# NTRU-based Bootstrapping for MK-FHEs without using Overstretched Parameters

Binwu Xiang[⋆1,2,3], Jiang Zhang[2(✉)], Kaixing Wang[2,4], Yi Deng[1,3], and Dengguo Feng[2]

[1] Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, CAS, Beijing, China.
`{xiangbinwu,deng}@iie.ac.cn`

[2] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China.
`zhangj@sklc.org,fengdg@263.net,wangkaixing22@mails.ucas.ac.cn`

[3] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China.

[4] School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, China.

**Abstract.** Recent attacks on NTRU lattices given by Ducas and van Woerden (ASIACRYPT 2021) showed that for moduli $q$ larger than the so-called fatigue point $n^{2.484+o(1)}$, the security of NTRU is noticeably less than that of (ring)-LWE. Unlike NTRU-based PKE with $q$ typically lying in the secure regime of NTRU lattices (i.e., $q < n^{2.484+o(1)}$), the security of existing NTRU-based multi-key FHEs (MK-FHEs) requiring $q = O(n^k)$ for $k$ keys could be significantly affected by those attacks.

In this paper, we first propose a (matrix) NTRU-based MK-FHE for super-constant number $k$ of keys without using overstretched NTRU parameters. Our scheme is essentially a combination of two components following the two-layer framework of TFHE/FHEW:

– a simple first-layer matrix NTRU-based encryption that naturally supports multi-key NAND operations with moduli $q = O(k \cdot n^{1.5})$ only linear in the number $k$ of keys;
– and a crucial second-layer NTRU-based encryption that supports an efficient hybrid product between a single-key ciphertext and a multi-key ciphertext for gate bootstrapping.

Then, by replacing the first-layer with a more efficient LWE-based multi-key encryption, we obtain an improved MK-FHE scheme with better performance. We also employ a light key-switching technique to reduce the key-switching key size from the previous $O(n^2)$ bits to $O(n)$ bits.

A proof-of-concept implementation shows that our two MK-FHE schemes outperform the state-of-the-art TFHE-like MK-FHE schemes in both computation efficiency and bootstrapping key size. Concretely, for $k = 8$ at the same 100-bit security level, our improved MK-FHE scheme can bootstrap a ciphertext in 0.54s on a laptop and only has a bootstrapping key of size 13.89MB, which are respectively 2.2 times

[⋆] This work was done while I was a visiting student in the group of Dr. Jiang Zhang during 2021-2024 at State Key Laboratory of Cryptology, Beijing, China.

faster and 7.4 times smaller than the MK-FHE scheme (which relies on a second-layer encryption from the ring-LWE assumption) due to Chen, Chillotti and Song (ASIACRYPT 2019).

## 1 Introduction

Multi-key Fully Homomorphic Encryption (MK-FHE) has emerged as a critical cryptographic primitive, facilitating secure computations on encrypted data contributed by multiple users in cloud environments. At STOC 2012, López-Alt, Tromer and Vaikuntanathan constructed the first MK-FHE scheme [27] which supported a-priori bounded number of keys. Clear and McGoldrick presented a GSW-type MK-FHE based on the Learning With Errors (LWE) problem [14]. Subsequently, Mukherjee and Wichs streamlined this approach and obtained a two-round Multi-Party Computation (MPC) protocol in the common random string model by proposing another LWE-based MK-FHE [30]. Both [30] and [14] are static in the sense that the keys involved in the homomorphic computation have to be determined at the beginning and do not support homomorphic computation on ciphertexts under new keys.

Peikert and Shiehian [31] introduced the concept of dynamic MK-FHE, where the resulting ciphertexts from previous homomorphic evaluations can be employed in subsequent homomorphic computations involving additional keys. A related concept, known as fully dynamic MK-FHE, was proposed by Brakerski and Perlman [8], which even does not need to know the total number of all possible keys during the setup. Chen, Chillotti and Song [9] proposed a dynamic TFHE-like MK-FHE scheme with efficient bootstrapping, but it requires a large number of bootstrapping keys (approximately 90MB for each party in the two-party case), limiting the applicability in resource-constrained environments such as the Internet of Things (IoT), blockchain, and GPU acceleration [32,34,29]. Recently, Kwak et al. [25] presented another variant of the TFHE-like MK-FHE scheme with asymptotically better computational efficiency but with much worse noise growth and larger bootstrapping keys (more than 214MB for each party in the two-party case).

Even if the first MK-FHE was based on NTRU lattices [27], the research on NTRU-based MK-FHE schemes was indeed suffered from the sublattice attacks in [20,2,10,23,16]. [5] In particular, the scheme in [27], adhering to the frameworks outlined in BGV [7] or BFV [18], had an error of magnitude about $\tilde{O}(n^{\tau k})$ and a moduli $q > \tilde{O}(n^{\tau k})$ for correctness, where $\tau$ is a small constant. However, Ducas and van Woerden [16] showed that the NTRU problem (including its matrix version [19]) with large moduli $q > n^{2.484+o(1)}$ and ternary secret could be significantly easier than its ring-LWE counterpart using similar parameters. Recently, Hough, Sandsbråten and Silde [21] showed that this conclusion still holds for Gaussian secret of NTRU with such overstretched parameter, which

[5] One way to avoid the sublattice attacks is to increase key size, but this can cause noise to escalate rapidly, potentially preventing even a single homomorphic multiplication. Therefore, ternary secret keys are typically preferred for efficiency.

**Table 1.** Experimental comparison between our MK-FHEs and two related ones [9,25].

| Scheme | Security | First layer | | Second layer | Runtime(s) | | | | Boot. Key(MB) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Assum. | Key Dist. | Assum. | $k = 2$ | $k = 4$ | $k = 8$ | $k = 16$ | $k = 2$ | $k = 4$ | $k = 8$ | $k = 16$ |
| CCS [9] | 100 | LWE | binary | RLWE | 0.07 | 0.33 | 1.19 | $\backslash$ | 89.82 | 96.38 | 102.94 | $\backslash$ |
| KMS [25] | 100 | LWE | binary | RLWE | 0.14 | 0.44 | 1.17 | 2.86 | 214.61 | 285.22 | 250.06 | 285.31 |
| Ours (Alg. 1) | 100 | MNTRU | ternary | NTRU | 0.07 | 0.28 | 0.82 | 2.74 | 38.31 | 38.31 | 38.31 | 38.31 |
| Ours (Alg. 2) | 100 | LWE | binary | NTRU | 0.05 | 0.21 | 0.54 | 2.61 | 13.89 | 13.89 | 13.89 | 13.89 |
| Ours (Alg. 1) | 128 | MNTRU | ternary | NTRU | 0.14 | 0.40 | 1.55 | 6.84 | 72.5 | 72.5 | 92.69 | 112.87 |
| Ours (Alg. 2) | 128 | LWE | binary | NTRU | 0.06 | 0.23 | 0.76 | 4.21 | 17.45 | 17.45 | 17.45 | 25.83 |

puts a crucial limitation on the number $k$ of keys in existing NTRU-based MK-FHEs [27,13]. Although the problem of obtaining MK-FHE with larger $k$ can be somehow eased by using very sparse secrets [36], the construction of NTRU-based MK-FHE schemes with super-constant $k = \omega(1)$ in the standard and secure parameter regime of NTRU lattices (e.g., uniform ternary keys with $q < n^{2.484+o(1)}$), to the best of our knowledge, is still open.

## 1.1 Our results

In this work, we first propose a (matrix) NTRU-based MK-FHE for super-constant number $k$ of keys without using overstretched NTRU parameters. Our construction basically consists of two components following the two-layer framework of the TFHE/FHEW scheme in [15,11] and the MK-FHE scheme in [9]: a simple first-layer encryption for homomorphic NAND operations and a more complex second-layer encryption for gate bootstrapping. Specifically, our first-layer is an encryption scheme based on the matrix version of the NTRU problem (a.k.a., MNTRU [19]) which naturally supports multi-key NAND operation with moduli $q = O(k \cdot n^{1.5})$ only linear in the number $k$ of keys, and can thus support a sub-linear number $k$ of keys for $q < n^{2.484+o(1)}$. Our second-layer is a new and crucial NTRU-based encryption (more precisely, a Hint-NTRU based encryption, see technical overview below) which supports an efficient hybrid product between a single-key ciphertext and a multi-key ciphertext for gate bootstrapping. Then, by using our second-layer encryption as a building block, we also present an improved MK-FHE scheme with better performance by replacing the first layer with an LWE-based multi-key encryption. We further reduce the key-switching key size from previous $O(n^2)$ to $O(n)$ bits by using a light key-switching technique. As the TFHE-like MK-FHE in [9], our proposed two MK-FHEs inherently support dynamic homomorphic computation on ciphertexts under new keys.

We implement our two schemes in experiment using the OpenFHE library [4]. In Table 1, we provide an experimental comparison of our schemes with the two MK-FHEs by Chen, Chillotti and Song (CCS) [9] and by Kwak, Min and Song (KMS) [25]. Since both CCS [9] and KMS [25] only choose parameters achieving 100-bit security, we select two sets of parameters achieving both 100-bit and 128-bit security for a fair comparison, where the concrete security of all set of parameters (as well as the ones in [9,25]) is estimated by using the LWE estimator in [3] and the NTRU estimator in [16] (see Sec. 7.2 for details). From Table 1 one can see that our two MK-FHE schemes are faster and have smaller bootstrapping keys than the MK-FHE schemes in [9,25]. The efficiency improvement over [9,25] is mainly because our NTRU-based second-layer encryption supports a more efficient hybrid product. One can also see that our first MK-FHE scheme with MNTRU-based first-layer ciphertexts (i.e., Alg. 1) is less efficient than our second one with LWE-based first-layer ciphertexts (i.e., Alg. 2). The main reason is that Alg. 1 uses uniform ternary keys which requires a more complex bootstrapping algorithm than that of Alg. 2 using binary keys (see the technical overview below for more details). For a concrete comparison at the same 100-bit security level, our Alg. 2 is about 1.4, 1.6 and 2.2 times faster than CCS [9], and 2.8, 2.1 and 2.2 times faster than KMS [25] for $k = 2$, 4 and 8, respectively. Correspondingly, the bootstrapping key size of our Alg. 2 is 6.5, 6.9 and 7.4 smaller than that of CCS [9], and is 15.5, 20.5 and 18 times smaller than that of KMS [25]. For $k = 16$, our Alg. 2 is about 1.1 times faster and 20.5 times smaller than KMS [25] (note that CCS [9] did not provide the parameters for $k = 16$).

### 1.2 Technical Overview

Before diving into the technical details of our constructions, we begin by recalling the two-layer framework of FHEW/TFHE in [15,11], which was later extended to the multi-key setting in [9]. Basically, the framework consists of two layers of different encryption schemes: the first layer only supports a single homomorphic NAND computation and the second layer is designed to bootstrap the first layer ciphertext to support more NAND operations (namely, the gate bootstrapping). For simplicity, we restricted our attention to the setting where the first layer is an LWE-based encryption, and the second layer is an RLWE-based encryption (namely, the whole scheme is essentially a hybrid scheme under two assumptions). Formally, let $n, q$ be the dimension and moduli of the LWE problem in the first layer, respectively. Let $\mathbf{z} \in \{0,1\}^n$ be the LWE secret of the first layer, and the ciphertext of the first layer has the form of $(b, \mathbf{a}) \in \mathbb{Z}_q^{n+1}$ such that $b + \langle \mathbf{a}, \mathbf{z} \rangle \approx \lfloor q/4 \rceil \cdot m$ for $m \in \{0,1\}$. Given two first-layer ciphertexts $ct_1 = (b_1, \mathbf{a}_1) \in \mathbb{Z}_q^{n+1}$ and $ct_2 = (b_2, \mathbf{a}_2) \in \mathbb{Z}_q^{n+1}$ encrypting $m_1 \in \{0,1\}$ and $m_2 \in \{0,1\}$, respectively, the homomorphic NAND operation can be simply done by computing $ct' = (b', \mathbf{a}') = (\frac{5q}{8}, \mathbf{0}) - ct_1 - ct_2$ such that $b' + \langle \mathbf{a}', \mathbf{z} \rangle \approx \lfloor q/2 \rceil \cdot m$ where $m = m_1 \bar{\wedge} m_2$. Moreover, let $N, Q$ be the dimension and moduli of the RLWE problem in the second layer such that $N$ is a power of two and $q = 2N$. Let $R = \mathbb{Z}[X]/(X^N + 1)$ and its quotient ring $R_Q = \mathbb{Z}_Q[X]/(X^N + 1)$. The second-layer RLWE-based encryption is basically a GSW-like encryption under secret key $s \in R$, which given an encryption of the first-layer secret key $\mathbf{z} \in \{0,1\}^n$ under $s \in R$, supports the homomorphic computation of $b' + \langle \mathbf{a}', \mathbf{z} \rangle \approx \lfloor q/2 \rceil \cdot m$ on the exponent of $X$ (namely, $X^{b' + \langle \mathbf{a}', \mathbf{z} \rangle}$) such that the modulo $q$ operation is free in $R_Q$ (this is because the order of $X$ in $R$ is exactly $q$ due to the choice of $q = 2N$). By multiplying a carefully designed polynomial $r(X) \in R_Q$ to $X^{b' + \langle \mathbf{a}', \mathbf{z} \rangle}$, one can exactly extract an encryption of

$m$ on the constant term of the resulting polynomial [6]. The above procedure is also known as blind rotation [15,11,26,35,5].

To extend the above framework to the multi-key setting, it suffices to design a first-layer encryption that supports multi-key NAND operation and a second-layer GSW-like encryption that supports multi-key blind rotation. Note that given a ciphertext $ct_1 = (b_1, \mathbf{a}_1) \in \mathbb{Z}_q^{n+1}$ under a single-key secret key $\mathbf{z}_1$, by appending $(k-1)n$ zeros one can easily obtain a ciphertext $\overline{ct}_1 = (b, \mathbf{a}_1, \cdots, \mathbf{a}_k) = (b_1, \mathbf{a}_1, \mathbf{0}, \cdots, \mathbf{0}) \in \mathbb{Z}_q^{kn+1}$ that encrypts the same message under the set of keys $\{\mathbf{z}_i\}_{1\le i\le k}$ because of $b + \langle \mathbf{a}_1, \mathbf{z}_1 \rangle = b + \sum_{i=1}^{k} \langle \mathbf{a}_i, \mathbf{z}_i \rangle$. This means that the multi-key NAND operation can be trivially done as in the single-key setting.

The multi-key blind rotation is more complex and non-trivial because we essentially need a way to generate the blind rotation evaluation key that encrypts a set of first-layer secrets $\{\mathbf{z}_i\}_{1\le i\le k}$ under a set of second-layer secrets $\{s_i\}_{1\le i\le k}$ on the fly (namely, the set of keys involved in the computation is not known in the setup phase). To handle this, Chen et al. [9] presented an RGSW-like scheme (called uni-encryption) supporting the hybrid product between an MK-RLWE ciphertext and the uni-encryption. Specifically, for a base $B \in \mathbb{Z}$ and $d = \lceil \log_B Q \rceil$, the uni-encryption $\mathsf{UniEnc}(\mu, s) = (\mathbf{d}, \mathbf{f}_0, \mathbf{f}_1) \in R_Q^d \times R_Q^d \times R_Q^d$ that encrypts $\mu \in R_Q$ under the secret $s \in R$ is defined as

$$\mathbf{d} = r \cdot \mathbf{a} + \mu \cdot \mathbf{g} + \mathbf{e}_1 \in R_Q^d, \qquad \mathbf{f}_0 = -s \cdot \mathbf{f}_1 + r \cdot \mathbf{g} + \mathbf{e}_2 \in R_Q^d,$$

where $\mathbf{a} \in R_Q^d$ is a Common Reference String (CRS), $\mathbf{g} = \left[B^0, \ldots, B^{d-1}\right] \in \mathbb{Z}^d$ is a gadget vector, $r \in R_Q$ is a random polynomial and $\mathbf{f}_1, \mathbf{e}_1, \mathbf{e}_2 \in R_Q^d$ are $d$-dimension polynomials in $R_Q$.

To perform the multi-key blind rotation for an LWE-based first-layer ciphertext $\overline{ct}' = (b', \mathbf{a}'_1, \cdots, \mathbf{a}'_k)$ under the keys $\{\mathbf{z}_i\}_{1\le i\le k}$ for some $\mathbf{a}'_i = (a'_{i,j})_{0\le j<n}$ and $\mathbf{z}_i = (z_{i,j})_{0\le j<n}$, each party (of index $i$) independently generates a set of uni-encryption ciphertexts $\{\mathsf{UniEnc}(z_{i,j}, s_i)\}_{0\le j<n}$ that encrypts $\mathbf{z}_i \in \{0,1\}^n$ under the secret key $s_i \in R_Q$ as the evaluation key. Let

$$\mathsf{CMUX}(z_{i,j}) = \mathbf{1} + (X^{a'_{i,j}} - 1) \cdot \mathsf{UniEnc}(z_{i,j}, s_i)$$

for binary $z_{i,j}$. Since $\mathbf{1}$ can be designed as a noiseless uni-encryption of one and the uni-encryption ciphertext also has homomorphic properties (see Lemma 2), we have $\mathsf{CMUX}(z_{i,j}) = \mathsf{UniEnc}(X^{a'_{i,j} z_{i,j}}, s_i)$. Then, we can initialize the accumulator as a trivial MK-RLWE encryption $\mathsf{ACC}_{in} = (r(X)X^{b'}, \mathbf{0}) \in R_Q^{k+1}$ and recursively compute

$$\mathsf{ACC}_{out} = \mathsf{ACC}_{in} \cdot \prod_{i=1}^{k} \prod_{j=0}^{n-1} \mathsf{CMUX}(z_{i,j}).$$

By the fact that the order of $X$ is exactly $q$ for $q = 2N$, one can check that the output $\mathsf{ACC}_{out}$ is an MK-RLWE ciphertext that encrypts $r(X) \cdot X^{b' + \sum_{i=1}^{k} \langle \mathbf{a}'_i, \mathbf{z}_i \rangle}$

[6] The details on how to set such $r(X)$ can be found in [12,9,5,35]. We omit these details in this paper.

under the secret $(1, s_1, \cdots, s_k) \in R^{k+1}$. And by carefully designing the rotation polynomial $r(X)$, the message in the constant term can be exactly $\lfloor Q/4 \rceil \cdot m$. Finally, by using the sample extraction as in [22], we can extract an MK-LWE encryption $\overline{ct}'' = (b'', \mathbf{a}''_1, \ldots, \mathbf{a}''_k) \in \mathbb{Z}_Q^{kN+1}$ that encrypts $\lfloor Q/4 \rceil \cdot m$ under the secret $(\mathbf{s}_i)_{i\in[k]} \in \mathbb{Z}^{kN}$ where $\mathbf{s}_i$ is the coefficient vector of $s_i$. Then by performing the multi-key LWE modulus-switching from $Q$ to $q$ and key-switching from $(\mathbf{s}_i)_{i\in[k]} \in \mathbb{Z}^{kN}$ to $(\mathbf{z}_i)_{i\in[k]} \in \mathbb{Z}^{kn}$, we finish the bootstrapping and get a refreshed MK-LWE ciphertext.

As sketched above, in order to obtain a (matrix) NTRU-based MK-FHE, it suffices to design a first-layer MNTRU-based encryption that supports multi-key NAND operation and a second-layer NTRU-based GSW-like encryption that supports efficient hybrid product for multi-key blind rotation.

