Towards Practical Multi-key TFHE: Parallelizable, Key-Compatible, Quasi-linear Complexity

Hyesun Kwak, Seonhong Min, and Yongsoo Song

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

Abstract. Multi-key homomorphic encryption is a generalized notion of homomorphic encryption supporting arbitrary computation on ciphertexts, possibly encrypted under different keys. In this paper, we revisit the work of Chen, Chillotti and Song (ASIACRYPT 2019) and present yet another multi-key variant of the TFHE scheme.

The previous construction by Chen et al. involves a blind rotation procedure where the complexity of each iteration gradually increases as it continuously operates on ciphertexts under different keys. Hence, the complexity of gate bootstrapping grows quadratically with respect to the number of associated keys.

Our scheme is based on a new blind rotation algorithm which consists of two separate phases. We first split a given multi-key ciphertext into several single-key ciphertexts, take each of them as input to the blind rotation procedure, and obtain accumulators corresponding to individual keys. Then, we merge these single-key accumulators into a single multi-key accumulator. In particular, we develop a novel homomorphic operation between single-key and multi-key ciphertexts to instantiate our pipeline. Therefore, our construction achieves an almost linear time complexity since the gate bootstrapping is dominated by the first phase of blind rotation which requires only independent single-key operations. It also enjoys with great advantages of parallelizability and key-compatibility.

We implement the proposed scheme and provide its performance benchmark. For example, our experiment of 16-key gate bootstrapping demonstrates about 4.75x speedup without parallelization, and 55.53x speedup with parallelization over prior work.

Keywords: Homomorphic Encryption, Multi-Key Homomorphic Encryption, Fully Homomorphic Encryption over Torus

1 Introduction

Homomorphic encryption (HE) is a cryptosystem which allows us to evaluate arbitrary functions directly on encrypted data without decryption. For example, in a cloud environment, the user encrypts its message with its own key and send

it to the cloud. The desired computations are executed in the cloud side and finally the user receive the ciphertext that encrypts the result of the computation without any information leakage. Due to such an attribute, it has been regarded as one of the promising solutions to process privacy-sensitive data such as financial or medical data. After the very first construction of HE by Gentry [14], a variety of HE schemes have been proposed such as BFV [2, 13], GSW [15], BGV [3], TFHE [9] and CKKS [8].

However, the conventional HE technology has an intrinsic disadvantage in that the authority is concentrated to a single party, as it only supports operations between data encrypted under the same secret key. Thereby, the usage of HE is restricted to scenarios where all data owners commonly trust a party who owns the secret key. To resolve this problem, several variants of HE with distributed authority have been studied, such as threshold HE [1, 23, 25] and multi-key HE (MKHE) [20, 11, 24, 26, 5, 6]. The former acts like a single-key HE by encrypting data under a jointly constructed public key and the latter supports operations between data encrypted with different secret keys. Although threshold HE is generally more efficient in ciphertext size and computation cost, no additional party can join the computation once the joint key is generated. On the other hand, MKHE allows each user to independently generate its own keys and join the computation. In this paper, we focus on the TFHE scheme [9], and its first and the only multi-key variant by Chen, Chillotti and Song (CCS19) [5].

TFHE is a well-known homomorphic encryption based on the *Learning with Errors* (LWE) [27] and Ring-LWE (RLWE) [21] problems. It allows us to perform arbitrary binary gate operations via a costly operation called the *gate bootstrapping* which mainly consists of three steps: linear combination, blind rotation and key-switching. In the first step, it computes a linear combination of input LWE ciphertexts corresponding to the gate to be evaluated. In the following blind rotation step, it homomorphically decrypts the resulting LWE ciphertext from the linear combination step over the exponent of a monomial using the *external product* operation. By multiplying this monomial to the *test polynomial* with pre-assigned coefficient and extracting the constant term, we obtain an LWE ciphertext with ring dimension encrypting the output of the gate. Finally, the key-switching step reduces the ciphertext dimension back to the LWE dimension.

The MK variant of TFHE follows the same pipeline although the blind rotation step is realized in a multi-key manner. During the blind rotation step in the original TFHE, it recursively evaluates the homomorphic MUX gates on the accumulator ACC via the external product that multiplies an RGSW ciphertext to an RLWE ciphertext. When it comes to the multi-key situation, the multiplicand of the external product is an MK-RLWE ciphertext whereas the multiplier is an RGSW ciphertext generated by a single party. In CCS19 [5], the authors designed an RGSW-like cryptosystem and a multiplication method called the *hybrid product* which has the same functionality to external product with faster speed and small noise growth. However, the time complexity of blind rotation step from hybrid product is quadratic with respect to the number of associated parties.

1.1 Our Contributions

In this paper, we construct an improved multi-key TFHE scheme with a blind rotation algorithm that is (1) asymptotically faster, (2) parallelizable, and (3) key-compatible with the single-key scheme. We refactored the blind rotation algorithm to first perform single-key multiplications and then merge the results into a multi-key ciphertext. During the party-wise computation, we only make use of a single-key multiplication and thus its time complexity is linear to the number of parties k. Merging the resulting ciphertexts of each party requires $O(k^2)$ time complexity since we perform k multi-key multiplications, nevertheless it is relatively fast compared to the party-wise blind rotation. Consequently, we achieve quasi-linear time complexity which is dominated by the party-wise blind rotation.

However, this cannot be actualized with the existing building blocks in prior work since they only support a multiplication between a fresh (structured) single-key encryption and a multi-key ciphertext. Hence we instantiate our idea by introducing a new homomorphic multiplication method called the *generalized external product*. This generalized external product can be regarded as an improvement of the hybrid product from CCS19, but it exploits the hybrid product as a building block. It multiplies a single-key RLEV ciphertext, which is upper half of the RGSW ciphertext, directly to an MK-RLWE ciphertext, and then 'relinearize' the resulting ciphertext with a quadratic key structure using the hybrid product. As a side contribution, we improve the performance of hybrid product from the observation that we can rearrange the order of operations and reduce the execution time in almost half. The noise variance is slightly smaller than the original algorithm as well.

With the improved hybrid product and the generalized external product, we finally realize the asymptotically faster MK-TFHE scheme. In the blind rotation step, we first execute the blind rotation party-wise with the temporary accumulators ACC'_i of RLEV ciphertexts for $1 \le i \le k$ in a single-key manner. Then we merge the ACC'_is into a single MK-RLWE ciphertext ACC using our generalized external product. Fig. 1.1 depicts the blind rotation of CCS19 and our new algorithm.

We also remark that this party-wise blind rotation is parallelizable. Compared to the sequential multiplications in the blind rotation from CCS19, our algorithm can be executed party-wise in parallel and then merged sequentially. Furthermore, the bootstrapping key in our scheme is compatible to a single-party TFHE scheme [9] as well. Our scheme makes use of RGSW encryptions of the LWE secret key as blind rotation key, identical to the blind rotation key for the TFHE scheme with a single auxiliary key of the ring key.

Finally, we implement our multi-key variant of TFHE scheme and provide the basic benchmarks and the comparison between CCS19 and ours.

1.2 Related Works

After López-Alt et al. [20] first proposed the concept of MKHE, there have been several follow-up studies to construct multi-key HE schemes. Clear and

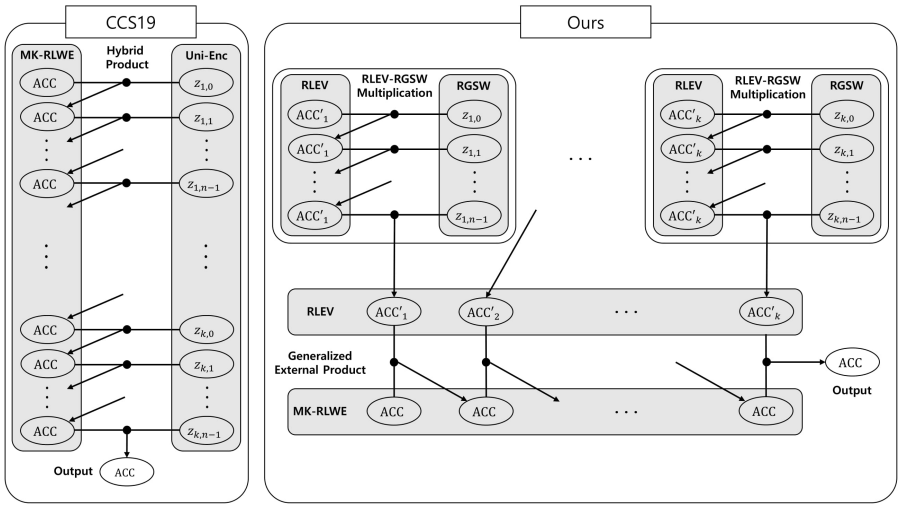

Fig. 1. High-level overview of the blind rotation algorithm of MK variant of TFHE from CCS19 and Ours.

McGoldrick [11] constructed MKHE from GSW [15] by introducing a *masking system* where a ciphertext, encrypted under an individual key, is converted to be encrypted under a master secret key. Mukherjee and Wichs [24] simplified the masking system and build a two-round MPC protocol from the MKHE scheme. These schemes support a single-hop evaluation where participants must be determined at the start. The contemporary studies Peikert and Shiehian [26] constructed a multi-hop scheme from GSW [15] that supports dynamic computation on ciphertexts encrypted under additional keys by expanding the ciphertext to be encrypted under the union of the original set of keys and the additional keys.

Then, there have been several studies on MK variants of batched HE schemes. Chen, Zhang and Wang [7] built an MKHE scheme based on BGV [3] with a compact ciphertext extension. Chen, Dai, Kim and Song [6] presented multi-key variants of BFV [2, 13] and CKKS [8] with quadratic complexity, and it was improved to have a linear complexity in a recent work of Kim et al. [17].

On the other hand, Brakerski and Perlman [4] presented an MK-LWE scheme whose bootstrapping process relies on the external product. A follow-up study by Chen, Chillotti and Song [5] improved its efficiency by introducing a hybrid product between single-key and multi-key encryptions and constructing an MK-variant of TFHE [9]. This is the most relevant work to ours, and its detailed description will be given in Sec. 3.

Recently, Klemsa et al. [18] proposed a variant of TFHE scheme for multiple parties, which is partially MK and partially Threshold. Its blind rotation keys are encrypted as n-out-of-n threshold HE ciphertexts under a joint key structure, while it encrypts data and key-switching keys in a multi-key manner in order to enhance the bootstrapping performance. However, this approach requires an

additional communication round in order to make the public RGSW blind rotation key, and even few more additional communication round in order to make the scheme fully-dynamic.

2 Background

2.1 Notation

The real torus $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ is the set of real numbers modulo 1. For a power-of-two integer N, we write $T = \mathbb{T}[X]/(X^N + 1)$. We denote vectors in lower-case bold (e.g. $\mathbf{a}$), and matrices in upper-case bold (e.g. $\mathbf{A}$). The inner product of two vectors $\mathbf{a}$, $\mathbf{b}$ is denoted by $\langle \mathbf{a}, \mathbf{b} \rangle$. For a positive integer k, we write $[k] = \{1, \ldots, k\}$.

