Delta}$. By the definition of external product we have that $\hat{c}_0(X) = \mathrm{NTRU}_{Q,f,\tau,\Delta}(r(X^{\frac{2N}{q} w'_1}) \cdot X^{-\frac{2N}{q} b w'_1} X^{w_0 w'_1 s_0})$.

Similarly, we can iteratively absorb X^{s_i} for $1 \le i \le n-1$ as follows. First, we compute $c_i(X) = \hat{c}_{i-1}(X) \odot \mathbf{evk}_i = \hat{c}_{i-1}(X) \odot \mathrm{NTRU}'_{Q,f,\tau}(X^{s_i})$ to obtain $c_i(X) = \mathrm{NTRU}_{Q,f,\tau,\Delta}(r(X^{\frac{2N}{q} w'_i}) \cdot X^{-\frac{2N}{q} b w'_i} X^{(\sum_{j=0}^{i-1} w_j s_j) w'_i + s_i})$. Then, apply the automorphism $X \to X^{w_i w'_{i+1}}$ to $c_i(X)$ and obtain

$$c'_i(X) = \mathrm{NTRU}_{Q,f(X^{w_i w'_{i+1}}),\tau,\Delta}(r(X^{\frac{2N}{q} w'_{i+1}}) \cdot X^{-\frac{2N}{q} b w'_{i+1}} X^{(\sum_{j=0}^{i} w_j s_j) w'_{i+1}}).$$

Next, compute

$$\hat{c}_i(X) = \begin{cases} c'_i(X), & \text{if } w_i w'_{i+1} = 1 \\ c'_i(X) \odot \mathbf{ksk}_{w_i w'_{i+1}}, & \text{otherwise,} \end{cases}$$

to obtain $\hat{c}_i(X) = \mathrm{NTRU}_{Q,f,\tau,\Delta}(r(X^{\frac{2N}{q} w'_{i+1}}) \cdot X^{-\frac{2N}{q} b w'_{i+1}} X^{(\sum_{j=0}^{i} w_j s_j) w'_{i+1}})$.

After the above iterative procedure, we compute and output $\hat{c}_{n-1}(X) \odot \mathbf{evk}_n$. Since $\mathbf{evk}_n = \mathrm{NTRU}'_{Q,f,\tau}(X^{-\sum_{i=0}^{n-1} s_i})$ and $w'_n = 1$, we finally have that

$$\hat{c}_{n-1}(X) \odot \mathbf{evk}_n = \mathrm{NTRU}_{Q,f,\tau,\Delta}\left(r(X^{\frac{2N}{q}}) \cdot X^{-\frac{2N}{q} b} X^{\sum_{j=0}^{n-1} w_j s_j} X^{-\sum_{j=0}^{n-1} s_j}\right),$$

which is the desired result for blind rotation. In all, our blind rotation only needs at most n automorphisms and $2n+1$ external products (i.e., $n+1$ for homomorphic multiplications and n for key switchings), which in turn only requires at most $d(2n+1)$ multiplications in R_Q (note that the automorphisms in R_Q can be done cheaply by permuting the coefficients of R_Q elements with signs).

Bootstrapping LWE-based first-layer ciphertexts. For ease of presentation, we only consider the case that the plaintext $m \in \mathbb{Z}_t$ is encoded in the higher bits of $\mathsf{noised}(m) = e + m \cdot \lfloor \frac{q}{t} \rceil$ for some noise e (but we note that our approach also applies to other kinds of encodings). Specifically, the first-layer ciphertext (in the secret-key setting) has a form of $(\mathbf{a}, b = \sum_{i=0}^{n-1} a_i s_i - (e + m \cdot \lfloor \frac{q}{t} \rceil)) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where $\mathbf{a} = (a_0, \ldots, a_{n-1})$ and $\mathbf{s} = (s_0, \ldots, s_{n-1})$. Given an evaluation key $\mathbf{EVK}_{\tau,\Delta}$ with $(\tau, \Delta) = (1, \lfloor \frac{Q}{t} \rceil)$ as defined before, we can use the above blind rotation algorithm to obtain a ciphertext $c = \mathrm{NTRU}_{Q,f,\tau,\Delta}(u) = g/f + \Delta \cdot u/f$, where $u = r(X^{\frac{2N}{q}}) \cdot X^{-\frac{2N}{q} b} X^{\sum_{j=0}^{n-1} w_j s_j} X^{-\sum_{i=0}^{n-1} s_i} = \mathsf{noised}(m) + \sum_{i \neq \mathsf{noised}(m)} i X^{\frac{2N}{q}(\mathsf{noised}(m) - i)}$. We can actually further remove the noise e in $\mathsf{noised}(m)$ almost for free by setting $r(X) = \sum_{i=0}^{q/2-1} (\lfloor i / \lfloor q/t \rceil \rceil \mod t) X^{-i}$. Since $m = \lfloor \mathsf{noised}(m) / \lfloor q/t \rceil \rceil \mod t$ by the correctness of the first-layer encryption, we have

$$u = m + \sum_{i \neq \mathsf{noised}(m)} (\lfloor i / \lfloor q/t \rceil \rceil \mod t) X^{\frac{2N}{q}(\mathsf{noised}(m) - i)}.$$

Now, it suffices to show how to convert $c = \mathrm{NTRU}_{Q,f,\tau,\Delta}(u) = g/f + \Delta \cdot u/f$ into an LWE-based ciphertext that encrypts m under the original secret key $\mathbf{s}$. Let $\mathbf{c} = (c_0, \ldots, c_{N-1})$, $\mathbf{f} = (f_0, \ldots, f_{N-1}), \mathbf{g} = (g_0, \ldots, g_{N-1}) \in \mathbb{Z}_Q^N$ be the corresponding coefficient vectors of the polynomials $c, f, g \in R_Q$. We have that the constant term of the polynomial $fc \in R_Q$ is equal to $g_0 + \Delta \cdot m \in \mathbb{Z}_Q$, which in turn is equal to $f_0 c_0 - \sum_{i=1}^{N-1} c_i f_{N-i} \mod Q$. By setting $\hat{\mathbf{c}} = (c_0, -c_{N-1}, \ldots, -c_1) \in \mathbb{Z}_Q^N$, we have that $0 = \langle \hat{\mathbf{c}}, \mathbf{f} \rangle - (g_0 + \Delta \cdot m) \in \mathbb{Z}_Q$. This means that $(\hat{\mathbf{c}}, 0) \in \mathbb{Z}_Q^N \times \mathbb{Z}_Q$ can actually be seen as an LWE-based ciphertext that encrypts m under the secret key $\mathbf{f} \in \mathbb{Z}_Q^N$ and noise g_0. Thus, we can finally obtain a ciphertext in $\mathbb{Z}_q$ that encrypts m under the original secret key $\mathbf{s}$ by first making a key-switching from $\mathbf{f}$ to $\mathbf{s}$ and then a modulus switching from Q to q.

In summary, to bootstrap the ciphertext $(\mathbf{a}, b = \sum_{i=0}^{n-1} a_i s_i - (e + m \cdot \lfloor \frac{q}{t} \rceil)) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, we only need to set the parameters such that $g_0/Q \ll e/q$ and create a key-switching key from $\mathbf{f}$ to $\mathbf{s}$, which is essentially a set of ciphertexts that encrypts $\mathbf{f} \in \mathbb{Z}^N$ under the secret key $\mathbf{s} \in \mathbb{Z}^n$.

Bootstrapping RLWE-based first-layer ciphertexts. Let $\hat{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$ be the ring for the first-layer encryption. Given an RLWE ciphertext $(a, b = a \cdot s - \mathsf{noised}(m)) \in \hat{R}_q \times \hat{R}_q$, where $s \in \hat{R}_q$ is the secret key. Let $\mathbf{a} = (a_0, \ldots, a_{n-1}), \mathbf{s} = (s_0, \ldots, s_{n-1}), \mathbf{m} = (m_0, \ldots, m_{n-1}) \in \mathbb{Z}_q^n$ be the corresponding coefficient vectors of the polynomials $a, s, m \in \hat{R}_q$. By the fact that

$$a \cdot s = \sum_{i=0}^{n-1} \left(\sum_{j=0}^{i} s_j \cdot a_{i-j} - \sum_{j=i+1}^{n-1} s_j \cdot a_{i-j+n} \right) X^i \in \hat{R}_q,$$

we can actually get n LWE ciphertexts $(\mathbf{a}_i, b_i)$ that encrypts m_i under the secret key $\mathbf{s}$ for all $0 \le i \le n-1$, where $\mathbf{a}_i = (a_i, \ldots, a_0, -a_{n-1}, \ldots, -a_{i+1})$.

Then, for each LWE ciphertext $(\mathbf{a}_i, b_i)$, we run our blind rotation algorithm to get an NTRU ciphertext $c_i = \mathrm{NTRU}_{Q,f,\tau,\Delta}(u_i) \in R_Q$ such that the constant coefficient of $u_i \in R_Q$ is exactly m_i. Let $\mathbf{c}_i = (c_{i,0}, c_{i,1}, \ldots, c_{i,N-1}) \in \mathbb{Z}_Q^N$ be the coefficient vector of $c_i \in R_Q$. Let $\hat{\mathbf{c}}_i = (c_{i,0}, -c_{i,N-1}, \ldots, -c_{i,1}) \in \mathbb{Z}_Q^N$, then $(\hat{\mathbf{c}}_i, 0) \in \mathbb{Z}_Q^N \times \mathbb{Z}_Q$ is actually an LWE-based ciphertext that encrypts m_i under the secret key $\mathbf{f} = (f_0, f_1, \ldots, f_{N-1}) \in \mathbb{Z}_Q^N$. Finally, we can directly use the technique in [40] to pack the resulting LWE-based ciphertexts into a single RLWE-based ciphertext.

1.3 Other related work and discussions

Bonte et al. [7] and Kluczniak [35] adapted the GINX-like blind rotation to the NTRU setting with better efficiency, also based on new second-layer GSW-like encryptions from the NTRU assumption, which are different from ours. In particular, our GSW ciphertext is designed to have the form of $(g + m)/f$ to support fast key-switching for ring automorphisms, while the ciphertext in [7] is $g/f + m$, and the ciphertext in [35] is $e_1 \cdot g/f + e_2 + m$. It seems that the NTRU-based GSW encryptions in both [7,35] do not support fast key-switching for ring automorphisms, which is very critical for our improvement.

In terms of concrete efficiency, for binary key distribution the blind rotation algorithms in [7,35] perform better than ours in terms of evaluation key size and computational efficiency. For ternary key distribution, Bonte et al. [7] presented a special CMux gate which results in a reduction of the computational efficiency of GINX-like blind rotation by half, while maintaining the evaluation key size unchanged. This makes the computational performance of their blind rotation asymptotically two times faster than ours in the setting of ternary key distribution. For other key distributions such as Gaussian distribution, they [7,35] have to use the general GINX approach to obtain blind rotations that have performance linear in the bit-decomposition size of the secret keys, and thus is asymptotically

worse than ours (offering asymptotic overhead with factor $|U|$ in runtime and evaluation key size). For example, for uniform key distribution over $[-2, 2]$ (that is slightly larger than ternary distribution), the computational efficiency of our blind rotation is already asymptotically 1.5 times faster than that in [7,35]. We note that many existing FHE libraries, e.g., Helib, use small binary or ternary secrets for performance consideration. However, the literature has reported several attacks on existing libraries using small secret keys [2,3,44,24]. In fact, there are three types key distributions recommended in [1], namely, uniform, Gaussian, and ternary distribution, which do not include the binary secrets for security concerns [1]. Thus, it is of great theoretical and practical interest to improve the bootstrapping performance for large secret keys.

More recently, Lee et al. [37] proposed the first blind rotation algorithm that uses ring automorphisms. A key difference between [37] and ours is that their construction is based on an existing RLWE-based GSW encryption, while we have to carefully design a new GSW-like encryption from the NTRU assumption. As discussed in Section 1.2 and above, the adaption from the RLWE setting to the NTRU setting is non-trivial. In particular, unlike other known GSW encryptions including the one in [37], our NTRU-based GSW ciphertexts designed for fast key-switching cannot be publicly generated even for known plaintexts, which requires a careful design to the generation of evaluation keys.

1.4 Organization

After giving some background in Sec. 2, we give an NTRU-based GSW-like encryption in Sec. 3. In Sec. 4, we present our new blind rotation. We show how to bootstrap LWE-based and RLWE-based first-layer ciphertexts in Sec. 5 and Sec. 6, respectively. Finally, we report our implementation for bootstrapping LWE-based first layer ciphertexts in Sec. 7.

2 Preliminaries

2.1 Notation

The set of real numbers (integers) is denoted by $\mathbb{R}$ ($\mathbb{Z}$, resp.). We use $\leftarrow$ to denote randomly choosing an element from some distribution (or the uniform distribution over some finite set). Vectors are denoted by bold lower-case letters (e.g., $\mathbf{a}$). By $\|\cdot\|_2$ and $\|\cdot\|_\infty$, we denote the ℓ_2 and ℓ_∞ norms. By $\lceil\cdot\rceil$, $\lfloor\cdot\rfloor$ and $\lfloor\cdot\rceil$ we denote the ceiling, floor and round functions, respectively.