*First-layer Matrix NTRU-based Multi-Key Encryption.* We begin by first recalling the single-key MNTRU ciphertext in [5]. Formally, let $n, q$ be two positive integers. The single-key MNTRU ciphertext that encrypts a message bit $m \in \{0,1\}$ in [5] is defined as $\mathbf{c} := (\mathbf{e} + \lfloor \frac{q}{4} \rceil \cdot \mathbf{m}) \cdot \mathbf{F}^{-1} \in \mathbb{Z}_q^n$ where the secret key $\mathbf{F} \in \mathbb{Z}^{n\times n}$ is an invertible matrix, $\mathbf{e} \in \mathbb{Z}_q^n$ is a random vector from $\mathbb{Z}_q^n$, and $\mathbf{m} = (m, 0, \ldots, 0) \in \mathbb{Z}^n$. Note that this can be actually viewed as a standard MNTRU ciphertext that encrypts a message vector $\lfloor \frac{q}{4} \rceil \cdot \mathbf{m} \cdot \mathbf{F}^{-1} = (*, 0, \ldots, 0) \in \mathbb{Z}^n$ with a single non-zero term, whose security can be based on either a KDM-form MNTRU assumption (which essentially assumes that the standard MNTRU encryption is KDM-secure as in [35,5], we refer to Sec. 2.4 for a formal definition), or the matrix inhomogeneous NTRU (MiNTRU) assumption with a more complex distribution for error $\mathbf{e}$ in [19]. We prefer to the former assumption because a circular-secure/KDM-secure assumption is common for constructing FHEs with bootstrapping (e.g., [35,5]), and designing FHEs without circular-secure/KDM-secure assumptions is actually a long-term open problem. We now extend the above single-key ciphertext to the multi-key setting, and naturally define the multi-key MNTRU ciphertext as a vector of the form $\overline{ct} = (\mathbf{c}_1, \cdots, \mathbf{c}_k) \in \mathbb{Z}_q^{kn}$ such that

$$\langle \mathbf{c}_1, \mathrm{col}_0(\mathbf{F}_1) \rangle + \cdots + \langle \mathbf{c}_k, \mathrm{col}_0(\mathbf{F}_k) \rangle = \left\lfloor \frac{q}{4} \right\rceil \cdot m + e$$

where $k$ denotes the number of involved keys, $e$ is a small noise and $\mathrm{col}_0(\mathbf{F}_i) = (f_{i,j})_{0\le j<n}$ is the first column of the secret matrix $\mathbf{F}_i$.

Since a constant cannot be treated as a noiseless ciphertext in our MNTRU-based multi-key encryption (which is unlike the LWE-based one in [9]), we have to create an additional evaluation key $evk_i = (\mathbf{e}_i + \lfloor \frac{5q}{8} \rceil \cdot (1, \mathbf{0})) \cdot \mathbf{F}_i^{-1} \in \mathbb{Z}_q^n$ encrypting the constant $\frac{5q}{8}$ for performing the NAND gate evaluation. Now, given two multi-key MNTRU ciphertexts $\overline{ct}_1 = (\mathbf{c}_1, \cdots, \mathbf{c}_{k_1}) \in \mathbb{Z}_q^{k_1 n}$ and $\overline{ct} = (\mathbf{c}_1, \cdots, \mathbf{c}_{k_2}) \in \mathbb{Z}_q^{k_2 n}$, let $k$ be the maximum number of different keys involved in $\overline{ct}_1$ and $\overline{ct}_2$. By reorganizing the components and padding empty slots with zeros, we can extend the ciphertexts $\overline{ct}_1$ and $\overline{ct}_2$ to $\overline{ct}'_1, \overline{ct}'_2 \in \mathbb{Z}^{kn}$ under the secret

$(\mathbf{F}_j)_{j\in[k]}$. Then, the NAND gate operation can be done by computing

$$\overline{ct}' = (\underbrace{\mathbf{0},\cdots,\mathbf{0}}_{(i-1)n\ \text{zeros}}, evk_i, \underbrace{\mathbf{0},\cdots,\mathbf{0}}_{(k-i)n\ \text{zeros}}) - \overline{ct}'_1 - \overline{ct}'_2 \in \mathbb{Z}_q^{kn}.$$

*Second-layer NTRU-based Uni-Encryption.* Now, we give our NTRU-based uni-encryption scheme that supports efficient hybrid product for multi-key blind rotation. Let $B, d$ be two integers and $d = \lceil \log_B Q \rceil$. Specifically, each party $i$ takes a uniformly random CRS $\mathbf{a} \in R_Q^d$ and sets the public key $\mathbf{b}_i = -\mathbf{a} \cdot s_i + \mathbf{e}_i \in R_Q^d$ where $s_i$ is the secret of the second layer for party $i$, $\mathbf{e}_i$ is a $d$-dimensional polynomial vectors. Party $i$ can encrypt a plaintext $\mu_i \in R_Q$ into a uni-encryption ciphertext $\mathsf{UniEnc}(\mu_i, s_i) = (\mathbf{d}_i, \mathbf{f}_i) \in R_Q^d \times R_Q^d$ under secret key $s_i$ such that

$$\mathbf{d}_i = r_i \cdot \mathbf{a} + \mu_i \cdot \mathbf{g} + \mathbf{e}_{i,1} \in R_Q^d, \qquad \mathbf{f}_i = \mathbf{e}_{i,2} \cdot s_i^{-1} + r_i \cdot \mathbf{g} \cdot s_i^{-1} \in R_Q^d,$$

where $r_i$ is a polynomial with small coefficients, $\mathbf{e}_{i,1}, \mathbf{e}_{i,2}$ are $d$-dimensional polynomial vectors and $\mathbf{g} = \left[B^0, \ldots, B^{d-1}\right] \in \mathbb{Z}^d$ is a gadget vector. Note that our NTRU-based uni-encryption ciphertext only contains two polynomials in $R_Q$ (while the MK-FHE scheme [9] and its variant [25] have to store three polynomials), which allows us to save both computation and storage for a hybrid product. Because the same $s_i$ is used in both $\mathbf{b}_i = -\mathbf{a} \cdot s_i + \mathbf{e}_i$ and $\mathbf{f}_i = \mathbf{e}_{i,2} \cdot s_i^{-1} + r_i \cdot \mathbf{g} \cdot s_i^{-1}$, the above construction essentially relies on the hardness of a KDM-secure assumption and a variant of the NTRU problem called Hint-NTRU [17]. As discussed in [17], with a suitable choice of secret and error distributions, the Hint-NTRU problem is at least as hard as the inhomogeneous NTRU problem [19].

The single-key KDM-form NTRU ciphertext that encrypts a message $\mu \in R_Q$ in [35] is defined as $c := (g + \mu)/s$ where the secret key $s \in R$ is an invertible polynomial, $g \in R_Q$ is a random polynomial from $R_Q$. We can extend the above single-key ciphertext to the multi-key setting, and naturally define the multi-key NTRU (MK-NTRU) ciphertext under the secret $(s_i)_{i\in[k]}$ as a vector of the form $(c_i)_{i\in[k]} \in R_Q^k$ such that $\sum_{i=1}^k c_i s_i = g' + \mu$ where $k$ denotes the number of involved keys, $g'$ is a small noise. To homomorphically multiply the MK-NTRU ciphertext by a uni-encryption that encrypts $\mu_i$ of party $i$, we compute the following inner products: $u_j = \left\langle \mathbf{g}^{-1}(c_j), \mathbf{d}_i \right\rangle$ for $1 \leqslant j \leqslant k$, and compute $v = \sum_{j=1}^k \left\langle \mathbf{g}^{-1}(c_j), \mathbf{b}_j \right\rangle$. Then we output $\bar{\mathbf{c}}' = (c'_1, \ldots, c'_k) \in R_Q^k$ where $c'_i = u_i + \left\langle \mathbf{g}^{-1}(v), \mathbf{f}_i \right\rangle$ and $c'_j = u_j$ for $j \neq i$. One can check that $\bar{\mathbf{c}}'$ is an MK-NTRU ciphertext that encrypts $\mu \cdot \mu_i$ (we defer the proof and security analysis to Sec. 4). It is important to note that our MK-NTRU ciphertext consists of $k$ polynomials in $R_Q$, which is one less than that of the MK-RLWE ciphertext in [9,25]. This means our hybrid product only needs $d(2k+1)$ multiplications in $R_Q$, which is $2d$ less than that of the MK-FHE schemes in [9,25].

*Bootstrapping First-layer matrix NTRU-based Multi-Key Ciphertexts.* Now, we show how to bootstrap first-layer matrix NTRU-based multi-key ciphertexts.

Formally, Given a multi-key MNTRU ciphertext $\overline{ct}' = (\mathbf{c}_1', \cdots, \mathbf{c}_k') \in \mathbb{Z}_q^{kn}$ satisfying $\sum_{i=1}^{k} \langle \mathbf{c}_i', \mathrm{col}_0(\mathbf{F}_i) \rangle \approx \left\lfloor \frac{q}{2} \right\rceil m$ for some $\mathbf{c}_i' = (c_{i,j}')_{0 \le j < n}$ where $\mathrm{col}_0(\mathbf{F}_i) = (f_{i,j})_{0 \le j < n} \in \{-1, 0, 1\}^n$ is the first column of the secret matrix $\mathbf{F}_i$, the gate bootstrapping consists of three steps: blind rotation, modulus switching and key-switching. Recall the previous multi-key blind rotation is to homomorphically decrypt an MK-LWE ciphertext with a binary secret key distribution on the exponent [9,25]. One issue we encounter is that in our MNTRU scheme, the secret key follows a ternary distribution. Therefore, we need to extend this method to accommodate ternary secret key distributions. To solve this issue, we employ the ternary CMUX gate $1 + \left(X^{c_{i,j}'} - 1\right) \cdot f_{i,j}^{+} + \left(X^{-c_{i,j}'} - 1\right) \cdot f_{i,j}^{-} = X^{c_{i,j}' f_{i,j}}$ first used in [5] where

$$\begin{cases} f_{i,j}^{+} = 1, & \text{if } f_{i,j} = 1 \\ f_{i,j}^{+} = 0, & \text{otherwise} \end{cases}, \quad \begin{cases} f_{i,j}^{-} = 1, & \text{if } f_{i,j} = -1 \\ f_{i,j}^{-} = 0, & \text{otherwise} \end{cases} \quad \text{for } 0 \le j < n.$$

Another issue is that in previous methods [9] the accumulator can be initialized as a trivial MK-RLWE encryption. However, this feature is not satisfied for our MK-NTRU ciphertext. Noticed that if we set the initial accumulator as $\mathsf{ACC}_{in} = (r(X), \mathbf{0}) \in R_Q^k$ for some rotation polynomial $r(X) \in R_Q$, we have the fact $\langle \mathsf{ACC}_{in}, \mathbf{s} \rangle = r(X) s_1$ where $\mathbf{s} = (s_1, \cdots, s_k) \in R^k$ is the secret keys of $k$ keys used in hybrid product. Fortunately, we can effectively address this challenge by carefully designing the evaluation key. Specifically, party $i$ creates a set of ciphertexts as follows:

- For $i = 1$, given secret key $\mathrm{col}_0(\mathbf{F}_1) = (f_{1,0}, \ldots, f_{1,n-1}) \in \{-1, 0, 1\}^n$, create a set of ciphertexts that encrypts $\mathrm{col}_0(\mathbf{F}_1)$ under $s_1$ as follows:

$$\begin{cases} \mathbf{evk}_{1,0}^{+} = \mathsf{UniEnc}(f_{1,0}^{+}/s_1, s_1) \\ \mathbf{evk}_{1,0}^{-} = \mathsf{UniEnc}(f_{1,0}^{-}/s_1, s_1) \end{cases}, \quad \begin{cases} \mathbf{evk}_{1,j}^{+} = \mathsf{UniEnc}(f_{1,j}^{+}, s_1) \\ \mathbf{evk}_{1,j}^{-} = \mathsf{UniEnc}(f_{1,j}^{-}, s_1) \end{cases} \quad \text{for } 1 \le j < n,$$

- For $i \neq 1$, given secret key $\mathrm{col}_0(\mathbf{F}_i) = (f_{i,0}, \ldots, f_{i,n-1}) \in \{-1, 0, 1\}^n$, create a set of ciphertext that encrypts $\mathrm{col}_0(\mathbf{F}_i)$ under $s_i$ as follows:

$$\mathbf{evk}_{i,j}^{+} = \mathsf{UniEnc}(f_{i,j}^{+}, s_i), \mathbf{evk}_{i,j}^{-} = \mathsf{UniEnc}(f_{i,j}^{-}, s_i) \text{ for } 0 \le j < n.$$

Recall that in the previous scheme [9,25], the uni-encryption ciphertexts can be publicly generated even when the plaintext is known. This feature is crucial for the CMUX gate evaluation. But our KDM-form evaluation keys $\mathbf{evk}_{1,0}^{+}$ and $\mathbf{evk}_{1,0}^{-}$ given above don't satisfy this requirement. We address this issue with minimal additional cost by additionally constructing an auxiliary ciphertext $\mathbf{evk}_{1,0}^{*} = \mathsf{UniEnc}(1/s_1, s_1)$. In this case, the CMUX gate in the first iteration can be computed as

$$\mathsf{CMUX}(f_{1,0}) = \mathbf{evk}_{1,0}^{*} + \left(X^{c_{1,0}'} - 1\right) \cdot \mathbf{evk}_{1,0}^{+} + \left(X^{-c_{1,0}'} - 1\right) \cdot \mathbf{evk}_{1,0}^{-},$$

which is a uni-encryption of $X^{c'_{1,0}f_{1,0}}/s_1$. In other iterations, we slightly revised the CMUX gate as

$$\mathsf{CMUX}(f_{i,j}) = \mathbf{1} + \left(X^{c'_{i,j}} - 1\right) \cdot \left(\mathbf{evk}^{+}_{i,j} - X^{-c'_{i,j}} \cdot \mathbf{evk}^{-}_{i,j}\right),$$

where $\mathbf{1}$ is a noiseless uni-encryption of one.

We now describe our bootstrapping algorithm. Firstly, we initialize the accumulator as $\mathsf{ACC}_{in} = (r(X), \mathbf{0}) \in R_Q^k$ and compute the hybrid product of $\mathsf{ACC}_{in}$ and $\mathsf{CMUX}(f_{1,0})$ to obtain $\mathsf{ACC}_{1,0}$. Since $\mathsf{CMUX}(f_{1,0}) = \mathsf{UniEnc}(X^{c'_{1,0}f_{1,0}}/s_1, s_1)$, one can easily check that $\mathsf{ACC}_{1,0}$ is an MK-NTRU ciphertext that encrypts $r(X) \cdot X^{c'_{1,0}f_{1,0}}$. Next we compute $\mathbf{evk}'_{1,1} = \mathbf{evk}^{+}_{1,1} - \mathbf{evk}^{-}_{1,1} \cdot X^{-c'_{1,1}}$ to obtain a uni-encryption for $f^{+}_{1,1} - f^{-}_{1,1} \cdot X^{-c'_{1,1}}$. Then we compute the hybrid product between $(X^{c'_{1,1}} - 1)\mathsf{ACC}_{1,0}$ and $\mathbf{evk}'_{1,1}$ , and $\mathsf{ACC}_{1,0}$ to obtain $\mathsf{ACC}_{1,1}$. We can check that $\mathsf{ACC}_{1,1}$ is an MK-NTRU ciphertext that encrypts

$$r(X) \cdot X^{c'_{1,0}f_{1,0}} \left(1 + (X^{c'_{1,1}} - 1) \cdot (f^{+}_{1,1} - f^{-}_{1,1} \cdot X^{-c'_{1,1}})\right) = r(X) \cdot X^{c'_{1,0}f_{1,0} + c'_{1,1}f_{1,1}}.$$

We can iteratively absorb $X^{c'_{i,j}f_{i,j}}$ for $i \in [k]$ and $0 \le j < n$ to obtain an MK-NTRU ciphertext $\mathsf{ACC}_{k,n-1}$ that encrypts $r(X) \cdot X^{\sum_{i=1}^{k}\sum_{j=0}^{n-1} c'_{i,j}f_{i,j}}$. Furthermore, by carefully designing $r(X)$ we can ensure that the constant term of $r(X) \cdot X^{\sum_{i=1}^{k}\sum_{j=0}^{n-1} c'_{i,j}f_{i,j}}$ equals $\left\lfloor \frac{Q}{4} \right\rceil \cdot m$.

Another issue is that after multi-key blind rotation, we obtain an MK-NTRU ciphertext under the modulus $Q$ instead of a multi-key MNTRU-based first-layer ciphertext. Therefore, we use the modulus switching technique to switch the modulus from $Q$ back to $q$ and we design a key-switching method to switch the MK-NTRU ciphertext back to MNTRU first-layer ciphertext, resulting in a refreshed multi-key MNTRU ciphertext of encryption $\left\lfloor \frac{q}{4} \right\rceil \cdot m$, which is convenient for the next NAND gate computation.

*Bootstrapping First-layer Multi-Key LWE Ciphertexts.* Recall that the multi-key LWE ciphertext (after NAND gate) that encrypts a message bit $m \in \{0, 1\}$ in [9] under the secret $(1, \mathbf{z}_1, \cdots, \mathbf{z}_k) \in \mathbb{Z}^{kn+1}$ is defined as a vector of the form $\overline{ct}' = (b', \mathbf{a}'_1, \ldots, \mathbf{a}'_k) \in \mathbb{Z}_q^{kn+1}$ such that $b' + \sum_{i=1}^{k} \langle \mathbf{a}'_i, \mathbf{z}_i \rangle \approx \lfloor q/2 \rceil \cdot m$ for some $\mathbf{a}'_i = (a'_{i,j})_{0 \leqslant j < n} \in \mathbb{Z}_q^n$. Our second-layer encryption scheme can also be modified to bootstrap such a standard first-layer multi-key LWE ciphertext $\overline{ct}'$. Similarly, it requires us to carefully design the evaluation key. Details can be found in Sec. 6 and we skip it here.

After blind rotation, we get an MK-NTRU ciphertext $\mathbf{c} = (c_1, \cdots, c_k) \in R_Q^k$ under the secret key $(s_1, \cdots, s_k) \in R^k$ instead of an MK-LWE ciphertext. Then we can extract a multi-key LWE ciphertext $\overline{ct}'' = (0, \mathbf{a}''_1, \cdots, \mathbf{a}''_k) \in \mathbb{Z}_Q^{kN+1}$ under the secret key $(1, \mathbf{s}_1, \cdots, \mathbf{s}_k) \in \mathbb{Z}^{kN+1}$ where $\mathbf{s}_i$ is the coefficient vector of $s_i$. Subsequently, by performing a modulus switching, we can switch the modulus from $Q$ back to $q$.

Finally, we only need to perform one key-switching to switch the secret key from $(1, \mathbf{s}_1, \cdots, \mathbf{s}_k) \in \mathbb{Z}^{kN+1}$ back to the secret key $(1, \mathbf{z}_1, \cdots, \mathbf{z}_k) \in \mathbb{Z}^{kn+1}$. However, in previous key-switching methods [9,25] the key-switching keys are substantially larger than the evaluation keys. For example, the evaluation key measures 19.7 MB, whereas the key-switching keys occupy a much larger 70.1 MB of storage space in [9]. This size discrepancy presents a significant challenge and warrants attention. To solve this issue, we propose a light key-switching method. Let $B_{ks}$ be an integer, $d_{ks} = \lceil \log_{B_{ks}} q \rceil$. To switch an MK-LWE ciphertext under the secret $(1, \mathbf{s}_1, \cdots, \mathbf{s}_k) \in \mathbb{Z}^{kN+1}$ to the secret $(1, \mathbf{z}_1, \cdots, \mathbf{z}_k) \in \mathbb{Z}^{kn+1}$, the previous method is to create a set of LWE ciphertexts that encrypts $vB_{ks}^l s_{i,j}$ under $\mathbf{z}_i$ for party $i$ where $i \in [k], j \in [0, \cdots, N-1], l \in [0, \cdots, d_{ks}-1], v \in [1, \cdots, B_{ks}-1]$. Instead, we pack these LWE ciphertexts of party $i$ into a small number of RLWE ciphertexts, and the server can extract them as LWE ciphertexts almost for free using sample extraction. This approach reduces the key-switching key size from $\tilde{O}(kn^2)$ bits to $\tilde{O}(kn)$ bits, which can be of independent interest to other schemes (e.g., the single-key FHEs [15,11,12,26,35]) as well.

### 1.3 Organization

After giving some background in Sec. 2, we present a multi-key matrix NTRU-based encryption scheme in Sec. 3. In Sec. 4, we present our new hybrid product. We show how to use our new hybrid product to bootstrap the multi-key matrix NTRU-based and LWE-based ciphertexts in Sec. 5 and Sec. 6, respectively. Finally, we report our implementation in Sec. 7.