We use $x \leftarrow S$ to denote that x is sampled uniformly from a set S. For a real $\alpha \geq 0$, ψ' denotes the Gaussian distribution of variance α^2. When sampling a polynomial from T, we use ψ to denote a distribution over T which samples N coefficients of the output polynomial independently from the Gaussian distribution of variance β^2 for a real $\beta \geq 0$.

2.2 LWE and RLWE assumptions

The security of TFHE relies on the torus variants of LWE and RLWE assumptions [9].

Definition 1 (The LWE assumption). *Let n be a positive integer, $\alpha > 0$ a noise parameter, and χ' a key distribution over $\mathbb{Z}^n$. An LWE instance of a secret $\mathbf{z} \in \mathbb{Z}^n$ is a tuple $(b, \mathbf{a}) \in \mathbb{T}^{n+1}$ generated by $\mathbf{a} \leftarrow \mathbb{T}^n$, $e \leftarrow \psi'$ and $b = -\langle \mathbf{a}, \mathbf{z} \rangle + e \pmod 1$. The LWE assumption states that the LWE distribution of a secret $\mathbf{z} \leftarrow \chi'$ is computationally indistinguishable from the uniform distribution over $\mathbb{T}^{n+1}$.*

Definition 2 (The RLWE assumption). *Let N be a power of two, $\beta > 0$ a noise parameter, and χ a key distribution over R. An RLWE instance of a secret $s \in R$ is a pair $(b, a) \in T^2$ generated by $a \leftarrow T$, $e \leftarrow \psi$ and $b = -a \cdot s + e \pmod 1$. The RLWE assumption states that the RLWE distribution of a secret $s \leftarrow \chi$ is computationally indistinguishable from the uniform distribution over T^2.*

Under these assumptions, we can define the (R)LWE cryptosystem. An LWE ciphertext is a vector of torus elements, in a form of $(b, \mathbf{a}) \in \mathbb{T}^{n+1}$ and an RLWE ciphertext is a tuple $(b, a) \in T^2$. Now we introduce *phase*, a randomized encoding of (R)LWE ciphertexts. The phase for LWE ciphertext, $\varphi_{\mathbf{z}}(\cdot) : \mathbb{T}^{n+1} \to \mathbb{T}$ is defined by $\varphi_{\mathbf{z}}(b, \mathbf{a}) = b + \langle \mathbf{a}, \mathbf{z} \rangle \pmod 1$ and the phase for RLWE ciphertext $\varphi_s(\cdot) : T^2 \to T$ is defined by $\varphi_s(b, a) = b + a \cdot s \pmod 1$. We remark that the phase preserves the linear combinations between the (R)LWE ciphertexts.

2.3 Multi-Key Homomorphic Encryption

MKHE is a variant of HE that enables computation on ciphertexts encrypted under different keys. It contains five PPT algorithms ($\mathtt{Setup}$, $\mathtt{KeyGen}$, $\mathtt{Enc}$, $\mathtt{Eval}$, $\mathtt{Dec}$).

- **Setup**: $pp \leftarrow \mathtt{Setup}(1^\lambda)$. Given the security parameter λ, set the public parameter set pp.
- **Key Generation**: $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathtt{KeyGen}(i)$. A party i generates its secret key sk_i and public key pk_i.
- **Encryption**: $\mathsf{ct} \leftarrow \mathtt{Enc}(\mathsf{pk}_i; m)$. A party i encrypts its message m with its public key pk_i and output a ciphertext ct.
- **Evaluation**: $\overline{\mathsf{ct}} \leftarrow \mathtt{Eval}(\mathcal{C}, \mathsf{pk}_1, \ldots, \mathsf{pk}_k; \overline{\mathsf{ct}}_1, \ldots, \overline{\mathsf{ct}}_l)$. Given a circuit $\mathcal{C}$, ciphertexts $\overline{\mathsf{ct}}_1, \ldots, \overline{\mathsf{ct}}_l$, and public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_k$ of associated parties, output a ciphertext ct.
- **Decryption**: $m \leftarrow \mathtt{Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_k; \overline{\mathsf{ct}})$. Given a ciphertext $\overline{\mathsf{ct}}$ and secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_k$ of associated parties, output a message m.

Let $\overline{\mathsf{ct}}_1, \ldots, \overline{\mathsf{ct}}_l$ be ciphertexts encrypting $m_1, \ldots, m_l$, respectively, and $\mathsf{pk}_1, \ldots, \mathsf{pk}_k$ be public keys of associated parties. An MKHE scheme is considered to be secure if its encryption is semantically secure. A valid MKHE scheme satisfies that

$$\mathtt{Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_k; \mathtt{Eval}(\mathcal{C}, \mathsf{pk}_1, \ldots, \mathsf{pk}_k; \overline{\mathsf{ct}}_1, \ldots, \overline{\mathsf{ct}}_l)) = \mathcal{C}(m_1, \ldots, m_l)$$

with an overwhelming probability.

We also compare the MKHE to n out of n threshold HE as well. The key difference is that given the secret keys of each party s_i $(1 \le i \le k)$, MK ciphertexts are encrypted under a concatenated key structure of $(s_1, \ldots, s_k)$ whereas threshold ciphertexts are encrypted under a joint key structure $\sum_{i \in [k]} s_i$. Due to such key structure, MKHE scheme is less efficient than Threshold HE schemes, however it does not require any communications between the parties in the setup phase whereas Threshold HE scheme requires an additional communication in the setup phase. In addition, MKHE schemes are fully dynamic but Threshold HE schemes are not dynamic in general *i.e.,* any additional party cannot join the computation once the setup phase is completed.

2.4 Gadget Decomposition

A *gadget decomposition* is a map $h : \mathbb{T} \to \mathbb{Z}^d$ with a *gadget vector* $\mathbf{g} \in \mathbb{T}^d$ that satisfies $\|h(a)\|_\infty \le \delta$ and $|\langle h(a), \mathbf{g}\rangle - a| \le \varepsilon$ for some small constants $\varepsilon, \delta > 0$. It is a widely used technique to manage noise growth in HE schemes. The *digit decomposition* is an example of gadget decomposition corresponding to the gadget vector $\mathbf{g} = [B^{-1}, \ldots, B^{-d}] \in \mathbb{T}^d$, defined by $h(a) = (a_1, \ldots, a_d)$ where a_i is the ith-digit of a in base B. We can also balance the output $h(a)$ by decomposing a by $a = \sum_{p=1}^{d} a_i \cdot B^{-p}$ where $a_i \in (-B/2, B/2]$, which minimizes the decomposition error $|\langle h(a), \mathbf{g}\rangle - a|$.

The definition of a gadget decomposition is naturally extended to T as $h : T \to R^d$ by identifying an element of T to the vector of its coefficients in $\mathbb{T}^N$. In TFHE [9], the digit decomposition is used for an element in $\mathbb{T}$ and the balanced version for T.

2.5 RLEV and RGSW

In this section, we describe the RLEV [10] and RGSW [15] encryptions, and multiplication operations between ciphertexts of different types. For a gadget decomposition $h : T \to R^d$ corresponding to a gadget vector $\mathbf{g} \in \mathbb{T}^d$, we define encryption algorithms as follows:

- $\underline{\texttt{RLEV.Enc}(s;\mu)}$: Given a secret key s and a message $\mu \in R$, sample $\mathbf{a} \leftarrow T^d$ and $\mathbf{e} \leftarrow \psi^d$. return $\mathbf{C} \leftarrow [-s \cdot \mathbf{a} + \mathbf{e} + \mu \cdot \mathbf{g} \pmod 1, \mathbf{a}] \in T^{d\times 2}$.
- $\underline{\texttt{RGSW.Enc}(s;\mu)}$: Sample $\mathbf{a} \leftarrow T^{2d}$ and $\mathbf{e} \leftarrow \psi^{2d}$. Given a secret key s and a message $\mu \in R$, return $\overline{\mathbf{C}} \leftarrow [-s \cdot \mathbf{a} + \mathbf{e}, \mathbf{a}] + \mu \cdot \begin{bmatrix} \mathbf{g} & \mathbf{0} \\ \mathbf{0} & \mathbf{g} \end{bmatrix} \pmod 1 \in T^{2d\times 2}$.

We also define the phase of an RLEV encryption $\mathbf{C} = (\mathbf{b}, \mathbf{a}) \in T^{d\times 2}$ by $\varphi_s(\mathbf{C}) = \mathbf{b} + s \cdot \mathbf{a} \pmod 1$. Note that an RLEV encryption $\mathbf{C} \leftarrow \texttt{RLEV.Enc}(s;\mu)$ satisfies that $\varphi_s(\mathbf{C}) \approx \mu \cdot \mathbf{g} \pmod 1$.

Now we define three homomorphic multiplications between RLWE, RLEV and RGSW ciphertexts. For convenience, we generalize the definition of gadget decomposition to RLWE and RLEV ciphertexts by decomposing individual entries in T. For example, we write $h(\mathbf{c}) = (h(c_0), h(c_1)) \in R^{2d}$ for an RLWE ciphertext $\mathbf{c} = (c_0, c_1) \in T^2$, and $h(\mathbf{C}) = \begin{bmatrix} h(c_{0,0}) & h(c_{0,1}) \\ \vdots & \\ h(c_{d-1,0}) & h(c_{d-1,1}) \end{bmatrix} \in R^{d\times 2d}$ for an RLEV ciphertext $\mathbf{C} = \begin{bmatrix} c_{0,0} & c_{0,1} \\ \vdots & \\ c_{d-1,0} & c_{d-1,1} \end{bmatrix} \in T^{d\times 2}$.

Definition 3 (T-RLEV multiplication). *Let $\mathbf{C} \in T^{d\times 2}$ be an RLEV ciphertext and $c \in T$ be a torus polynomial. We define the T-RLEV multiplication $\odot : T \times T^{d\times 2} \to T^2$ as $c \odot \mathbf{C} = h(c) \cdot \mathbf{C} \pmod 1$.*

If $\mathbf{C}$ is an RLWE encryption of μ under s, then the T-RLWE multiplication outputs an RLWE ciphertext whose phase is

$$\varphi_s(c \odot \mathbf{C}) = \langle h(c), \varphi_s(\mathbf{C}) \rangle \approx \langle h(c), \mu \cdot \mathbf{g} \rangle \approx \mu \cdot c \pmod 1.$$

Definition 4 (RLWE-RGSW multiplication). *Let $\mathbf{c} \in T^2$ be an RLWE ciphertext and $\overline{\mathbf{C}} \in T^{2d\times 2}$ be an RGSW ciphertext. We define the RLWE-RGSW multiplication $\otimes : T^2 \times T^{2d\times 2} \to T^2$ as $\mathbf{c} \otimes \overline{\mathbf{C}} = h(\mathbf{c}) \cdot \overline{\mathbf{C}} \pmod 1$.*

Definition 5 (RLEV-RGSW multiplication). *Let $\mathbf{C} \in T^{d\times 2}$ be an RLEV ciphertext and $\overline{\mathbf{C}} \in T^{2d\times 2}$ be an RGSW ciphertext. The RLEV-RGSW multiplication $\circledast : T^{d\times 2} \times T^{2d\times 2} \to T^{d\times 2}$ is defined as $\mathbf{C} \circledast \overline{\mathbf{C}} = h(\mathbf{C}) \cdot \overline{\mathbf{C}} \pmod 1$.*

If $\overline{\mathbf{C}}$ is RGSW encryption of μ under s, then the RLWE-RGSW multiplication outputs an RLWE ciphertext whose phase is

$$\varphi_s(\mathbf{c} \otimes \overline{\mathbf{C}}) = \left\langle h(\mathbf{c}), \varphi_s(\overline{\mathbf{C}}) \right\rangle \approx \langle h(\mathbf{c}), \mu \cdot (\mathbf{g}, s \cdot \mathbf{g}) \rangle \approx \mu \cdot \varphi_s(\mathbf{c}) \pmod 1.$$

The RLWE-RGSW multiplication is also called the *external product* [9]. Similarly, the RLWE-RGSW multiplication outputs an RLEV ciphertext whose phase is

$$\varphi_s(\mathbf{C} \circledast \overline{\mathbf{C}}) = h(\mathbf{C}) \cdot \varphi_s(\overline{\mathbf{C}}) \approx \langle h(\mathbf{C}), \mu \cdot (\mathbf{g}, s \cdot \mathbf{g}) \rangle \approx \mu \cdot \varphi_s(\mathbf{C}) \pmod 1.$$

3 Overview of Chen et al. (2019)

In 2016, Chillotti et al. [9] designed TFHE, which is a fully homomorphic encryption scheme based on the LWE and RLWE assumptions. The TFHE scheme can encrypt a single bit in each LWE ciphertext, and evaluate an arbitrary binary gate homomorphically using the "gate bootstrapping". The basic idea of TFHE bootstrapping is to homomorphically compute the phase of an LWE ciphertext on the exponent of a ring polynomial and extract the pre-assigned coefficient.