Let n, N, q, Q be positive integers with n, N being powers of 2. By R (resp., $\hat{R}$) and R_Q we denote the $2N$-th (resp., $2n$-th) cyclotomic ring $R = \mathbb{Z}[X]/\left(X^N + 1\right)$ (resp., $\hat{R}$) and its quotient ring $R_Q = R/QR$ (resp., $\hat{R}/q\hat{R}$). The norm of a ring element $a(X) \in R$ (or $\hat{R}$) is defined as the norm of its coefficient vector. There are N (resp., n) automorphisms ψ_t over R (resp., $\hat{R}$) given by $\psi_t : a(X) \to a(X^t)$ for all t that is coprime to $2N$ (resp., $2n$). Since $\psi_t(a(X))$ is essentially equivalent to a permutation with signs on the coefficient vector of $a(X)$, the norm of a polynomial is invariant under those N (resp., n) automorphisms.

2.2 Hard Problems and Ciphertexts

In this subsection, we first recall some backgrounds on hard problems.

Definition 1 (Decisional LWE Problem [42]). *For positive integers q and n, let χ be a noise distribution over $\mathbb{Z}$, the decisional LWE problem is to distinguish between $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle - e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ and a pair uniformly chosen at random from $\mathbb{Z}_q^n \times \mathbb{Z}_q$, where $\mathbf{a}$ is drawn uniformly at random from $\mathbb{Z}^n$, $\mathbf{s}$ is drawn from some key distribution over $\mathbb{Z}^n$, and e is drawn from χ.*

Definition 2 (Decisional RLWE Problem [38]). *For positive integers N and Q, let $R = \mathbb{Z}[X]/(X^N + 1)$ and $R_Q = \mathbb{Z}_Q[X]/(X^N + 1)$. Let χ be a noise distribution over R. The decisional RLWE problem is to distinguish between $(a, b = a \cdot s - e) \in R_Q \times R_Q$ from a pair uniformly chosen at random from $R_Q \times R_Q$, where a is drawn uniformly at random from R_Q, s is drawn from some key distribution over R_Q, and e is drawn according to χ.*

Definition 3 (Decisional NTRU Problem [34]). *For positive integers N and Q, let $R = \mathbb{Z}[X]/(X^N + 1)$ and $R_Q = \mathbb{Z}_Q[X]/(X^N + 1)$. Let χ be a noise distribution over R. Let $f, g \leftarrow \chi$ and f has an inverse in R_Q, the decisional NTRU problem is to distinguish between $g/f \in R_Q$ and a uniform polynomial drawn uniformly at random from R_Q.*

We also need a variant of the NTRU assumption, which we call Decisional Vectorized-NTRU Problem.

Definition 4 (Decisional Vectorized-NTRU Problem). *For positive integers N and Q, let $R = \mathbb{Z}[X]/(X^N + 1)$ and $R_Q = \mathbb{Z}_Q[X]/(X^N + 1)$. Let χ be a noise distribution over R. For a positive integer d, let $f, g_0, \cdots, g_{d-1} \leftarrow \chi$ and f has an inverse in R_Q, the decisional Vectorized-NTRU problem is to distinguish between $(g_0/f, \cdots, g_{d-1}/f) \in R_Q^d$ and a d-dimensional uniform polynomial vector drawn uniformly at random from R_Q^d.*

As commented in [27, Section 1.2], the above problem is merely a syntactic extension of the standard NTRU problem and facilitates the encryption of longer messages, without an essential change in the security assumption.

Definition 5 (LWE Ciphertexts). *For positive integers q and n, an LWE-based encryption of $m \in \mathbb{Z}$ under secret key $\mathbf{s} \in \mathbb{Z}^n$ is defined as*

$$\mathrm{LWE}_{q,\mathbf{s}}(m) := (\mathbf{a}, b) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle - \mathsf{noised}(m)) \in \mathbb{Z}_q^n \times \mathbb{Z}_q,$$

where $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ is the randomness, and $\mathsf{noised}(m) \in \mathbb{Z}_q$ is an noised encoding of $m \in \mathbb{Z}$ using some noise $e \in \mathbb{Z}$ chosen from some distribution χ over $\mathbb{Z}$.

Definition 6 (RLWE ciphertexts). *For positive integers Q and N, an RLWE-based encryption of $m \in R$ under secret key $s \in R$ is defined as*

$$\mathrm{RLWE}_{Q,s}(m) := (a, a \cdot s - \mathsf{noised}(m)) \in R_Q \times R_Q,$$

where $a \leftarrow R_Q$ is the randomness, and $\mathsf{noised}(m) \in R_Q$ is an noised encoding of $m \in R$ using some noise $e \in R$ chosen from some distribution χ over R.

3 NTRU-based GSW-like Encryption

In this section, we give an NTRU-based GSW-like encryption that supports fast key switching for ring automorphisms on (scalar) ciphertexts.

Definition 7 (scalar NTRU ciphertexts). *Let τ, Δ be two integer parameters that will be determined later. Our scalar NTRU encryption of $u \in R_Q$ under a secret key $f \in R_Q$ (that is invertible in R_Q) is defined as*

$$\mathrm{NTRU}_{Q,f,\tau,\Delta}(u) := \tau \cdot g/f + \Delta \cdot u/f \in R_Q,$$

where both $f, g \in R_Q$ are polynomials with small coefficients, which are usually taken from a ternary distribution in practice.

Looking ahead, we will set the two integer parameters (τ, Δ) depending on the encodings (namely, Regev-like [42], BGV-like [10], and CKKS-like [16]) of $m \in \mathbb{Z}_t$ in $\mathsf{noised}(m)$ of the first-layer encryption over $\mathbb{Z}_q$ as follows:

$$(\tau, \Delta) = \begin{cases} \left(1, \left\lfloor \frac{Q}{t} \right\rceil\right), & \text{if } \mathsf{noised}(m) = e + \left\lfloor \frac{q}{t} \right\rceil \cdot m; \\ (t, 1), & \text{if } \mathsf{noised}(m) = t \cdot e + m; \\ (1, 1), & \text{if } \mathsf{noised}(m) = e + m. \end{cases}$$

Definition 8 (vector NTRU ciphertexts). *Let B be an integer parameter, our vector NTRU encryption of $v \in R_Q$ under a secret key $f \in R$ (that is invertible in R_Q) is defined as*

$$\mathrm{NTRU}'_{Q,f,\tau}(v) := (\tau \cdot g_0/f + B^0 \cdot v, \cdots, \tau \cdot g_{d-1}/f + B^{d-1} \cdot v) \in R_Q^d,$$

where $f, g_0, \cdots, g_{d-1} \in R_Q$ have small coefficients, and $d = \lceil \log_B Q \rceil$.

For any $a \in R_Q$, by $\mathrm{BitDecom}_B(a) \in R_B^d$ we denote the decomposition vector $(a_0, \ldots, a_{d-1}) \in \mathbb{Z}_B^d$ of a in base B such that $a = \sum_{i=0}^{d-1} a_i \cdot B^i$. Now, we define the external product "$\odot$" between a polynomial $c \in R_Q$ and a vector NTRU ciphertext $\mathbf{c}' = \mathrm{NTRU}'_{Q,f,\tau}(v) \in R_Q^d$ as the inner product between $\mathrm{BitDecom}_B(c) = (c_0, \ldots, c_{d-1})$ and $\mathbf{c}' = (c'_0, \cdots, c'_{d-1})$:

$$c \odot \mathbf{c}' = \langle \mathrm{BitDecom}_B(c), \mathbf{c}' \rangle = \sum_{i=0}^{d-1} c_i c'_i = \tau \cdot \sum_{i=0}^{d-1} c_i g_i/f + cv \in R_Q$$

Note that if $c \in \mathrm{NTRU}_{Q,f,\tau,\Delta}(u)$ is a scalar NTRU ciphertext and v has small coefficients, then we have that the result of $c \odot \mathbf{c}'$ is a scalar NTRU ciphertext that encrypts uv, namely,

$$c \odot \mathbf{c}' = \mathrm{NTRU}_{Q,f,\tau,\Delta}(u) \odot \mathrm{NTRU}'_{Q,f,\tau}(v) = \mathrm{NTRU}_{Q,f,\tau,\Delta}(uv).$$

Formally, we have the following lemma.

Lemma 1 (Homomorphic Multiplication). *Let $c = \mathrm{NTRU}_{Q,f,\tau,\Delta}(u) \in R_Q$ be a scalar NTRU ciphertext using some noise distribution with variance $Var(g)$. Let $\mathbf{c}' = \mathrm{NTRU}'_{Q,f,\tau}(v)$ be a vector NTRU ciphertext using some noise distribution with variance $Var(g')$. Then, we have that $\hat{c} = c \odot \mathbf{c}' = \tau \cdot \hat{g}/f + \Delta \cdot uv/f$ is a scalar NTRU ciphertext for uv, and the variance of $\hat{g}$ satisfies*

$$Var(\hat{g}) \leqslant dN\frac{B^2}{12}Var(g') + \|v\|_2^2 \cdot Var(g).$$

In particular, if v is a monomial with binary coefficient, then we have

$$Var(\hat{g}) \leqslant dN\frac{B^2}{12}Var(g') + Var(g).$$

Proof. By definition, we have $c = \tau \cdot g/f + \Delta \cdot u/f = \mathrm{NTRU}_{Q,f,\tau,\Delta}(u)$, and $\mathbf{c}' = (c'_0, \ldots, c'_{d-1}) = \mathrm{NTRU}'_{Q,f,\tau}(v)$ for some $c'_i = \tau \cdot g_i/f + B^i \cdot v$. Let $(c_0, \cdots, c_{d-1}) = \mathrm{BitDecom}_B(c)$ be the decomposition of c in base B, i.e., $c = \sum_{i=0}^{d-1} = c_i \cdot B^i$. Then, we have

$$\begin{aligned}\hat{c} = c \odot \mathbf{c}' &= \langle \mathrm{BitDecom}_B(c), \mathbf{c}' \rangle = \sum_{i=0}^{d-1} c_i(\tau \cdot g_i/f + B^i \cdot v) \\ &= \tau \cdot \sum_{i=0}^{d-1} c_i g_i/f + cv = \tau \cdot (\sum_{i=0}^{d-1} c_i g_i + gv)/f + \Delta \cdot uv/f \\ &= \tau \cdot g'/f + \Delta \cdot uv/f \in R_Q\end{aligned}$$

where $g' = \sum_{i=0}^{d-1} c_i g_i + gv$. Hence $\hat{c}$ is a scalar NTRU ciphertext for uv. Moreover, since $\|c_i\|_\infty < B$ for all $0 \leq i \leq d-1$, by assumption that g has variance $Var(g)$ and that all g_i's have the same variance $Var(g')$, we have that $Var(\hat{g}) \leqslant Var(\sum_{i=0}^{d-1} c_i g_i) + Var(gv) = dN\frac{B^2}{12}Var(g') + \|v\|_2^2 \cdot Var(g)$.[7] This proves the first claim. The second claim directly follows from the fact that $\|v\|_2^2 \leq 1$ for any monomial v with binary coefficient. This completes the proof.

In the rest of this paper, we denote $\sigma_\odot^2 = dN\frac{B^2}{12}Var(g')$ as the variance of the increased noise (with respect to the input scalar NTRU ciphertext) for homomorphic multiplication.

3.1 Key Switching for Scalar NTRU Ciphertexts

In this subsection, we define a pair of algorithms $(\mathsf{KSKGen}, \mathsf{KS})$ for key switching scalar NTRU ciphertexts as follows:

- $\mathsf{KSKGen}(f_1, f_2)$: Given two secret keys $f_1, f_2 \in R_Q$ that are invertible in R_Q as inputs, the key-switching key from f_1 to f_2 is defined as

$$\mathbf{ksk}_{f_1,f_2} = (\tau \cdot g_0/f_2 + B^0 \cdot f_1/f_2, \cdots, \tau \cdot g_{d-1}/f_2 + B^{d-1} \cdot f_1/f_2).$$

[7] This formula uses a simplification assumption that each coefficient of c_i is uniformly distributed over $[0, B-1]$.

Note that $\mathbf{ksk}_{f_1,f_2}$ is essentially a vector NTRU ciphertext $\mathrm{NTRU}'_{Q,f_2,\tau}(f_1/f_2)$ that encrypts f_1/f_2 under the secret key f_2.

- $\mathsf{KS}_{f_1\to f_2}(c, \mathbf{ksk}_{f_1,f_2})$: Given a scalar ciphertext $\mathrm{NTRU}_{Q,f_1,\tau,\Delta}(u)$ that encrypts u under the secret key f_1, and a key-switching key $\mathbf{ksk}_{f_1,f_2}$ as inputs, compute and output

$$\hat{c} = c \odot \mathbf{ksk}_{f_1,f_2}.$$

Clearly, the above key-switching algorithm only needs a single external product, which is as efficient as a single homomorphic multiplication, and can be done by using d multiplications in R_Q. In the following lemma, we show that $\hat{c}$ is indeed a scalar NTRU ciphertexts that encrypts u under the secret key f_2.