## 2 Preliminaries

### 2.1 Notation

The set of real numbers (integers) is denoted by $\mathbb{R}$ ($\mathbb{Z}$, resp.). Vectors and matrices are denoted as lowercase bold letters (e.g. $\mathbf{a}$, $\mathbf{b}$) and uppercase bold letters (e.g. $\mathbf{A}$, $\mathbf{B}$), respectively. The $i+1$-th column of a matrix $\mathbf{A}$ is denoted by $\mathrm{col}_i(\mathbf{A})$ and $\mathbf{A}_{i,j}$ denotes the element in the $i$-th row and $j$-th column.

The inner product of two vectors $\mathbf{a}$ and $\mathbf{b}$ is denoted by the symbol $\langle \mathbf{a}, \mathbf{b} \rangle$. The index set $[k]$ represents the set of integers $\{1, 2, ..., k\}$. For positive integers $q, Q$ and power of two $N$, by $R$ and $R_q$ (resp., $R_Q$) we denote the $2N$-th cyclotomic ring $R = \mathbb{Z}[X]/(X^N+1)$ and its quotient ring $R_q = R/qR$ (resp., $R_Q = R/QR$).

For polynomial $a = \sum_{i=0}^{N-1} a_i X^i \in R$, we use $\mathsf{Cof}(a)$ to denote the coefficient vector of $a$, and we denote the $N$-dimensional anti-circulant matrix of $a$ by

$$\mathcal{A}(a) := \begin{pmatrix} a_0 & a_1 & \cdots & a_{N-1} \\ -a_{N-1} & a_0 & \cdots & a_{N-2} \\ \vdots & \ddots & \ddots & \vdots \\ -a_1 & -a_2 & \cdots & a_0 \end{pmatrix} = \begin{pmatrix} (\mathsf{Cof}(a)) \\ (\mathsf{Cof}(a \cdot X)) \\ \vdots \\ (\mathsf{Cof}(a \cdot X^{N-1})) \end{pmatrix}.$$

We use $\leftarrow$ to denote sampling an element uniformly at random from some distribution. By $\|\cdot\|$ and $\|\cdot\|_\infty$, we denote the $\ell_2$ and $\ell_\infty$ norms. By $\lceil\cdot\rceil$, $\lfloor\cdot\rfloor$ and $\lfloor\cdot\rceil$ we denote the ceiling, floor and round function, respectively.

## 2.2 Gadget Decomposition

For integers $q$, $B$, let $d = \lceil \log_q B \rceil$ and $\mathbf{g}_{q,B} = \left[B^0, \ldots, B^{d-1}\right] \in \mathbb{Z}^d$ be a gadget vector. We write $\mathbf{g}$ when $q$ and $B$ are clear from the context. For an integer $k \geq 1$, the gadget matrix is defined as

$$\mathbf{G}_k = \mathbf{I}_k \otimes \mathbf{g}^t = \begin{bmatrix} \mathbf{g}^t & 0 & \ldots & 0 \\ 0 & \mathbf{g}^t & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \mathbf{g}^t \end{bmatrix} \in \mathbb{Z}^{dk \times k}.$$

For any $a \in \mathbb{Z}$, the gadget decomposition of $a$ is defined as its signed decomposition in base $B$ as $\mathbf{g}^{-1}(a) = (a_0, \cdots, a_{d-1})$ with each $a_i \in (-B/2, B/2]$ such that $a = \langle \mathbf{g}^{-1}(a), \mathbf{g} \rangle$. The definition can be naturally extended to any polynomial in $R_q$. For any $a = \sum_{i=0}^{N-1} a_i X^i$, we define $\mathbf{g}^{-1}(a) = \sum_{i=0}^{N-1} \mathbf{g}^{-1}(a_i) X^i$.

In practice, we can ignore the first element of $\mathbf{g}^{-1}(a)$ and $\mathbf{g}$ to obtain $\mathbf{g}^{-1}(a) = (a_1, \cdots, a_{d-1})$ which satisfies $a \approx \langle \mathbf{g}^{-1}(a), \mathbf{g} \rangle$, thereby achieving optimization in both computation and storage.

## 2.3 Multi-key fully homomorphic encryption

Let $\mathcal{M}$ be the message space with arithmetic structure. Let $k$ be the bound of the involved keys. A multi-key FHE scheme is a tuple of PPT algorithms ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}$) having the following properties:

- $\mathsf{Setup}\left(1^\lambda\right)$ : Given the security parameter $\lambda$, outputs a public parameter $pp$.
- $\mathsf{KeyGen}(pp)$: Outputs a public key $pk$ and secret key $sk$.
- $\mathsf{Enc}(m, pk)$: Given the public key $pk$ and a message $m \in \mathcal{M}$, output a ciphertext $ct$. For convenience, we assume that $ct$ contains an index to $pk$.
- $\mathsf{Dec}\left(\overline{ct}_j, \{sk_i\}_{i \in T_j}\right)$ : Let $T_j \subseteq [k]$ be a set. Given a ciphertext $\overline{ct}_j$ corresponding to a set of keys $T_j \subseteq [k]$ and a tuple of secret keys $\{sk_i\}_{i \in T_j}$, outputs the message $m \in \mathcal{M}$.
- $\mathsf{Eval}\left(\mathcal{C}, (\overline{ct}_1, \cdots, \overline{ct}_l), \{pk_i\}_{i \in T}\right)$ : Let $T = T_1 \cup \cdots \cup T_\ell$. Given a circuit $\mathcal{C}$, a set of public keys $\{pk_i\}_{i \in T}$ and a tuple of multi-key ciphertexts $\overline{ct}_1, \cdots, \overline{ct}_l$ where each ciphertext $\overline{ct}_j$ is evaluated using $pk_{T_j} = \{pk_d, \forall d \in T_j\}$ for $j \in [l]$, outputs a ciphertext $\overline{ct}$.

**Compactness.** We say that a multi-key FHE scheme is compact if there exists a polynomial $poly(\cdot, \cdot)$ such that the length of a ciphertext associated with $k$ keys is bounded by a polynomial $poly(\lambda, k)$.

**Correctness.** Let $\overline{ct}_j$ be a ciphertext (associated with the set $T_j$) such that $\mathsf{Dec}\left(\overline{ct}_j, \{sk_i\}_{i\in T_j}\right) = m_j$ for $1 \leq j \leq l$. Let $\mathcal{C} : \mathcal{M}^\ell \to \mathcal{M}$ be a circuit and $\overline{ct} \leftarrow \mathsf{Eval}\left(\mathcal{C}, (\overline{ct}_1, \cdots, \overline{ct}_l), \{pk_i\}_{i\in T}\right)$ for $T = T_1 \cup \cdots \cup T_\ell$. We say that a multi-key FHE scheme is correct if

$$\Pr\left[\mathsf{Dec}\left(\overline{ct}, \{sk_i\}_{i\in T}\right) \neq \mathcal{C}\left(m_1, \ldots, m_l\right)\right] = \text{negl}(\lambda).$$

### 2.4 Hard Problems

Let $q, n, Q$ be three integers, and $N$ a power of two. Let $R_Q = \mathbb{Z}_Q[X]/\left(X^N + 1\right)$. Let $\chi_e$(resp.,$\chi'_e$) be a noise distribution over $\mathbb{Z}$ (resp., $R_Q$). Let $\chi_s$ (resp., $\chi'_s$) be a secret distribution over $\mathbb{Z}$ (resp., $R$).

**The LWE Problem [33].** The decisional $\text{LWE}_{n,q,\chi_s,\chi_e}$ problem is to distinguish the following two distributions:

– $\{(\mathbf{A}, \mathbf{b} = \mathbf{A}\cdot\mathbf{s} + \mathbf{e}) | \mathbf{A} \leftarrow \mathbb{Z}_q^{k_1\times n}, \mathbf{s} \leftarrow \chi_s^n, \mathbf{e} \leftarrow \chi_e^{k_1}\}$,
– $\{(\mathbf{U}, \mathbf{v}) | \mathbf{U} \leftarrow \mathbb{Z}_q^{k_1\times n}, \mathbf{v} \leftarrow \mathbb{Z}_q^{k_1}\}$.

The decisional $\text{LWE}_{n,q,\chi_s,\chi_e}$ assumption says that it is hard for any PPT algorithms to solve decisional $\text{LWE}_{n,q,\chi_s,\chi_e}$ with non-negligible advantage over a random guess.

**The RLWE Problem [28].** The decisional $\text{RLWE}_{N,Q,\chi'_s,\chi'_e}$ problem is to distinguish the following two distributions:

$$\{(\mathbf{a}, \mathbf{b} = \mathbf{a}\cdot s + \mathbf{e}) | \mathbf{a} \leftarrow R_Q^{k_2}, s \leftarrow \chi'_s, \mathbf{e} \leftarrow \chi_e'^{k_2} \quad \text{and} \quad \{(\mathbf{u}, \mathbf{v}) | \mathbf{u}, \mathbf{v} \leftarrow R_Q^{k_2}\}\}.$$

The decisional $\text{RLWE}_{N,Q,\chi'_s,\chi'_e}$ assumption says that it is hard for any PPT algorithms to solve decisional $\text{RLWE}_{N,Q,\chi'_s,\chi'_e}$ with non-negligible advantage over a random guess.

**The NTRU Problem [5,35]** (in the vector form). For an integer $d$, the decisional $\text{NTRU}_{N,Q,\chi'_s,\chi'_e}$ problem is to distinguish the following two distributions:

– $\{(g_0/f, \cdots, g_{d-1}/f) | f \leftarrow \chi'_s, g_0, \cdots, g_{d-1} \leftarrow \chi'_e\}$,
– $\{(u_1, \cdots, u_d) | u_1, \cdots, u_d \leftarrow R_Q\}$.

The decisional NTRU assumption (in the vector form) says that it is hard for any PPT algorithms to solve decisional $\text{NTRU}_{N,Q,\chi'_s,\chi'_e}$ with non-negligible advantage over a random guess.
**The matrix NTRU Problem [19,16].** The decisional matrix $\text{NTRU}_{n,q,\chi_s,\chi_e}$ problem is to distinguish the following two distributions:

$$\{\mathbf{G}\cdot\mathbf{F}^{-1} | \mathbf{F} \leftarrow \chi_s^{n\times n}, \mathbf{G} \leftarrow \chi_e^{m\times n}\} \quad \text{and} \quad \{\mathbf{U} | \mathbf{U} \leftarrow \mathbb{Z}_q^{m\times n}\}.$$

The decisional matrix $\text{NTRU}_{n,q,\chi_s,\chi_e}$ assumption says that it is hard for any PPT algorithms to solve decisional matrix $\text{NTRU}_{n,q,\chi_s,\chi_e}$ with non-negligible advantage over a random guess.

**The Hint-NTRU problem [17]** (in the vector form). The decisional Hint-NTRU$_{N,Q,\chi'_s,\chi'_e}$ problem is to distinguish the following two distributions:

- $\{(\mathbf{h} = \mathbf{e}_1/s, \mathbf{a}, \mathbf{b} = \mathbf{a} \cdot s + \mathbf{e}_2) | s \leftarrow \chi'_s, \mathbf{e}_1, \mathbf{e}_2 \leftarrow \chi'^d_e, \mathbf{a} \leftarrow R_Q^d\}$,
- $\{(\mathbf{u}, \mathbf{v}, \mathbf{w}) | \mathbf{u}, \mathbf{v}, \mathbf{w} \leftarrow R_Q^d\}$.

The decisional Hint-NTRU$_{N,Q,\chi'_s,\chi'_e}$ assumption says that it is hard for any PPT algorithms to solve decisional Hint-NTRU$_{N,Q,\chi'_s,\chi'_e}$ with non-negligible advantage over a random guess.

In Eurocrypt 2024, Esgin, Espitau, Niot et al. [17] first introduced the Hint-NTRU problem, asserting that an adversary against Hint-NTRU can break the indistinguishability of the inhomogeneous NTRU instance [19] for appropriate choices of parameters.

As the FHE schemes in [19,35,5], our MK-FHEs essentially rely on the KDM-security of the (matrix) NTRU problems and Hint-NTRU problems. We formally define the problems below.

**The KDM-form NTRU Problem [35]** (in the vector form). For an arbitrarily chosen (and public known) $m \in R_Q$ and integers $B, d$, the decisional KDM-form NTRU$_{N,Q,\chi'_s,\chi'_e}$ problem is to distinguish the following two distributions:

- $\{\big((g_0 + B^0 \cdot m)/f, \cdots, (g_{d-1} + B^{d-1} \cdot m)/f\big) | f \leftarrow \chi'_s, g_0, \cdots, g_{d-1} \leftarrow \chi'_e\}$,
- $\{(u_1, \cdots, u_d) | u_1, \cdots, u_d \leftarrow R_Q\}$.

The decisional KDM-form NTRU$_{N,Q,\chi'_s,\chi'_e}$ assumption says that it is hard for any PPT algorithms to solve decisional KDM-form NTRU$_{N,Q,\chi'_s,\chi'_e}$ with non-negligible advantage over a random guess.

**The KDM-form matrix NTRU Problem.** For an arbitrarily chosen (and public known) $\mathbf{M} \in \mathbb{Z}_q^{m\times n}$, the decisional KDM-form matrix NTRU$_{n,q,\chi_s,\chi_e}$ problem is to distinguish the following two distributions:

$$\{(\mathbf{G} + \mathbf{M}) \cdot \mathbf{F}^{-1} | \mathbf{F} \leftarrow \chi_s^{n\times n}, \mathbf{G} \leftarrow \chi_e^{m\times n}\} \quad \text{and} \quad \{\mathbf{U} | \mathbf{U} \leftarrow \mathbb{Z}_q^{m\times n}\}.$$

The decisional KDM-form matrix NTRU$_{n,q,\chi_s,\chi_e}$ assumption says that it is hard for any PPT algorithms to solve decisional KDM-form matrix NTRU$_{n,q,\chi_s,\chi_e}$ with non-negligible advantage over a random guess.

Note that the standard (matrix) NTRU problem is essentially a special case of the KDM-form (matrix) NTRU problem with $m = 0$ or $\mathbf{M} = 0$. Intuitively, the hardness of the KDM-form (matrix) NTRU problem is equivalent to the KDM-security of the standard (matrix) NTRU encryption which encrypts $m/f$ or $\mathbf{G} \cdot \mathbf{F}^{-1}$ [35,5]. We also note that the matrix inhomogeneous NTRU (MiNTRU) problem considered in [19] is basically a special case of our KDM-form matrix NTRU problem with $\mathbf{M}$ being fixed to a gadget matrix. As shown in [19], for an appropriate choice of error distributions, the above KDM-form matrix NTRU problem is polynomially equivalent to the MiNTRU problem, which in turn is not easier than a trapdoor version of the standard LWE problem [19].

**The KDM-form Hint-NTRU problem** (in the vector form). For an arbitrarily chosen (and public known) $m \in R_Q$, integers $B, d$ and gadget vector $\mathbf{g} = (B^0, \cdots, B^{d-1})$, the decisional KDM-form Hint-NTRU$_{N,Q,\chi'_s,\chi'_e}$ problem is to distinguish the following two distributions:

– $\{((\mathbf{e}_1 + \mathbf{g} \cdot m)/s, \mathbf{a}, \mathbf{b} = \mathbf{a} \cdot s + \mathbf{e}_2) | s \leftarrow \chi'_s, \mathbf{e}_1, \mathbf{e}_2 \leftarrow \chi'^d_e, \mathbf{a} \leftarrow R_Q^d\}$,
– $\{(\mathbf{u}, \mathbf{v}, \mathbf{w}) | \mathbf{u}, \mathbf{v}, \mathbf{w} \leftarrow R_Q^d\}$.

The decisional KDM-form Hint-NTRU$_{N,Q,\chi'_s,\chi'_e}$ assumption says that it is hard for any PPT algorithms to solve decisional KDM-form Hint-NTRU$_{N,Q,\chi'_s,\chi'_e}$ with non-negligible advantage over a random guess.

Note that the standard Hint-NTRU problem is essentially a special case of the KDM-form Hint-NTRU problem with $m = 0$. Essentially, the hardness of the KDM-form Hint-NTRU problem is equivalent to the KDM-security of the standard Hint-NTRU encryption which encrypts $m/s$.

## 3 First-layer Matrix NTRU-based Multi-Key Encryption

In this section, we propose a first-layer multi-key encryption $\mathsf{HE}$ that supports multi-key NAND operation in a natural way based on the matrix NTRU assumption. Formally, our construction $\mathsf{HE} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{MK\text{-}Dec}, \mathsf{MK\text{-}NAND})$ consists of five algorithms below:

- $\mathsf{HE.Setup}(1^\lambda)$: Given the security parameter $\lambda$, set the matrix dimension $n$, ciphertext modulus $q$, secret distribution $\chi_s$ and error distribution $\chi_e$ over $\mathbb{Z}$. Return the public parameter $pp = (n, q, \chi_s, \chi_e)$.
- $\mathsf{HE.KG}(pp)$: Sample $\mathbf{F} \leftarrow \chi_s^{n \times n}$ until $\mathbf{F}^{-1}$ exists in $\mathbb{Z}_q^{n \times n}$. Define $sk := \mathbf{F}$. Create a public evaluation key as $evk := (\mathbf{e} + \lfloor 5 \cdot q/8 \rceil \cdot (1, \mathbf{0})) \cdot \mathbf{F}^{-1} \in \mathbb{Z}_q^n$, where $\mathbf{e} \leftarrow \chi_e^n$. Output $(evk, sk)$.
- $\mathsf{HE.Enc}(m, \mathbf{F})$: Given $m \in \{0, 1\}$, sample $\mathbf{e}' \leftarrow \chi_e^n$. Let $\Delta := \lfloor q/4 \rceil$ and output
$$\mathbf{c} = (\mathbf{e}' + \Delta \cdot (m, \mathbf{0})) \cdot \mathbf{F}^{-1} \in \mathbb{Z}_q^n.$$
- $\mathsf{HE.MK\text{-}Dec}(\overline{ct}, \{\mathbf{F}_i\}_{i \in [k]})$ : Given a ciphertext $\overline{ct} = (\mathbf{c}_1, \cdots, \mathbf{c}_k) \in \mathbb{Z}_q^{kn}$ after NAND gate evaluation under the secret key $(\mathbf{F}_1, \cdots, \mathbf{F}_k) \in (\mathbb{Z}_q^{n \times n})^k$, which satisfies $\sum_{i=1}^{k} \langle \mathbf{c}_i, \mathrm{col}_0(\mathbf{F}_i) \rangle \approx \lfloor \frac{q}{2} \rceil m$. Compute
$$\left\lfloor \frac{2 \cdot \sum_{i=1}^{k} \mathbf{c}_i \cdot \mathrm{col}_0(\mathbf{F}_i)}{q} \right\rceil \in \mathbb{Z}_2.$$
- $\mathsf{HE.MK\text{-}NAND}(\overline{ct}_1, \overline{ct}_2, evk_i)$ : Given $\overline{ct}_1 = (\mathbf{c}_{1,j_1}, \cdots, \mathbf{c}_{1,j_{k_1}}) \in \mathbb{Z}_q^{k_1 n}$ satisfying $\sum_{i=j_1}^{j_{k_1}} \langle \mathbf{c}_{1,i}, \mathrm{col}_0(\mathbf{F}_i) \rangle \approx \lfloor \frac{q}{4} \rceil m_1$ , $\overline{ct}_2 = (\mathbf{c}_{2,j_1}, \cdots, \mathbf{c}_{2,j_{k_2}}) \in \mathbb{Z}_q^{k_2 n}$ satisfying $\sum_{i=j_1}^{j_{k_2}} \langle \mathbf{c}_{2,i}, \mathrm{col}_0(\mathbf{F}_i) \rangle \approx \lfloor \frac{q}{4} \rceil m_2$ and the evaluation key $evk_i$ of

party $i \in [k]$ as inputs, let $k$ be the maximum number of different keys involved in $\overline{ct}_1$ and $\overline{ct}_2$. Extend $\overline{ct}_i = (\mathbf{c}_{i,j_1}, \cdots, \mathbf{c}_{i,j_{k_i}}) \in \mathbb{Z}_q^{k_i n}$ to the ciphertext $\overline{ct}'_i = \left(\mathbf{c}'_{i,1}, \ldots, \mathbf{c}'_{i,k}\right) \in \mathbb{Z}_q^{kn}$ where

$$\mathbf{c}'_{i,j} = \begin{cases} \mathbf{c}_{i,\ell} & \text{if } j = j_\ell \text{ for some } \ell \in [k_i] \\ \mathbf{0} & \text{otherwise} \end{cases}$$

for $i \in \{1,2\}$ and $j \in [k]$. Compute the homomorphic NAND gate homomorphically as follows:

$$\overline{ct} = (\underbrace{\mathbf{0}, \cdots, \mathbf{0}}_{(i-1)n \text{ zeros}}, evk_i, \underbrace{\mathbf{0}, \cdots, \mathbf{0}}_{(k-i)n \text{ zeros}}) - \overline{ct}'_1 - \overline{ct}'_2 \in \mathbb{Z}_q^{kn}.$$

By Lemma 1, we show the correctness of the NAND gate evaluation.