The gate bootstrapping of TFHE consists of three steps: linear combination, blind rotation and key-switching. Let $\mathbf{z} = (z_0, \dots z_{n-1})$ be the LWE secret. Given LWE ciphertexts ct_1 and ct_2 such that $\varphi_{\mathbf{z}}(\mathsf{ct}_i) \approx \frac{1}{4} m_i \pmod 1$, the linear combination step computes an LWE ciphertext ct such that $\varphi_{\mathbf{z}}(\mathsf{ct}) \approx \frac{1}{2} m \pmod 1$ where m is the resulting bit of a binary operation between m_1 and m_2. In the next step, the ciphertext ct is scaled by $2N$ and converted into $(\tilde{b}, \tilde{\mathbf{a}} = (\tilde{a}_0, \dots, \tilde{a}_{n-1}))$ such that $\tilde{b} + \langle \tilde{\mathbf{a}}, \mathbf{z} \rangle \approx N \cdot m \pmod{2N}$. The blind rotation algorithm initializes an "accumulator" as a trivial RLWE encryption $(v \cdot X^{\tilde{b}}, 0)$, where v is a fixed torus polynomial called the test vector, and then multiplies $X^{\tilde{a}_i z_i}$ recursively for $0 \le i < n$ using the external product to obtain an encryption of $v \cdot X^{\tilde{b} + \sum_{i=0}^{n-1} \tilde{a}_i z_i}$. The test vector v has pre-assigned coefficients so that we can extract an LWE ciphertext that is decryptable by the RLWE key into the constant term of the message polynomial of the output accumulator. Finally, the key-switching procedure is used to produce an LWE encryption of the same message under $\mathbf{z}$.

In 2019, Chen, Chillotti and Song [5] presented the first MK variant of TFHE (which we will refer to as CCS19 throughout the paper). Its gate bootstrapping follows a similar pipeline but uses MK variants of LWE and RLWE. The main challenge was to re-design the blind rotation algorithm in an MK manner, which requires substitution of the external product. To resolve the issue, the authors introduced a variant of RGSW (called "uni-encryption"), together with a dyadic operation (called "hybrid product") for multiplying a uni-encryption to an MK-RLWE ciphertext. In this section, we give a brief overview of CCS19.

3.1 Uni-encryption and Hybrid Product

We first present basic setup and key generation algorithms, then describe *uni-encryption* and *hybrid product*. Uni-encryption is an RGSW-like single-key structured encryption scheme, while the hybrid product is a binary operation that

takes its input as a pair of uni-encryption and MK-RLWE encryption and returns an MK-RLWE ciphertext. In general, an MK-RLWE ciphertext is in the form of $\overline{\mathsf{ct}} = (c_0, \ldots, c_k) \in T^{k+1}$ with an index set $\{1, \ldots, k\}$ of the associated parties. An MK-RLWE ciphertext corresponds to the concatenated secret key $\overline{\mathbf{s}} = (s_1, \ldots, s_k)$, and its phase is defined as $\varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}) = c_0 + c_1 s_1 + \cdots + c_k s_k \pmod 1$.

• $\underline{\mathtt{CCS.Setup}(1^\lambda)}$: Given the security parameter λ, return the following parameters:

- An LWE dimension n, a key distribution χ' over $\mathbb{Z}^n$, an error parameter $\alpha > 0$.
- A base B' and a degree d' to set a gadget vector $\mathbf{g}' = [B'^{-1}, \ldots, B^{-d'}]$ and a gadget decomposition $h' : \mathbb{T} \to \mathbb{Z}^{d'}$ for LWE.
- An RLWE dimension N, a key distribution χ over $R = \mathbb{Z}[X]/(X^N + 1)$, and an error parameter $\beta > 0$.
- A base B and a degree d to set a gadget vector $\mathbf{g} = [B^{-1}, \ldots, B^{-d}]$ and a gadget decomposition $h : T \to R^d$ for ring-based schemes.
- A CRS $\mathbf{a} \leftarrow T^d$.

We set the LWE error distribution ψ' as a Gaussian distribution over $\mathbb{R}$ of variance α^2, and the RLWE error distribution ψ as a distribution over T which samples N coefficients independently from a Gaussian distribution of variance β^2.

• $\underline{\mathtt{CCS.KeyGen}(i)}$: A party i generates its secret and public keys as follows:

- Sample an LWE secret key $\mathbf{z}_i = (z_{i,0}, \ldots, z_{i,n-1}) \leftarrow \chi'$.
- Sample an RLWE secret key $s_i = s_{i,0} + s_{i,1} X + \cdots + s_{i,N-1} X^{N-1} \leftarrow \chi$ and an error $\mathbf{e} \leftarrow \psi^d$. Compute $\mathbf{b}_i = -s_i \cdot \mathbf{a} + \mathbf{e} \pmod 1$ and set the public key as $\mathsf{pk}_i = \mathbf{b}_i$.

For simplicity, we write $s_0 = 1$ and $\mathbf{b}_0 = -\mathbf{a}$.

• $\underline{\mathtt{CCS.UniEnc}(s_i; \mu)}$: A party i samples $r_i \leftarrow \chi$, $\mathbf{f}_{i,1} \leftarrow T^d$, and $\mathbf{e}_1, \mathbf{e}_2 \leftarrow \psi^d$. Given a plaintext $\mu \in R$ and a secret s_i, return $\mathbf{d}_i = r_i \cdot \mathbf{a} + \mu \cdot \mathbf{g} + \mathbf{e}_1 \pmod 1$ and $\mathbf{F}_i = [\mathbf{f}_{i,0} | \mathbf{f}_{i,1}]$ where $\mathbf{f}_{i,0} = -s_i \cdot \mathbf{f}_{i,1} + r_i \cdot \mathbf{g} + \mathbf{e}_2 \pmod 1$.

• $\underline{\mathtt{CCS.HbProd}(\{\mathbf{b}_j\}_{j \in [k]}; \overline{\mathsf{ct}}, (\mathbf{d}_i, \mathbf{F}_i))}$: Given an MK-RLWE ciphertext $\overline{\mathsf{ct}} = (c_0, \ldots, c_k) \in T^{k+1}$, a uni-encryption $(\mathbf{d}_i, \mathbf{F}_i)$ of party i and the public keys $\{\mathbf{b}_j\}_{j \in [k]}$ of the parties associated with $\overline{\mathsf{ct}}$, compute and output an MK-RLWE ciphertext $\overline{\mathsf{ct}}'$ as follows:

1. For $0 \le j \le k$, let

$$
\begin{aligned}
u_j &= \langle h(c_j), \mathbf{d}_i \rangle , \\
v_j &= \langle h(c_j), \mathbf{b}_j \rangle , \\
w_{j,0} &= \langle h(v_j), \mathbf{f}_{i,0} \rangle , \\
w_{j,1} &= \langle h(v_j), \mathbf{f}_{i,1} \rangle .
\end{aligned}
$$

2. Output $\overline{\mathsf{ct}}' = (c'_0, \ldots, c'_k) \in T^{k+1}$ where

$$c'_j = \begin{cases} u_0 + \sum_{j=0}^{k} w_{j,0} \pmod 1 & \text{if } j = 0, \\ u_i + \sum_{j=0}^{k} w_{j,1} \pmod 1 & \text{if } j = i, \\ u_j & \text{otherwise;} \end{cases}$$

Below, we describe the correctness of the hybrid product. We refer the reader to [5] for a more detailed analysis. Suppose that $\overline{\mathsf{ct}}$ is an MK-RLWE ciphertext and $(\mathbf{d}_i, \mathbf{F}_i)$ is a uni-encryption of $\mu \in R$ of party i, and let $\overline{\mathsf{ct}}'$ be the resulting MK-RLWE ciphertext of the hybrid product algorithm. Then, we have

$$\begin{aligned} \varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}') &\approx \sum_{j=0}^{k} \langle h(c_j), r_i \cdot \mathbf{a} + \mu \cdot \mathbf{g} \rangle \cdot s_j + \sum_{j=0}^{k} \langle h(v_j), r_i \cdot \mathbf{g} \rangle \pmod 1 \\ &\approx \mu \cdot \varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}) + \sum_{j=0}^{k} \langle h(c_j), \mathbf{a} \cdot s_j \rangle \cdot r_i + \sum_{j=0}^{k} \langle h(c_j), \mathbf{b}_j \rangle \cdot r_i \pmod 1 \\ &\approx \mu \cdot \varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}) \pmod 1. \end{aligned}$$

In other words, the phase of $\overline{\mathsf{ct}}$ is multiplied by μ with a small noise.

3.2 Gate Bootstrapping

We now describe the gate bootstrapping of CCS19 that is based on the uni-encryption and hybrid product algorithms. It requires additional generations of blind rotation and key-switching keys.

• $\underline{\mathtt{CCS.BootKeyGen}(i)}$: Each party i generates and publishes a blind rotation key brk_i and a key-switching key ksk_i as follows:

- Generate $\mathsf{brk}_{i,j} = (\mathbf{d}_{i,j}, \mathbf{F}_{i,j}) \leftarrow \mathtt{CCS.UniEnc}(s_i; z_{i,j})$ for $0 \le j < n$. Set the blind rotation key as $\mathsf{brk}_i = \{\mathsf{brk}_{i,j}\}_{0 \le j < n}$.
- Let $(s^*_{i,0}, \ldots, s^*_{i,N-1}) = (s_{i,0}, -s_{i,N-1}, \ldots, -s_{i,1})$. Sample $\mathbf{A}_{i,j} \leftarrow \mathbb{T}^{d' \times n}$ and $\mathbf{e}_{i,j} \leftarrow \psi'^{d'}$ for $0 \le j < N$, and let $\mathsf{ksk}_{i,j} = [\mathbf{b}_{i,j} | \mathbf{A}_{i,j}]$ where $\mathbf{b}_{i,j} = -\mathbf{A}_{i,j} \cdot \mathbf{z}_i + \mathbf{e}_{i,j} + s^*_{i,j} \cdot \mathbf{g}' \pmod 1$. Set the key-switching key as $\mathsf{ksk}_i = \{\mathsf{ksk}_{i,j}\}_{0 \le j < N}$.