Lemma 2 (Key Switching). *Let $c = \mathrm{NTRU}_{Q,f_1,\tau,\Delta}(u) \in R_Q$ be a scalar NTRU ciphertext that encrypts u using secret key f_1 and some noise distribution with variance $Var(g)$. Let $\mathbf{ksk}_{f_1,f_2} = \mathsf{KSKGen}(f_1, f_2) = \mathrm{NTRU}'_{Q,f_2,\tau}(f_1/f_2)$ be the key-switching key from f_1 to f_2 using some noise distribution with variance $Var(g')$. Then, we have that*

$$\hat{c} = \mathsf{KS}_{f_1\to f_2}(c, \mathbf{ksk}_{f_1,f_2}) = c \odot \mathbf{ksk}_{f_1,f_2} = \tau \cdot \hat{g}/f_2 + \Delta \cdot u/f_2$$

is a scalar NTRU ciphertext that encrypts u under the secret key f_2, and the variance of $\hat{g}$ satisfies

$$Var(\hat{g}) \leqslant dN\frac{B^2}{12}Var(g') + Var(g).$$

Proof. By definition, we have $c = \tau \cdot g/f_1 + \Delta \cdot u/f_1 = \mathrm{NTRU}_{Q,f_1,\tau,\Delta}(u)$, and $\mathbf{ksk}_{f_1,f_2} = (c'_0, \ldots, c'_{d-1}) = \mathrm{NTRU}'_{Q,f_2,\tau}(f_1/f_2)$ for some $c'_i = \tau \cdot g_i/f_2 + B^i \cdot f_1/f_2$. Let $(c_0, \cdots, c_{d-1}) = \mathrm{BitDecom}_B(c)$ be the decomposition of c in base B, i.e., $c = \sum_{i=0}^{d-1} = c_i \cdot B^i$. Then, we have

$$\begin{aligned}
\hat{c} &= \mathrm{NTRU}_{Q,f_1,\tau,\Delta}(u) \odot \mathbf{ksk}_{f_1,f_2} = \sum_{i=0}^{d-1} c_i(\tau \cdot g_i/f_2 + B^i \cdot f_1/f_2) \\
&= \tau \cdot \sum_{i=0}^{d-1} c_i g_i/f_2 + c f_1/f_2 = \tau \cdot (\sum_{i=0}^{d-1} c_i g_i + g)/f_2 + \Delta \cdot u/f_2 \\
&= \tau \cdot g'/f_2 + \Delta \cdot u/f_2 \in R_Q
\end{aligned}$$

where $g' = \sum_{i=0}^{d-1} c_i g_i + g$. Hence $\hat{c}$ is a scalar NTRU ciphertext that encrypts u under secret key f_2. Moreover, since $\|c_i\|_\infty < B$ for all $0 \leq i \leq d-1$, by assumption that g has variance $Var(g)$ and that all g_i's have the same variance $Var(g')$, we have that $Var(g') \leqslant dN\frac{B^2}{12}Var(g') + Var(g)$.[8]

By $\sigma^2_{KS} = dN\frac{B^2}{12}Var(g') = \sigma^2_{\odot}$ we denote the variance of the increased noise (with respect to the input scalar NTRU ciphertext) for key switching.

[8] Again, this formula uses a simplification assumption that each coefficient of c_i is uniformly distributed over $[0, B-1]$.

3.2 Automorphisms on Scalar NTRU Ciphertexts

Let N be a power of 2, and $R = \mathbb{Z}[X]/(X^N + 1)$. Recall that there are N automorphisms $\psi_t : R \to R$ given by $a(X) \to a(X^t)$ for all odd integers in $\mathbb{Z}_{2N}$. In this subsection, we show how to apply automorphism ψ_t on a plaintext $u \in R_Q$ that is encrypted in a scalar NTRU ciphertext $c = \mathrm{NTRU}_{Q,f,\tau,\Delta}(u)$ to get a new ciphertext $\hat{c} = \mathrm{NTRU}_{Q,f,\tau,\Delta}(\psi_t(u)) = \mathrm{NTRU}_{Q,f,\tau,\Delta}(u(X^t))$ that encrypts $\psi(u)$ under the same secret key f. Looking ahead, we will achieve this by first applying ψ_t on c, and then making a key switching from $\psi_t(f) = f(X^t)$ to $f(X)$ on the resulting ciphertexts $\psi_t(c) = c(X^t)$. Formally, we define a pair of algorithms $(\mathsf{AutoKGen}, \mathsf{EvalAuto})$ for automorphisms as follows:

- $\mathsf{AutoKGen}(t, f)$: Given an odd integer $t \in \mathbb{Z}_{2N}$ and a secret key $f \in R_Q$, compute $\mathbf{ksk}_t = \mathsf{KSKGen}(\psi_t(f), f) = \mathsf{KSKGen}(f(X^t), f(X))$.
- $\mathsf{EvalAuto}_t(c, \mathbf{ksk}_t)$: Given a scalar NTRU ciphertext $c = \mathrm{NTRU}_{Q,f,\tau,\Delta}(u)$ that encrypts $u \in R_Q$ under the secret key $f \in R_Q$, and an automorphism key $\mathbf{ksk}_t$, compute $\psi_t(c) = c(X^t)$ and return

$$\hat{c} = \mathsf{KS}_{f(X^t)\to f(X)}(c(X^t), \mathbf{ksk}_t) = c(X^t) \odot \mathbf{ksk}_t.$$

Since the automorphism on $\psi_t : X \to X^t$ on the ciphertext c can be achieved by simply permuting the coefficient vector of $c \in R_Q$ with signs, the cost of the above automorphism algorithm is dominated by the single external product for key switching. We now show that $\hat{c}$ is a desired ciphertext that encrypts $\psi_t(u) = u(X^t)$ under the secret key f in the following lemma.

Lemma 3 (Automorphisms on Scalar NTRU Ciphertexts). *Let N be a power of 2, and let t be an odd integer. Let $c = \mathrm{NTRU}_{Q,f,\tau,\Delta}(u) \in R_Q$ be a scalar NTRU ciphertext that encrypts u using secret key f and some noise with variance $Var(g)$. Let $\mathbf{ksk}_t = \mathsf{AutoKGen}(t, f) = \mathsf{KSKGen}(f(X^t), f(X))$ be the key-switching key from $f(X^t)$ to $f(X)$ using some noise distribution with variance $Var(g')$. Then, we have that*

$$\hat{c} = \mathsf{EvalAuto}_t(c, \mathbf{ksk}_t) = \mathsf{KS}_{f(X^t)\to f(X)}(c(X^t), \mathbf{ksk}_t) = \tau \cdot \hat{g}/f + \Delta \cdot u(X^t)/f$$

is a scalar NTRU ciphertext that encrypts $u(X^t)$ under the secret key f, and the variance of $\hat{g}$ satisfies

$$Var(\hat{g}) \leqslant dN\frac{B^2}{12}Var(g') + Var(g).$$

Proof. By the definition of $\mathsf{AutoKGen}(t, f)$, we have that $\mathbf{ksk}_t$ is essentially a key-switching key from $f(X^t)$ to $f(X)$. Thus, by Lemma 2, it suffices to show that $\psi_t(c) = c(X^t)$ is a ciphertext that encrypts $u(X^t)$ under the secret key $f(X^t)$ with the same noise variance $Var(g)$ as c. Note that $c = \mathrm{NTRU}_{Q,f,\tau,\Delta}(u) = \tau \cdot g/f + \Delta \cdot u/f$. By the property of the automorphism ψ_t, we have that $\psi_t(c) = c(X^t) = \tau \cdot g(X^t)/f(X^t) + \Delta \cdot u(X^t)/f(X^t)$. Clearly, $c(X^t)$ is a ciphertext that encrypts $u(X^t)$ under secret key $\psi_t(f) = f(X^t)$ and noise $g(X^t)$. Since the coefficient vector of $g(X^t)$ is essentially a permutation of the coefficient vector of $g(X)$ with signs, we have $Var(g(X^t)) = Var(g(X))$. This completes the proof.

4 Fast Blind Rotation in the NTRU setting

Given an LWE-based first-layer ciphertext $\mathrm{LWE}_{q,\mathbf{s}}(m) = (\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ that encrypts some plaintext $m \in \mathbb{Z}_q$ under the secret key $\mathbf{s} \in \mathbb{Z}_q^n$, a rotation polynomial $r(X) \in R_Q = \mathbb{Z}_Q[X]/(X^N + 1)$, the goal of blind rotation is to create a second-layer (scalar) NTRU ciphertext that encrypts $r(Y) \cdot Y^{-b+\langle \mathbf{a}, \mathbf{s} \rangle} \in R_Q$ for some monomial $Y \in R_Q$ of order exactly q.

In this paper, we will set the parameter N such that N is a power of 2 and $q|N$. In this setting, the monomial $Y = X^{\frac{2N}{q}}$ has an order of exactly q. Before giving our construction of blind rotation, we first define a set $S = \left\{ \frac{2N}{q} i + 1 : 1 \leq i \leq q-1 \right\} \subset \mathbb{Z}_{2N}$ of size $q-1$ that will be used later. Since $q|N$, it is easy to check that all the numbers in S are odd integers, and thus have inverses in $\mathbb{Z}_{2N}$. Moreover, we have that $S \cup \{1\}$ is actually a multiplicative subgroup of $\mathbb{Z}_{2N}$. In fact, given any two numbers $w, \hat{w} \in S \cup \{1\}$, we have that their inverses $w^{-1}, \hat{w}^{-1} \in \mathbb{Z}_{2N}$ and their multiplication $w\hat{w} \in \mathbb{Z}_{2N}$ are also in the set $S \cup \{1\}$. This is because we always have $w^{-1} = \hat{w}^{-1} = w\hat{w} = 1 \mod \frac{2N}{q}$. Now, we are ready to give our blind rotation construction.

4.1 The Construction

Let τ, Δ be two integers depending on the encoding of m in the first-layer ciphertext. We define two algorithms $(\mathsf{BRKGen}, \mathsf{BREval})$ for blind rotation as follows:

- $\mathsf{BRKGen}(\mathbf{s}, f)$: Given a secret key $\mathbf{s} = (s_0, \ldots, s_{n-1}) \in \mathbb{Z}_q^n$ for the first-layer encryption, and a secret key $f \in R_Q$ for the second-layer NTRU-based GSW-like encryption, the algorithm first computes a set of ciphertexts:

  $$\mathbf{evk}_0 = \mathrm{NTRU}'_{Q,f,\tau}(X^{s_0}/f), \qquad \mathbf{evk}_i = \mathrm{NTRU}'_{Q,f,\tau}(X^{s_i}) \text{ for } 1 \leq i < n,$$
  $$\mathbf{evk}_n = \mathrm{NTRU}'_{Q,f,\tau}(X^{-\sum_{i=0}^{n-1} s_i}).$$

  Then, it computes a set of ciphertexts for automorphisms as follows:

  $$\mathbf{ksk}_j = \mathsf{AutoKGen}(j, f) = \mathrm{NTRU}'_{Q,f,\tau}\left(f(X^j)/f(X)\right) \text{ for all } j \in S.$$

  Finally, the algorithm outputs $\mathbf{EVK}_{\tau,\Delta} = (\mathbf{evk}_0, \ldots, \mathbf{evk}_n, \{\mathbf{ksk}_j\}_{j \in S})$ as the evaluation key for blind rotation.
- $\mathsf{BREval}((\mathbf{a}, b), r, \mathbf{EVK}_{\tau,\Delta})$: Given an LWE-based ciphertext $\mathrm{LWE}_{q,\mathbf{s}}(m) = (\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, a rotation polynomial $r(X) \in R_Q$ and an evaluation key $\mathbf{EVK}_{\tau,\Delta}$ at inputs, computes and returns ACC as described in algorithm 1.

In Theorem 1, we show that the output ACC of the blind rotation is indeed a scalar NTRU ciphertext that encrypts $r(X^{\frac{2N}{q}}) \cdot X^{\frac{2N}{q}(-b+\sum_{i=0}^{n-1} a_i s_i)}$.

Theorem 1 (Correctness of Blind Rotation). *Let* $\mathrm{LWE}_{q,\mathbf{s}}(m) = (\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ *be an LWE-based ciphertext under secret key* $\mathbf{s} = (s_1, \ldots, s_{n-1}) \in \mathbb{Z}_q^n$, *where* $\mathbf{a} = (a_0, \ldots, a_{n-1}) \in \mathbb{Z}_q^n$. *Let* N *be a power of 2 and* $q|N$. *Let* $r(X) \in$

Algorithm 1 $\mathsf{BREval}((\mathbf{a}, b), r, \mathbf{EVK}_{\tau,\Delta})$

Input:

An LWE ciphertext $\mathrm{LWE}_{\mathbf{s},q}(m) = (\mathbf{a} = (a_0, \ldots, a_{n-1}), b) \in \mathbb{Z}_q^{n+1}$;
A rotation polynomial $r(X) \in R_Q$;
An evaluation key $\mathbf{EVK}_{\tau,\Delta} = (\mathbf{evk}_0, \ldots, \mathbf{evk}_n, \{\mathbf{ksk}_j\}_{j\in S}) \in R_Q^{d(n+q)}$.