**Lemma 1.** *For $i \in \{1,2\}$, let $\overline{ct}_i = (\mathbf{c}_{i,j_1}, \ldots, \mathbf{c}_{i,j_{k_i}}) \in \mathbb{Z}_q^{k_i n}$ be the ciphertext that encrypts $m_i$ under the secret key $(\mathbf{F}_{j_1}, \ldots, \mathbf{F}_{j_{k_i}}) \in (\mathbb{Z}_q^{n\times n})^{k_i}$ with noise $\mathbf{e}_i$, satisfying $\langle \mathbf{c}_{i,j_1}, \mathrm{col}_0(\mathbf{F}_{j_1})\rangle + \cdots + \langle \mathbf{c}_{i,j_{k_i}}, \mathrm{col}_0(\mathbf{F}_{j_{k_i}})\rangle = \left\lfloor \frac{q}{4} \right\rceil \cdot m_i + e_{i,0}$, where $e_{i,0}$ is the first element of $\mathbf{e}_i$. Let $evk_i$ be the evaluation key of party $i$ with a noise $\mathbf{e}$. Let $e_0$ be the first element of $\mathbf{e}$. Let $\overline{ct} = \mathsf{HE.MK\text{-}NAND}(\overline{ct}_1, \overline{ct}_2, evk_i)$. If $|e_0 - e_{1,0} - e_{2,0}| < \frac{q-12}{8}$ then $\mathsf{HE.MK\text{-}Dec}(\overline{ct}, \{\mathbf{F}_i\}_{i\in[k]})$ outputs $\mathsf{NAND}(m_1, m_2) = 1 - m_1 m_2$.*

*Proof.* Let $\overline{\mathbf{F}} = (\mathrm{col}_0(\mathbf{F}_j))_{j\in[k]}$. For $1 \leqslant i \leqslant 2$, let $\overline{ct}'_i = (\mathbf{c}'_{i,1}, \cdots, \mathbf{c}'_{i,k})$ be the extended ciphertext of $\overline{ct}_i$. By definition, we have $\left\langle \overline{ct}'_i, \overline{\mathbf{F}} \right\rangle = e_{i,0} + \left\lfloor \frac{q}{4} \right\rceil m_i$. Let $e_0$ be the first element of $\mathbf{e}$, then

$$\begin{aligned} \langle \overline{ct}', \overline{\mathbf{F}} \rangle &= \left\langle \left(\mathbf{0}, \cdots, \mathbf{0}, \mathbf{e} + \lfloor 5 \cdot q/8 \rceil \cdot (1, \mathbf{0}) \cdot \mathbf{F}_i^{-1}, \mathbf{0}, \cdots, \mathbf{0}\right) - \overline{ct}'_1 - \overline{ct}'_2, \overline{\mathbf{F}} \right\rangle \\ &= \frac{5q}{8} + e_0 + \epsilon - \left(e_{1,0} + \frac{q}{4} m_1 + m_1 \epsilon_1\right) - \left(e_{2,0} + \frac{q}{4} m_2 + m_2 \epsilon_2\right) \\ &= e_0 - e_{1,0} - e_{2,0} \pm \frac{q}{8} + \epsilon - m_1 \epsilon_1 - m_2 \epsilon_2 + \frac{q}{2}(1 - m_1 m_2), \end{aligned}$$

where $\epsilon, \epsilon_1, \epsilon_2$ are round-off errors and $|\epsilon| \leqslant \frac{1}{2}$, $|\epsilon_1| \leqslant \frac{1}{2}$, $|\epsilon_2| \leqslant \frac{1}{2}$, respectively. Let $e = \epsilon - m_1\epsilon_1 - m_2\epsilon_2$, we have $|e| \leqslant \frac{3}{2}$. The output of $\mathsf{HE.MK\text{-}Dec}(\overline{ct}, \{\mathbf{F}_i\}_{i\in[k]})$ is

$$\left\lfloor \frac{2}{q} \cdot \left(e + e_0 - e_{1,0} - e_{2,0} \pm \frac{q}{8} + \frac{q}{2}(1 - m_1 m_2)\right) \right\rceil.$$

Thus, the output is equal to $1 - m_1 m_2$ as long as $|\frac{2}{q} \cdot (e + e_0 - e_{1,0} - e_{2,0} \pm \frac{q}{8})| < \frac{1}{2}$, which implies $|e_0 - e_{1,0} - e_{2,0}| < \left(\frac{1}{2} - \frac{1}{4} - \frac{3}{q}\right) \cdot \frac{q}{2} = \frac{q-12}{8}$. $\square$

**Theorem 1 (Security of HE).** *Let $pp = (n, q, \chi_s, \chi_e)$ be some parameters such that the KDM-form matrix NTRU problem is hard. Then, for any $m \in \mathbb{Z}_q$, if $(evk, \mathbf{F}) \leftarrow \mathsf{HE.KG}(pp)$, $\mathbf{c} \leftarrow \mathsf{HE.Enc}(m, \mathbf{F})$, it holds that the joint distribution $(evk, \mathbf{c})$ is computationally indistinguishable from uniform over $\mathbb{Z}_q^n \times \mathbb{Z}_q^n$.*

Note that our multi-key construction is essentially a natural extension of the single-key MNTRU encryption based on the KDM-form MNTRU problems in [5]. The proof is similar and directly, we omit the details.

## 4 Second-layer NTRU-based Uni-Encryption

In this section, we present a second-layer NTRU-based uni-encryption that supports an efficient hybrid product.

Our uni-encryption $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{UniEnc}, \mathsf{HbProd})$ consists of four algorithms:

- $\mathsf{Setup}(1^\lambda)$: Given the security parameter $\lambda$, set the polynomial degree $N$, ciphertext modulus $Q$, gadget vector dimension $d$, secret distribution $\chi'_s$ and error distribution $\chi'_e$ over $R$. Generate a random vector $\mathbf{a} \leftarrow R_Q^d$. Return the public parameter $pp' = (N, Q, d, \mathbf{a}, \chi'_s, \chi'_e)$.
- $\mathsf{KG}(pp')$: Sample a polynomial $s \leftarrow \chi'_s$ uniformly at random until $s^{-1}$ exists in $R_Q$ and a noise $\mathbf{e} \leftarrow \chi'^d_e$. Compute the public key $\mathbf{b} = -\mathbf{a} \cdot s + \mathbf{e}$ and output $(s, \mathbf{b})$.
- $\mathsf{UniEnc}(\mu, s)$: For a message $\mu \in R_Q$ and a secret key $s$, sample $r \leftarrow \chi'_s$ uniformly at random, and noise $\mathbf{e}_1$, $\mathbf{e}_2 \leftarrow \chi'^d_e$. Compute the ciphertext $\mathbf{d} = r \cdot \mathbf{a} + \mu \cdot \mathbf{g} + \mathbf{e}_1 \in R_Q^d$ and $\mathbf{f} = \mathbf{e}_2 \cdot s^{-1} + r \cdot \mathbf{g} \cdot s^{-1} \in R_Q^d$ and output $(\mathbf{d}, \mathbf{f}) \in R_Q^d \times R_Q^d$.
- $\mathsf{HbProd}(\overline{\mathbf{c}}, \{\mathbf{b}_j\}_{j\in[k]}, (\mathbf{d}_i, \mathbf{f}_i))$: Given an NTRU-based multi-key ciphertext $\overline{\mathbf{c}} = (c_1, \cdots, c_k) \in R_Q^k$, the public keys $\{\mathbf{b}_j\}_{j\in[k]}$ of $k$ keys involved in $\overline{\mathbf{c}}$, and a uni-encryption $(\mathbf{d}_i, \mathbf{f}_i)$ of party $i$ as inputs, return an MK-NTRU ciphertext $\overline{\mathbf{c}}' \in R_Q^k$ as follows:
  (a) Compute the following inner products:
$$u_j = \left\langle \mathbf{g}^{-1}(c_j), \mathbf{d}_i \right\rangle \text{ for } 1 \leqslant j \leqslant k,$$
$$v = \sum_{j=1}^{k} \left\langle \mathbf{g}^{-1}(c_j), \mathbf{b}_j \right\rangle.$$
  (b) Output $\overline{\mathbf{c}}' = (c'_1, \ldots, c'_k) \in R^k$ where
$$c'_j = \begin{cases} u_i + \left\langle \mathbf{g}^{-1}(v), \mathbf{f}_i \right\rangle & \text{if } j = i, \\ u_j & \text{otherwise.} \end{cases}$$

By Lemma 2, we show that our uni-encryption supports homomorphic additions. In Lemma 3, we establish the correctness and estimate noise bound.

**Lemma 2.** *Let* $\mathsf{UniEnc}(\mu_1, s) = (\mathbf{d}_1, \mathbf{f}_1), \mathsf{UniEnc}(\mu_2, s) = (\mathbf{d}_2, \mathbf{f}_2) \in R_Q^d \times R_Q^d$ *be two uni-encryption ciphertexts. We define the homomorphic addition between* $\mathsf{UniEnc}(\mu_1, s)$ *and* $\mathsf{UniEnc}(\mu_1, s)$ *as*

$$\mathsf{UniEnc}(\mu_1, s) + \mathsf{UniEnc}(\mu_2, s) = (\mathbf{d}_1 + \mathbf{d}_2, \mathbf{f}_1 + \mathbf{f}_2),$$

*which is also a uni-encryption ciphertext that encrypts $\mu_1 + \mu_2$ under the secret $s$. And for a monomial $u$ with ternary coefficient, $u \cdot \mathsf{UniEnc}(\mu_i, s) = (u \cdot \mathbf{d}_i, u \cdot \mathbf{f}_i)$ is also a uni-encryption ciphertext that encrypts $u \cdot \mu_i$ under the secret $s$.*

*Moreover, if the variance of the noise distribution used in generating uni-encryption is $Var(e)$, then the noise variance of the ciphertext $(\mathbf{d}_1 + \mathbf{d}_2, \mathbf{f}_1 + \mathbf{f}_2)$ and $u \cdot \mu_i$ is bounded by $2Var(e)$ and $Var(e)$, respectively.*

*Proof.* By definition, we have that

$$\mathbf{d}_1 = r_1 \cdot \mathbf{a} + \mu_1 \cdot \mathbf{g} + \mathbf{e}_{1,1} \in R_Q^d, \qquad \mathbf{f}_1 = \mathbf{e}_{1,2} \cdot s^{-1} + r_1 \cdot \mathbf{g} \cdot s^{-1} \in R_Q^d$$

$$\mathbf{d}_2 = r_2 \cdot \mathbf{a} + \mu_2 \cdot \mathbf{g} + \mathbf{e}_{2,1} \in R_Q^d, \qquad \mathbf{f}_2 = \mathbf{e}_{2,2} \cdot s^{-1} + r_2 \cdot \mathbf{g} \cdot s^{-1} \in R_Q^d$$

Therefore, $\mathbf{d}_1 + \mathbf{d}_2 = (r_1 + r_2) \cdot \mathbf{a} + (\mu_1 + \mu_2) \cdot \mathbf{g} + \mathbf{e}_{1,1} + \mathbf{e}_{2,1} \in R_Q^d$ and $\mathbf{f}_1 + \mathbf{f}_2 = (\mathbf{e}_{1,2} + \mathbf{e}_{2,2}) \cdot s^{-1} + (r_1 + r_2) \cdot \mathbf{g} \cdot s^{-1} \in R_Q^d$. We have that $(\mathbf{d}_1 + \mathbf{d}_2, \mathbf{f}_1 + \mathbf{f}_2)$ is also a uni-encryption ciphertext that encrypts $\mu_1 + \mu_2$ and the noise variance of $(\mathbf{d}_1 + \mathbf{d}_2, \mathbf{f}_1 + \mathbf{f}_2)$ is bounded by $2Var(e)$.

By definition, we also have that

$$(u\mathbf{d}_i, u\mathbf{f}_i) = (r_i' \cdot \mathbf{a} + u \cdot \mu_i \cdot \mathbf{g} + \mathbf{e}_{i,1}', \mathbf{e}_{i,2}' \cdot s^{-1} + r_i' \cdot \mathbf{g} \cdot s^{-1})$$

is also a uni-encryption ciphertext that encrypts $u \cdot \mu_i$ where $r_i' = u \cdot r_i$, $\mathbf{e}_{i,1}' = u \cdot \mathbf{e}_{i,1}, \mathbf{e}_{i,2}' = u \cdot \mathbf{e}_{i,2}$ for $i \in \{1, 2\}$. By the fact $u$ is a monomial with a ternary coefficient, we have $Var(\mathbf{e}_{i,1}') \leq Var(e)$ and $Var(\mathbf{e}_{i,2}') \leq Var(e)$. □

**Lemma 3 (Hybrid Product).** *Let $\overline{\mathbf{c}} = (c_1, \cdots, c_k) \in R_Q^k$ be an MK-NTRU ciphertext under the secret key $\mathbf{s} = (s_1, \cdots, s_k) \in R^k$, and let $\{\mathbf{b}_j\}_{j \in [k]}$ be the public keys of $k$ keys associated with $\overline{\mathbf{c}}$. Let $(\mathbf{d}_i, \mathbf{f}_i) \leftarrow \mathsf{UniEnc}(\mu_i, s_i)$ be the uni-encryption of $\mu_i \in R_Q$ for party $i$, and let $\overline{\mathbf{c}}' \leftarrow \mathsf{HbProd}(\overline{\mathbf{c}}, \{\mathbf{b}_j\}_{j \in [k]}, (\mathbf{d}_i, \mathbf{f}_i))$ be the output of the hybrid product, we have that $\langle \overline{\mathbf{c}}', \mathbf{s} \rangle \approx \mu_i \langle \overline{\mathbf{c}}, \mathbf{s} \rangle$.*

*Moreover, if the noise variance in generating $(\mathbf{d}_i, \mathbf{f}_i)$ and $\{\mathbf{b}_j\}_{j \in [k]}$ is $Var(e)$ and the variance of the secret is $Var(s)$, then the variance of the increased noise in the resulting ciphertext is upper bounded by $(2kNVar(s) + 1)\frac{B^2}{12}dNVar(e)$.*

*Proof.* By definition, we have that $\overline{\mathbf{c}}'$ is generated by adding $\left\langle \mathbf{g}^{-1}(v), \mathbf{f}_i \right\rangle$ to the $i-$th components of $(u_1, \dots, u_k)$. Thus, we have

$$\begin{aligned}
\langle \overline{\mathbf{c}}', \mathbf{s} \rangle &= \sum_{j=1}^{k} u_j \cdot s_j + \left\langle \mathbf{g}^{-1}(v), \mathbf{f}_i \right\rangle \cdot s_i = \sum_{j=1}^{k} \left\langle \mathbf{g}^{-1}(c_j), \mathbf{d}_i \right\rangle \cdot s_j + \langle \mathbf{g}^{-1}(v), \mathbf{f}_i \cdot s_i \rangle \\
&= \sum_{j=1}^{k} \left\langle \mathbf{g}^{-1}(c_j), r_i \cdot \mathbf{a} + \mu_i \cdot \mathbf{g} + \mathbf{e}_{i,1} \right\rangle \cdot s_j + \langle \mathbf{g}^{-1}(v), r_i \cdot \mathbf{g} + \mathbf{e}_{i,2} \rangle \\
&= \mu_i \sum_{j=1}^{k} c_j s_j + r_i \cdot \sum_{j=1}^{k} \langle \mathbf{g}^{-1}(c_j), s_j \cdot \mathbf{a} \rangle + \sum_{j=1}^{k} \langle \mathbf{g}^{-1}(c_j), \mathbf{e}_{i,1} \rangle \cdot s_j \\
&+ r_i \cdot \sum_{j=1}^{k} \left\langle \mathbf{g}^{-1}\left(c_j\right), -\mathbf{a} \cdot s_j + \mathbf{e}_j \right\rangle + \langle \mathbf{g}^{-1}(v), \mathbf{e}_{i,2} \rangle \\
&= \mu_i \sum_{j=1}^{k} c_j s_j + \sum_{j=1}^{k} \langle \mathbf{g}^{-1}(c_j), \mathbf{e}_{i,1} \rangle \cdot s_j + r_i \cdot \sum_{j=1}^{k} \left\langle \mathbf{g}^{-1}\left(c_j\right), \mathbf{e}_j \right\rangle + \langle \mathbf{g}^{-1}(v), \mathbf{e}_{i,2} \rangle
\end{aligned}$$

and the noise variance is bounded by

$$\begin{aligned}
&(kNVar(s_j) + 1)\frac{B^2}{12} dNVar(e) + kN\mathsf{Var}(r_i)\frac{B^2}{12} dNVar(e) \\
&= (2kNVar(s) + 1)\frac{B^2}{12} dNVar(e)
\end{aligned}$$

This completes the proof. □

In the following, we will use $\sigma^2_{HP1} = (Var(s)kN + 1)\frac{dNB^2}{12}Var(e)$, and $\sigma^2_{HP2} = Var(s)kN\frac{dNB^2}{12}Var(e)$ to denote the variance of the increased noise (with respect to the input ciphertext) for the hybrid product.

By Theorem 2, we show that our NTRU-based Uni-Encryption is provably IND-CPA secure in the standard model.

**Theorem 2 (Security of the second-layer NTRU-based Uni-Encryption).** *Let $pp' = (N, Q, d, \mathbf{a}, \chi'_s, \chi'_e) \leftarrow \mathsf{Setup}(1^\lambda)$ be the parameters such that standard* $\mathrm{RLWE}_{N,Q,\chi'_s,\chi'_e}$ *and the KDM-form Hint-*$\mathrm{NTRU}_{N,Q,\chi'_s,\chi'_e}$ *assumptions hold, we have that our second-layer NTRU-based Uni-Encryption is provably IND-CPA secure in the standard model.*

*Proof.* We prove Theorem 2 by using a series of games $G_0 \sim G_2$, where $G_0$ is the standard IND-CPA game, and $G_2$ is a random one. Suppose that there exists an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ which can break the IND-CPA security of our hybrid product with advantage $\varepsilon$. Let $H_i$ be the event that $\mathcal{A}$ correctly guesses in game $G_i$, then the adversary's advantage $\mathrm{Adv}_{H_i}[\mathcal{A}]$ in game $G_i$ is exactly $|\Pr[H_i] - 1/2|$.

*Game $G_0$.* This game is the real IND-CPA security game. Formally, the challenger $\mathcal{C}$ works as follows:

**KeyGen.** First randomly choose $s \leftarrow \chi'_s$, $\mathbf{e} \leftarrow \chi'^d_e$, compute $\mathbf{b} = -\mathbf{a} \cdot s + \mathbf{e}$. Then, return $\mathbf{b}$ to the adversary $\mathcal{A}_1$, and keep the secret key $s$ private.