• $\underline{\mathtt{CCS.Enc}(\mathbf{z}_i; m)}$: A party i samples $\mathbf{a}_i \leftarrow \mathbb{T}^n$ and $e \leftarrow \psi'$. Given a message bit $m \in \{0,1\}$ and its secret key $\mathbf{z}_i$, return the ciphertext $\mathsf{ct} = (b_i, \mathbf{a}_i)$ where $b_i = -\langle \mathbf{a}_i, \mathbf{z}_i \rangle + \frac{1}{4}m + e \pmod 1$.

• $\underline{\mathtt{CCS.Dec}(\{\mathbf{z}_i\}_{i \in [k]}; \overline{\mathsf{ct}})}$: Given a ciphertext $\overline{\mathsf{ct}} \in \mathbb{T}^{kn+1}$ and secret keys $\{\mathbf{z}_i\}_{i \in [k]}$, return the bit $m \in \{0,1\}$ which minimizes $|b + \sum_{i \in [k]} \langle \mathbf{a}_i, \mathbf{z}_i \rangle - \frac{1}{4}m|$.

A fresh encryption of CCS19 returns a usual (single-key) LWE ciphertext, but an MK-LWE ciphertext is generally written as a vector of the form $\mathsf{ct} =$

Algorithm 1 Blind Rotation of CCS [5]

Input: $\overline{\mathsf{ct}} = (b, \mathbf{a}_1, \ldots, \mathbf{a}_k), \{(\mathsf{pk}_i, \mathsf{brk}_i)\}_{i\in[k]}$
Output: ACC

Let $\tilde{b} := \lfloor 2Nb \rceil$, $\tilde{a}_{i,j} := \lfloor 2Na_{i,j} \rceil$ for $1 \le i \le k$, $0 \le j < n$
  and $v := -\frac{1}{8} \cdot (1 + X + \cdots + X^{N/2-1} - X^{N/2} - \cdots - X^{N-1})$.
$\mathsf{ACC} \leftarrow (v \cdot X^{\tilde{b}}, 0, \ldots, 0)$
for $1 \le i \le k$ **do**
  for $0 \le j < n$ **do**
    $\mathsf{ACC} \leftarrow \mathsf{ACC} + \mathtt{HbProd}(\{\mathsf{pk}_\ell\}_{\ell\in[k]}; (X^{\tilde{a}_{i,j}} - 1) \cdot \mathsf{ACC}, \mathsf{brk}_{i,j})$
  end for
end for

$(b, \mathbf{a}_1, \ldots, \mathbf{a}_k) \in \mathbb{T}^{kn+1}$ where k denotes the number of associated parties. It is decrypted using the concatenated key $\overline{\mathbf{z}} = (\mathbf{z}_1, \ldots, \mathbf{z}_k)$ of k parties, *i.e.,* $\varphi_{\overline{\mathbf{z}}}(\mathsf{ct}) = b + \sum_{i=1}^{k} \langle \mathbf{a}_i, \mathbf{z}_i \rangle \approx \frac{1}{4}m \pmod 1$. In the encryption phase, each party locally encrypts its message without knowing any information about other parties. The ciphertexts are extended before evaluation to be encrypted under the concatenated secret key of associated parties.

• $\mathtt{CCS.HomNAND}(\{(\mathsf{pk}_i, \mathsf{brk}_i, \mathsf{ksk}_i)\}_{i\in[k]}; \overline{\mathsf{ct}}_1, \overline{\mathsf{ct}}_2)$: Given two ciphertexts $\overline{\mathsf{ct}}_1, \overline{\mathsf{ct}}_2$ and key-triple $\{(\mathsf{pk}_i, \mathsf{brk}_i, \mathsf{ksk}_i)\}_{i\in[k]}$ of associated parties, perform the following steps:

1. Compute $\overline{\mathsf{ct}} \leftarrow (\frac{5}{8}, 0, \ldots, 0) - \overline{\mathsf{ct}}_1 - \overline{\mathsf{ct}}_2 \pmod 1$.
2. Compute $\mathsf{ACC} \leftarrow \mathtt{CCS.BlindRotate}(\{(\mathsf{pk}_i, \mathsf{brk}_i)\}_{i\in[k]}; \overline{\mathsf{ct}})$ using the blind rotation algorithm (Alg. 1).
3. Compute $\overline{\mathsf{ct}} \leftarrow (\frac{1}{8}, 0, \ldots, 0) + (b, \mathbf{a}_1, \ldots, \mathbf{a}_k) \pmod 1 \in \mathbb{T}^{kN+1}$ where b is the constant term of $\mathsf{ACC}[0]$ and $\mathbf{a}_i$ is the coefficient vector of $\mathsf{ACC}[i]$ for $i \in [k]$.
4. Perform the key-switching process: Compute $(b'_i, \mathbf{a}'_i) = \sum_{j=0}^{N-1} h'(a_{i,j}) \cdot \mathsf{ksk}_{i,j} \pmod 1$ for $i \in [k]$ and $b' = b + \sum_{i\in[k]} b'_i$. Return $\overline{\mathsf{ct}}' = (b', \mathbf{a}'_1, \ldots, \mathbf{a}'_k) \in \mathbb{T}^{kn+1}$.

4 Accelerating Multi-key TFHE

In this section, we present a new MK variant of the TFHE scheme. Our scheme is asymptotically faster than CCS19, and its bootstrapping procedure is parallelizable. In addition, its key structure is almost compatible with the original TFHE as each party only needs to publish a single auxiliary key. At the heart of our construction, we design a generalized external product to re-design the blind rotation algorithm.

Recall that the blind rotation algorithm (Alg. 1) of CCS19 takes nk hybrid products to homomorphically multiply $X^{\tilde{a}_{i,j}z_{i,j}}$ to the accumulator. The hybrid product algorithm operates on multi-key ciphertexts and achieves linear complexity in terms of the number of parties involved. Thus, this linear complexity results in a quadratic overall complexity.

In our scheme, we minimize the cost of operations on MK ciphertexts and exploit single-key multiplication to reduce the overall complexity. This is based on our observation that an encryption of $X^{\langle \tilde{\mathbf{a}}_i, \mathbf{z}_i \rangle}$ for each $1 \le i \le k$ can be obtained in a 'single-key' manner since the secret $\mathbf{z}_i = (z_{i,0}, \ldots, z_{i,n-1})$ is related solely to party i. However, this approach cannot be achieved by known techniques in CCS19 since the hybrid product can only multiply a fresh single-key uni-encryption. To realize our framework, we introduce a novel homomorphic operation called the *generalized external product*, which enables us to multiply a single-key RLEV ciphertext to MK-RLWE accumulator. This operation does not require an input RLEV ciphertext to have a special structure like uni-encryption, so can be generally used for operation between possibly noisy ciphertexts.

In Sec. 4.1 and Sec 4.2, we introduce our improved hybrid product and generalized external product as a building block. In Sec. 4.3, we describe the overall scheme.

4.1 Improved Hybrid Product

We present an improved hybrid product that enjoys better performance in terms of speed and noise growth. In the correctness proof of hybrid product in CCS19, we have

$$\begin{aligned}\sum_{j=0}^{k}(w_{j,0} + w_{j,1}s_i) &= \sum_{j=1}^{k}\langle h(v_j), \mathbf{f}_{i,0} + s_i \cdot \mathbf{f}_{i,1}\rangle \\ &\approx \sum_{j=1}^{k}\langle h(v_j), r_i \cdot \mathbf{g}\rangle \approx r_i \cdot \sum_{j=0}^{k} v_j \pmod 1.\end{aligned}$$

We observe that since

$$\sum_{j=1}^{k}\langle h(v_j), \mathbf{f}_{i,0} + s_i \cdot \mathbf{f}_{i,1}\rangle \approx \left\langle h\Big(\sum_{j=1}^{k} v_j\Big), \mathbf{f}_{i,0} + s_i \cdot \mathbf{f}_{i,1}\right\rangle,$$

the computation of $h(v_j)$ for $1 \le j \le k$ can be replaced by a single decomposition $h\big(\sum_{j=1}^{k} v_j\big)$. Below, we provide a formal description of the new hybrid product operation.

• $\underline{\texttt{NewHbProd}(\{\mathbf{b}_j\}_{j\in[k]}; \overline{\mathsf{ct}}, (\mathbf{d}_i, \mathbf{F}_i))}$: Given an MK-RLWE ciphertext $\overline{\mathsf{ct}} = (c_0, \ldots, c_k) \in T^{k+1}$, a uni-encryption $(\mathbf{d}_i, \mathbf{F}_i)$ of party i and the public keys $\{\mathbf{b}_j\}_{j\in[k]}$ of parties associated with $\overline{\mathsf{ct}}$, return an MK-RLWE ciphertext $\overline{\mathsf{ct}}'$ as follows:

1. Compute

$$\begin{aligned}u_j &= \langle h(c_j), \mathbf{d}_i\rangle \quad (0 \le j \le k)\\ v &= \sum_{j=0}^{k}\langle h(c_j), \mathbf{b}_j\rangle\end{aligned}$$

2. Output $\overline{\mathsf{ct}}' = (c'_0, \ldots, c'_k) \in T^{k+1}$ where

$$c'_j = \begin{cases} u_0 + \langle h(v), \mathbf{f}_{i,0} \rangle \pmod 1 & \text{if } j = 0, \\ u_i + \langle h(v), \mathbf{f}_{i,1} \rangle \pmod 1 & \text{if } j = i, \\ u_j & \text{otherwise;} \end{cases}$$

Correctness. Let $\overline{\mathsf{ct}} = (c_0, \ldots, c_k)$ be an MK-RLWE encryption and $(\mathbf{d}_i, \mathbf{F}_i)$ be a uni-encryption of μ of party i. The output $\overline{\mathsf{ct}}' = (c'_0, \ldots, c'_k)$ satisfies that

$$\begin{aligned} \varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}') &= \sum_{j=0}^{k} \langle h(c_j), \mathbf{d}_i \rangle \cdot s_j + \langle h(v), \mathbf{f}_{i,0} \rangle + \langle h(v), \mathbf{f}_{i,1} \rangle \cdot s_i \\ &\approx \sum_{j=0}^{k} \langle h(c_j), r_i \cdot \mathbf{a} + \mu \cdot \mathbf{g} \rangle \cdot s_j + \langle h(v), \mathbf{f}_{i,0} + s_i \cdot \mathbf{f}_{i,1} \rangle \\ &\approx \mu \cdot \varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}) + r_i \cdot \sum_{j=0}^{k} \langle h(c_j), s_j \cdot \mathbf{a} \rangle + r_i \cdot \sum_{j=0}^{k} \langle h(c_j), \mathbf{b}_j \rangle \\ &\approx \mu \cdot \varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}) \pmod 1. \end{aligned}$$

Performance. We estimate the number of $\langle h(\cdot), \cdot \rangle$, say gadget product, to analyze the time complexity. The hybrid product of CCS19 requires $4(k+1)$ gadget products to compute u_j, v_j, $w_{j,0}$, and $w_{j,1}$ for $0 \le j \le k$. Meanwhile, our algorithm takes only $2k + 4$ gadget products in total.