Output:

An NTRU ciphertext $\mathrm{NTRU}_{Q,f,\tau,\Delta}\left(r(X^{\frac{2N}{q}}) \cdot X^{\frac{2N}{q}(-b+\sum_{i=0}^{n-1} a_i s_i)}\right)$.

for $i = 0; i < n; i = i + 1$ **do**
  $w_i = \frac{2N}{q} a_i + 1$
  $w_i' = w_i^{-1} \mod 2N$
end for
$w_n' = 1$
$\mathrm{ACC} \leftarrow \Delta \cdot r(X^{\frac{2N}{q} w_0'}) \cdot X^{-\frac{2N}{q} b w_0'}$
for $i = 0; i < n; i = i + 1$ **do**
  $\mathrm{ACC} \leftarrow \mathrm{ACC} \odot \mathbf{evk}_i$
  If $w_i w_{i+1}' \neq 1$
    $\mathrm{ACC} \leftarrow \mathsf{EvalAuto}_{w_i w_{i+1}'}(\mathrm{ACC}, \mathbf{ksk}_{w_i w_{i+1}'})$
end for
$\mathrm{ACC} \leftarrow \mathrm{ACC} \odot \mathbf{evk}_n$
return ACC

$R_Q = \mathbb{Z}_Q[X]/(X^N + 1)$ be any rotation polynomial. Let $f \in R_Q$ be a polynomial that is invertible in R_Q. Then, for any $\mathbf{EVK}_{\tau,\Delta} = \mathsf{BRKGen}(\mathbf{s}, f)$, and $\mathsf{ACC} = \mathsf{BREval}((\mathbf{a}, b), r, \mathbf{EVK}_{\tau,\Delta})$, we have that ACC is a scalar NTRU ciphertext $\mathrm{NTRU}_{Q,f,\tau,\Delta}(r(X^{\frac{2N}{q}}) \cdot X^{\frac{2N}{q}(-b+\sum_{i=0}^{n-1} a_i s_i)})$.

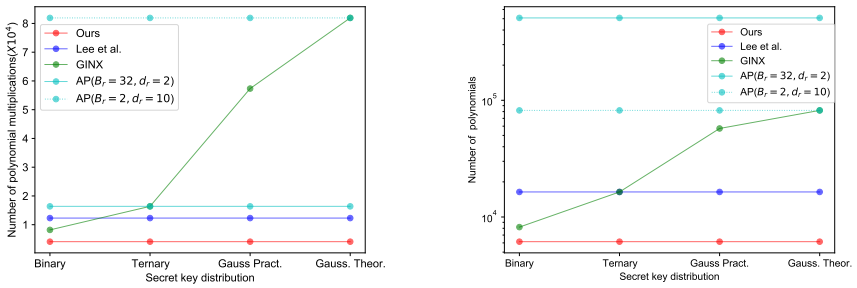
We can prove Theorem 1 following the sketch idea from Section 1.2 and we defer the proof to Appendix A.

4.2 Analysis and Comparisons

In this subsection, we analyze the performance of our blind rotation, and give a comparison with AP [22,5,41], GINX [19,26], and Lee et al.'s blind rotation [37].

Note that the evaluation key $\mathbf{EVK}_{\tau,\Delta}$ for our blind rotation consists of $n+1$ vector NTRU ciphertexts for encrypting $\mathbf{s}$ and $q-1$ vector NTRU ciphertexts for automorphisms. This leads to a total $n+q$ vector NTRU ciphertexts, and thus $d(n+q)$ R_Q elements in the evaluation key. Moreover, it is easy to check that starting from the initial value $\mathsf{ACC} = \Delta \cdot r(X^{\frac{2N}{q} w_0'}) \cdot X^{-\frac{2N}{q} b w_0'}$ in Algorithm 1, we need $n+1$ homomorphic multiplications and at most n automorphisms for the whole blind rotation. Since each automorphism consists of a permutation on the coefficient vector of the input ciphertext (with signs) and an external product, we need at most $2n+1$ external products, which leads to a total up to $d(2n+1)$ multiplications in R_Q. Finally, let σ_e^2 be the noise variance for the second-layer GSW-like encryption. By Lemma 2 and Lemma 3, the variance of the increased noise introduced by a homomorphic multiplication is $\sigma_\odot^2 = dN\frac{B^2}{12}\sigma_e^2$ and by an

automorphism is equal to $\sigma_{KS}^2 = \sigma_{\odot}^2 = dN\frac{B^2}{12}\sigma_e^2$. By the fact that the initial value ACC essentially has zero noise, we have that the final ciphertext of our blind rotation contains a noise with variance $\sigma_{BR}^2 = (2n+1)\sigma_{\odot}^2 = (2n+1)dN\frac{B^2}{12}\sigma_e^2$.



(a) The number of ring multiplications for blind rotation.

(b) The number of ring elements in the evaluation key for blind rotation.

Fig. 1. Comparison of blind rotations using different first-layer key distributions

In Table 1, we give a comparison of our blind rotation with AP/FHEW [22,5,41], GINX/TFHE [19,26], and Lee et al.'s blind rotation [37] in terms of the number $\#R_Q$ of R_Q elements in the evaluation key and the number $\#mul$ of multiplications in R_Q for computation, where n, q are the dimension and modulus for the first-layer encryption; N, Q are the dimension and modulus of the ring $R_Q = \mathbb{Z}_Q[X]/(X^N+1)$ for the second-layer GSW-like encryption; d is bit-decomposition dimension for multiplying a second-layer GSW-like ciphertext; B is a base parameter in bit-decomposing the first-layer ciphertexts for AP; and $|U|$ denote the size of the public set U in bit-decomposing the first-layer secret keys for GINX. Since q is usually less than $2n$ for typically choices of parameters [39], and $|U| \geq 1$ for all key distributions, one can see from Table 1 that our blind rotation has asymptotically the best performance in both evaluation key size and computational efficiency.

In Figure 1, we give a comparison of different blind rotations using different first-layer key distributions. As in [39,37], we fix all other parameters, and only consider the use of different key distributions for the first-layer encryption. Specifically, we fix the parameters $(n, q, d) = (512, 1024, 4)$ for AP, GINX and Lee et al. that is recommended in STD128 [39] and $(n, q, d) = (512, 1024, 5)$ for our blind rotation using parameter P128T. We consider three types of key distributions, namely, binary distribution, ternary distribution, and Gaussian distribution. In Figure 1, "Gauss Pract." and "Gauss. Theor." refer to Gaussian key distributions with standard deviation $\sigma = 3.19$ and $\sigma = \sqrt{n}$, respectively. We bound the size of an element chosen from Gaussian key distribution by using 12σ as in [39], which leads to a basis set U with size $|U| = 7$ and $|U| = 10$ corresponding to "Gauss Pract." and "Gauss. Theor." for GINX, respectively.

Table 3. Estimation of noise variance for different blind rotations.

Methods	Noise variance	Ratio
AP [22,5,41]	$d_r dnN\frac{B^2}{6}\sigma_e^2$	$\frac{2d_r n}{2n+1} \approx d_r$
GINX [19,26]	$\lvert U\rvert dnN\frac{B^2}{3}\sigma_e^2$	$\frac{4\lvert U\rvert n}{2n+1} \approx 2\lvert U\rvert$
Lee et al. [37]	$(3n+2)dN\frac{B^2}{12}\sigma_e^2$	$\frac{3n+2}{2n+1} \approx 1.5$
Ours	$(2n+1)dN\frac{B^2}{12}\sigma_e^2$	\

As shown in Figure 1, our bind rotation has performance asymptotically better than others for all key distributions.

In Table 3, we give a comparison of different blind rotations in terms of noise variance (i.e., the variance of the noise contained in the ciphertext output by the blind rotation). For AP, GINX and Lee et al.'s blind rotation, we use the formula given in [37]:

$$\sigma_{AP}^2 = d_r dnN\frac{B^2}{6}\sigma_e^2, \sigma_{GINX}^2 = 2|U|dnN\frac{B^2}{6}\sigma_e^2, \text{and } \sigma_{Lee}^2 = (3n+2)dN\frac{B^2}{12}\sigma_e^2.$$

Here $d_r = \lceil \log_{B_r} q \rceil$ for some integer parameter B_r in bit-decomposing the first-layer ciphertexts for AP. The column "Ratio" denotes the ratio between the considered blind rotation in that row and ours. As shown in Table 3, our blind rotation has the smallest noise variance, which makes it possible to use smaller parameters (e.g., N, Q) for the second-layer GSW-like encryption.

5 Bootstrapping LWE-based First-layer Ciphertexts

In this section, we describe how to apply our new blind rotation to bootstrap an LWE-based first-layer ciphertext. Our bootstrapping has a similar high-level structure as that in [22,5,41,19,26,37], which can be described as follows:

$$\mathrm{LWE}_{q,s}(m) \xrightarrow[\text{Alg .1}]{\mathsf{BREval}} \mathsf{ACC} \xrightarrow[\text{[33]}]{\mathsf{Ext}} \mathrm{LWE}_{Q,\mathbf{f}}(m) \xrightarrow[\text{Lemma 4+Lemma 5}]{\mathsf{ModSwitch+KeySwitch}} \mathrm{LWE}_{q,\mathbf{s}}(m).$$

Given an LWE ciphertext $(\mathbf{a}, b) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s}\rangle - \mathsf{noised}(m)) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where $\mathsf{noised}(m) = e + \lfloor \frac{q}{t} \rceil m$ (here we take Regev-like ciphertexts [42] as an example, the same idea can be extended to BGV-like [10] and CKKS-like [16] ciphertexts by using different choice of (τ, Δ) and rotation polynomial $r(X) \in R_Q$). By first applying our blind rotation in Algorithm 1, we get a scalar NTRU ciphertext $c = \mathsf{ACC}$ that encrypts $r(X^{\frac{2N}{q}}) \cdot X^{\frac{2N}{q}\mathsf{noised}(m)}$ under secret key $f \in R_Q$. Let $r(X) = \sum_{i=0}^{q/2-1}(\lfloor i/\lfloor q/t \rceil \rceil \bmod t)X^{-i}$, one can easily check that the constant coefficient of $r(X^{\frac{2N}{q}}) \cdot X^{\frac{2N}{q}\mathsf{noised}(m)}$ is m by the correctness of the first-layer encryption that $m = \lfloor \mathsf{noised}(m)/\lfloor q/t \rceil \rceil$. Let $\mathbf{c} = (c_0, \ldots, c_{N-1})$, $\mathbf{f} = (f_0, \ldots, f_{N-1}) \in \mathbb{Z}_Q^N$ be the coefficient vectors of the polynomials $c, f \in R_Q$. Then, we can directly set $\hat{\mathbf{c}} = (c_0, -c_{N-1}, \ldots, -c_1) \in \mathbb{Z}_Q^N$ the same as the LWE sample extraction algorithm Ext in [33] to get a ciphertext $\mathrm{LWE}_{Q,\mathbf{f}}(m) = (\hat{\mathbf{c}} = (c_0, -c_{N-1}, \ldots, -c_1), 0)$ that

encrypts m under the secret key $\mathbf{f} \in \mathbb{Z}_Q^N$. Finally, we can use modulus-switching and key-switching to convert $\mathrm{LWE}_{Q,\mathbf{f}}(m)$ to a desired $\mathrm{LWE}_{q,\mathbf{s}}(m)$.

5.1 Modulus Switching for LWE-based Ciphertexts

The modulus switching from Q to q can be easily achieved by multiplying the targeted ciphertext with q/Q, and rounding the results to the nearest integer.

Lemma 4 (Modulus switching for LWE, adapted from [22]). *Given a ciphertext* $\mathrm{LWE}_{Q,\mathbf{f}}(m) = (\mathbf{a}, b = \langle \mathbf{a}, \mathbf{f} \rangle - (e + \lfloor \frac{Q}{t} \rceil m)) \in \mathbb{Z}_Q^N \times \mathbb{Z}_Q$ *using some key distribution with variance* $Var(\mathbf{f})$ *and some noise distribution with variance* $Var(e)$ *and a targeted modulus* q*, define the modulus switching as*

$$(\mathbf{a}', b') = \mathsf{ModSwitch}(\mathrm{LWE}_{Q,\mathbf{f}}(m), q) = \left\lfloor \mathrm{LWE}_{Q,\mathbf{f}}(m) \cdot \frac{q}{Q} \right\rceil \mod q.$$

Then, $(\mathbf{a}', b')$ *is an LWE ciphertext that encrypts the same* m *under the same secret key* $\mathbf{f} \in \mathbb{Z}^N$*. Moreover, the noise variance of the resulting ciphertext* $(\mathbf{a}', b')$ *is bounded by* $\frac{q^2}{Q^2} Var(e) + \frac{1 + N Var(\mathbf{f})}{12}$*.*

We denote $\sigma_{MS}^2 = \frac{1 + N Var(\mathbf{f})}{12}$ as the variance of the increased noise (with respect to the input ciphertext) for modulus switching.