**Challenge.** After receiving two equal length plaintexts $M_0, M_1 \in \mathcal{M}$ from the adversary $\mathcal{A}_1$, the challenger $\mathcal{C}$ first randomly chooses a bit $\mu^* \leftarrow \{0, 1\}$, and

$$r \leftarrow \chi'_s, \mathbf{e}_1 \leftarrow \chi'^d_e, \mathbf{e}_2 \leftarrow \chi'^d_e.$$

Then, it defines

$$\mathbf{d} = r \cdot \mathbf{a} + M_{\mu^*} \cdot \mathbf{g} + \mathbf{e}_1 \in R^d_Q, \mathbf{f} = \mathbf{e}_2 \cdot s^{-1} + r \cdot \mathbf{g} \cdot s^{-1} \in R^d_Q$$

and returns the challenge ciphertext $(\mathbf{d}, \mathbf{f})$ to $\mathcal{A}_2$.

**Finalize.** Upon receiving a guess $\mu' \in \{0, 1\}$ from $\mathcal{A}_2$, if $\mu' = \mu^*$, the challenger $\mathcal{C}$ outputs 1, else outputs 0.

**Lemma 4.** $\left|\Pr[H_0] - \frac{1}{2}\right| = \epsilon$.

*Proof.* This lemma immediately follows the fact that $\mathcal{C}$ honestly simulates the attack environment for $\mathcal{A}$, and only outputs 1 if and only if $\mu' = \mu^*$.

*Game $G_1$.* This game is identical to $G_0$ except that the challenger $\mathcal{C}$ changes the KeyGen phase and Challenge phase as follows:

**KeyGen.** Randomly choose $s \leftarrow \chi'_s$, $\mathbf{b}' \leftarrow \chi'^d_e$. Then, return $\mathbf{b}'$ to the adversary $\mathcal{A}_1$, and keep the secret key $s$ private.

**Challenge.** After receiving two equal length plaintexts $M_0, M_1 \in \mathcal{M}$ from the adversary $\mathcal{A}_1$, the challenger $\mathcal{C}$ first randomly chooses a bit $\mu^* \leftarrow \{0, 1\}$, and

$$r \leftarrow \chi'_s, \mathbf{e}_1 \leftarrow \chi'^d_e, \mathbf{f}' \leftarrow R^d_Q.$$

Then, it defines

$$\mathbf{d} = r \cdot \mathbf{a} + M_{\mu^*} \cdot \mathbf{g} + \mathbf{e}_1 \in R^d_Q$$

and returns the challenge ciphertext $(\mathbf{d}, \mathbf{f}')$ to $\mathcal{A}_2$.

**Lemma 5.** *Game $G_0$ and $G_1$ are computationally indistinguishable in the adversary's view. Moreover,* $|\Pr[H_1] - \Pr[H_0]| = negl(\lambda)$.

*Proof.* Since the differences between Game $G_0$ and $G_1$ is that $\mathcal{C}$ replaces $\mathbf{b} = -\mathbf{a} \cdot s + \mathbf{e} \in R^d_Q$ and $\mathbf{f} = \mathbf{e}_2 \cdot s^{-1} + r \cdot \mathbf{g} \cdot s^{-1}$ in Game $G_0$ with a randomly chosen $\mathbf{b}' \in R^d_Q$ and $\mathbf{f}'$ in Game $G_1$. Under the KDM-form Hint-NTRU$_{N,Q,\chi'_s,\chi'_e}$ assumption, we have that $|\Pr[H_1] - \Pr[H_0]| = negl(\lambda)$.

*Game $G_2$.* This game is similar to game $G_1$ except that the challenger $\mathcal{C}$ changes the Challenge phase as follows:

**Challenge.** After receiving two equal length plaintexts $M_0, M_1 \in \mathcal{M}$ from the adversary $\mathcal{A}_1$, the challenger $\mathcal{C}$ first randomly chooses a bit $\mu^* \leftarrow \{0,1\}$, and

$$\mathbf{h} \leftarrow R_Q^d, \mathbf{f}' \leftarrow R_Q^d.$$

Then, it defines

$$\mathbf{d}' = \mathbf{h} + M_{\mu^*} \cdot \mathbf{g} \in R_Q^d$$

and returns the challenge ciphertext $(\mathbf{d}', \mathbf{f}')$ to $\mathcal{A}_2$.

**Lemma 6.** *Under the* $\mathrm{RLWE}_{N,Q,\chi'_s,\chi'_e}$ *assumption, we have that Game* $G_1$ *and* $G_2$ *are computationally indistinguishable in the adversary's view. Moreover,* $|\Pr[H_2] - [H_1]| = negl(\lambda)$.

*Proof.* This lemma follows from that the only difference between Games $G_1$ and $G_2$ is that $\mathcal{C}$ replaces $r \cdot \mathbf{a} + \mathbf{e}_1$ in $G_1$ with a random one $\mathbf{h} \leftarrow R_Q^d$ in $G_2$.

**Lemma 7.** $\left|\Pr[H_2] - \frac{1}{2}\right| \leqslant negl(\lambda)$.

*Proof.* This lemma directly follows from that $\mathbf{h}$ in Game $G_2$ is uniformly random, and statistically hides the information of $M_{\mu^*}$ in $\mathbf{h} + M_{\mu^*} \cdot \mathbf{g}$.

By Lemmas 4∽7, we have that $\varepsilon = \left|\Pr[H_0] - \frac{1}{2}\right| \leqslant negl(\lambda)$. This completes the proof of Theorem 2. □

## 5 Bootstrapping First-layer matrix NTRU-based Multi-Key Ciphertexts.

In this section, we describe how to apply our second-layer NTRU-based hybrid product to bootstrap a first-layer matrix NTRU-based multi-key ciphertext. In Sec.5.1, we described the multi-key NTRU modulus switching technique. In Sec.5.2, a key-switching technique is introduced to transform an MK-NTRU ciphertext back into a multi-key matrix NTRU ciphertext. In Sec.5.3, we present the bootstrapping algorithm and analyze its correctness.

### 5.1 Modulus Switching for MK-NTRU ciphertext

To transform an MK-NTRU ciphertext from modulus $Q$ to $q$, we can multiply it by $q/Q$ and round the result to the nearest integer.

**Lemma 8.** *Given an MK-NTRU ciphertext* $\overline{\mathbf{c}} = (c_1, \cdots, c_k) \in R_Q^k$ *that encrypts* $m \in \{0,1\}$ *with secret key* $\mathbf{s} = (s_1, \cdots, s_k) \in R^k$*, where* $\sum_{i=1}^k c_i s_i = \left\lfloor \frac{Q}{4} \right\rceil m + e$*, the MK-NTRU modulus switching procedure* $\mathsf{NModSwitch}(\overline{\mathbf{c}}, q)$ *is defined as*

$$\mathsf{NModSwitch}(\overline{\mathbf{c}}, q) = (c'_1, \cdots, c'_k) = (\sum_{i=0}^{N-1} \left\lfloor \frac{q}{Q} c_{1,i} \right\rceil X^i, \cdots, \sum_{i=0}^{N-1} \left\lfloor \frac{q}{Q} c_{k,i} \right\rceil X^i),$$

*where $c_{j,i}$ denotes the i-th coefficient for $c_j$ ($j \in [k]$). Then, $(c'_1, \cdots, c'_k)$ is an MK-NTRU ciphertext that encrypts the same message under the secret key $\mathbf{s} \in R^k$. Moreover, if the noise variance of $\overline{\mathbf{c}}$ is $Var(e)$, then the noise variance of the ciphertext after modulus switching is bounded by $(\frac{q}{Q})^2 Var(e) + 1 + \frac{\sum_{i=1}^k \|s_i\|}{12}$.*

*Proof.* Let $\sum_{i=1}^k c_i s_i = \left\lfloor \frac{Q}{4} \right\rceil m + e$. By definition, we have

$$\begin{aligned}
\sum_{i=1}^k c'_i \cdot s_i &= \sum_{i=1}^k \left\lfloor \frac{q}{Q} c_i \right\rceil \cdot s_i \\
&= \sum_{i=1}^k \frac{q}{Q} c_i \cdot s_i + \sum_{i=1}^k \epsilon_i \cdot s_i \\
&= \left\lfloor \frac{q}{4} \right\rceil m + \frac{q}{Q} \cdot e + \epsilon_0 + \sum_{i=1}^k \epsilon_i \cdot s_i
\end{aligned}$$

where $\epsilon_0, \epsilon_1$ is a polynomial with infinite norm bounded by 1 and $\frac{1}{2}$, respectively. The noise variance of the ciphertext after modulus switching is bounded by

$$(\frac{q}{Q})^2 Var(e) + 1 + \frac{\sum_{i=1}^k \|s_i\|}{12}.$$

This completes the proof. □

For this modulus switching, we denote $\sigma_{NMS}^2 = 1 + \frac{\sum_{i=1}^k \|s_i\|}{12}$ as the variance of the increased noise (relative to the input ciphertext).

### 5.2 Key-switching from MK-NTRU Ciphertext to the base scheme

In this subsection, we define a pair of two algorithms ($\mathsf{MN.KSKG}$, $\mathsf{MN.KS}$) for key switching the MK-NTRU ciphertext to the MNTRU-based first layer multi-key ciphertext as follows:

- $\mathsf{MN.KSKG}(s_i, \mathbf{F}_i)$: Given matrice $\mathbf{F}_i \in \mathbb{Z}_q^{n \times n}$, polynomial $s_i \in R$ as inputs, the algorithm first samples matrice $\mathbf{E}_i \leftarrow \mathbb{Z}_q^{N d_{ks} \times n}$ from some noise distribution over $\mathbb{Z}_q$ and outputs

$$\mathbf{KSK}_i = (\mathbf{E}_i + \mathbf{G}_N \cdot \mathcal{A}(s_i) \cdot \mathbf{M}) \cdot \mathbf{F}_i^{-1} \in \mathbb{Z}_q^{(N \cdot d_{ks}) \times n},$$

  where $\mathcal{A}(s_i)$ denotes the anti-circulant matrix of $s_i$ and $\mathbf{M} \in \mathbb{Z}_q^{N \times n}$ is a matrix with entries being all zeros except for $\mathbf{M}_{0,0} = 1$.
- $\mathsf{MN.KS}(\overline{\mathbf{c}}, \{\mathbf{KSK}_i\}_{i \in [k]})$: Input an NTRU-based multi-key ciphertext $\overline{\mathbf{c}} = (c_1, \cdots, c_k) \in R_q^k$ that encrypts a polynomial with constant coefficient $m \in \{0, 1\}$ and the key-switching keys $\{\mathbf{KSK}_i\}_{i \in [k]}$ of keys associated with $\overline{ct}$, it first computes

$$\hat{\mathbf{c}}_i = \left(\mathbf{g}^{-1}(c_{i,0}), \cdots, \mathbf{g}^{-1}(c_{i,N-1})\right)$$

where $(c_{i,0}, \cdots, c_{i,N-1})$ is the coefficient vector of $c_i$. Then it computes

$$\mathbf{c}_i = \hat{\mathbf{c}}_i \cdot \mathbf{KSK}_i$$

for $i \in [k]$ and outputs $\overline{ct} = (\mathbf{c}_1, \cdots, \mathbf{c}_k)$.

**Lemma 9 (Key-switching for MK-NTRU ciphertext).** *Let $\mathbf{F}_1, \cdots, \mathbf{F}_k \in \mathbb{Z}_q^{n\times n}$ and $s_1, \cdots, s_k \in R$ be $k$ matrices and polynomials, respectively. Let $\overline{\mathbf{c}} = (c_1, \cdots, c_k) \in R_q^k$ be an MK-NTRU ciphertext that encrypts a polynomial with constant coefficient $m \in \{0,1\}$ under the secret key $(s_1, \cdots, s_k) \in R^k$. Then, for any $\mathbf{KSK}_i = \mathsf{MN.KSKG}(s_i, \mathbf{F}_i)$, we have that the output of $\mathsf{MN.KS}(\overline{\mathbf{c}}, \{\mathbf{KSK}_i\}_{i\in[k]})$ is a matrix NTRU based ciphertext that encrypts $m \in \{0,1\}$ under the secret key $\{\mathbf{F}_1, \cdots, \mathbf{F}_k\} \in \mathbb{Z}_q^{(n\times n)k}$.*

*Moreover, if the variance of the noise in $\overline{\mathbf{c}}$ is $Var(e)$, and the variance of the noise distribution used in generating $\mathbf{KSK}_i$ is $Var(e_{ks})$, then the variance of the noise in the resulting ciphertext $\overline{ct}$ is upper bounded by*

$$k\frac{B_{ks}^2}{12}Nd_{ks}Var(e_{ks}) + Var(e).$$

*Proof.* By definition, it is clear that $\sum_{i=1}^{k} c_i s_i = \left\lfloor \frac{q}{4} \right\rceil m + e$ and

$$\begin{aligned}\hat{\mathbf{c}}_i \cdot \mathbf{KSK}_i &= \left(\mathbf{g}^{-1}(c_{i,0}), \cdots, \mathbf{g}^{-1}(c_{i,N-1})\right) \cdot (\mathbf{E}_i + \mathbf{G}_N \cdot \mathcal{A}(s_i) \cdot \mathbf{M}) \cdot \mathbf{F}_i^{-1} \\ &= \left((\mathbf{g}^{-1}(c_{i,0}), \cdots, \mathbf{g}^{-1}(c_{i,N-1})) \cdot \mathbf{E}_i + (\mathsf{Cof}_0(c_i s_i), 0, \cdots, 0)\right) \cdot \mathbf{F}_i^{-1},\end{aligned}$$

where $\mathsf{Cof}_0(c_i s_i)$ denotes the constant term of the polynomial $c_i s_i \in R_q$. Therefore, each $\mathbf{c}_i$ is a matrix NTRU based ciphertext that encrypts the constant term of $c_i s_i$ under the secret $\mathbf{F}_i \in \mathbb{Z}_q^{n\times n}$. Thus, the output $\overline{ct} = (\mathbf{c}_1, \cdots, \mathbf{c}_k)$ is a MNTRU-based multi-key ciphertext that encrypts $m$ under the secret $\{\mathbf{F}_i\}_{i\in[k]}$.

Moreover, since the variance of the noise distribution used in generating $\mathbf{KSK}_i$ is $Var(e_{ks})$, we have $Var(\hat{\mathbf{c}}_i \cdot \mathbf{E}_i) \le \frac{B_{ks}^2}{12}Nd_{ks}Var(e_{ks})$ where $\hat{\mathbf{c}}_i = \left(\mathbf{g}^{-1}(c_{i,0}), \cdots, \mathbf{g}^{-1}(c_{i,N-1})\right)$, and the variance of the noise in the resulting ciphertext $\overline{ct}$ is upper bounded by $k\frac{B_{ks}^2}{12}Nd_{ks}Var(e_{ks}) + Var(e)$. This completes the proof. □

We use the symbol $\sigma_{NKS}^2 = k\frac{B_{ks}^2}{12}Nd_{ks}Var(e_{ks})$ to denote the variance of the increased noise (with respect to the input ciphertext) for key-switching.

## 5.3 Bootstrapping

In this subsection, we define a pair of algorithms $(\mathsf{MN.BSKG}, \mathsf{MN.BSEval})$ for bootstrapping an MNTRU-based first-layer multi-key ciphertext as follows:

- $\mathsf{MN.BSKG}(\mathbf{F}_i)$: Given a matrix $\mathbf{F}_i \in \mathbb{Z}_q^{n\times n}$ as input, run $(s_i, \mathbf{b}_i) \leftarrow \mathsf{KG}(pp')$ and set the public key as $\mathbf{PK}_i = \mathbf{b}_i$. Let $(f_{i,j})_{0\le j<n} = \mathrm{col}_0(\mathbf{F}_i)$ be the first column of the secret matrix $\mathbf{F}_i$. For $(f_{i,j})_{0\le j<n}$, let

$$\begin{cases} f_{i,j}^{+} = 1, & \text{if } f_{i,j} = 1 \\ f_{i,j}^{+} = 0, & \text{otherwise} \end{cases}, \quad \begin{cases} f_{i,j}^{-} = 1, & \text{if } f_{i,j} = -1 \\ f_{i,j}^{-} = 0, & \text{otherwise} \end{cases} \quad \text{for } 0 \le j < n.$$

  - For $i = 1$, given secret key $\mathrm{col}_0(\mathbf{F}_1) = (f_{1,0}, \ldots, f_{1,n-1}) \in \{-1, 0, 1\}^n$, create a set of ciphertexts that encrypts $\mathrm{col}_0(\mathbf{F}_1)$ under $s_1$ as follows:

$$\begin{cases} \mathbf{evk}^+_{1,0} = \mathsf{UniEnc}(f^+_{1,0}/s_1, s_1) \\ \mathbf{evk}^-_{1,0} = \mathsf{UniEnc}(f^-_{1,0}/s_1, s_1) \end{cases}, \begin{cases} \mathbf{evk}^+_{1,j} = \mathsf{UniEnc}(f^+_{1,j}, s_1) \\ \mathbf{evk}^-_{1,j} = \mathsf{UniEnc}(f^-_{1,j}, s_1) \end{cases} \text{for } j \neq 0,$$

    and creates an auxiliary ciphertext $\mathbf{evk}^*_{1,0} = \mathsf{UniEnc}(1/s_1, s_1)$. The evaluation key is defined as $\mathbf{EVK}_1 = (\mathbf{evk}^*_{1,0}, \{\mathbf{evk}^+_{1,j}, \mathbf{evk}^-_{1,j}\}_{0 \leq j < n})$.
  - For $i \neq 1$, given secret key $\mathrm{col}_0(\mathbf{F}_i) = (f_{i,0}, \ldots, f_{i,n-1}) \in \{-1, 0, 1\}^n$, create a set of ciphertext that encrypts $\mathrm{col}_0(\mathbf{F}_i)$ under $s_i$ as follows:

$$\mathbf{evk}^+_{i,j} = \mathsf{UniEnc}(f^+_{i,j}, s_i), \mathbf{evk}^-_{i,j} = \mathsf{UniEnc}(f^-_{i,j}, s_i) \text{ for } 0 \leq j < n.$$

    The evaluation key is defined as $\mathbf{EVK}_i = (\{\mathbf{evk}^+_{i,j}, \mathbf{evk}^-_{i,j}\}_{0 \leq j < n})$.

  Then it computes the key-switching key $\mathbf{KSK}_i = \mathsf{MN.KSKG}(s_i, \mathbf{F}_i)$, and outputs $(\mathbf{EVK}_i, \mathbf{KSK}_i)$.
- $\mathsf{MN.BSEval}(\overline{ct}, \{\mathbf{PK}_i, \mathbf{EVK}_i, \mathbf{KSK}_i\}_{i\in[k]}, r)$: Given a multi-key MNTRU ciphertext $\overline{ct} = (\mathbf{c}_1, \cdots, \mathbf{c}_k) \in \mathbb{Z}_q^{kn}$ that encrypts $m \in \{0, 1\}$ under the secret key $\mathbf{F}_1, \cdots, \mathbf{F}_k \in \mathbb{Z}_q^{n\times n}$, the key-triple $\{\mathbf{PK}_i, \mathbf{EVK}_i, \mathbf{KSK}_i\}_{i\in[k]}$, and a rotation polynomial $r \in R_Q$ as inputs, computes and returns $\overline{ct}'$ as described in Algorithm 1.