Noise growth. As we compute $\langle h(v), \mathbf{f}_{i,0} \rangle$ and $\langle h(v), \mathbf{f}_{i,1} \rangle$ for $v = \sum_{j=0}^{k} \langle h(c_j), \mathbf{b}_j \rangle$ where it previously computed $\sum_{j=0}^{k} \langle h(v_j), \mathbf{f}_{i,0} \rangle$ and $\sum_{j=0}^{k} \langle h(v_j), \mathbf{f}_{i,1} \rangle$, the error introduced in this part has reduced by a factor of $k+1$. Nevertheless, the difference is negligibly small and thus the two hybrid product algorithms show similar error variance. We refer the reader to Sec. 5.2 for thorough noise analysis.

4.2 Generalized External Product

We introduce a new multiplication operation that multiplies an arbitrary single-key RLEV ciphertext to an MK-RLWE ciphertext. To understand the underlying idea, we first recall the external product: given an RLWE ciphertext $\mathbf{c}$ and an RGSW encryption $\overline{\mathbf{C}}$ of μ under the secret t, $\mathbf{c} \otimes \overline{\mathbf{C}}$ outputs an RLWE ciphertext with $\varphi_t(\mathbf{c} \otimes \overline{\mathbf{C}}) \approx \mu \cdot \varphi_t(\mathbf{c}) = \varphi_t(\mu \cdot \mathbf{c})$. Our key observation is that the external product can be comprehended as multiplying the message μ homomorphically to each component of $\mathbf{c}$.

Now let us 'generalize' the external product to the multi-key setting. Suppose that we are given an MK-RLWE ciphertext $\mathsf{ct} = (c_0, \ldots, c_k)$ under the concatenated key $\overline{\mathbf{s}} = (1, s_1, \ldots, s_k)$ and a single-key RLEV encryption $\mathbf{C}$ of μ under another key t_i of party i. Inspired by the external product, we aim to multiply μ to ct homomorphically. This goal can be achieved by executing $(k+1)$ T-RLEV multiplications: $c_j \odot \mathbf{C}$ for $0 \le j \le k$. However, the resulting ciphertext is not decryptable by $\mathbf{s}$, but it is encrypted under the tensor product of two keys, namely

$(1,t)\otimes\overline{\mathbf{s}} = (\overline{\mathbf{s}}, t\cdot\overline{\mathbf{s}})$. To change the secret key back to $\overline{\mathbf{s}}$, we exploit the relinearization technique, which is used in a variety of HE schemes such as [2, 13, 3, 8]. Let i-th party publish a relinearization key, a uni-encryption of t under the key s_i. Then we can obtain an MK-RLWE ciphertext $\overline{\mathsf{ct}}'$ with $\varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}') \approx \mu \cdot \varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}})$ by multiplying t homomorphically to the corresponding components to $t\cdot\overline{\mathbf{s}}$ with hybrid product and adding it to the rest of the components. The exact algorithm is given below.

• $\mathtt{ExtProd}(\{\mathbf{b}_j\}_{j\in[k]}, \mathsf{rlk}_i; \overline{\mathsf{ct}}, \mathbf{C}_i)$: Given an MK-RLWE ciphertext $\overline{\mathsf{ct}} = (c_0, \ldots, c_k) \in T^{k+1}$, the public keys $\{\mathbf{b}_j\}_{j\in[k]}$ of parties associated with $\overline{\mathsf{ct}}$, an RLEV ciphertext $\mathbf{C}_i \in T^{d\times 2}$ and the relinearization key rlk_i of party $i \in [k]$, it returns an MK-RLWE ciphertext $\overline{\mathsf{ct}}'$ as follows:

1. Compute $(x_j, y_j) \leftarrow c_j \odot \mathbf{C}_i$ for $0 \le j \le k$. Let $\overline{\mathbf{x}} = (x_0, \ldots, x_k)$ and $\overline{\mathbf{y}} = (y_0, \ldots, y_k)$
2. Compute $\overline{\mathsf{ct}}' \leftarrow \mathtt{NewHbProd}(\{\mathbf{b}_j\}_{j\in[k]}; \overline{\mathbf{y}}, \mathsf{rlk}_i) + \overline{\mathbf{x}}$ and return $\overline{\mathsf{ct}}'$.

Correctness. Suppose that $\overline{\mathsf{ct}} = (c_0, c_1 \ldots, c_k)$ is a MK-RLWE ciphertext under the secret $\overline{\mathbf{s}} = (s_1, \cdots, s_k)$ and $\mathbf{C}_i$ is an RLEV encryption of μ under the secret t_i. Now, $0 \le j \le k$, $(x_j, y_j) = c_j \odot \mathbf{C}_i$ satisfies $x_j \cdot t_i + y_j \approx \mu \cdot c_j$. Let $\overline{\mathsf{ct}}' = (c'_0, \ldots, c'_k) \leftarrow \mathtt{NewHbProd}(\{\mathbf{b}_j\}_{j\in[k]}; \overline{\mathbf{y}}, \mathsf{rlk}_i) + \overline{\mathbf{x}}$ where $\mathsf{rlk}_i = \mathtt{UniEnc}(s_i; t_i)$. Then we have

$$\begin{aligned}\varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}') &\approx \sum_{j=0}^{k} x_j s_j + t_i \cdot \sum_{j=0}^{k} y_j s_j \\ &= \sum_{j=0}^{k} (x_j + t_i y_j) \cdot s_j \approx \mu \cdot \sum_{j=0}^{k} c_j s_j = \mu \cdot \varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}) \pmod 1\end{aligned}$$

where $s_0 = 1$. Note that this algorithm is exact for any RGSW ciphertext $\overline{\mathbf{C}}$ as well, by replacing $c_j \odot \mathbf{C}_i$ to $(0, c_j) \otimes \overline{\mathbf{C}}$.

General-Purpose Utility. We remark that this generalized external product is a general-purpose multiplication in that it multiplies a commonly used single-key ciphertext to a multi-key ciphertext. Compared to the previous (R)GSW-like MKHEs [11, 24, 4, 26] which construct multiplications on multi-key ciphertexts, CCS19 [5] and our scheme introduces multiplications, hybrid product and external product, between single-key and multi-key ciphertexts. These multiplications enable better performance in both time and memory. However, the hybrid product requires fresh uni-encryption of a multiplicand. For example, if one wants to evaluate arbitrary operations between uni-encryptions from the same party before they are multiplied to an MK ciphertext, they should be expanded to an MK-RGSW ciphertext and then evaluated via MK-RGSW operations. In contrast, our generalized external product enables us to perform arbitrary operations as a single-key ciphertext and then multiply the resulting RLEV or RGSW ciphertext to a multi-key ciphertext at any time, with the relinearization key generated once in the key generation phase.

Performance. In the first step of the external product, it executes $k+1$ T-RLEV multiplications, which takes $2(k+1)$ gadget products in total. Then in the second step, the new hybrid product consumes $2k+4$ gadget products as explained in Sec. 4.1. To sum up, the external product requires $4k+6$ gadget products.

Noise Growth. The error variance of our external product will be discussed in Sec. 5.2.

4.3 Our Scheme

In this section, we combine all building blocks to construct yet another MK-variant of TFHE. Similar to CCS19, our scheme shares the same blueprint for gate bootstrapping as TFHE. However, the major difference is that our blind rotation algorithm has a different structure consisting of two distinguished phases involving single-key and multi-key computation, respectively.

More precisely, the first phase of our blind rotation aims to perform blind rotation party-wise with the accumulator staying as a single-key ciphertext. In other words, we compute $X^{\langle \tilde{\mathbf{a}}_i, \mathbf{z}_i \rangle} (1 \le i \le k)$ simultaneously. In the second phase of blind rotation, we merge k accumulators, which are single-key RLEV encryptions of $X^{\langle \tilde{\mathbf{a}}_i, \mathbf{z}_i \rangle}$ under s_i, into a trivial MK-RLWE ciphertext of $v \cdot X^{\tilde{b}}$ under k secrets $s_1, \ldots, s_k$. This is achieved by using the generalized external product introduced in the previous section.

Below we provide a formal description of our MK-TFHE scheme. We remark that its setup, basic key generation and ciphertext structure are the same as CCS19.

- $\underline{\texttt{Setup}(1^\lambda)}$: Given the security parameter λ, return the following parameters:
  - An LWE dimension n, a key distribution χ' over $\mathbb{Z}^n$, an error parameter $\alpha > 0$.
  - A base B' and a degree d' to set a gadget vector $\mathbf{g}' = [B'^{-1}, \ldots, B^{-d'}]$ and a gadget decomposition $h' : \mathbb{T} \to \mathbb{Z}^{d'}$ for LWE.
  - An RLWE dimension N, a key distribution χ over $R = \mathbb{Z}[X]/(X^N+1)$, and an error parameter $\beta > 0$.
  - A base B and a degree d to set a gadget vector $\mathbf{g} = [B^{-1}, \ldots, B^{-d}]$ and a gadget decomposition $h : T \to R^d$ for ring-based schemes.
  - A CRS $\mathbf{a} \leftarrow T^d$.

- $\underline{\texttt{KeyGen}(i)}$: A party i generates its secret and public keys as follows.
  - Sample an LWE secret key $\mathbf{z}_i = (z_{i,0}, \ldots, z_{i,n-1}) \leftarrow \chi'$.
  - Sample an RLWE secret key $s_i = s_{i,0} + s_{i,1}X + \cdots + s_{i,N-1}X^{N-1} \leftarrow \chi$ and an error $\mathbf{e} \leftarrow \psi^d$. Compute $\mathbf{b}_i = -s_i \cdot \mathbf{a} + \mathbf{e} \pmod 1$ and set the public key as $\mathsf{pk}_i = \mathbf{b}_i$.

- $\underline{\texttt{BootKeyGen}(i)}$: A party i generates and publishes a blind rotation key brk_i, a relinearization key rlk_i and a key-switching key ksk_i as follows.