5.2 Key Switching for LWE-based Ciphertexts

In this subsection, we define a pair of two algorithms $(\mathsf{LWE\_KSKG}, \mathsf{LWE\_KS})$ for key switching LWE ciphertexts as follows:

- $\mathsf{LWE\_KSKG}(\mathbf{f}, \mathbf{s}, q, B_{ks})$: Given two vectors $\mathbf{f} = (f_0, \ldots, f_{N-1}) \in \mathbb{Z}_q^N, \mathbf{s} = (s_0, \ldots, s_{n-1}) \in \mathbb{Z}_q^n$ and two integers q, B_{ks} as inputs, the algorithm first computes $d_{ks} = \lceil \log_{B_{ks}} q \rceil$. Then, it creates a set of ciphertexts:

  $$\mathbf{lksk}_{i,j,v} = (\mathbf{a}_{i,j,v}, b_{i,j,v} = \langle \mathbf{a}_{i,j,v}, \mathbf{s} \rangle - e_{i,j,v} - v B_{ks}^j f_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$$

  for all $i \in \mathbb{Z}_N, j \in \mathbb{Z}_{d_{ks}}, 1 \leq v < B_{ks}$ by randomly choosing $\mathbf{a}_{i,j,v} \leftarrow \mathbb{Z}_q^n$ and noise $e_{i,j,v}$ from some noise distribution over $\mathbb{Z}$. Finally, it outputs $\mathbf{LKSK}_{\mathbf{f},\mathbf{s},q,B_{ks}} = \{\mathbf{lksk}_{i,j,v}\}_{i \in \mathbb{Z}_N, j \in \mathbb{Z}_{d_{ks}}, 1 \leq v < B_{ks}}$ as the key-switching key.
- $\mathsf{LWE\_KS}((\hat{\mathbf{a}}, \hat{b}), \mathbf{LKSK}_{\mathbf{f},\mathbf{s},q,B_{ks}})$: Given as inputs a ciphertext $\mathrm{LWE}_{q,\mathbf{f}}(m) = (\hat{\mathbf{a}}, \hat{b} = \langle \hat{\mathbf{a}}, \mathbf{f} \rangle - (\hat{e} + \lfloor \frac{q}{t} \rceil m)) \in \mathbb{Z}_q^N \times \mathbb{Z}_q$ for some $\hat{\mathbf{a}} = (\hat{a}_0, \ldots, \hat{a}_N) \in \mathbb{Z}_q^N$ and a key-switching key $\mathbf{LKSK}_{\mathbf{f},\mathbf{s},q,B_{ks}} = \{\mathbf{lksk}_{i,j,v}\}_{i \in \mathbb{Z}_N, j \in \mathbb{Z}_{d_{ks}}, 1 \leq v < B_{ks}}$ for some $\mathbf{lksk}_{i,j,v} = (\mathbf{a}_{i,j,v}, b_{i,j,v})$ as inputs, the algorithm first decomposes each $\hat{a}_i = \sum_{j=0}^{d_{ks}-1} v_{i,j} B_{ks}^j$ into a vector $(v_{i,0}, \ldots, v_{i,d_{ks}-1}) \in \mathbb{Z}_{B_{ks}}^{d_{ks}}$. Then, it computes

  $$\hat{\mathbf{a}}' = \sum_{i=0}^{N-1} \sum_{j=0, v_{i,j} \neq 0}^{d_{ks}-1} \mathbf{a}_{i,j,v_{i,j}}, \quad \hat{b}' = \sum_{i=0}^{N-1} \sum_{j=0, v_{i,j} \neq 0}^{d_{ks}-1} b_{i,j,v_{i,j}} + \hat{b}.$$

  Finally, it outputs $(\hat{\mathbf{a}}', \hat{b}') \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.

Lemma 5 (Key Switching for LWE Ciphertexts). *Let* $\mathbf{f} \in \mathbb{Z}_q^N, \mathbf{s} \in \mathbb{Z}_q^n$ *be two vectors. Let* $\mathrm{LWE}_{q,\mathbf{f}}(m) = (\hat{\mathbf{a}}, \hat{b}) \in \mathbb{Z}_q^N \times \mathbb{Z}_q$ *be a ciphertext that encrypts* $m \in \mathbb{Z}_q$ *under the secret key* $\mathbf{f} \in \mathbb{Z}_q^N$*. Then, for any* $\mathbf{LKSK}_{\mathbf{f},\mathbf{s},q,B_{ks}} = \mathsf{LWE\_KSKG}(\mathbf{f}, \mathbf{s}, q, B_{ks})$*, we have that* $(\hat{\mathbf{a}}', \hat{b}') = \mathsf{LWE\_KS}((\hat{\mathbf{a}}, \hat{b}), \mathbf{LKSK}_{\mathbf{f},\mathbf{s},q,B_{ks}}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ *is a ciphertext that encrypts* $m \in \mathbb{Z}_q$ *under the secret key* $\mathbf{s} \in \mathbb{Z}_q^n$*.*

Moreover, if the variance of the noise in $(\hat{\mathbf{a}}, \hat{b})$ *is* $Var(\hat{e})$*, and the variance of the noise distribution used in generating* $\mathbf{LKSK}_{\mathbf{f},\mathbf{s},q,B_{ks}}$ *is* $Var(e)$*, we have that the variance* $Var(\hat{e}')$ *of the noise* $\hat{e}'$ *in* $(\hat{\mathbf{a}}', \hat{b}')$ *is upper bounded by* $Var(\hat{e}') \leqslant Nd_{ks}Var(e) + Var(\hat{e})$*, where* $d_{ks} = \lceil \log_{B_{ks}} q \rceil$*.*

We defer the proof of lemma 5 to Appendix B, as it is also presented in [22]. Similarly, we use the symbol $\sigma_{LKS}^2 = Nd_{ks}Var(e)$ to denote the variance of the increased noise (with respect to the input ciphertext) for LWE key-switching.

5.3 The Bootstrapping Algorithm

In this section, we give the full description of our bootstrapping algorithm for LWE-based first-layer ciphertexts in Algorithm 2 (note that we use a modulus switching from Q to Q_{ks} for some $q < Q_{ks} < Q$ before key-switching to optimize the parameters for generating the LWE key-switching key).

Algorithm 2 Bootstrapping LWE-based First-layer Ciphertexts

Input:
An LWE ciphertext $\mathrm{LWE}_{q,\mathbf{s}}(m) = (\mathbf{a} = (a_0, \ldots, a_{n-1}), b) \in \mathbb{Z}_q^{n+1}$;
A rotation polynomial $r(X) \in R_Q$;
Evaluation key $\mathbf{EVK}_{\tau,\Delta} = (\mathbf{evk}_0, \ldots, \mathbf{evk}_n, \{\mathbf{ksk}_j\}_{j \in S}) \in R_Q^{d(n+q)}$;
LWE key-switching key $\mathbf{LKSK}_{\mathbf{f},\mathbf{s},Q_{ks},B_{ks}} = \{\mathbf{lksk}_{i,j,v}\}_{i \in \mathbb{Z}_N, j \in \mathbb{Z}_{d_{ks}}, 1 \leq v < B_{ks}}$;
Output:
A refreshed LWE ciphertext $\mathrm{LWE}_{q,\mathbf{s}}(m) \in \mathbb{Z}_q^{n+1}$;
1: $\mathsf{ACC} \leftarrow \mathsf{BREval}((\mathbf{a}, b), r, \mathbf{EVK}_{\tau,\Delta})$
2: $\mathrm{LWE}_{Q,\mathbf{f}}(m) \leftarrow \mathsf{Ext}(\mathsf{ACC})$
3: $\mathrm{LWE}_{Q_{ks},\mathbf{f}}(m) \leftarrow \mathsf{ModSwitch}(\mathrm{LWE}_{Q,\mathbf{f}}(m), Q_{ks})$
4: $\mathrm{LWE}_{Q_{ks},\mathbf{s}}(m) \leftarrow \mathsf{LWE\_KS}(\mathrm{LWE}_{Q_{ks},\mathbf{f}}(m), \mathbf{LKSK}_{\mathbf{f},\mathbf{s},Q_{ks},B_{ks}})$
5: $\mathrm{LWE}_{q,\mathbf{s}}(m) \leftarrow \mathsf{ModSwitch}(\mathrm{LWE}_{Q_{ks},\mathbf{s}}(m), q)$

Theorem 2 (Bootstrapping LWE-based First-layer Ciphertexts). *Let* $q < Q_{ks} < Q$ *be positive integers. Given an LWE ciphertext* $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle - \mathsf{noised}(m)) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ *encrypting a plaintext* $m \in \mathbb{Z}_t$ *with* $\mathsf{noised}(m) = \lfloor \frac{q}{t} \rceil m + e$*, Algorithm 2 outputs a refreshed LWE ciphertext* $(\mathbf{a}', b' = \mathbf{a}'\mathbf{s} - (\lfloor \frac{q}{t} \rceil m + e')) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ *encrypting the same plaintext. And the noise* e' *of the refreshed ciphertext is bounded by a Gaussian with standard deviation*

$$\sqrt{\frac{q^2}{Q_{ks}^2}\left(\frac{Q_{ks}^2}{Q^2}\sigma_{BR}^2 + \sigma_{MS1}^2 + \sigma_{LKS}^2\right) + \sigma_{MS2}^2}$$

where $\sigma^2_{BR} = (2n+1)dN\frac{B^2}{12}Var(g)$, $\sigma^2_{MS1} = \frac{NVar(f)+1}{12}$, $\sigma^2_{LKS} = Nd_{ks}Var(e)$, *and* $\sigma^2_{MS2} = \frac{nVar(\mathbf{s})+1}{12}$ *are the noise variances contributed respectively by the blind rotation, the first modulus switching, the key switching, and the second modulus switching in Algorithm 2, and* $Var(g), Var(f), Var(e), Var(\mathbf{s})$ *are respectively the noise variance and secret key variance of NTRU ciphertext and the noise variance and secret key variance of LWE ciphertext.*

Proof. The correctness of Algorithm 2 directly follows from the correctness of the blind rotation in Theorem 1, the modulus switching in Lemma 4, and the LWE key switching in Lemma 5.

Next we analyze the noise of the resulting ciphertext. After the blind rotation in step 1, we get a scalar NTRU ciphertext with noise variance bounded by $\sigma^2_{BR} = (2n+1)dN\frac{B^2}{12}Var(g)$ according to the analysis in Section 4.2. By a trivial Ext in step 2 we get an LWE ciphertext without increasing the noise variance. By Lemma 4, we have the noise variance after the first modulus switching from Q to Q_{ks} in step 3 is bounded by $\frac{Q^2_{ks}}{Q^2}\cdot\sigma^2_{BR} + \sigma^2_{MS1}$ where $\sigma^2_{MS1} = \frac{NVar(f)+1}{12}$. By Lemma 5, the noise variance after the key switching in step 4 is bounded by $\frac{Q^2_{ks}}{Q^2}\cdot\sigma^2_{BR} + \sigma^2_{MS1} + \sigma^2_{LKS}$, where $\sigma^2_{LKS} = Nd_{ks}Var(e)$. Again, by Lemma 4, we have the noise variance after the second modulus switching in step 5 from Q_{ks} to q is bounded by $\frac{q^2}{Q^2_{ks}}\left(\frac{Q^2_{ks}}{Q^2}\sigma^2_{BR} + \sigma^2_{MS1} + \sigma^2_{LKS}\right) + \sigma^2_{MS2}$ where $\sigma^2_{MS2} = \frac{nVar(\mathbf{s})+1}{12}$. This finally completes the proof.

6 Bootstrapping RLWE-based First-layer Ciphertexts

We now describe how to apply our blind rotation to bootstrap an RLWE-based first-layer ciphertext. Our bootstrapping has a high-level structure as follows:

$$\mathrm{RLWE}_{q,s}(m) \xrightarrow[[33]]{\mathsf{Ext}} \{\mathrm{LWE}_{q,\mathbf{s}}(m_i)\} \xrightarrow[\text{Alg.1+[33]}]{\mathsf{BREval+Ext}} \{\mathrm{LWE}_{Q,\mathbf{f}}(m_i)\} \xrightarrow[\text{Lemma 6}]{\mathsf{Pack}} \mathrm{RLWE}_{Q,s}(m).$$

Given an RLWE ciphertext $\mathrm{RLWE}_{q,s}(m) = (a,b) \in \hat{R}_q \times \hat{R}_q$, and let $\mathbf{a} = (a_0,\ldots,a_{n-1}), \mathbf{s} = (s_0,\ldots,s_{n-1}), \mathbf{m} = (m_0,\ldots,m_{n-1}) \in \mathbb{Z}_q^n$ be the coefficients of $a, s, m \in \hat{R}_q$, one can easily extract n LWE ciphertexts $(\mathbf{a}_i, b_i)$ that encrypts m_i under secret $\mathbf{s}$ by the LWE sample extraction algorithm Ext in [33]. Then, for each LWE ciphertext $(\mathbf{a}_i, b_i)$, we apply our blind rotation algorithm BREval to obtain an NTRU ciphertext c_i that encrypts $r(X^{\frac{2N}{q}}) \cdot X^{\frac{2N}{q}\mathsf{noised}(m_i)}$ under the secret key f. Let $\mathbf{f} = (f_0, f_1, \ldots, f_{N-1}) \in \mathbb{Z}_Q^N$ and $\mathbf{c}_i = (c_{i,0}, c_{i,1}, \cdots, c_{i,N-1}) \in \mathbb{Z}_Q^N$ be the coefficient vectors of $f, c_i \in R_Q$. Let $\hat{\mathbf{c}}_i = (c_{i,0}, -c_{i,N-1}, \ldots, -c_{i,1}) \in \mathbb{Z}_Q^N$, then $(\hat{\mathbf{c}}_i, 0) \in \mathbb{Z}_Q^N \times \mathbb{Z}_Q$ is actually an LWE ciphertext that encrypts m_i under the secret key $\mathbf{f} = (f_0, f_1, \ldots, f_{N-1}) \in \mathbb{Z}_Q^N$. In particular, by setting appropriate rotation polynomial $r(X)$, we can always have that $\langle \hat{\mathbf{c}}_i, \mathbf{f}\rangle = \tau \cdot e_i + \Delta \cdot m_i$ for some noise e_i and integers (τ, Δ) depending on the encoding of m in $\mathrm{RLWE}_{q,s}(m)$. Finally, we can use the packing algorithm Pack in [40] pack the LWE ciphertexts into an RLWE ciphertext $\mathrm{RLWE}_{Q,s}(m) = (a', b') \in \hat{R}_Q \times \hat{R}_Q$. For completeness, we describe the packing algorithm in [40] in Section 6.1.