**Theorem 3 (Bootstrapping MNTRU-based Ciphertexts).** *Let $q, Q$ be two positive integers. Given a multi-key MNTRU ciphertext $\overline{ct} = (\mathbf{c}_1, \cdots, \mathbf{c}_k) \in \mathbb{Z}_q^{kn}$ that encrypts $m \in \{0, 1\}$ under the secret key $\mathbf{F}_1, \cdots, \mathbf{F}_k \in \mathbb{Z}_q^{n\times n}$, Algorithm 1 outputs a refreshed multi-key MNTRU ciphertext that encrypts the same message $m \in \{0, 1\}$. And the noise of the refreshed ciphertext is bounded by a Gaussian with standard deviation*

$$\beta = \sqrt{\frac{q^2}{Q^2}((2kn + 3)\sigma^2_{HP1} + kn\sigma^2_{HP2}) + \sigma^2_{NMS} + \sigma^2_{NKS}}$$

*where $\sigma^2_{HP1}$, $\sigma^2_{HP2}$ are the variance of the increased noise for the hybrid product described in Sec. 4 , $\sigma^2_{NMS}$ and $\sigma^2_{NKS}$ are the variance of the increased noise for modulus switching and key switching for NTRU described in Sec. 5.1 and 5.2, respectively.*

*Proof.* To aid in description, we use $\mathrm{MKNTRU}_{Q,\mathbf{s}}(m)$ to represent an MK-NTRU ciphertext encrypts $m \in R$ under the secret $\mathbf{s} = (s_1, \cdots, s_k) \in R^k$. To perform the correctness of the Algorithm 1, we first show that the output $\mathsf{ACC}$ of the loop (step 3 $\sim$ 13) is an MK-NTRU ciphertext

$$\mathrm{MKNTRU}_{Q,\mathbf{s}}(rX^{\sum_{i=1}^{k}\sum_{j=0}^{n-1}\hat{c}_{i,j}f_{i,j}}).$$

Let $\mathsf{ACC}_{i,j}$ be the value of $\mathsf{ACC}$ after evaluating the $i$-th iteration of the outer loop and the $j$-th iteration of the inner loop. By Lemma 2 and the correctness of the CMUX gate, it's clear that the value of $\mathbf{evk}_{1,0}$ is a uni-encryption

**Algorithm 1** $\mathsf{MN.BSEval}(\overline{ct}, \{\mathbf{PK}_i, \mathbf{EVK}_i, \mathbf{KSK}_i\}_{i\in[k]}, r)$

```
Input:
    A multi-key MNTRU ciphertext $\overline{ct} = (\mathbf{c}_1, \cdots, \mathbf{c}_k) \in \mathbb{Z}_q^{kn}$;
    The key-triple $\{\mathbf{PK}_i, \mathbf{EVK}_i, \mathbf{KSK}_i\}_{i\in[k]}$;
    A rotation polynomial $r(X) \in R_Q$.
Output:
    A multi-key MNTRU ciphertext $\overline{ct}' \in \mathbb{Z}_q^{kn}$.
1: $\hat{ct} = (\hat{\mathbf{c}}_1, \cdots, \hat{\mathbf{c}}_k) \leftarrow \left\lfloor \frac{2N\cdot\overline{ct}}{q} \right\rceil \in \mathbb{Z}_{2N}^{kn}$
2: $\mathsf{ACC} \leftarrow (r(X), \mathbf{0}) \in R_Q^k$
3: for $i = 1; i < k+1; i = i+1$ do
4:     for $j = 0; j < n; j = j+1$ do
5:         if $i = 1, j = 0$ then
6:             $\mathbf{evk}_{1,0} \leftarrow \mathbf{evk}^*_{1,0} + (X^{\hat{c}_{1,0}} - 1)\mathbf{evk}^+_{1,0} + (X^{-\hat{c}_{1,0}} - 1)\mathbf{evk}^-_{1,0}$
7:             $\mathsf{ACC} \leftarrow \mathsf{HbProd}(\mathsf{ACC}, \{\mathbf{PK}_l\}_{l\in[k]}, \mathbf{evk}_{1,0})$
8:         else
9:             $\mathbf{evk}_{i,j} \leftarrow \mathbf{evk}^+_{i,j} - \mathbf{evk}^-_{i,j} \cdot X^{-\hat{c}_{i,j}}$
10:            $\mathsf{ACC} \leftarrow \mathsf{ACC} + \mathsf{HbProd}((X^{\hat{c}_{i,j}} - 1)\mathsf{ACC}, \{\mathbf{PK}_l\}_{l\in[k]}, \mathbf{evk}_{i,j})$
11:        end if
12:    end for
13: end for
14: $\mathsf{ACC} \leftarrow \mathsf{NModSwitch}(\mathsf{ACC}, q)$
15: $\overline{ct}' \leftarrow \mathsf{MN.KS}(\mathsf{ACC}, \{\mathbf{KSK}_i\}_{i\in[k]})$
16: return $\overline{ct}'$
```

ciphertext of $(1 + (X^{\hat{c}_{1,0}} - 1)\cdot f^+_{1,0} + (X^{-\hat{c}_{1,0}} - 1)\cdot f^-_{1,0})/s_1 = X^{\hat{c}_{1,0}f_{1,0}}/s_1$, and we can easily check that $\mathsf{ACC}_{1,0}$ is $\mathrm{MKNTRU}_{Q,\mathbf{s}}(rX^{\hat{c}_{1,0}f_{1,0}})$ by the property of hybrid product. Then, in the next iteration, $\mathbf{evk}_{1,1} = \mathbf{evk}^+_{1,1} - \mathbf{evk}^-_{1,1}\cdot X^{-\hat{c}_{1,1}} = \mathsf{UniEnc}((f^+_{1,1} - X^{-\hat{c}_{1,1}}\cdot f^-_{1,1}), s_1)$ by Lemma 2 and $\mathsf{ACC}_{1,1} = \mathsf{ACC}_{1,0} + \mathsf{HbProd}((X^{\hat{c}_{1,1}} - 1)\mathsf{ACC}_{1,0}, \{\mathbf{b}_l\}_{l\in[k]}, \mathbf{evk}_{1,1})$. By Lemma 3, we have that

$$\langle \mathsf{ACC}_{1,1}, \mathbf{s}\rangle \approx rX^{\hat{c}_{1,0}f_{1,0}}(1 + (X^{\hat{c}_{1,1}} - 1)(f^+_{1,1} - X^{-\hat{c}_{1,1}}\cdot f^-_{1,1})) = rX^{\hat{c}_{1,0}f_{1,0}+\hat{c}_{1,1}f_{1,1}}$$

which is also an MKNTRU ciphertext $\mathrm{MKNTRU}_{Q,\mathbf{s}}(rX^{\hat{c}_{1,0}f_{1,0}+\hat{c}_{1,1}f_{1,1}})$.

Now, it suffices to show that if $\mathsf{ACC}_{h,m} = \mathrm{MKNTRU}_{Q,\mathbf{s}}(rX^{\sum_{i=1}^{h}\sum_{j=0}^{m}\hat{c}_{i,j}f_{i,j}})$, then $\mathsf{ACC}_{h,m+1}$ also has the same formula. Note that at the $h$-th iteration of the outer loop and the $m+1$-th iteration of the inner loop, the algorithm will first compute $\mathbf{evk}_{h,m+1} = \mathbf{evk}^+_{h,m+1} - \mathbf{evk}^-_{h,m+1}\cdot X^{-\hat{c}_{h,m+1}}$ and then compute $\mathsf{ACC}_{h,m+1} = \mathsf{ACC}_{h,m} + \mathsf{HbProd}((X^{\hat{c}_{h,m+1}} - 1)\mathsf{ACC}_{h,m}, \{\mathbf{b}_l\}_{l\in[k]}, \mathbf{evk}_{h,m+1})$. And we can easily check $\mathbf{evk}_{h,m+1}$ is $\mathsf{UniEnc}(f^+_{h,m+1} - X^{-\hat{c}_{h,m+1}}\cdot f^-_{h,m+1})$. By Lemma 3, we have that

$$\begin{aligned}\langle \mathsf{ACC}_{h,m}, \mathbf{s}\rangle &\approx rX^{\sum_{i=1}^{h}\sum_{j=0}^{m}\hat{c}_{i,j}f_{i,j}}(1 + (f^+_{h,m+1} - X^{-\hat{c}_{h,m+1}}\cdot f^-_{h,m+1}))\\ &= rX^{\sum_{i=1}^{h}\sum_{j=0}^{m+1}\hat{c}_{i,j}f_{i,j}}.\end{aligned}$$

Also, we can check that if $\mathsf{ACC}_{h,n-1} = \text{MKNTRU}_{Q,\mathbf{s}}(rX^{\sum_{i=1}^{h}\sum_{j=0}^{n-1}\hat{c}_{i,j}f_{i,j}})$, then $\mathsf{ACC}_{h+1,n-1}$ also has the same formula. We omit the details here since it's the same as the previous analysis. Thus, after the loop, we have that $\mathsf{ACC}_{k,n-1} = \text{MKNTRU}_{Q,\mathbf{s}}(rX^{\sum_{i=1}^{k}\sum_{j=0}^{n-1}\hat{c}_{i,j}f_{i,j}})$. And by Lemma 8 and Lemma 9, the output of Algorithm 1 is a refreshed multi-key MNTRU ciphertext.

Now, we analyze the noise of the resulting ciphertext. We begin by analyzing the noise in $\mathsf{ACC}$ during the loop (steps $3 \sim 13$). During the first iteration of the outer loop and the zeroth iteration of the inner loop, the variance of the noise in $\mathbf{evk}_{1,0}$ is $5Var(e)$, where the factor of 5 arises from the CMux gate. Consequently, the variance of the noise in $\mathsf{ACC}_{1,0}$ is $5\sigma_{HP1}^2 + \sigma_{HP2}^2$. In the next iteration, the variance of the noise in $\mathbf{evk}_{1,1}$ reduces to $2Var(e)$, while the variance of the additional noise in $\mathsf{ACC}_{1,1}$ becomes $2\sigma_{HP1}^2 + \sigma_{HP2}^2$. Therefore, after the loop (step $3 \sim 13$), the variance of the noise is bounded by

$$5\sigma_{HP1}^2 + \sigma_{HP2}^2 + (kn-1)(2\sigma_{HP1}^2 + \sigma_{HP2}^2) = (2kn+3)\sigma_{HP1}^2 + kn\sigma_{HP2}^2.$$

By Lemma 8, we have the noise variance after the modulus switching from $Q$ to $q$ in step 14 is bounded by $\frac{q^2}{Q^2}((2kn+3)\sigma_{HP1}^2 + kn\sigma_{HP2}^2) + \sigma_{NMS}^2$. By Lemma 9, we have the noise variance after the key switching in step 15 is bounded by

$$\frac{q^2}{Q^2}((2kn+3)\sigma_{HP1}^2 + kn\sigma_{HP2}^2) + \sigma_{NMS}^2 + \sigma_{NKS}^2.$$

This finally completes the proof. □

**Theorem 4.** *If the standard deviation of refreshed noise for the output in Algorithm 1 satisfies Theorem 3 except with negligible probability and the modulus satisfies $q = \tilde{O}(kn^{1.5})$, then the output of Algorithm 1 can be correctly decrypted except with negligible probability.*

*Proof.* Since $\sigma_{HP1}^2 = (Var(s)kN+1)\frac{dNB^2}{12}Var(e)$, $\sigma_{HP2}^2 = Var(s)kN\frac{dNB^2}{12}Var(e)$, $\sigma_{NMS}^2 = 1 + \frac{\sum_{i=1}^{k}\|s_i\|}{12}$ and $\sigma_{NKS}^2 = k\frac{B_{ks}^2}{12}Nd_{ks}Var(e_{ks})$ where $Var(s) = O(1)$ for uniform ternary distribution, $q/Q$, $B$, $B_{ks}, Var(e), Var(e_{ks}) \in O(1)$, $d, d_{ks} \in O(\log n)$, and $\|\mathbf{s}_i\| \in O(n)$ we have the final noise of the output ciphertext in Algorithm 1 is $\tilde{O}(kn^{1.5})$ except with negligible probability. By lemma 1, the correctness will hold as long as the standard deviation of refreshed noise after bootstrapping is less than $\frac{q-12}{24}$. Using six standard deviations as a high-probability bound of the refreshed noise after bootstrapping, we have that $q = \tilde{O}(kn^{1.5})$. □

## 6 Bootstrapping First-layer Multi-Key LWE Ciphertexts

In this section, we describe how to modify our second-layer scheme to bootstrap a standard first-layer multi-key LWE ciphertext. In Sec. 6.1, we describe the modulus switching for multi-key LWE ciphertext. In Sec. 6.2, we propose a light-key switching technique. In Sec. 6.3, we present the bootstrapping algorithm and analyze its correctness.

### 6.1 Modulus Switching for Multi-Key LWE ciphertext

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 10 (Modulus Switching for Multi-Key LWE ciphertext).** *Given as input a multi-key LWE ciphertext* $\overline{ct} = (b, \mathbf{a}_1, \cdots, \mathbf{a}_k) \in \mathbb{Z}_Q^{kN+1}$ *under secret key* $\mathbf{s} = (1, \mathbf{s}_1, \cdots, \mathbf{s}_k) \in \mathbb{Z}^{kN+1}$*, the LWE modulus switching procedure is defined as*

$$b' = \left\lfloor \frac{q}{Q} b \right\rceil \quad \textit{and} \quad \mathbf{a}'_i = \left\lfloor \frac{q}{Q} \mathbf{a}_i \right\rceil \quad \textit{for } i \in [k].$$

*Then,* $(b', \mathbf{a}'_1, \cdots, \mathbf{a}'_k)$ *is a multi-key LWE ciphertext that encrypts the same message under the same secret key* $\mathbf{s} \in \mathbb{Z}^{kN+1}$*. Moreover, if the noise variance of* $\overline{ct}$ *is* $\alpha^2$*, then the noise variance of the ciphertext after modulus switching is bounded by* $(\frac{q}{Q})^2\alpha^2 + \frac{1+\sum_{i=1}^k \|\mathbf{s}_i\|}{12}$.

*Proof.* Let $b + \sum_{i=1}^k \langle \mathbf{a}_i, \mathbf{s}_i \rangle = \left\lfloor \frac{Q}{4} \right\rceil m + e$. By definition, we have

$$\begin{aligned} b' + \sum_{i=1}^k \langle \mathbf{a}'_i, \mathbf{s}_i \rangle &= \left\lfloor \frac{q}{Q} \cdot b \right\rceil + \sum_{i=1}^k \left\lfloor \frac{q}{Q} \langle \mathbf{a}_i, \mathbf{s}_i \rangle \right\rceil \\ &= \frac{q}{Q} \cdot b + \sum_{i=1}^k \frac{q}{Q} \langle \mathbf{a}_i, \mathbf{s}_i \rangle + \epsilon_0 + \sum_{i=1}^k \langle \epsilon_i, \mathbf{s}_i \rangle \\ &= \frac{q}{4} \cdot m + \frac{q}{Q} \cdot e + \epsilon_0 + \sum_{i=1}^k \langle \epsilon_i, \mathbf{s}_i \rangle \end{aligned}$$

where $\epsilon_0 \in [-1/2, 1/2]$ and $\epsilon_i \in [-1/2, 1/2]^N$ are uniformly random. The noise variance of the ciphertext after modulus switching is bounded by

$$(\frac{q}{Q})^2\alpha^2 + \frac{1 + \sum_{i=1}^k \|\mathbf{s}_i\|}{12}.$$

This completes the proof. □

For modulus switching, we denote $\sigma_{LMS}^2 = \frac{1+\sum_{i=1}^k \|\mathbf{s}_i\|}{12}$ as the variance of the increased noise (relative to the input ciphertext).

### 6.2 Light Key Switching for Multi-key LWE ciphertext

Let $(b, \mathbf{a}_1, \cdots, \mathbf{a}_k) \in \mathbb{Z}_q^{kN+1}$ be an MK-LWE ciphertext under $(1, \mathbf{s}_1, \cdots, \mathbf{s}_k) \in \mathbb{Z}^{kN+1}$. To switch the secret from $(1, \mathbf{s}_1, \cdots, \mathbf{s}_k) \in \mathbb{Z}^{kN+1}$ to $(1, \mathbf{z}_1, \cdots, \mathbf{z}_k) \in \mathbb{Z}^{kn+1}$ for some $\mathbf{s}_i = (s_{i,j})_{0 \le j \le N-1}$ and $\mathbf{z}_i = (z_{i,j})_{0 \le j < n}$. In prior methods [9,25], each party $i$ independently generates the key switching key as a set of LWE ciphertexts that encrypts each value $v \cdot B_{ks}^l \cdot s_{i,j}$ under the secret $\mathbf{z}_i = (z_{i,j})_{0 \le j < n}$

where $j \in \mathbb{Z}_N$, $l \in \mathbb{Z}_{d_{ks}}$, $v \in [B_{ks}-1]$ (See Appendix A). Instead, we sequentially decode those values

$$\{B_{ks}^0 s_{i,0}, 2B_{ks}^0 s_{i,0}, \cdots, (B_{ks}-1)B_{ks}^{d_{ks}-1} s_{i,N-1}\}$$

into the coefficients of $t$ polynomials $m_1(X), \cdots, m_t(X)$ in $R_q = \mathbb{Z}_q[X]/(X^N+1)$ where $t = (B_{ks}-1)d_{ks}$.

In this setting, for any value $v \cdot B_{ks}^l \cdot s_{i,j}$, one can check that it be decoded in $x$-th coefficient $m_y(X)$ polynomial where $\left\lceil y = \frac{(B_{ks}-1)(jd_{ks}+l)+v}{N} \right\rceil$ and $x = (B_{ks}-1)(jd_{ks}+l)+v-1 \mod N$. To perform key-switching, we only need to employ the sample extraction algorithm in [22] (Sec. IV, step 1), which we refer to as $\mathsf{SamExt}$ for clarity, to extract the corresponding LWE ciphertext from the RLWE ciphertext at the desired position. Notably, we only need to extract the first $n$ coefficients.

Formally, we define two algorithms ($\mathsf{ML.PKSKG}$, $\mathsf{ML.KS}$) as follows.

- $\mathsf{ML.PKSKG}(\mathbf{z}_i, \mathbf{s}_i)$: Given two vectors $\mathbf{z}_i = (z_{i,0}, \cdots, z_{i,n-1}) \in \mathbb{Z}^n$, $\mathbf{s} = (s_{i,0}, \cdots, s_{i,N-1}) \in \mathbb{Z}^N$ and two integers $q$, $B_{ks}$ as input, the algorithm computes $d_{ks} = \lceil \log_{B_{ks}} q \rceil$ and sets $z_i = \sum_{j=0}^{n-1} z_{i,j}X^j + 0X^n \cdots + 0X^{N-1} \in R_q$. Then it decodes the values $v \cdot B_{ks}^l s_{i,j}$ into $t$ polynomials $m_1(X), \cdots, m_t(X)$ where $j \in \mathbb{Z}_N$, $l \in \mathbb{Z}_{d_{ks}}$, $v \in [B_{ks}-1]$ as described above. Then it samples polynomial $a'_{i,j} \in R_q$ uniformly at random and $e_{i,j} \in R_q$ from some noise distribution and computes $\mathbf{PKSK}_{i,j} = (b'_{i,j}, a'_{i,j})$ where $b'_{i,j} = a'_{i,j} \cdot z_i + e_{i,j} + m_j(X) \in R_q$ for $j \in [t]$. Finally, it outputs $\mathbf{PKSK}_i = \{\mathbf{PKSK}_{i,j}\}_{j\in[t]}$ as the key-switching key of party $i$.
- $\mathsf{ML.KS}(\overline{ct}, \{\mathbf{PKSK}_i\}_{i\in[k]})$: Given as input a multi-key LWE ciphertext

$$\overline{ct} = \left( b = \sum_{i=1}^{k} \langle \mathbf{a}_i, \mathbf{s}_i \rangle + e + \lfloor \frac{q}{4} \rceil \cdot m, \mathbf{a}_1, \cdots, \mathbf{a}_k \right) \in \mathbb{Z}_q^{kN+1}$$

for $\mathbf{a}_i = (a_{i,j})_{j\in\mathbb{Z}_N} \in \mathbb{Z}_q^N$ and the key-switching keys $\{\mathbf{PKSK}_i\}_{i\in[k]}$ of keys associated with $\overline{ct}$, the algorithm first computes $\mathbf{g}^{-1}(a_{i,j}) = (v_{i,j,l})_{l\in\mathbb{Z}_{d_{ks}}}$ for each $i \in [k]$ and $j \in \mathbb{Z}_N$ and then computes as follows:
  - Key reconstruction. Extrac LWE ciphertexts

$$\{(b'_{i,j,l,v_{i,j,l}}, \mathbf{a}'_{i,j,l,v_{i,j,l}})\}_{i\in[k], j\in\mathbb{Z}_n, l\in\mathbb{Z}_{d_{ks}}}$$

  where $b'_{i,j,l,v_{i,j,l}} = \left\langle \mathbf{a}'_{i,j,l,v_{i,j,l}}, \mathbf{z}_i \right\rangle + e_{i,j,l,v_{i,j,l}} + v_{i,j,l} \cdot s_{i,j} \cdot B_{ks}^l$ from $\{\mathbf{PKSK}_i\}_{i\in[k]}$.
  - Key switching. Compute

$$b'_i = \sum_{j=0}^{N-1} \sum_{l=0, v_{i,j,l}\neq 0}^{d_{ks}-1} b'_{i,j,l,v_{i,j,l}} \quad \text{and} \quad \mathbf{a}'_i = \sum_{j=0}^{N-1} \sum_{l=0, v_{i,j,l}\neq 0}^{d_{ks}-1} \mathbf{a}'_{i,j,l,v_{i,j,l}},$$

  and let $b' = b + \sum_{i=1}^{k} b'_i$. Finally, the algorithm outputs a multi-key LWE ciphertext $\overline{ct}' = (b', \mathbf{a}'_1, \ldots, \mathbf{a}'_k) \in \mathbb{Z}^{kn+1}$.