Algorithm 2 New Blind Rotation

Input: $\overline{\mathsf{ct}} = (b, \mathbf{a}_1, \ldots, \mathbf{a}_k), \{(\mathsf{pk}_i, \mathsf{brk}_i, \mathsf{rlk}_i)\}_{i\in[k]}$
Output: ACC

Let $\tilde{b} := \lfloor 2Nb \rceil$, $\tilde{a}_{i,j} := \lfloor 2Na_{i,j} \rceil$ for $1 \le i \le k$, $0 \le j < n$
    and $v := -\frac{1}{8} \cdot (1 + X + \cdots + X^{N/2-1} - X^{N/2} - \cdots - X^{N-1})$.
$\mathsf{ACC} \leftarrow (v \cdot X^{\tilde{b}}, 0, \ldots, 0)$
for $1 \le i \le k$ **do**
    $\mathsf{ACC}'_i \leftarrow (\mathbf{g}, \mathbf{0}) \in T^{d\times 2}$
    for $0 \le j < n$ **do**
        $\mathsf{ACC}'_i \leftarrow \mathsf{ACC}'_i + \left[(X^{\tilde{a}_{i,j}} - 1) \cdot \mathsf{ACC}'_i\right] \circledast \mathsf{brk}_{i,j}$
    end for
end for
for $1 \le i \le k$ **do**
    $\mathsf{ACC} \leftarrow \mathtt{ExtProd}(\{\mathsf{pk}_\ell\}_{\ell\in[k]}, \mathsf{rlk}_i; \mathsf{ACC}, \mathsf{ACC}'_i)$
end for

- Sample $t_i \leftarrow \chi$ and generate $\mathsf{brk}_{i,j} \leftarrow \mathtt{RGSW.Enc}(t_i; z_{i,j})$ for $0 \le j < n$. Set the blind rotation key $\mathsf{brk}_i = \{\mathsf{brk}_{i,j}\}_{0\le j<n}$.
- Generate the relinearization key $\mathsf{rlk}_i \leftarrow \mathtt{CCS.UniEnc}(s_i; t_i)$.
- Let $(s^*_{i,0}, \ldots, s^*_{i,N-1}) = (s_{i,0}, -s_{i,N-1}, \ldots, -s_{i,1})$. Sample $\mathbf{A}_{i,j} \leftarrow \mathbb{T}^{d'\times n}$ and $\mathbf{e}_{i,j} \leftarrow \psi'^{d'}$ for $0 \le j < N$, and let $\mathsf{ksk}_{i,j} = [\mathbf{b}_{i,j} | \mathbf{A}_{i,j}]$ where $\mathbf{b}_{i,j} = -\mathbf{A}_{i,j} \cdot \mathbf{z}_i + \mathbf{e}_{i,j} + s^*_{i,j} \cdot \mathbf{g}'$. Set the key-switching key $\mathsf{ksk}_i = \{\mathsf{ksk}_{i,j}\}_{0\le j<N}$.

• $\underline{\mathtt{Enc}(\mathbf{z}_i; m)}$: A party i samples $\mathbf{a}_i \leftarrow \mathbb{T}^n$ and $e \leftarrow \psi'$. Given a message bit $m \in \{0,1\}$ and its secret key $\mathbf{z}_i$, return the ciphertext $\mathsf{ct} = (b_i, \mathbf{a}_i)$ where $b_i = -\langle \mathbf{a}_i, \mathbf{z}_i \rangle + \frac{1}{4}m + e \pmod 1$.

• $\underline{\mathtt{Dec}(\{\mathbf{z}_i\}_{i\in[k]}; \overline{\mathsf{ct}})}$: Given a ciphertext $\overline{\mathsf{ct}} \in \mathbb{T}^{kn+1}$ and secret keys $\{\mathbf{z}_i\}_{i\in[k]}$, return the bit $m \in \{0,1\}$ which minimizes $|b + \sum_{i\in[k]} \langle \mathbf{a}_i, \mathbf{z}_i \rangle - \frac{1}{4}m|$.

• $\underline{\mathtt{HomNAND}(\{(\mathsf{pk}_i, \mathsf{brk}_i, \mathsf{rlk}_i, \mathsf{ksk}_i)\}_{i\in[k]}; \overline{\mathsf{ct}}_1, \overline{\mathsf{ct}}_2)}$: Given two ciphertexts $\overline{\mathsf{ct}}_1$, $\overline{\mathsf{ct}}_2$ and key-quadruple $\{(\mathsf{pk}_i, \mathsf{brk}_i, \mathsf{rlk}_i, \mathsf{ksk}_i)\}_{i\in[k]}$ of associated parties, perform the following steps:

1. Compute $\overline{\mathsf{ct}} \leftarrow (\frac{5}{8}, 0, \ldots, 0) - \overline{\mathsf{ct}}_1 - \overline{\mathsf{ct}}_2 \pmod 1$.
2. Compute $\mathsf{ACC} \leftarrow \mathtt{BlindRotate}(\{(\mathsf{pk}_i, \mathsf{brk}_i, \mathsf{rlk}_i)\}_{i\in[k]}; \overline{\mathsf{ct}})$ where $\mathtt{BlindRotate}(\cdot)$ is the blind rotation algorithm in Alg. 2.
3. Compute $\overline{\mathsf{ct}} \leftarrow (\frac{1}{8}, 0, \ldots, 0) + (b, \mathbf{a}_1, \ldots, \mathbf{a}_k) \pmod 1 \in \mathbb{T}^{kN+1}$ where b is the constant term of $\mathsf{ACC}[0]$ and $\mathbf{a}_i$ is the coefficient vector of $\mathsf{ACC}[i]$ for $i \in [k]$.
4. Perform the key-switching process: Compute $(b'_i, \mathbf{a}'_i) = \sum_{j=0}^{N-1} h'(a_{i,j}) \cdot \mathsf{ksk}_{i,j} \pmod 1$ for $i \in [k]$ and $b' = b + \sum_{i\in[k]} b'_i$. Return $\overline{\mathsf{ct}}' = (b', \mathbf{a}'_1, \ldots, \mathbf{a}'_k) \in \mathbb{T}^{kn+1}$.

Security. In the bootstrapping key generation, each party publishes the blind rotation key brk_i encrypting the elements of $\mathbf{z}_i$ under t_i, the relinearization key

rlk_i encrypting t_i under s_i, and key-switching key ksk_i encrypting the coefficients of s_i under $\mathbf{z}_i$. As the previous TFHE [9] and multi-key TFHE [5] schemes, we require a circular security assumption along with the (R)LWE assumption to have our scheme semantically secure.

Correctness. We show that the output ACC of our blind rotation in Alg. 2 is an MK-RLWE encryption of $v \cdot X^{\tilde{b}+\sum_{i=1}^{k}\langle \tilde{\mathbf{a}}_i, \mathbf{z}_i\rangle}$ under $\overline{\mathbf{s}} = (s_1, \ldots, s_k) \in T^k$. Initially, ACC is an MK-RLWE encryption of $v \cdot X^{\tilde{b}}$ under $\overline{\mathbf{s}}$ (line 3). In the i-th iteration of the first loop, it computes ACC'_i which is an RLEV encryption of $X^{\langle \tilde{\mathbf{a}}_i, \mathbf{z}_i\rangle}$ (line 5-8). In line 5, ACC'_i is initialized to a trivial RLEV encryption of $1 \in T$. Then for $0 \le j < n$ (line 6-8), ACC' is updated by $\mathsf{ACC}'_i + \left[(X^{\tilde{a}_{i,j}} - 1) \cdot \mathsf{ACC}'_i\right] \circledast \mathsf{brk}_{i,j}$. Since $\mathsf{brk}_{i,j}$ is the RGSW encryption of $z_{i,j}$, it implies that ACC'_i is multiplied by $X^{\tilde{a}_{i,j}}$ if $z_{i,j} = 1$, or else ($z_{i,j} = 0$), stays the same. As a result, we get k different $\mathsf{ACC}'_i (1 \le i \le k)$ encrypting $X^{\langle \tilde{\mathbf{a}}_i, \mathbf{z}_i\rangle}$ under t_i. Finally, in the i-th iteration of the second loop (line 10-12), ACC is homomorphically multiplied by $X^{\langle \tilde{\mathbf{a}}_i, \mathbf{z}_i\rangle}$ with external product. Consequently, ACC is an MK-RLWE encryption of $v \cdot X^{\tilde{b}+\sum_{i=1}^{k}\langle \tilde{\mathbf{a}}_i, \mathbf{z}_i\rangle}$ under $\overline{\mathbf{s}}$.

Our new scheme provides an asymptotically faster NAND algorithm as we perform single-key, parallelizable operations in the first phase by which the time complexity is dominated. Furthermore, the bootstrapping keys are almost compatible with the single-key TFHE [9], which allows to perform multi-key evaluation from the original (single-key) TFHE scheme with a small number of auxiliary keys. We describe the advantages in detail below.

- **Performance.** As will be analyzed in Sec. 5.1, our blind rotation algorithm Alg. 2 requires $O(nkd + k^2)$ gadget decompositions, while the blind rotation algorithm Alg. 1 of CCS19 requires $O(nk^2)$. In typical settings, n is much bigger than k, therefore the time complexity of our algorithm is quasi-linear to the number of parties.
- **Parallelization.** Our blind rotation generates k different single-key RLEV encryptions ACC'_i of $X^{\langle \tilde{\mathbf{a}}_i, \mathbf{z}_i\rangle}$ and then merge them into a single MK-RLWE ciphertext ACC by the generalized external product. Since ACC'_is are independently generated, we can evaluate them in parallel. However, merging the RLEV ciphertext cannot be parallelized since they should be sequentially multiplied, thus the time complexity of the parallelized algorithm becomes $O(nd + k^2)$.
- **Key Compatibility.** Recall that our scheme generates three bootstrapping keys: the blind rotation key brk_i, the relinearization rlk_i and the key-switching key ksk_i. We note that the blind rotation key and the key-switching key is identical to the single-key TFHE [9]. Thus, the single-key TFHE scheme can be easily expanded to the multi-key scheme with each party generating a key $\mathsf{rlk}_i = \mathtt{UniEnc}(s_i; t_i) (1 \le i \le k)$.

On the other hand, our scheme consumes two levels (one for each phase) so that the noise blows up to an extent which cannot be handled in the ring dimension $N = 1024$ used in CCS19. We use a larger ring dimension $N = 2048$ in spite

of performance degradation, but still, it is negligible as the number of parties increases. We compare CCS19 and our scheme using $N = 1024$ and $N = 2048$, respectively, in Sec. 6.

4.4 Using Different Gadget Decompositions

So far we have used the same gadget decomposition h for elements in T, but in fact, different gadgets can be applied for different encryptions in our scheme. Let h_{lev} and h_{uni} be two different gadget decompositions corresponding to gadget vectors $\mathbf{g}_{lev}$ and $\mathbf{g}_{uni}$, respectively. In the external product, for example, we can use h_{lev} in T-RLEV multiplication (step 1) and use h_{uni} in the hybrid product (step 2). More precisely, let the input RLEV ciphertext $\mathbf{C}$ of μ under a secret s involve the gadget vector $\mathbf{g}_{lev}$ *i.e.,* $\mathbf{C} = (\mathbf{b} = -s \cdot \mathbf{a} + \mathbf{e} + \mu \cdot \mathbf{g}_{lev} \pmod 1, \mathbf{a})$, and compute the T-RLEV multiplication as

$$c \odot \mathbf{C} = (\langle h_{lev}(c), \mathbf{b} \rangle, \langle h_{lev}(c), \mathbf{a} \rangle) \pmod 1$$

for $c \in T$. Then it satisfies that

$$\phi_s(c \odot \mathbf{C}) = \langle h_{lev}(c), \mathbf{b} \rangle + \langle h_{lev}(c), \mathbf{a} \rangle \cdot s \approx \langle h_{lev}(c), \mu \cdot \mathbf{g}_{lev} \rangle \approx \mu \cdot c \pmod 1.$$

In a similar argument, we can compute the hybrid product using the decomposition h_{uni} when the uni-encryption as input involves $\mathbf{g}_{uni}$.