6.1 Packing LWE Ciphertexts to RLWE Ciphertexts

Let n, B, Q, τ, Δ be positive integers, and let $d = \lceil \log_B Q \rceil$. Without loss of generality, we assume that an LWE ciphertext $\mathrm{LWE}_{Q,\mathbf{f}}(m)$ has the form of $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{f} \rangle - (\tau \cdot e + \Delta \cdot m))$. Let $\hat{R}_Q = \mathbb{Z}[X]/(X^n + 1)$. We recall the algorithm for packing LWE ciphertexts into RLWE ciphertexts in [40]. Formally, we define two algorithms $(\mathsf{PackKGen}, \mathsf{Pack})$ as follows:

- $\mathsf{PackKGen}(\mathbf{f}, s)$: Given a vector $\mathbf{f} = (f_0, \cdots, f_{N-1}) \in \mathbb{Z}_Q^N$ and a ring element $s \in \hat{R}_Q$ as inputs, the algorithm first computes a set of ciphertexts

$$\mathbf{rpk}_{j,k} = (\hat{a}_{j,k}, \hat{b}_{j,k} = \hat{a}_{j,k} s - \tau \cdot \hat{e}_{j,k} - B^k f_j) \in \hat{R}_Q \times \hat{R}_Q$$

  for all $j \in \mathbb{Z}_N, k \in \mathbb{Z}_d$ by randomly choosing $\hat{a}_{j,k} \leftarrow \hat{R}_Q$ and noise $\hat{e}_{j,k}$ from some distribution over $\hat{R}_Q$. Then, it outputs $\mathbf{RPK}_\tau = \{\mathbf{rpk}_{j,k}\}$.
- $\mathsf{Pack}(\{\mathrm{LWE}_{Q,\mathbf{f}}(m_i)\}, \mathbf{RPK}_\tau)$: Given ciphertexts $\mathrm{LWE}_{Q,\mathbf{f}}(m_i) = (\mathbf{a}_i, b_i) \in \mathbb{Z}_Q^N \times \mathbb{Z}_Q$ for all $0 \leqslant i \leqslant n-1$ and $\mathbf{a}_i = (a_{i,0}, a_{i,1}, \cdots, a_{i,N-1}) \in \mathbb{Z}_Q^N$, and the packing key $\mathbf{RPK}_\tau = \{\mathbf{rpk}_{j,k}\}$ for some $\mathbf{rpk}_{j,k} = (\hat{a}_{j,k}, \hat{b}_{j,k})$ as inputs, the algorithm first computes $u_j = \sum_{i=0}^{n-1} a_{i,j} X^i$ for $0 \leq j \leq N-1$, and $\mathrm{BitDecom}_B(u_j) = (u_{j,0}, \cdots, u_{j,d-1}) \in \hat{R}_B^d$ for some $u_j = \sum_{k=0}^{d-1} u_{j,k} B^k$. Then, it computes

$$u = \sum_{j=0}^{N-1} \sum_{k=0}^{d-1} u_{j,k} \hat{a}_{j,k}, \qquad v = \sum_{j=0}^{N-1} \sum_{k=0}^{d-1} u_{j,k} \hat{b}_{j,k} + \sum_{i=0}^{n-1} b_i X^i.$$

  Finally, it returns $(u, v) \in \hat{R}_Q \times \hat{R}_Q$ as the output.

Lemma 6 (Correctness of Packing). *Let $\mathbf{f} = (f_0, \cdots, f_{N-1}) \in \mathbb{Z}_Q^N$ be a vector, and let $s \in \hat{R}_Q$ be a ring element. For all $0 \leq i \leq n-1$, let $\mathrm{LWE}_{Q,\mathbf{f}}(m_i) = (\mathbf{a}_i, b_i = \langle \mathbf{a}_i, \mathbf{f} \rangle - (\tau \cdot e_i + \Delta \cdot m_i)) \in \mathbb{Z}_Q^N \times \mathbb{Z}_Q$ be an LWE ciphertext that encrypts m_i under secret key $\mathbf{f} = (f_0, \cdots, f_{N-1}) \in \mathbb{Z}_Q^N$. Then, for any $\mathbf{RPK}_\tau = \{\mathbf{rpk}_{j,k}\} = \mathsf{PackKGen}(\mathbf{f}, s)$, we have that $(u, v) = \mathsf{Pack}(\{\mathrm{LWE}_{Q,\mathbf{f}}(m_i)\}, \mathbf{RPK}_\tau) \in \hat{R}_Q \times \hat{R}_Q$ is a ciphertext that encrypts $m = \sum_{i=0}^{n-1} m_i X^i$ under the secret key $s \in \hat{R}_Q$.*

Moreover, if the variance of the noise in $\mathrm{LWE}_{Q,\mathbf{f}}(m_i)$ is $Var(e)$, and the variance of the noise distribution in generating $\mathbf{RPK}_\tau$ is $Var(\hat{e})$, then the variance $Var(e')$ of the noise in the resulting ciphertext (u, v) is upper bounded by $Var(e') \leqslant nNd\frac{B^2}{12} Var(\hat{e}) + Var(e)$.

We defer the proof of Lemma 6 to Appendix C, as it is also presented in [40].

6.2 The Bootstrapping Algorithm

In this subsection, we give the bootstrapping algorithm for RLWE-based first-layer ciphertexts in Algorithm 3, and prove the following theorem.

Algorithm 3 Bootstrapping RLWE-based First-layer Ciphertexts

Input:

RLWE ciphertext $\mathrm{RLWE}_{q,\mathbf{s}}(m) = (a, b) \in \hat{R}_q \times \hat{R}_q$;
A rotation polynomial $r(X) \in R_Q$;
Evaluation keys $\mathbf{EVK}_{\tau,\Delta} = (\mathbf{evk}_0, \ldots, \mathbf{evk}_n, \{\mathbf{ksk}_j\}_{j\in S}) \in R_Q^{d(n+q)}$;
Packing keys $\mathbf{RPK}_\tau$;

Output:

A refreshed RLWE ciphertext $(a', b') \in \hat{R}_Q \times \hat{R}_Q$;

1: **for** $i = 0, i < n, i = i + 1$ **do**
2: $(\mathbf{a}_i, b_i) \leftarrow \mathsf{Ext}(\mathrm{RLWE}_{q,\mathbf{s}}(m))$
3: $\mathsf{ACC}_i \leftarrow \mathsf{BREval}((\mathbf{a}_i, b_i), r, \mathbf{EVK}_{\tau,\Delta})$
4: $\mathrm{LWE}_{Q,\mathbf{f}}(m_i) \leftarrow \mathsf{Ext}(\mathsf{ACC}_i)$
5: **end for**
6: $\mathrm{RLWE}_{Q,\mathbf{s}}(m) = (a', b') \leftarrow \mathsf{Pack}(\{\mathrm{LWE}_{Q,\mathbf{f}}(m_i)\}_{i\in[0,n-1]}, \mathbf{RPK}_\tau)$

Theorem 3 (Bootstrapping RLWE-based First-layer Ciphertexts). *Given an RLWE ciphertext $(a, b) = (a, as - \mathsf{noised}(m)) \in \hat{R}_q \times \hat{R}_q$ encrypting a plaintext $m \in \hat{R}$, Algorithm 3 outputs a refreshed RLWE ciphertext $(a', b' = a's - (\tau \cdot e' + \Delta \cdot m)) \in \hat{R}_Q \times \hat{R}_Q$ encrypting the same plaintext. And the noise e' of the refreshed ciphertext is bounded by a Gaussian with standard deviation*

$$\sqrt{nNd\frac{B^2}{12}Var(e) + Var(e) + (2n+1)dN\frac{B^2}{12}Var(g)}$$

where $Var(g), Var(e)$ represents the noise variance of NTRU ciphertext and the noise variance of RLWE ciphertext, respectively.

Proof. The correctness of Algorithm 3 directly follows from the correctness of the blind rotation in Theorem 1 and the LWE packing in Lemma 6.

Now, we analyze the noise of the resulting ciphertext. After each blind rotation in step 2, we get a scalar NTRU ciphertext with noise variance bounded by $Var(e)+\sigma_{BR}^2 = Var(e)+(2n+1)dN\frac{B^2}{12}Var(g)$. Then we extract the coefficients of NTRU ciphertext as LWE ciphertext, and use the packing algorithm to obtain a refreshed RLWE ciphertext in $\hat{R}_Q$ under large modulus Q. By Lemma 6, the refreshed noise is a Gaussian with standard deviation

$$\sqrt{nNd\frac{B^2}{12}Var(e) + Var(e) + (2n+1)dN\frac{B^2}{12}Var(g)}.$$

This completes the proof.

7 Security, Parameters and Implementation

In this section, we present the necessary security analysis and the experimental results of our bootstrapping algorithm for LWE-based first-layer ciphertexts, and compare its performance with that of FHEW/AP [22] and TFHE/GINX [19] running on the same laptop.

7.1 Security Analysis

The ciphertext of our first-layer encryption basically consists of (R)LWE samples, and thus it is indistinguishable from uniform based on the decisional (R)LWE assumptions. The ciphertext of our second-layer encryption contains scalar NTRU ciphertexts and vector NTRU ciphertexts. Our scalar NTRU encryption has form of $c = (\tau \cdot g + \Delta \cdot u)/f$. Since τ and Δ are publicly defined constants, this can be simplified as $c = (t \cdot g + u)/f$, where t is an integer. Because the standard NTRU encryption (in the symmetric key setting) has the form of $c = t \cdot g/f + m$, our ciphertext can be seen as a standard NTRU ciphertext that encrypts a key-dependent message $m = u/f$. Thus, the security of our second-layer encryption requires the circular-security/KDM-security assumption of the standard NTRU scheme, which is the same as that in [7,27,41]. Note that the construction of FHEs without such kind of assumptions is currently unknown. Moreover, the currently best algorithm solving $c = (t \cdot g + u)/f$, to the best of our knowledge, is to treat it as an RLWE instance (c, u) (as we can rewrite it as $u = cf - t \cdot g$), and the lattice used to solve this problem is very close to the standard NTRU lattice. Based on known algorithms solving (R)LWE and NTRU problems, we believe that this type of encryption $c = (t \cdot g + u)/f$ will not provide extra advantage to the adversary from the point of concrete attacks. Our vector NTRU ciphertext (for generating the evaluation keys $\mathbf{EVK}_{\tau,\Delta} = (\mathbf{evk}_0, \ldots, \mathbf{evk}_n, \{\mathbf{ksk}_j\}_{j \in S})$) basically encrypts different messages using the same secret key as that in [7,41], whose security is essentially guaranteed by the decisional Vectorized-NTRU assumption.

In all, the semantically (IND-CPA) security of our two-layer FHE scheme can be directly proved under the decisional (R)LWE problem, the decisional NTRU problem (with KDM security) and the decisional Vectorized-NTRU problem via a standard hybrid argument.

7.2 Parameters

In Table 4, we give all the parameters used in our experiment. The parameter STD128 (resp. STD192) is used for FHEW/AP [22] and TFHE/GINX [19], which is the default parameter for 128-bit (resp., 192-bit) security provided in the OpenFHE 0.9.1 library [6] and recommended in [39]. [9]

For 128-bit security, we choose two types of parameters, namely, P128T and P128G, for our algorithms: P128T is set to have the same parameters (n, q) and ternary key distribution with STD128 for the first-layer LWE encryption; while P128G is set to use a Gaussian key distribution with variance 4/3. We use the LWE estimator [4] to estimate the security of (R)LWE instances, and use the NTRU estimator offered by Ducas and van Woerden [23] to find the BKZ block size β needed by Dense Sublattice Discovery(DSD) attack. Then, we use the cost

[9] Our code is publicly verifiable at `https://github.com/SKLC-FHE/CHIFHE`. We also observed that increasing the secret key and noise variance settings for NTRU allows selection of more competitive parameters. Optimized parameters and experimental results can be found in Appendix D.

Table 4. Parameters for bootstrapping LWE-based first-layer ciphertexts.

Parameters	Key distrib.	n	q	N	Q	B	Q_{ks}	B_{ks}
STD128 [39]	Ternary	512	1024	1024	2^{27}	2^7	2^{14}	2^7
P128T	Ternary	512	1024	1024	$995329 \approx 2^{19.9}$	2^4	2^{14}	2^7
P128G	Gaussian	465	1024	1024	$995329 \approx 2^{19.9}$	2^4	2^{14}	2^7
STD192 [39]	Ternary	1024	1024	2048	2^{37}	2^{13}	2^{19}	28
P192T	Ternary	1024	1024	2048	$44421121 \approx 2^{25.4}$	2^9	2^{19}	28
P192G	Gaussian	870	1024	2048	$44421121 \approx 2^{25.4}$	2^9	2^{17}	28

model $T(d, \beta) := 2^{0.292 \cdot \beta + 16.4 + \log_2(8 \cdot d)}$(in the NTRU setting $d = 2N$) to estimate the concrete security. Both P128T and P128G are set to provide at least 128-bit security. Similar to [22,17,39], they are also set to satisfy the requirement that the decryption failure upper bound is less than 2^{-32}. We choose P192T and P192G to provide at least 192-bit security as STD192 using the same way as P128T and P128G.