**Lemma 11 (Key-switching for MK-LWE ciphertext).** *Let $\mathbf{s}_1, \cdots, \mathbf{s}_k \in \mathbb{Z}^N$ and $\mathbf{z}_1, \cdots, \mathbf{z}_k \in \mathbb{Z}^n$ be some vectors. Let $\overline{ct} = (b, \mathbf{a}_1, \cdots, \mathbf{a}_k) \in \mathbb{Z}^{kN+1}$ be a multi-key LWE ciphertext that encrypts $m \in \{0,1\}$ under the secret key $(1, \mathbf{s}_1, \cdots, \mathbf{s}_k) \in \mathbb{Z}^{kN+1}$. Let $\mathbf{PKSK}_i = \mathsf{ML.PKSKG}(\mathbf{z}_i, \mathbf{s}_i)$ be the key-switching key of party $i$. We have that $\overline{\mathbf{c}}' = \mathsf{ML.KS}(\overline{ct}, \{\mathbf{PKSK}_i\}_{i\in[k]}) \in \mathbb{Z}_q^{kn+1}$ is a valid multi-key LWE ciphertext that encrypts the same message $m \in \{0,1\}$ under the secret key $(1, \mathbf{z}_1, \cdots, \mathbf{z}_k) \in \mathbb{Z}^{kn+1}$.*

*Moreover, if the variance of the noise in $\overline{ct}$ is $Var(e)$ and the variance of the noise distribution used in generating $\mathbf{PKSK}_i$ is $Var(e_{ks})$, we have that the variance of the noise after key-switching is upper bounded by $Var(e) + kd_{ks}NVar(e_{ks})$.*

*Proof.* By definition, we have that $\overline{ct} = (b = \sum_{i=1}^{k} \langle \mathbf{a}_i, \mathbf{s}_i \rangle + e + \lfloor \frac{q}{4} \rceil \cdot m, \mathbf{a}_1, \cdots, \mathbf{a}_k)$ for some $\mathbf{a}_i = (a_{i,0}, \cdots, a_{i,N-1}) \in \mathbb{Z}_q^N$ and that $\mathbf{PKSK}_i = \{\mathbf{PKSK}_{i,j}\}_{j\in[t]}$ for some $\mathbf{PKSK}_{i,j} = (b'_{i,j} = a'_{i,j} \cdot z_i + e_{i,j} + m_j(X), a'_{i,j})$ where $\{m_j(X)\}_{j\in[t]}$ are polynomial that sequently decode the values

$$\{B_{ks}^0 s_{i,0}, 2B_{ks}^0 s_{i,0}, \cdots, (B_{ks}-1)B_{ks}^{d_{ks}-1} s_{i,N-1}\}$$

into their coefficients. By a trivial sample extraction, we get some LWE ciphertexts $\text{LWE}(v_{i,j,l}) = \left(b'_{i,j,l,v_{i,j,l}}, \mathbf{a}'_{i,j,l,v_{i,j,l}}\right)$ where

$$b'_{i,j,l,v_{i,j,l}} = \left\langle \mathbf{a}'_{i,j,l,v_{i,j,l}}, \mathbf{z}_i \right\rangle + e_{i,j,l,v_{i,j,l}} + v_{i,j,l} \cdot s_{i,j} \cdot B_{ks}^l$$

without increasing the noise variance. This lemma directly follows from the fact that $a_{i,j} = \sum_{l=0}^{d_{ks}-1} v_{i,j,l} B_{ks}^l$ and that

$$\begin{aligned}
b' &= \sum_{i=1}^{k} \sum_{j=0}^{N-1} \sum_{l=0, v_{i,j,l}\neq 0}^{d_{ks}-1} b'_{i,j,l,v_{i,j,l}} + b \\
&= \sum_{i=1}^{k} \sum_{j=0}^{N-1} \sum_{l=0, v_{i,j,l}\neq 0}^{d_{ks}-1} (\left\langle \mathbf{a}'_{i,j,l,v_{i,j,v}}, \mathbf{z}_i \right\rangle + e_{i,j,l,v_{i,j,v}} + v_{i,j,l} \cdot s_{i,j} \cdot B_{ks}^l) + b \\
&= \sum_{i=1}^{k} \langle \mathbf{a}'_i, \mathbf{z}_i \rangle + \sum_{i=1}^{k} \sum_{j=0}^{N-1} \sum_{l=0, v_{i,j,l}\neq 0}^{d_{ks}-1} e_{i,j,l,v_{i,j,v}} + \sum_{i=1}^{k} \sum_{j=0}^{N-1} a_{i,j} s_{i,j} + b \\
&= \sum_{i=1}^{k} \langle \mathbf{a}'_i, \mathbf{z}_i \rangle + \sum_{i=1}^{k} \sum_{j=0}^{N-1} \sum_{l=0, v_{i,j,l}\neq 0}^{d_{ks}-1} e_{i,j,l,v_{i,j,v}} + e + \lfloor \frac{q}{4} \rceil \cdot m.
\end{aligned}$$

Therefore, the variance of the noise for the output ciphertext is bounded by $Var(e) + kd_{ks}NVar(e_{ks})$. This completes the proof. □

For key switching, we denote $\sigma_{LKS}^2 = kd_{ks}NVar(e_{ks})$ as the variance of the increased noise.

The correctness of our light key switching for multi-key LWE ciphertexts is guaranteed by Lemma 11. We now proceed to provide the necessary security analysis.

*Security.* Our $\mathsf{ML.PKSKG}$'s security can be theoretically based on the standard RLWE and NTRU assumptions as long as the secret in **PKSK** has sufficient large entropy for appropriate choices of parameters.

Specifically,the RLWE ciphertexts generated by our ML.PKSKG basically use a secret key $z_i$ chosen from a distribution always having zeros in the last $N - n$ coefficients over $R_q$, which is different from standard RLWE ciphertexts. However, as long as $z_i$ has sufficient large entropy for appropriate choices of parameters, the corresponding RLWE instances well fit the setting of Entropic RLWE problem, which in turn is provably hard under the standard RLWE and NTRU (a.k.a DSPR) assumptions [6].

Regarding concrete security, one often translates an RLWE instance in dimension $N$ into $N$ samples of LWE instances in the same dimension $N$, as the best-known lattice attacks do not seem to offer additional advantages in solving RLWE compared to standard LWE, and one can naturally treat a single RLWE instance as $N$ samples of LWE instances corresponding to a public anti-circular matrix defined by the ring multiplication.

In our case, as the last $N - n$ coefficients of the secret key are zeros, it is equivalent to treat the last $N - n$ columns of the public (anti-circular) matrix as zeros when we translate the RLWE instance into LWE instances. In particular, our RLWE instance in $\mathsf{ML.PKSKG}$ can be translated into $N$ samples of LWE instances in dimension $n$ (instead of $N$ for a standard RLWE instance). We have actually taken this into account when choosing our concrete parameters using the LWE estimator [3] in Sec. 7.

## 6.3 Bootstrapping

In this subsection, we define a pair of algorithms $(\mathsf{ML.BSKG}, \mathsf{ML.BSEval})$ for bootstrapping an MK-LWE ciphertext as follows:

- $\mathsf{ML.BSKG}(\mathbf{z}_i)$: Given a secret key $\mathbf{z}_i = (z_{i,j})_{0\le j<n} \in \{0,1\}^n$ as input, the algorithm first runs $(s_i, \mathbf{b}_i) \leftarrow \mathsf{KG}(pp')$ and sets the public key as $\mathbf{PK}_i = \mathbf{b}_i$.
  - For $i = 1$, given secret key $\mathbf{z}_1 = (z_{1,j})_{0\le j<n} \in \{0,1\}^n$, create a set of ciphertexts that encrypts $\mathbf{z}_1$ under $s_1$ as follows:
  $$\mathbf{evk}_{1,0} = \mathsf{UniEnc}(z_{1,0}/s_1, s_1), \quad \mathbf{evk}_{1,j} = \mathsf{UniEnc}(z_{1,j}, s_1) \text{ for } j \neq 0$$
  and creates an auxiliary ciphertext $\mathbf{evk}^*_{1,0} = \mathsf{UniEnc}(1/s_1, s_1)$. The evaluation key is defined as $\mathbf{EVK}_1 = (\mathbf{evk}^*_{1,0}, \{\mathbf{evk}_{1,j}\}_{0\le j<n})$.
  - For $i \neq 1$, given secret key $\mathbf{z}_i = (z_{i,j})_{0\le j<n} \in \{0,1\}^n$, create a set of ciphertexts as follows
  $$\mathbf{evk}_{i,j} = \mathsf{UniEnc}(z_{i,j}, s_i) \text{ for } 1 \le j < n.$$
  The evaluation key is defined as $\mathbf{EVK}_i = (\{\mathbf{evk}_{i,j}\}_{0\le j<n})$.

  Then it computes the key-switching key $\mathbf{PKSK}_i = \mathsf{ML.PKSKG}(\mathbf{z}_i, \mathbf{s}_i)$, and outputs $(\mathbf{EVK}_i, \mathbf{PKSK}_i)$.

- $\mathsf{ML.BSEval}(\overline{ct}, \{\mathbf{PK}_i, \mathbf{EVK}_i, \mathbf{PKSK}_i\}_{i\in[k]}, r)$: Given a multi-key LWE ciphertext $\overline{ct} = (b, \mathbf{a}_1, \cdots, a_k) \in \mathbb{Z}_q^{kn+1}$ under the secret $(1, \mathbf{z}_1, \cdots, \mathbf{z}_k) \in \mathbb{Z}^{kn+1}$, the key-triple $\{\mathbf{PK}_i, \mathbf{EVK}_i, \mathbf{PKSK}_i\}_{i\in[k]}$, and a rotation polynomial as inputs, computes and returns $\overline{ct}'$ as described in algorithm 2.

**Algorithm 2** $\mathsf{ML.BSEval}(\overline{ct}, \{\mathbf{PK}_i, \mathbf{EVK}_i, \mathbf{PKSK}_i\}_{i\in[k]}, r)$

```
Input:
    A multi-key LWE ciphertext $\overline{ct} = (b, \mathbf{a}_1, \cdots, \mathbf{a}_k) \in \mathbb{Z}_q^{kn+1}$;
    The key-triple $\{\mathbf{PK}_i, \mathbf{EVK}_i, \mathbf{PKSK}_i\}_{i\in[k]}$;
    A rotation polynomial $r(X) \in R_Q$.
Output:
    A multi-key LWE ciphertext $\overline{ct}' \in \mathbb{Z}_q^{kn+1}$.
1: $\hat{ct} = (\hat{b}, \hat{\mathbf{a}}_1, \cdots, \hat{\mathbf{a}}_k) \leftarrow \left\lfloor \frac{2N\cdot\overline{ct}}{q} \right\rceil \in \mathbb{Z}_{2N}^{kn+1}$
2: $\mathsf{ACC} \leftarrow (r(X)X^{\hat{b}}, \mathbf{0}) \in R_Q^k$
3: for $i = 1; i < k+1; i = i+1$ do
4:     for $j = 0; j < n; j = j+1$ do
5:         if $i = 1, j = 0$ then
6:             $\mathbf{evk}_{1,0} \leftarrow \mathbf{evk}^*_{1,0} + (X^{\hat{a}_{1,0}} - 1)\mathbf{evk}_{1,0}$
7:             $\mathsf{ACC} \leftarrow \mathsf{HbProd}(\mathsf{ACC}, \{\mathbf{PK}_l\}_{l\in[k]}, \mathbf{evk}_{1,0})$
8:         else
9:             $\mathsf{ACC} \leftarrow \mathsf{ACC} + \mathsf{HbProd}((X^{\hat{a}_{i,j}} - 1)\mathsf{ACC}, \{\mathbf{PK}_l\}_{l\in[k]}, \mathbf{evk}_{i,j})$
10:        end if
11:    end for
12: end for
13: $\overline{ct}_1 \leftarrow \mathsf{SamExt}(\mathsf{ACC})$
14: $\overline{ct}_2 \leftarrow \mathsf{LModSwitch}(\overline{ct}_1, q)$
15: $\overline{ct}' \leftarrow \mathsf{MN.KS}(\overline{ct}_2, \{\mathbf{PKSK}_i\}_{i\in[k]})$
16: return $\overline{ct}'$
```

**Theorem 5 (Bootstrapping LWE-based Ciphertexts).** *Let $q, Q$ be two positive integers. Given a multi-key LWE ciphertext $\overline{ct} = (b, \mathbf{a}_1, \cdots, \mathbf{a}_k) \in \mathbb{Z}_q^{kn+1}$ that encrypts $m \in \{0,1\}$ under the secret key $(1, \mathbf{z}_1, \cdots, \mathbf{z}_k) \in \mathbb{Z}^{kn+1}$, Algorithm 2 outputs a refreshed multi-key LWE ciphertext that encrypts the same message $m \in \{0,1\}$. The noise of the refreshed ciphertext is bounded by a Gaussian with standard deviation*

$$\beta = \sqrt{\frac{q^2}{Q^2}((kn+1)\sigma_{HP1}^2 + kn\sigma_{HP2}^2) + \sigma_{LMS}^2 + \sigma_{LKS}^2},$$

*where $\sigma_{HP1}^2$, $\sigma_{HP2}^2$ are the variance of the increased noise for hybrid product described in Sec. 4, $\sigma_{LMS}^2$ and $\sigma_{LKS}^2$ are the variance of the increased noise for modulus switching and key switching for LWE ciphertexts described in Sec. 6.1 and Sec. 6.2, respectively.*

*Proof.* The correctness of Algorithm 2 directly follows from the properties of uni-encryption in Lemma 2, the correctness of the hybrid product in Lemma 3, the modulus switching in Lemma 10, and the LWE key switching in Lemma 11. The analysis process parallels that of Theorem 3, we omit it for brevity.

Next, we analyze the noise of the resulting ciphertext. We begin our analysis by evaluating the noise within $\mathsf{ACC}$ during the loop (steps $3 \sim 13$). In the first iteration of the outer loop and the initial iteration of the inner loop, the variance of noise in $\mathbf{evk}_{1,0}$ is $2Var(e)$, with the factor of 2 attributed to the CMUX gate. Therefore, the variance of the noise in $\mathsf{ACC}_{1,0}$ is $2\sigma^2_{HP1} + \sigma^2_{HP2}$. In the next iteration, the variance of the noise in $\mathbf{evk}_{1,1}$ reduces to $Var(e)$, while the variance of the additional noise in $\mathsf{ACC}_{1,1}$ becomes $\sigma^2_{HP1} + \sigma^2_{HP2}$. Therefore, after the loop (step $3 \sim 12$), the variance of the noise is bounded by

$$2\sigma^2_{HP1} + \sigma^2_{HP2} + (kn-1)(\sigma^2_{HP1} + \sigma^2_{HP2}) = (kn+1)\sigma^2_{HP1} + kn\sigma^2_{HP2}.$$

By Lemma 8, we have the noise variance after the modulus switching from $Q$ to $q$ in step 14 is bounded by $\frac{q^2}{Q^2}((kn+1)\sigma^2_{HP1} + kn\sigma^2_{HP2}) + \sigma^2_{LMS}$. By Lemma 9, we have the noise variance after the key switching in step 15 is bounded by

$$\frac{q^2}{Q^2}((kn+1)\sigma^2_{HP1} + kn\sigma^2_{HP2}) + \sigma^2_{LMS} + \sigma^2_{LKS}.$$

This finally completes the proof. □

**Theorem 6.** *If the standard deviation of the noise for the output in Algorithm 2 satisfies Theorem 5 except with negligible probability and the modulus satisfies $q = \tilde{O}(kn^{1.5})$, then the output of Algorithm 2 can be correctly decrypted except with negligible probability.*

*Proof.* Since $\sigma^2_{HP1} = (Var(s)kN+1)\frac{dNB^2Var(e)}{12}$, $\sigma^2_{HP2} = Var(s)kN\frac{dNB^2Var(e)}{12}$, $\sigma^2_{LMS} = \frac{1+\sum_{i=1}^{k}\|\mathbf{s}_i\|}{12}$ and $\sigma^2_{LKS} = kd_{ks}NVar(e_{ks})$ where $Var(s) = O(1)$ for uniform ternary distribution, $q/Q$, $B$, $B_{ks}, Var(e), Var(e_{ks}) \in O(1)$, $d, d_{ks} \in O(\log n)$, and $\|\mathbf{s}_i\| \in O(n)$ we have that the standard deviation of the noise in Algorithm 2 after bootstrapping is $\tilde{O}(kn^{1.5})$ except with negligible probability. By the reference [15], the NAND gate correctness of MK-LWE ciphertext will hold as long as the refreshed noise after bootstrapping is less than $\frac{q}{16}$. Using six standard deviations as a high-probability bound of the refreshed noise after bootstrapping, we have that $q = \tilde{O}(kn^{1.5})$. □

## 7 Analysis and Implementation

In this section, we first analyze the computational complexity and key size of our algorithms proposed in Sec. 5 and Sec. 6, and then give a comparison to the prior works. Finally, we present the implementation results.

## 7.1 Analysis and Comparison

In Table 2, we give a theoretical comparison of the bootstrapping algorithms among our MK-FHEs, CCS [9] and its variant KMS [25], where $n$ is lattice dimension, $k$ is the number of keys, $d$ is the gadget decomposition dimension, #mul denotes the number of multiplications in $R_Q$ for performing the bootstrapping, and $\#R_Q$ (resp.,$\#bits$) denotes the number of $R_Q$ elements (resp., bits) for storing the evaluation key (resp., key-switching key) at each party. One can see that our Alg. 2 outperforms CCS [9] and KMS [25] in both the evaluation key size and the computational efficiency. Alg. 1 and Alg. 2 have the same computational complexity, but the evaluation key size of Alg. 1 is almost twice that of Alg. 2. This is primarily because the secret key of the first layer in Alg. 1 is ternary, which requires us to employ a more complex CMUX gate. Moreover, the noise magnitude in the KMS scheme is larger, which means that larger parameters need to be chosen in implementation, resulting in higher costs.

**Table 2.** Comparison of different bootstrapping methods among ours vs. [9,25]

| Method | #mul | $\#R_Q$ | $\#bits$ | noise |
|---|---|---|---|---|
| CCS [9] | $4k(k+1)nd$ | $3dn$ | $nNB_{ks}d_{ks}\log_2 Q_{ks}$ | $\tilde{O}(kn^{1.5})$ |
| KMS [25] | $4nkd^2+2k(2k+3)d$ | $(4n+3)d$ | $nNB_{ks}d_{ks}\log_2 Q_{ks}$ | $\tilde{O}(kn^2)$ |
| Ours (Alg. 1) | $(2k+1)knd$ | $(4n+2)d$ | $nNd_{ks}\log_2 Q_{ks}$ | $\tilde{O}(kn^{1.5})$ |
| Ours (Alg. 2) | $(2k+1)knd$ | $(2n+2)d$ | $2N(B_{ks}-1)d_{ks}\log_2 Q_{ks}$ | $\tilde{O}(kn^{1.5})$ |

## 7.2 Recommended Parameters

We observe that the state-of-the-art works CCS [9] and KMS [25] only achieve a security level of 100 bits, as noticed in [1]. For a fair comparison, we first select parameters that support 2, 4, 8, and 16 participants with a minimum security level of 100 bits. We also give a set of parameters that support up to 2, 4, 8, and 16 participants with the standard 128-bit security. The recommended parameter sets are presented in Table 3.