In the rest of the paper, we use different gadget decompositions for RGSW, RLEV ciphertexts and uni-encryption respectively, each of which contains the corresponding gadget vector. We write $\{gsw, lev, uni\}$ by subscript to distinguish the gadget decompositions, i.e., h_{gsw} is the gadget decomposition corresponding to the gadget vector $\mathbf{g}_{gsw}$ of dimension d_{gsw}. The scheme using different gadget decompositions is provided in Appendix A.

4.5 Distributed Decryption

The decryption process of MKHE or Threshold HE schemes requires an online computation between different parties. For instance, to decrypt an MK-TLWE ciphertext $(b, \mathbf{a}_1, \ldots, \mathbf{a}_k)$, we evaluate the decryption circuit $\lfloor 4(b + \sum_{i \in [k]} \langle \mathbf{a}_i, \mathbf{s}_i \rangle) \rceil$ where $\mathbf{s}_i$ is the secret of i-th party. The easiest way of realizing this is to use the noise flooding technique [14], which is that each i-th party can publish $\langle \mathbf{a}_i, \mathbf{s}_i \rangle + e_i$ for some exponentially large noise e_i in order to ensure that there is no leak to the distribution of the secret key structur of the ciphertext. This technique is valid for variants of CKKS, BGV or B/FV scheme since these schemes use loose parameter sets which leave a room for an additional noise term that is exponentially big.

In contrary, TFHE scheme makes use of an extremely tight parameter to provide the best performance and thus the noise flooding technique might lead to a decryption failure. Therefore, we suggest using a garbled circuit for the distributed decryption. Kraitsberg et al. [19] proposed a distributed decryption

method of two-out-of-three threshold FV ciphertexts, which exploits a garbled circuit scheme for honest majority in three party setting. Their method is based on the observation that the decryption function has a binary circuit representation. We shall extend it to multi-key setting below.

Let q be some power-of-two unsigned integer which guarantees the correct functionality of the decryption circuit. Then the distributed decryption procedure is as follows.

1. Each party computes a $\log q$-bit integer $\langle \tilde{\mathbf{a}}_i, \mathbf{s}_i \rangle \pmod q$ where $\tilde{\mathbf{a}}_i = \lfloor q \cdot \mathbf{a}_i \rceil$.
2. Perform MPC to get $\lfloor q \cdot b \rceil + \sum_{i=1}^{k} \langle \tilde{\mathbf{a}}_i, \mathbf{s}_i \rangle \pmod q$.
3. If the binary representation of the result starts with $111,000$ then output 0 and if it starts with $001,010$ then output 1.

The first step and the second step requires n and k modular addition respectively, and the last step can be evaluated with two XNOR gates and a NAND gate. Since the first step can be computed locally, we will only focus on the second step and the last step. For q being a power-of-two unsigned integer, the addition mod q can be computed with $3 \log q$ AND gates, therefore we only need $3k \log q + 3$ binary gates for the decryption. Recall that scaling the ciphertext to modulus $2N$ before the blind rotation does not affect the correctness of the homomorphic decryption over exponents, we can optimize the number of the gate evaluations by setting $q = 2N$, or possibly smaller modulus as long as the correctness is guaranteed.

5 Performance Analysis

5.1 Time and Space Complexity

We remark that the hybrid product of CCS19 and our novel hybrid product require $4(k+1)$ and $2k+4$ uni-gadget products, respectively. Furthermore, the external product performs $k+1$ T-RLEV multiplications and one novel hybrid product to require $2(k+1)$ lev-gadget and $2k+4$ uni-gadget products. As the previous blind rotation in Alg. 1 performs nk hybrid products of CCS19, it takes $4nk(k+1)$ uni-gadget products. In our novel blind rotation Alg. 2, it performs nk RLEV-RGSW multiplications and k external products to have $2nkd_{lev}$ gsw-gadget, $2k(k+1)$ lev-gadget, and $k(2k+4)$ uni-gadget decompositions. Since the gadget decomposition takes by a factor of its degree, we have the complexity of about $O(nkd_{lev}d_{gsw} + k^2 d_{lev} + k^2 d_{uni})$. As the time complexity almost depends on $d_{lev}d_{gsw}$, we minimize $d_{lev}d_{gsw}$ when setting parameters in Sec. 6.

In the blind rotation, previous algorithm Alg. 1 takes the blind rotation keys brk_i for $1 \le i \le k$ as input where brk_i consists of n uni-encryptions. However, our algorithm Alg. 2 takes the blind rotation keys brk_i and the relinearization keys rlk_i for $1 \le i \le k$, where brk_i consists of n RGSW encryptions and rlk_i is a uni-encryption. Since a uni-encryption is in $T^{d\times 3}$ and an RGSW encryption is in $T^{2d\times 2}$, the size of the key used in our blind rotation is about $\frac{4}{3}$ times bigger than the previous one.

We remark that the blind rotation key size of our scheme can be reduced using key-compression methods for the TFHE scheme. For example, we can halve the size of the blind rotation key using the key compression method proposed by Kim et al. [16]. Or, we can replace the RGSW keys with uni-encryptions since the hybrid product is exact for a single party as well. However, there is a trade-off between the size of the key and the execution time for key-compression tricks in general.

5.2 Noise Growth

In this section, we provide an average-case noise analysis of homomorphic operations and analyze the noise growth from our gate bootstrapping procedure. We focus on the new blind rotation algorithm since other algorithms such as key-switching have been studied already in CCS19.

We start from introducing several assumptions and terminologies which we use in our analysis.

- For an RLEV encryption $\mathbf{C}$ of μ under secret s, the error of $\mathbf{C}$ is defined as $\mathsf{Err}(\mathbf{C}) = \varphi_s(\mathbf{C}) - \mu \cdot \mathbf{g}_{lev} \in T^{d_{lev}}$.
- For an RGSW encryption $\overline{\mathbf{C}}$ of μ under secret s, the error of $\overline{\mathbf{C}}$ is defined as $\mathsf{Err}(\overline{\mathbf{C}}) = \varphi_s(\overline{\mathbf{C}}) - \mu \cdot \begin{bmatrix} \mathbf{g} & \mathbf{0} \\ \mathbf{0} & \mathbf{g} \end{bmatrix} \in T^{2d_{gsw}}$.
- In our scheme, all entries of the error vector of an RLEV (RGSW) ciphertext have the same variance. Therefore, we use $\mathsf{VarErr}(\cdot)$ to denote the common variance of error components.
- For the gadget decomposition h with the gadget base B (a power-of-two) and the degree d, we define $\epsilon^2 = 1/(12B^{2d})$, the variance of uniform distribution over $(-\frac{1}{2}B^{-d}, \frac{1}{2}B^{-d}]$, and $V = \frac{1}{12}(B^2+2)$, the mean square of a uniform distribution over $\mathbb{Z}_B = \mathbb{Z} \cap (-B/2, B/2]$. We use $\{gsw, lev, uni\}$ as subscript to distinguish the variance and the mean square of specific gadget decompostions, e.g., we write ϵ_{gsw}, V_{gsw} for the gadget decomposition h_{gsw}.
- We assume that each component of an RLWE, RLEV, or RGSW ciphertext behaves as if it is a uniform random variable on T. Hence, the entries of the gadget decompositions are uniformly distributed over the set of polynomials of coefficients in $\mathbb{Z}_B$.

We provide the lemmas, corollaries and theorem on the error of the operations and algorithms used in CCS19 and our scheme. The proofs for the following lemmas, corollaries and theorem are given in Appendix B.

Lemma 1 (T-RLEV Multiplication). *Let c be a torus polynomial and $\mathbf{C}$ be an RLEV encryption of μ under secret s. Then $\mathbf{c} \leftarrow c \odot \mathbf{C}$ is an RLWE ciphertext such that $\varphi_s(\mathbf{c}) = \mu \cdot c + e \pmod 1$ for some error $e \in R$ whose variance is*

$$\mathsf{Var}(e) = \|\mu\|_2^2 \epsilon_{lev}^2 + d_{lev} N V_{lev} \mathsf{VarErr}(\mathbf{C}).$$

Lemma 2 (RLWE-RGSW Multiplication). *Let* $\mathbf{c}$ *be an RLWE ciphertext and* $\overline{\mathbf{C}}$ *an RGSW encryption of* μ *under secret* s*. Then* $\mathbf{c}' \leftarrow \mathbf{c} \otimes \overline{\mathbf{C}}$ *is an RLWE ciphertext such that* $\varphi_s(\mathbf{c}') = \mu \cdot \varphi_s(\mathbf{c}) + e \pmod 1$ *for some error* $e \in R$ *whose variance is*

$$\mathsf{Var}(e) = (1 + N/2)\|\mu\|_2^2 \epsilon_{gsw}^2 + 2d_{gsw} N V_{gsw} \mathsf{VarErr}(\overline{\mathbf{C}}).$$

Corollary 1 (RLEV-RGSW Multiplication). *Let* $\mathbf{C}$ *be an RLEV ciphertext and* $\overline{\mathbf{C}}$ *be an RGSW encryption of* μ *under secret* s*. Then* $\mathbf{C}' \leftarrow \mathbf{C} \circledast \overline{\mathbf{C}}$ *is an RLEV ciphertext with* $\varphi_s(\mathbf{C}') = \mu \cdot \varphi_s(\mathbf{C}) + \mathbf{e}$ *for some error* $\mathbf{e} = \begin{bmatrix} e_1 \\ \vdots \\ e_{d_{lev}} \end{bmatrix} \in T^{d_{lev}}$ *with*

$$\mathsf{Var}(e_i) = (1 + N/2)\|\mu\|_2^2 \epsilon_{gsw}^2 + 2d_{gsw} N V_{gsw} \mathsf{VarErr}(\overline{\mathbf{C}}) (1 \le i \le d_{lev}).$$

We provide a noise analysis on the hybrid product and blind rotation algorithms in CCS19.

Lemma 3 (Hybrid Product). *Let* $\overline{\mathsf{ct}}$ *be an MK-RLWE ciphertext and* $(\mathbf{d}_i, \mathbf{F}_i)$ *be a uni-encryption of* μ *of party* i*. Then* $\overline{\mathsf{ct}}' \leftarrow \mathtt{HbProd}(\{\mathbf{b}_j\}_{j\in[k]}; \mathsf{ct}, (\mathbf{d}_i, \mathbf{F}_i))$ *is an MK-RLWE ciphertext such that* $\varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}') = \mu \cdot \varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}) + e \pmod 1$ *for some error* $e \in R$ *with*

$$\mathsf{Var}(e) \approx \frac{k}{2}\|\mu\|_2^2 N^2 \epsilon_{uni}^2 + k d_{uni} N^2 V_{uni} \beta^2.$$

Corollary 2 (Blind Rotation of CCS19). *Let* ACC *be the resulting MK-RLWE ciphertext from the blind rotate algorithm 1. Then* $\varphi_{\overline{\mathbf{s}}}(\mathsf{ACC}) = X^{\sum_{i=1}^{k} \langle \tilde{\mathbf{a}}_i, \mathbf{z}_i \rangle + \tilde{b}} \cdot v + e \pmod 1$ *for some error* $e \in T$ *with*

$$\mathsf{Var}(e) \approx \frac{k(k+1)}{8} n N^2 (\epsilon_{uni}^2 + 4 d_{uni} V_{uni} \beta^2).$$

Now, we provide a noise analysis of our new hybrid product and generalized external product and the blind rotation.