For a NAND gate, we estimate the failure probability of decryption using the same formula $P = 1 - \mathrm{erf}\left(\frac{q/8}{2\beta'}\right)$ (β' represents the standard deviation of the final noise) as that in [22,17,39]. The estimated decryption failure probabilities for our parameters are given by $P_{\mathrm{P128T}} = 2^{-32}$, $P_{\mathrm{P128G}} = 2^{-34}$, $P_{\mathrm{P192T}} = 2^{-53}$ and $P_{\mathrm{P192G}} = 2^{-42}$. Note that our parameters have modulus $Q < N^2$ (resp., $Q < N^{2.31}$) for both P128T and P128G (resp., P192T and P192G), which is smaller than the fatigue point $Q = N^{2.484}$ in [23] to avoid sublattice attacks on NTRU problems.

7.3 Experimental Results

We implement our bootstrapping algorithm for LWE-based first-layer ciphertexts in C++ on a laptop with an Intel(R) Core(TM) i5 CPU @ 2.30GHz and 16 GB RAM, running Linux 4.4.0-19041-Microsoft. We compare the performance with FHEW/AP [22] and TFHE/GINX [19] that were implemented in the OpenFHE library [6]. All the experiments are running with a single thread at a single CPU core. In Table 5, we present the experimental results, where the timing figures are averaged over 1000 times running of the algorithms.

From Table 5, one can see that for parameters with ternary first-layer key distribution providing 128-bit security, we only need to store 18.65MB for blind rotation, which is about 89.8 times smaller than FHEW/AP [22,6] and 2.9 times smaller than TFHE/GINX [19,6]. Moreover, our algorithm only needs 112ms to evaluate a NAND gate followed by a bootstrapping, which is about 3.2 times faster than FHEW/AP and 2.1 times faster than TFHE/GINX. By using P128G with Gaussian key distribution, we can obtain an extra 10% improvement over P128T (we note that the performance of TFHE/GINX would become worse by roughly a factor of 1.5 when using the same Gaussian key distribution as

Table 5. Timings and key sizes for bootstrapping (where the column "Timings" denotes the timing for performing a NAND gate followed by a bootstrapping, the column "EVK" denotes the evaluation key size for blind rotation; the column "KSK" denotes the key size for switching a second-layer ciphertext back to a first-layer one; the last column "Boots. key" denotes the whole bootstrapping key size.)

Algorithms	Parameters	Key distrib.	Timings (ms)	EVK (MB)	KSK (MB)	Boots. key (MB)
FHEW/AP [22,6]	STD128 [39]	Ternary	359	1674	224	1898
TFHE/GINX [19,6]	STD128 [39]	Ternary	234	54	224	278
Ours	P128T	Ternary	112	18.65	224	242.65
	P128G	Gaussian	100	17.90	203.44	221.34
FHEW/AP [22,6]	STD192 [39]	Ternary	1200	6682	532	7214
TFHE/GINX [19,6]	STD192 [39]	Ternary	859	222	532	754
Ours	P192T	Ternary	320	38.10	532	570.10
	P192G	Gaussian	273	34.30	404.41	438.71

ours). For parameters with ternary first-layer key distribution providing 192-bit security, the evaluation key size of our blind rotation is about 175 times smaller than FHEW/AP and 5.8 times smaller than TFHE/GINX; Our bootstrapping algorithm is about 3.7 times faster than FHEW/AP and 2.7 times faster than TFHE/GINX. Again, by using P192G with Gaussian key distribution, we can obtain an extra 17% improvement over P192T.

Finally, we note that the whole bootstrapping key consists of the evaluation key for blind rotation and the key-switching key for converting a second-layer ciphertext (obtained from blind rotation) back to a first-layer ciphertext. Moreover, we also note that all the algorithms need a key-switching key of $nNB_{ks}d_{ks}\log_2 Q_{ks}$ bits. In Table 5, one can see that the size of the bootstrapping keys for TFHE/GINX and ours is mainly dominated by the key-switching key size. However, even if we use a key-switching key of the same size (namely, P128T and P192T) as TFHE/GINX, our blind rotation still provides a reduction in the total bootstrapping key size by about 12.7% for 128-bit security and 24.4% for 192-bit security. More improvements are available when using Gaussian key distributions.

Acknowledgements We thank the anonymous reviewers of CRYPTO 2023 for their helpful comments and suggestions on earlier versions of our paper. Jiang Zhang, the corresponding author, is supported by the National Key Research and Development Program of China (Grant No. 2022YFB2702000), and by the National Natural Science Foundation of China (Grant Nos. 62022018, 61932019). Yi Deng is supported by the National Natural Science Foundation of China (Grant Nos. 61932019) and by Beijing Natural Science Foundation (Grant No. M22003).

References

1. Albrecht, M., Chase, M., Chen, H., Ding, J., Goldwasser, S., Gorbunov, S., Halevi, S., Hoffstein, J., Laine, K., Lauter, K., et al.: Homomorphic encryption standard. In: Protecting Privacy through Homomorphic Encryption, pp. 31–62. Springer (2021). https://doi.org/10.1007/978-3-030-77287-1_2
2. Albrecht, M.R.: On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In: EUROCRYPT 2017. pp. 103–129. Springer (2017). https://doi.org/10.1007/978-3-319-56614-6_4
3. Albrecht, M.R., Göpfert, F., Virdia, F., Wunderer, T.: Revisiting the expected cost of solving uSVP and applications to LWE. In: ASIACRYPT 2017. pp. 297–322. Springer (2017). https://doi.org/10.1007/978-3-319-70694-8_11
4. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology **9**(3), 169–203 (2015). https://doi.org/10.1515/jmc-2015-0016
5. Alperin-Sheriff, J., Peikert, C.: Faster bootstrapping with polynomial error. In: CRYPTO 2014. pp. 297–314. Springer (2014). https://doi.org/10.1007/978-3-662-44371-2_17
6. Badawi, A.A., Bates, J., Bergamaschi, F., Cousins, D.B., Erabelli, S., Genise, N., Halevi, S., Hunt, H., Kim, A., Lee, Y., Liu, Z., Micciancio, D., Quah, I., Polyakov, Y., R.V., S., Rohloff, K., Saylor, J., Suponitsky, D., Triplett, M., Vaikuntanathan, V., Zucca, V.: OpenFHE: Open-source fully homomorphic encryption library. Cryptology ePrint Archive, Paper 2022/915 (2022)
7. Bonte, C., Iliashenko, I., Park, J., Pereira, H.V., Smart, N.P.: Final: Faster fhe instantiated with NTRU and LWE pp. 188–215 (2022). https://doi.org/10.1007/978-3-031-22966-4_7
8. Bossuat, J., Mouchet, C., Troncoso-Pastoriza, J.R., Hubaux, J.: Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. In: EUROCRYPT 2021. vol. 12696, pp. 587–617. Springer (2021). https://doi.org/10.1007/978-3-030-77870-5_21
9. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical gapsvp. In: CRYPTO 2012. pp. 868–886. Springer (2012). https://doi.org/10.1007/978-3-642-32009-5_50
10. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: ITCS 2012. pp. 309–325. ACM (2012). https://doi.org/10.1145/2090236.2090262
11. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: CRYPTO 2011. pp. 505–524. Springer (2011). https://doi.org/10.1007/978-3-642-22792-9_29
12. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. SIAM Journal on computing **43**(2), 831–871 (2014). https://doi.org/10.1137/120868669
13. Chen, H., Chillotti, I., Song, Y.: Improved bootstrapping for approximate homomorphic encryption. In: EUROCRYPT 2019. pp. 34–54. Springer (2019). https://doi.org/10.1007/978-3-030-17656-3_2
14. Chen, H., Han, K.: Homomorphic lower digits removal and improved FHE bootstrapping. In: EUROCRYPT 2018. pp. 315–337. Springer (2018). https://doi.org/10.1007/978-3-319-78381-9_12
15. Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: Bootstrapping for approximate homomorphic encryption. In: EUROCRYPT 2018. pp. 360–384. Springer (2018). https://doi.org/10.1007/978-3-319-78381-9_14

16. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: ASIACRYPT 2017. pp. 409–437. Springer (2017). https://doi.org/10.1007/978-3-319-70694-8_15
17. Chillotti, I., Gama, N., Georgieva, M., Izabachene, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: ASIACRYPT 2016. pp. 3–33. Springer (2016). https://doi.org/10.1007/978-3-662-53887-6_1
18. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In: ASIACRYPT 2017. Lecture Notes in Computer Science, vol. 10624, pp. 377–408. Springer (2017). https://doi.org/10.1007/978-3-319-70694-8_14
19. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: Fast fully homomorphic encryption over the torus. Journal of Cryptology **33**(1), 34–91 (2020). https://doi.org/10.1007/s00145-019-09319-x
20. Coron, J.S., Mandal, A., Naccache, D., Tibouchi, M.: Fully homomorphic encryption over the integers with shorter public keys. In: CRYPTO 2011. pp. 487–504. Springer (2011). https://doi.org/10.1007/978-3-642-22792-9_28
21. Dijk, M.v., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: EUROCRYPT 2010. pp. 24–43. Springer (2010). https://doi.org/10.1007/978-3-642-13190-5_2
22. Ducas, L., Micciancio, D.: FHEW: bootstrapping homomorphic encryption in less than a second. In: EUROCRYPT 2015. vol. 9056, pp. 617–640. Springer (2015). https://doi.org/10.1007/978-3-662-46800-5_24
23. Ducas, L., van Woerden, W.: NTRU fatigue: How stretched is overstretched? In: ASIACRYPT 2021. pp. 3–32. Springer (2021). https://doi.org/10.1007/978-3-030-92068-5_1
24. Espitau, T., Joux, A., Kharchenko, N.: On a dual/hybrid approach to small secret LWE: A dual/enumeration technique for learning with errors and application to security estimates of fhe schemes. In: INDOCRYPT 2020. pp. 440–462. Springer (2020). https://doi.org/10.1007/978-3-030-65277-7_20
25. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive (2012)
26. Gama, N., Izabachene, M., Nguyen, P.Q., Xie, X.: Structural lattice reduction: generalized worst-case to average-case reductions and homomorphic cryptosystems. In: EUROCRYPT 2016. pp. 528–558. Springer (2016). https://doi.org/10.1007/978-3-662-49896-5_19
27. Genise, N., Gentry, C., Halevi, S., Li, B., Micciancio, D.: Homomorphic encryption for finite automata. In: ASIACRYPT 2019. pp. 473–502. Springer (2019). https://doi.org/10.1007/978-3-030-34621-8_17
28. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC 2009. pp. 169–178 (2009). https://doi.org/10.1145/1536414.1536440
29. Gentry, C., Halevi, S., Smart, N.P.: Better bootstrapping in fully homomorphic encryption. In: PKC 2012. vol. 7293, pp. 1–16. Springer (2012). https://doi.org/10.1007/978-3-642-30057-8_1
30. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: CRYPTO 2013. pp. 75–92. Springer (2013). https://doi.org/10.1007/978-3-642-40041-4_5
31. Ha, J., Kim, S., Lee, B., Lee, J., Son, M.: Rubato: Noisy ciphers for approximate homomorphic encryption. In: EUROCRYPT 2022. vol. 13275, pp. 581–610. Springer (2022). https://doi.org/10.1007/978-3-031-06944-4_20

32. Halevi, S., Shoup, V.: Bootstrapping for Helib. Journal of Cryptology **34**(1), 1–44 (2021). https://doi.org/10.1007/s00145-020-09368-7
33. Kim, A., Deryabin, M., Eom, J., Choi, R., Lee, Y., Ghang, W., Yoo, D.: General bootstrapping approach for RLWE-based homomorphic encryption. Cryptology ePrint Archive (2021)
34. Kirchner, P., Fouque, P.A.: Revisiting lattice attacks on overstretched NTRU parameters. In: EUROCRYPT 2017. pp. 3–26. Springer (2017). https://doi.org/10.1007/978-3-319-56620-7_1
35. Kluczniak, K.: NTRU-ν-um: Secure fully homomorphic encryption from NTRU with small modulus pp. 1783–1797 (2022). https://doi.org/10.1145/3548606.3560700
36. Lee, J.W., Lee, E., Lee, Y., Kim, Y.S., No, J.S.: High-precision bootstrapping of RNS-CKKS homomorphic encryption using optimal minimax polynomial approximation and inverse sine function. In: EUROCRYPT. pp. 618–647. Springer (2021). https://doi.org/10.1007/978-3-030-77870-5_22
37. Lee, Y., Micciancio, D., Kim, A., Choi, R., Deryabin, M., Eom, J., Yoo, D.: Efficient FHEW bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption pp. 227–256 (2023). https://doi.org/10.1007/978-3-031-30620-4_8
38. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: EUROCRYPT 2010. pp. 1–23. Springer (2010). https://doi.org/10.1007/978-3-642-13190-5_1
39. Micciancio, D., Polyakov, Y.: Bootstrapping in FHEW-like cryptosystems. In: Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography. pp. 17–28 (2021). https://doi.org/10.1145/3474366.3486924
40. Micciancio, D., Sorrell, J.: Ring packing and amortized FHEW bootstrapping **107**, 100:1–100:14 (2018). https://doi.org/10.4230/LIPIcs.ICALP.2018.100
41. Pereira, H.V.L.: Bootstrapping fully homomorphic encryption over the integers in less than one second. In: PKC 2021. pp. 331–359. Springer (2021). https://doi.org/10.1007/978-3-030-75245-3_13
42. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM) **56**(6), 1–40 (2009). https://doi.org/10.1145/1568318.1568324
43. Smart, N.P., Vercauteren, F.: Fully homomorphic encryption with relatively small key and ciphertext sizes. In: PKC 2010. pp. 420–443 2010. Springer (2010). https://doi.org/10.1007/978-3-642-13013-7_25
44. Son, Y., Cheon, J.H.: Revisiting the hybrid attack on sparse secret LWE and application to HE parameters. In: Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography. pp. 11–20 (2019). https://doi.org/10.1145/3338469.3358941
45. Stehlé, D., Steinfeld, R.: Faster fully homomorphic encryption. In: ASIACRYPT 2010. pp. 377–394. Springer (2010). https://doi.org/10.1007/978-3-642-17373-8_22