For LWE instances, we use a uniform binary key distribution and set the noise standard deviation to 1.9. We use the LWE estimator [3] to determine the concrete security[7]. For MNTRU instances, we use a uniform ternary key distribution and set the noise standard deviation to 0.5 (resp., 0.75) for 128-bit (resp., 100-bit). For the second-layer NTRU ciphertexts, we fix the noise standard deviation to 0.4 (resp., 0.25) for 128-bit (resp., 100-bit) security with a uniform ternary secret key. To determine the concrete security, two recent types of attacks are considered. One is the Dense Sublattice Discovery (DSD) attack, which focuses on recovering a basis vector from the dense sublattice within the NTRU lattice. The other one is the Secret Key Recovery (SKR) attack, aiming

[7] https://github.com/malb/lattice-estimator

to directly recover a vector of the short lattice basis by the lattice attacks. Ducas and van Woerden identified the fatigue point at $Q = N^{2.484+o(1)}$, where the modulus $Q$ is such that for values above $Q$, the DSD attack becomes more efficient than SKR [16]. They have also provided an NTRU estimator[8]. One can use it to determine the BKZ block size $\beta'$ required for the DSD attack or SKR attack to break the (M)NTRU problem. To convert $\beta'$ to the concrete security, we use the cost model $T(d, \beta) := 2^{0.292 \cdot \beta' + 16.4 + \log_2(8 \cdot d)}$ (in the NTRU setting $d = 2N$) as in [5,24].

**Table 3.** Parameters Sets for Our MK-FHE.

| $k$ | First layer | | | | | Second layer | | | | Estimate |
|---|---|---|---|---|---|---|---|---|---|---|
| | Assumption | $(n, q, B_{ks}, d_{ks})$ | $\log_n q$ | Key Dist. | Noise Dist | $(N, Q, B, d)$ | $\log_N Q$ | Key Dist. | Noise Dist. | Security |
| 2 | LWE | $(500, 32749, 32, 3)$ | $1.67 < 2.484$ | binary | $\sigma = 1.9$ | $(2048, 2^{27}, 2^9, 3)$ | $2.45 < 2.484$ | ternary | $\sigma = 0.25$ | 100 |
| 4 | | | | | | $(2048, 2^{27}, 2^9, 3)$ | | | | |
| 8 | | | | | | $(2048, 2^{27}, 2^9, 3)$ | | | | |
| 16 | | | | | | $(2048, 2^{27}, 2^9, 3)$ | | | | |
| 2 | LWE | $(635, 32749, 32, 3)$ | $1.61 < 2.484$ | binary | $\sigma = 1.9$ | $(2048, 2^{27}, 2^9, 3)$ | $2.45 < 2.484$ | ternary | $\sigma = 0.4$ | 128 |
| 4 | | | | | | $(2048, 2^{27}, 2^9, 3)$ | | | | |
| 8 | | | | | | $(2048, 2^{27}, 2^9, 3)$ | | | | |
| 16 | | | | | | $(2048, 2^{27}, 2^7, 4)$ | | | | |
| 2 | MNTRU | $(560, 45181, 32, 4)$ | $1.69 < 2.484$ | ternary | $\sigma = 0.75$ | $(2048, 2^{27}, 2^9, 3)$ | $2.45 < 2.484$ | ternary | $\sigma = 0.25$ | 100 |
| 4 | | | | | | $(2048, 2^{27}, 2^9, 3)$ | | | | |
| 8 | | | | | | $(2048, 2^{27}, 2^9, 3)$ | | | | |
| 16 | | | | | | $(2048, 2^{27}, 2^9, 3)$ | | | | |
| 2 | MNTRU | $(765, 45181, 32, 4)$ | $1.61 < 2.484$ | ternary | $\sigma = 0.5$ | $(2048, 2^{27}, 2^7, 4)$ | $2.45 < 2.484$ | ternary | $\sigma = 0.4$ | 128 |
| 4 | | | | | | $(2048, 2^{27}, 2^7, 4)$ | | | | |
| 8 | | | | | | $(2048, 2^{27}, 2^6, 5)$ | | | | |
| 16 | | | | | | $(2048, 2^{27}, 2^5, 6)$ | | | | |

## 7.3 Experimental Results

All experiments run on the same laptop with a 12th Gen Intel(R) Core(TM) i9-12900H @2.50 GHz and 32 GB RAM, running Ubuntu 20.04.6 LTS. We use the OpenFHE library (v1.1.1) [4] to implement the proposed algorithms. Our codes are publicly available at `https://github.com/SKLC-FHE/MKFHE`. In Table 4, we present the implementation results with the state-of-the-art works CCS [9] and KMS [25], as detailed below. CCS* is implemented in C++ with the TFHE library[9], while KMS and CCS** are implemented in Julia[10].

From Table 4 one can see that, for parameters at 100-bit security, the bootstrapping time of our Alg. 1 is the same as CCS [9], and 2 times faster than KMS [25]. Correspondingly, the bootstrapping key size of our Alg. 1 is 2.3 times smaller than that of CCS [9], and 5.6 times smaller than that of KMS [25]. For $k = 4$ and 8, our Alg. 1 is about 1.2 and 1.5 times faster than CCS [9], and 1.6 and 1.4 times faster than KMS [25]. Correspondingly, the bootstrapping key size of our Alg. 1 is 2.5 and 2.7 smaller than that of CCS [9], and is 7.4 and 6.5 times smaller than that of KMS [25]. For $k = 16$, our Alg. 1 is approximately

[8] https://github.com/WvanWoerden/NTRUFatigue
[9] https://github.com/ilachill/MK-FHE
[10] https://github.com/SNUCP/MKTFHE

**Table 4.** Timings and key sizes for bootstrapping

| Scheme | $\lambda$ | Hybrid | Runtime(s) | | | | EVK(MB) | | | | KSK |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Product | 2 | 4 | 8 | 16 | 2 | 4 | 8 | 16 | (MB) |
| CCS* [9] | 100 | RLWE | 0.07 | 0.33 | 1.09 | \ | 19.69 | 26.25 | 32.81 | \ | 70.13 |
| CCS** [9] | 100 | RLWE | 0.12 | 0.42 | 1.61 | 11.36 | 39.85 | 53.12 | 66.38 | 159.21 | 54.38 |
| KMS [25] | 100 | RLWE | 0.14 | 0.44 | 1.17 | 2.86 | 105.86 | 176.47 | 141.31 | 176.56 | 108.75 |
| Alg. 1 | 100 | **NTRU** | 0.07 | 0.28 | 0.82 | 2.74 | 29.56 | 29.56 | 29.56 | 29.56 | 8.75 |
| Alg. 2 | 100 | **NTRU** | **0.05** | **0.21** | **0.54** | **2.61** | **13.21** | **13.21** | **13.21** | **13.21** | **0.68** |
| Alg. 1 | 128 | **NTRU** | 0.14 | 0.40 | 1.55 | 6.84 | 60.55 | 60.55 | 80.74 | 100.92 | 11.95 |
| Alg. 2 | 128 | **NTRU** | 0.06 | 0.23 | 0.76 | 4.21 | 16.77 | 16.77 | 16.77 | 25.15 | 0.68 |

the same as KMS [25] and 7.4 times smaller than KMS [25]. For $k = 16$, we did not report CCS's implementation in C++ because the parameters they provided only support up to $k = 8$. Moreover, our Alg. 2 is about 1.4, 1.6 and 2.2 times faster than CCS [9], and 2.8, 2.1 and 2.2 times faster than KMS [25] for $k = 2$, 4 and 8, respectively. Correspondingly, the bootstrapping key size of our Alg. 2 is 6.5, 6.9 and 7.4 smaller than that of CCS [9], and is 15.5, 20.5 and 18 times smaller than that of KMS [25]. For $k = 16$, our Alg. 2 is about 1.1 times faster and 20.5 times smaller than KMS [25]. Moreover, due to the influence of the first-layer ternary key, Alg. 1 exhibits slightly inferior performance compared to Alg. 2.

**Acknowledgements** We thank the anonymous reviewers of ASIACRYPT 2024 and Huanhuan Chen for their helpful comments and suggestions on earlier versions of our paper. This paper is supported by the National Key Research and Development Program of China (Grant No. 2022YFB2702000), the National Natural Science Foundation of China (Grant Nos. 62022018, 61932019), and the Strategic Priority Research Program of Chinese Academy of Sciences (Grant No. XDB0690200).

## References

1. Akin, Y., Klemsa, J., Önen, M.: A practical TFHE-based multi-key homomorphic encryption with linear complexity and low noise growth. In: ESORICS 2023. LNCS, vol. 14344, pp. 3–23. Springer (2023), https://doi.org/10.1007/978-3-031-50594-2_1
2. Albrecht, M., Bai, S., Ducas, L.: A subfield lattice attack on overstretched NTRU assumptions. In: CRYPTO 2016. LNCS, vol. 9814, pp. 153–178. Springer (2016), https://doi.org/10.1007/978-3-662-53018-4_6
3. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. J. Math. Cryptol. 9(3), 169–203 (2015), http://www.degruyter.com/view/j/jmc.2015.9.issue-3/jmc-2015-0016/jmc-2015-0016.xml

4. 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., Saraswathy, R.V., Rohloff, K., Saylor, J., Suponitsky, D., Triplett, M., Vaikuntanathan, V., Zucca, V.: OpenFHE: Open-source fully homomorphic encryption library. In: Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography. pp. 53–63. ACM (2022), https://doi.org/10.1145/3560827.3563379
5. Bonte, C., Iliashenko, I., Park, J., Pereira, H.V., Smart, N.P.: Final: Faster FHE instantiated with NTRU and LWE. In: ASIACRYPT 2022. LNCS, vol. 13792, pp. 188–215. Springer (2022), https://doi.org/10.1007/978-3-031-22966-4_7
6. Brakerski, Z., Döttling, N.: Lossiness and entropic hardness for ring-LWE. In: TCC 2020. LNCS, vol. 12550, pp. 1–27. Springer (2020), https://doi.org/10.1007/978-3-030-64375-1_1
7. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: ACM Trans. Comput. Theory. vol. 6, pp. 13:1–13:36 (2014), https://doi.org/10.1145/2633600
8. Brakerski, Z., Perlman, R.: Lattice-based fully dynamic multi-key FHE with short ciphertexts. In: CRYPTO 2016. LNCS, vol. 9814, pp. 190–213. Springer (2016), https://doi.org/10.1007/978-3-662-53018-4_8
9. Chen, H., Chillotti, I., Song, Y.: Multi-key homomorphic encryption from TFHE. In: ASIACRYPT 2019. LNCS, vol. 11922, pp. 446–472. Springer (2019), https://doi.org/10.1007/978-3-030-34621-8_16
10. Cheon, J.H., Jeong, J., Lee, C.: An algorithm for NTRU problems and cryptanalysis of the GGH multilinear map without a low-level encoding of zero. LMS J. Comput. Math. 19(A), 255–266 (2016), https://doi.org/10.1112/s1461157016000371
11. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: ASIACRYPT 2016. LNCS, vol. 10031, pp. 3–33 (2016), https://doi.org/10.1007/978-3-662-53887-6_1
12. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: fast fully homomorphic encryption over the torus. J. Cryptol. 33(1), 34–91 (2020), https://doi.org/10.1007/s00145-019-09319-x
13. Chongchitmate, W., Ostrovsky, R.: Circuit-private multi-key FHE. In: PKC 2017. LNCS, vol. 10175, pp. 241–270. Springer (2017), https://doi.org/10.1007/978-3-662-54388-7_9
14. Clear, M., McGoldrick, C.: Multi-identity and multi-key leveled FHE from learning with errors. In: CRYPTO 2015. LNCS, vol. 9216, pp. 630–656. Springer (2015), https://doi.org/10.1007/978-3-662-48000-7_31
15. Ducas, L., Micciancio, D.: FHEW: bootstrapping homomorphic encryption in less than a second. In: EUROCRYPT 2015. LNCS, vol. 9056, pp. 617–640. Springer (2015), https://doi.org/10.1007/978-3-662-46800-5_24
16. Ducas, L., van Woerden, W.: NTRU fatigue: How stretched is overstretched? In: ASIACRYPT 2021. LNCS, vol. 13093, pp. 3–32. Springer (2021), https://doi.org/10.1007/978-3-030-92068-5_1
17. Esgin, M.F., Espitau, T., Niot, G., Prest, T., Sakzad, A., Steinfeld, R.: Plover: Masking-friendly hash-and-sign lattice signatures. In: EUROCRYPT 2024. LNCS, vol. 14656, pp. 316–345. Springer (2024), https://doi.org/10.1007/978-3-031-58754-2_12
18. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptol. ePrint Arch. p. 144 (2012), http://eprint.iacr.org/2012/144

19. Genise, N., Gentry, C., Halevi, S., Li, B., Micciancio, D.: Homomorphic encryption for finite automata. In: ASIACRYPT 2019. LNCS, vol. 11922, pp. 473–502. Springer (2019), https://doi.org/10.1007/978-3-030-34621-8_17
20. Gentry, C., Szydlo, M.: Cryptanalysis of the revised NTRU signature scheme. In: EUROCRYPT 2002. LNCS, vol. 2332, pp. 299–320. Springer (2002), https://doi.org/10.1007/3-540-46035-7_20
21. Hough, P., Sandsbråten, C., Silde, T.: Concrete NTRU security and advances in practical lattice-based electronic voting. Cryptology ePrint Archive p. 933 (2023), https://eprint.iacr.org/2023/933
22. Kim, A., Deryabin, M., Eom, J., Choi, R., Lee, Y., Ghang, W., Yoo, D.: General bootstrapping approach for rlwe-based homomorphic encryption. IEEE Trans. Computers 73(1), 86–96 (2024), https://doi.org/10.1109/TC.2023.3318405
23. Kirchner, P., Fouque, P.A.: Revisiting lattice attacks on overstretched NTRU parameters. In: EUROCRYPT 2017. LNCS, vol. 10210, pp. 3–26 (2017), https://doi.org/10.1007/978-3-319-56620-7_1
24. Kluczniak, K.: NTRU-v-um: Secure fully homomorphic encryption from NTRU with small modulus. In: CCS 2022. pp. 1783–1797. ACM (2022), https://doi.org/10.1145/3548606.3560700
25. Kwak, H., Min, S., Song, Y.: Towards practical multi-key TFHE: parallelizable, key-compatible, quasi-linear complexity. In: PKC 2024. LNCS, vol. 14604, pp. 354–385. Springer (2024), https://doi.org/10.1007/978-3-031-57728-4_12
26. Lee, Y., Micciancio, D., Kim, A., Choi, R., Deryabin, M., Eom, J., Yoo, D.: Efficient FHEW bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023. LNCS, vol. 14006, pp. 227–256. Springer (2023), https://doi.org/10.1007/978-3-031-30620-4_8
27. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: STOC 2012. pp. 1219–1234. ACM (2012), https://doi.org/10.1145/2213977.2214086
28. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer (2010), https://doi.org/10.1007/978-3-642-13190-5_1
29. Morshed, T., Aziz, M.M.A., Mohammed, N.: CPU and GPU accelerated fully homomorphic encryption. In: HOST 2020. pp. 142–153. IEEE (2020), https://doi.org/10.1109/HOST45689.2020.9300288
30. Mukherjee, P., Wichs, D.: Two round multiparty computation via multi-key FHE. In: EUROCRYPT 2016. LNCS, vol. 9666, pp. 735–763. Springer (2016), https://doi.org/10.1007/978-3-662-49896-5_26
31. Peikert, C., Shiehian, S.: Multi-key FHE from LWE, revisited. In: TCC 2016-B. LNCS, vol. 9986, pp. 217–238 (2016), https://doi.org/10.1007/978-3-662-53644-5_9
32. Peralta, G., Cid-Fuentes, R.G., Bilbao, J., Crespo, P.M.: Homomorphic encryption and network coding in IoT architectures: Advantages and future challenges. Electronics 8(8), 827 (2019), https://doi.org/10.3390/electronics8080827
33. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM 56(6), 34:1–34:40 (2009), https://doi.org/10.1145/1568318.1568324
34. Shrestha, R., Kim, S.: Chapter ten - integration of IoT with blockchain and homomorphic encryption: Challenging issues and opportunities. vol. 115, pp. 293–331 (2019), https://doi.org/10.1016/bs.adcom.2019.06.002

35. Xiang, B., Zhang, J., Deng, Y., Dai, Y., Feng, D.: Fast blind rotation for bootstrapping FHEs. In: CRYPTO 2023. LNCS, vol. 14084, pp. 3–36. Springer (2023), https://doi.org/10.1007/978-3-031-38551-3_1
36. Xu, K., Tan, B.H.M., Wang, L., Aung, K.M.M., Wang, H.: Multi-key fully homomorphic encryption from NTRU and (R)LWE with faster bootstrapping. Theor. Comput. Sci. 968, 114026 (2023), https://doi.org/10.1016/j.tcs.2023.114026

## A Key-Switching for Multi-key LWE ciphertext in [9,25]

- $\mathsf{LWE.KSKG}(\mathbf{z}_i, \mathbf{s}_i)$: Given two vectors $\mathbf{z}_i = (z_{i,0}, \cdots, z_{i,n-1}) \in \mathbb{Z}^n$, $\mathbf{s} = (s_{i,0}, \cdots, s_{i,N-1}) \in \mathbb{Z}^N$ and two integer $q$, $B_{ks}$ as input, the algorithm first computes $d_{ks} = \lceil \log_{B_{ks}} q \rceil$ and sets $\mathbf{g} = (B_{ks}^0, \cdots, B_{ks}^{d_{ks}-1})$. Then it samples vector $\mathbf{a}_{i,j,l,v} \leftarrow \mathbb{Z}^n$ uniformly at random and $e_{i,j,l,v} \in \mathbb{Z}_q$ from some noise distribution and computes $\mathsf{ksk}_{i,j,l,v} = (b_{i,j,l,v}, \mathbf{a}_{i,j,l,v})$ where $b_{i,j,l,v} = -\mathbf{a}_{i,j,l,v} \cdot \mathbf{z}_i + e_{i,j,l,v} + v \cdot s_{i,j} \cdot B_{ks}^l$ for all $j \in \mathbb{Z}_N$, $l \in \mathbb{Z}_{d_{ks}}$, $v \in [B_{ks} - 1]$. Finally, it outputs $\mathbf{KSK}_i = \{\mathsf{ksk}_{i,j,l,v}\}$ as the key-switching key of party $i$.
- $\mathsf{LWE.KS}(\overline{ct}, \{\mathbf{KSK}_i\}_{i\in[k]})$: Given as input a multi-key LWE ciphertext $\overline{ct} = (b, \mathbf{a}_1, \cdots, \mathbf{a}_k) \in \mathbb{Z}^{kN+1}$ and the key-switching keys $\{\mathbf{KSK}_i\}_{i\in[k]}$ of keys associated with $\overline{ct}$, the algorithm computes $\mathbf{g}^{-1}(a_{i,j}) = (v_{i,j,l})_{l\in\mathbb{Z}_{d_{ks}}}$ for each $i \in [k]$ and $j \in \mathbb{Z}_N$ and then compute

$$b'_i = \sum_{j=0}^{N-1} \sum_{l=0, v_{i,j,l}\neq 0}^{d_{ks}-1} b_{i,j,l,v_{i,j,l}} \quad \text{and} \quad \mathbf{a}'_i = \sum_{j=0}^{N-1} \sum_{l=0, v_{i,j,l}\neq 0}^{d_{ks}-1} \mathbf{a}_{i,j,l,v_{i,j,l}},$$

  and let $b' = b + \sum_{i=1}^{k} b'_i$. Finally, the algorithm outputs a multi-key LWE ciphertext $\bar{\mathbf{c}}' = (b', \mathbf{a}'_1, \ldots, \mathbf{a}'_k) \in \mathbb{Z}^{kn+1}$.