A Proof of Theorem 1

Proof. Note that by the definition of w_i and w'_i, we have that $w_i w'_{i+1} \in S \cup \{1\}$ for all $0 \leq i \leq n-1$ and that $X^{\frac{2N}{q}(\sum_{i=0}^{n-1} a_i s_i)} = X^{\sum_{i=0}^{n-1} w_i s_i - \sum_{i=0}^{n-1} s_i}$. Let $\hat{c}_i$ for

$0 \le i \le n-1$ be the value of ACC after evaluating the i-th loop in steps $7 \sim 10$ of Algorithm 1. Let $\hat{c}_n$ be the value of ACC in step 13 of Algorithm 1. For our purpose, it suffices to show that

$$\hat{c}_n = \mathrm{NTRU}_{Q,f,\tau,\Delta}(r(X^{\frac{2N}{q}}) \cdot X^{-\frac{2N}{q}b} X^{\sum_{i=0}^{n-1} w_i s_i - \sum_{i=0}^{n-1} s_i}).$$

We first show that for all $0 \le i \le n-1$, we have

$$\hat{c}_i = \mathrm{NTRU}_{Q,f,\tau,\Delta}(r(X^{\frac{2N}{q} w'_{i+1}}) \cdot X^{-\frac{2N}{q} b w'_{i+1}} X^{(\sum_{j=0}^{i} w_j s_j) w'_{i+1}}).$$

Note that $\mathsf{ACC} = \Delta \cdot r(X^{\frac{2N}{q} w'_0}) \cdot X^{-\frac{2N}{q} b w'_0}$ before entering the loop in steps $7 \sim 10$, and that $\mathbf{evk}_0 = \mathrm{NTRU}'_{Q,f,\tau}(X^{s_0}/f) = (\tau \cdot g_0/f + B^0 X^{s_0}/f, \cdots, \tau \cdot g_{d-1}/f + B^{d-1} X^{s_0}/f)$. By the property of external product, one can easily check that

$$c_0(X) = \mathsf{ACC} \odot \mathbf{evk}_0 = \mathrm{NTRU}_{Q,f,\tau,\Delta}(r(X^{\frac{2N}{q} w'_0}) \cdot X^{-\frac{2N}{q} b w'_0} X^{s_0}).$$

If $w_0 w'_1 = 1 \mod 2N$, then we immediately have

$$\hat{c}_0(X) = c_0(X) = \mathrm{NTRU}_{Q,f,\tau,\Delta}(r(X^{\frac{2N}{q} w'_1}) \cdot X^{-\frac{2N}{q} b w'_1} X^{w_0 s_0 w'_1}).$$

Otherwise, the algorithm will compute $\hat{c}_0(X) = \mathsf{EvalAuto}_{w_0 w'_1}(c_0(X), \mathbf{ksk}_{w_0 w'_1})$. By Lemma 2, we still have that

$$\hat{c}_0(X) = \mathrm{NTRU}_{Q,f,\tau,\Delta}(r(X^{\frac{2N}{q} w'_1}) \cdot X^{-\frac{2N}{q} b w'_1} X^{w_0 s_0 w'_1}).$$

Now, it is enough to show that if

$$\hat{c}_k = \mathrm{NTRU}_{Q,f,\tau,\Delta}(r(X^{\frac{2N}{q} w'_{k+1}}) \cdot X^{-\frac{2N}{q} b w'_{k+1}} X^{(\sum_{j=0}^{k} w_j s_j) w'_{k+1}}),$$

for some $0 \le k < n-1$, then $\hat{c}_{k+1}$ also has the same formula. Note that at the $(k+1)$-th loop, the algorithm will first compute $c_{k+1}(X) = \hat{c}_k \odot \mathbf{evk}_{k+1}$. Since $\mathbf{evk}_{k+1} = \mathrm{NTRU}'_{Q,f,\tau}(X^{s_{k+1}})$. By Lemma 1, we have that

$$c_{k+1} = \mathrm{NTRU}_{Q,f,\tau,\Delta}(r(X^{\frac{2N}{q} w'_{k+1}}) \cdot X^{-\frac{2N}{q} b w'_{k+1}} X^{(\sum_{j=0}^{k} w_j s_j) w'_{k+1} + s_{k+1}}).$$

Then, the algorithm will set $\hat{c}_{k+1} = c_{k+1}$ if $w_{k+1} w'_{k+2} = 1$, otherwise $\hat{c}_{k+1} = \mathsf{EvalAuto}_{w_{k+1} w'_{k+2}}(c_{k+1}(X), \mathbf{ksk}_{w_{k+1} w'_{k+2}})$. Again, by Lemma 2 we always have that

$$\hat{c}_{k+1} = \mathrm{NTRU}_{Q,f,\tau,\Delta}(r(X^{\frac{2N}{q} w'_{k+2}}) \cdot X^{-\frac{2N}{q} b w'_{k+2}} X^{(\sum_{j=0}^{k+1} w_j s_j) w'_{k+2}}).$$

After the loop, the algorithm will finally compute $\hat{c}_n = \hat{c}_{n-1} \odot \mathbf{ksk}_n$. Using the fact that $w'_n = 1$ and $\mathbf{evk}_n = \mathrm{NTRU}'_{Q,f,\tau}(X^{-\sum_{i=0}^{n-1} s_i})$, by Lemma 1 we have that

$$\hat{c}_n = \mathrm{NTRU}_{Q,f,\tau,\Delta}(r(X^{\frac{2N}{q}}) \cdot X^{-\frac{2N}{q}b} X^{\sum_{i=0}^{n-1} w_i s_i - \sum_{i=0}^{n-1} s_i}).$$

This finally completes the proof.

B Proof of Lemma 5

Proof. By definition, we have that $\mathrm{LWE}_{q,\mathbf{f}}(m) = (\hat{\mathbf{a}}, \hat{b} = \langle \hat{\mathbf{a}}, \mathbf{f} \rangle - (\hat{e} + \lfloor \frac{q}{t} \rceil m)) \in \mathbb{Z}_q^N \times \mathbb{Z}_q$ for some $\hat{\mathbf{a}} = (\hat{a}_0, \ldots, \hat{a}_N) \in \mathbb{Z}_q^N$, and that $\mathbf{LKSK}_{\mathbf{f},\mathbf{s},q,B_{ks}} = \{\mathbf{lksk}_{i,j,v}\}$ for some $\mathbf{lksk}_{i,j,v} = (\mathbf{a}_{i,j,v}, b_{i,j,v} = \langle \mathbf{a}_{i,j,v}, \mathbf{s} \rangle - e_{i,j,v} - vB_{ks}^j f_i)$. This lemma directly follows from the fact that $\hat{a}_i = \sum_{j=0}^{d_{ks}-1} v_{i,j} B_{ks}^j$ and that

$$
\begin{aligned}
\hat{b}' &= \sum_{i=0}^{N-1} \sum_{j=0, v_{i,j} \neq 0}^{d_{ks}-1} b_{i,j,v_{i,j}} + \hat{b} \\
&= \sum_{i=0}^{N-1} \sum_{j=0, v_{i,j} \neq 0}^{d_{ks}-1} (\langle \mathbf{a}_{i,j,v_{i,j}}, \mathbf{s} \rangle - e_{i,j,v_{i,j}} - v_{i,j} B_{ks}^j f_i) + \hat{b} \\
&= \langle \hat{\mathbf{a}}', \mathbf{s} \rangle - \sum_{i=0}^{N-1} \sum_{j=0, v_{i,j} \neq 0}^{d_{ks}-1} e_{i,j,v_{i,j}} - \sum_{i=0}^{N-1} \hat{a}_i f_i + \hat{b} \\
&= \langle \hat{\mathbf{a}}', \mathbf{s} \rangle - \sum_{i=0}^{N-1} \sum_{j=0, v_{i,j} \neq 0}^{d_{ks}-1} e_{i,j,v_{i,j}} - (\hat{e} + \left\lfloor \frac{q}{t} \right\rceil m)
\end{aligned}
$$

C Proof of Lemma 6

Proof. By definition, we have that $\mathrm{LWE}_{Q,\mathbf{f}}(m_i) = (\mathbf{a}_i, b_i = \langle \mathbf{a}_i, \mathbf{f} \rangle - \mathsf{noised}(m_i))$ for some $\mathbf{a}_i = (a_{i,0}, \ldots, a_{i,N-1}) \in \mathbb{Z}_Q^N$ and $\mathsf{noised}(m_i) = \tau \cdot e_i + \Delta \cdot m_i$ and that $\mathbf{RPK}_\tau = \{\mathbf{rpk}_{j,k}\}$ for some $\mathbf{rpk}_{j,k} = (\hat{a}_{j,k}, \hat{b}_{j,k})$ and $\hat{b}_{j,k} = \hat{a}_{j,k} s - \tau \cdot \hat{e}_{j,k} - B^k f_j$. By the fact that $u_j = \sum_{i=0}^{n-1} a_{i,j} X^i = \sum_{k=0}^{d-1} u_{j,k} B^k$, we have that

$$
\begin{aligned}
v &= \sum_{j=0}^{N-1} \sum_{k=0}^{d-1} u_{j,k} \hat{b}_{j,k} + \sum_{i=0}^{n-1} b_i X^i \\
&= \sum_{j=0}^{N-1} \sum_{k=0}^{d-1} u_{j,k} (\hat{a}_{j,k} s - (\tau \cdot \hat{e}_{j,k} + B^k f_j)) + \sum_{i=0}^{n-1} b_i X^i \\
&= us - \tau \cdot \sum_{j=0}^{N-1} \sum_{k=0}^{d-1} u_{j,k} \hat{e}_{j,k} - \sum_{j=0}^{N-1} \sum_{i=0}^{n-1} a_{i,j} f_j X^i + \sum_{i=0}^{n-1} b_i X^i \\
&= us - \tau \cdot \sum_{j=0}^{N-1} \sum_{k=0}^{d-1} u_{j,k} \hat{e}_{j,k} - \sum_{i=0}^{n-1} (\tau \cdot e_i + \Delta \cdot m_i) X^i \\
&= us - \tau \cdot e' - \Delta \cdot m
\end{aligned}
$$

where $e = \sum_{i=0}^{n-1} e_i X^i$, $m = \sum_{i=0}^{n-1} m_i X^i$ and $e' = \sum_{j,k} u_{j,k} \cdot \hat{e}_{j,k} + e$. This means that (u, v) is a ciphertext that encrypts m under the secret key $\mathbf{s} \in \hat{R}_Q$. Moreover, since $\|u_{j,k}\|_\infty < B$, we have that $Var(e') \leqslant nNd \frac{B^2}{12} Var(\hat{e}) + Var(e)$. This completes the proof.

D Optimized Parameters and Experimental Results

We have noted that the latest OpenFHE library [6] adopts approximate gadget decomposition techniques to optimize performance and storage, and also includes parameter optimizations. To ensure a fair comparison, we have implemented the same optimizations in our experiments and have also optimized the parameters by setting the standard deviation for NTRU secret key to 1.1. We also provide experimental comparisons with the LMKCDEY algorithm [37]. In Table 7,

Table 6. Optimized parameters for bootstrapping LWE-based first-layer ciphertexts..

Parameter set	Key distrib.	n	q	N	Q	B	Q_{ks}	B_{ks}
newSTD128 [6]	Ternary	503	1024	1024	2^{27}	2^{9}	2^{14}	2^{5}
LMKCDEY128 [37]	3.2	446	1024	1024	2^{28}	2^{10}	2^{13}	2^{5}
NewP128G	3.2	446	1024	1024	2^{21}	2^{7}	2^{13}	2^{5}

we present the experimental results. The evaluation environment is 12th Gen Intel(R) Core(TM) i9-12900H @2.50 GHz and 32 GB RAM, running Ubuntu 20.04.6 LTS.

Table 7. Timings and key sizes for bootstrapping (where the column "Timings" denotes the timing for performing a NAND gate followed by a bootstrapping, the column "EVK" denotes the evaluation key size for blind rotation; the column "KSK" denotes the key size for switching a second-layer ciphertext back to a first-layer one; the last column "Boots. key" denotes the whole bootstrapping key size.)

Algorithms	Parameters	Key distrib.	Timings (ms)	EVK (MB)	KSK (MB)	Boots. key (MB)
FHEW/AP [22,6]	newSTD128 [6]	Ternary	7.8	822.29	80.1	902.39
TFHE/GINX [19,6]	newSTD128 [6]	Ternary	7.6	26.53	80.1	106.63
LMKCDEY [37]	LMKCDEY128 [37]	Gaussian	5.8	12.35	65.97	78.32
Ours	NewP128G	Gaussian	4.7	7.54	65.97	73.51