Lemma 4 (New Hybrid Product). *Let* $\overline{\mathsf{ct}}$ *be an MK-RLWE ciphertext and* $(\mathbf{d}_i, \mathbf{F}_i)$ *be a uni-encryption of* μ *of party* i*. Then* $\overline{\mathsf{ct}}' \leftarrow \mathtt{NewHbProd}(\{\mathbf{b}_j\}_{j\in[k]}; \mathsf{ct}, (\mathbf{d}_i, \mathbf{F}_i))$ *is an MK-RLWE ciphertext such that* $\varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}') = \mu \cdot \varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}) + e \pmod 1$ *for some error* $e \in R$ *with*

$$\mathsf{Var}(e) \approx \frac{k}{2}\|\mu\|_2^2 N^2 \epsilon_{uni}^2 + k d_{uni} N^2 V_{uni} \beta^2.$$

Lemma 5 (Generalized External Product). *Let* $\overline{\mathsf{ct}}$ *be an MK-RLWE ciphertext,* $\mathbf{C}_i$ *be a single-key RLEV encryption of* μ *under secret key* t *and* $\mathsf{rlk}_i = (\mathbf{d}_i, \mathbf{F}_i)$ *be a uni-encryption of* t *of party* i*. Then* $\overline{\mathsf{ct}}' \leftarrow \mathtt{ExtProd}(\{\mathbf{b}_j\}_{j\in[k]}, \mathsf{rlk}_i; \overline{\mathsf{ct}}, \mathbf{C}_i)$ *is an MK-RLWE ciphertext such that* $\varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}') = \mu \cdot \varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}) + e \pmod 1$ *for some error* $e \in T$ *with*

$$\mathsf{Var}(e) \approx (1+kN/2)\left[\|\mu\|_2^2 \epsilon_{lev}^2 + d_{lev} N V_{lev} \mathsf{VarErr}(\mathbf{C}_i)\right] + \frac{k}{4} N^3 \epsilon_{uni}^2 + k d_{uni} N^2 V_{uni} \beta^2.$$

Theorem 1 (Our Blind Rotation). *Let* ACC *be the resulting MK-RLWE ciphertext from our new blind rotation algorithm [2]. Then* $\varphi_{\overline{\mathbf{s}}}(\mathsf{ACC}) = X^{\sum_{i=1}^{k}\langle \tilde{\mathbf{a}}_i, \mathbf{z}_i\rangle + \tilde{b}} \cdot v + e \pmod 1$ *for some error* $e \in T$ *with*

$$\mathsf{Var}(e) \approx \frac{k(k+1)}{8}\left[2d_{lev}nN^3V_{lev}\cdot\left(2d_{gsw}V_{gsw}\beta^2+\epsilon_{gsw}^2\right)+N^3\epsilon_{uni}^2+4d_{uni}N^2V_{uni}\beta^2\right].$$

6 Implementation

We provide a proof-of-concept of our MK-TFHE scheme and the previous work [5]. Note that in the implementation of CCS19, the underpinning algorithms for the original TFHE [9] such as external product are optimized, however the algorithms for the multi-key variant are not fully optimized. Since our algorithm exploits the algorithms from original TFHE, we implemented our scheme and CCS19 based in Julia for a fair comparison. All experiments were performed on a machine with Intel(R) Xeon(R) Platinum 8268 @ 2.90GHz CPU and 192GB RAM running Ubuntu 20.04.2 LTS.

Table 1 and Table 2 describe candidate parameter sets for our MK-TFHE scheme and CCS19, respectively. They achieve at least 110-bit of security level according to LWE-estimator [22] with the same LWE parameters in both schemes. However, we use different RLWE parameters as our scheme introduces high noise variance due to an additional level consumption in the generalized external product, which is intolerable by the conventional ring dimension $N = 1024$ and the standard deviation $3.72 \cdot 10^{-9}$ of TFHE. Using a larger ring dimension $N = 2048$ in our scheme, we then have smaller $\beta = 4.63 \cdot 10^{-18}$ and accordingly implement high-precision torus arithmetic (64-bit). The five parameters sets from I to V in Table 1 supports up to 2, 4, 8, 16, and 32 parties. In Table 2, the first three parameter sets I′, II′, and III′ of CCS19 are introduced in the original paper [5] that support at most 2, 4, and 8 parties, respectively. We note that we changed the gadget base for parameter set I′, to guarantee the correct functionality of fully homomorphic encryption. To compare the performance under more parties, we additionally use the parameter set IV′ to evaluate the scheme on 16 parties, but could not find an appropriate parameter set that handles 32 parties in ring dimension $N = 1024$ of CCS19.

We make use of a well-known optimization technique with space-time tradeoff used in [12, 9]. In the key-switching key generation step of party i, we publish LWE encryptions of $b \cdot s_{i,j}^* \cdot \mathbf{g}'$ for $0 \le j < n$ and $b \in \mathbb{Z} \cap (-B'/2, B'/2]$, instead of $s_{i,j}^* \cdot \mathbf{g}'$. With this technique, we do not need to perform any multiplication during the key-switching phase with $B' - 1$ times bigger key-switching key size.

As mentioned in Section 5.1, we aim to minimize $d_{gsw} \cdot d_{lev}$ with smallest error variance possible. Let us recall the error analysis of our blind rotation given in Section 5.

$$\frac{k(k+1)}{8}\left[2d_{lev}nN^3V_{lev}\cdot\left(2d_{gsw}V_{gsw}\beta^2+\epsilon_{gsw}^2\right)+N^3\epsilon_{uni}^2+4d_{uni}N^2V_{uni}\beta^2\right]$$

We note that the effect of the uni-encryption on both the noise variance and the performance of the blind rotation is almost negligible, therefore we mainly

Set	LWE				RLWE		RGSW		RLEV		UniEnc	
	n	α	B'	d'	N	β	B	d	B	d	B	d
I	560	$3.05 \cdot 10^{-5}$	2^2	8	2048	$4.63 \cdot 10^{-18}$	2^{13}	3	2^7	2	2^{10}	3
II							2^8	5	2^8		2^6	7
III							2^{11}	4	2^6	3	2^4	8
IV							2^9	5				9
V							2^8	6	2^7		2^2	16

Table 1. Recommended parameter settings for our scheme. n, α and N, β denote the dimension and the standard deviations for LWE and RLWE ciphertexts, respectively. B_{ksk} and d_{ksk} are the gadget decomposition parameter for key-switching key.

focus on the parameters of RGSW and RLEV ciphertexts. As the error variance is dominated by $d_{lev}V_{lev}(d_{gsw}V_{gsw}\beta^2 + \epsilon_{gsw}^2)$, it follows that the gadget base and the gadget length of both RGSW and RLEV ciphertexts affect the final noise variance. However, the decomposition error of RLEV ciphertexts has little influence whereas that of RGSW ciphertexts has a great influence on the final noise. Based on this observation, our strategy to find the suitable parameter set is to set $d_{lev} \cdot d_{gsw}$ first, and then set the gadget base of RGSW ciphertexts according to d_{gsw} with small decomposition noise, followed by setting the gadget base of RLEV ciphertexts with regard to other parameters. Although the effect of the parameters for uni-encryptions are almost negligible to the time complexity, the final error variance, and even the space complexity, we chose the parameter achieving the least space complexity.

Set	LWE				RLWE		UniEnc	
	n	α	B'	d'	N	β	B	d
I′	560	$3.05 \cdot 10^{-5}$	2^2	8	1024	$3.72 \cdot 10^{-9}$	2^8	3
II′							2^8	4
III′							2^6	5
IV′							2^2	12

Table 2. Recommended parameter settings for CCS19 scheme.

We describe the performance of our scheme and of CCS19 in Table 3. Our scheme is slower when the number of parties is small due to a larger ring dimension $N = 2048$. However, our algorithmic improvements overwhelm its disadvantage and outperform the previous scheme. Finally, our experiments verify that the running time of our NAND algorithm is almost linear with the number of parties as expected, compared to quadratic growth of CCS19 (see Fig. 2). We

also provide the execution time of our NAND and parallelized NAND algorithm and that of CCS19 in Table 3.

Ours						CCS19				
Set	brk	ksk	k	NAND	Parallelized	Set	brk	ksk	k	NAND
I	106MB	109MB	2	0.32s	0.21s	I′	40MB	54MB	2	0.32s
II	176MB	109MB	2	0.46s	0.32s	II′	53MB	54MB	2	0.39s
			4	1.10s	0.33s				4	1.14s
III	141MB	109MB	2	0.51s	0.39s	III′	66MB	54MB	2	0.46s
			4	1.26s	0.40s				4	1.36s
			8	2.82s	0.43s				8	4.43s
IV	176MB	109MB	2	0.62s	0.46s	IV′	159MB	54MB	2	0.98s
			4	1.55s	0.49s				4	2.80s
			8	3.35s	0.51s				8	9.21s
			16	6.90s	0.59s				16	32.76s
V	212MB	109MB	2	0.61s	0.46s	-	-	-	-	-
			4	1.51s	0.49s					
			8	3.34s	0.53s					
			16	6.98s	0.62s					
			32	14.84s	0.99s					

Table 3. The memory consumed by keys and the elapsed time of NAND algorithms in our scheme and the CCS19 scheme.

A Multi-key TFHE Variant using Different Gadget Decompositions

We provide the algorithms of our new MK-TFHE scheme with different gadget decompositions. The encryption and decryption algorithms are the same as given in Sec. 4.3.

• $\underline{\texttt{Setup}(1^\lambda)}$: Given the security parameter λ, return the following parameters:

- An LWE dimension n, a key distribution χ' over $\mathbb{Z}^n$, and an error variance $\alpha > 0$.
- An RLWE dimension N, a key distribution χ over $R = \mathbb{Z}[X]/(X^N + 1)$, and an error variance $\beta > 0$.
- A CRS $\mathbf{a} \leftarrow T^{d_{uni}}$.
- 4 pairs of gadget decompositions and gadget vectors.

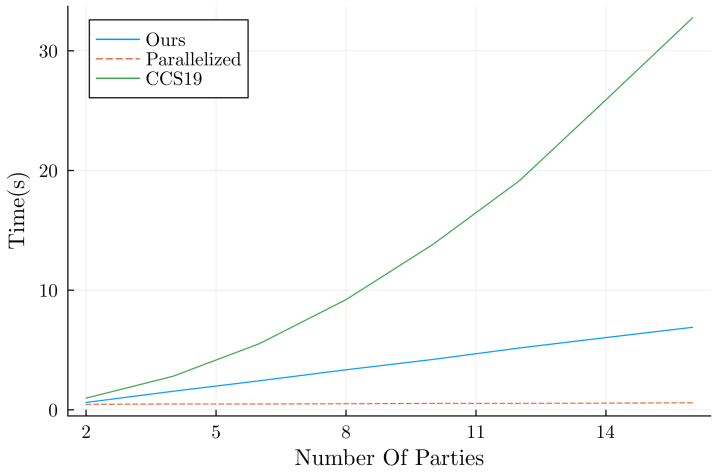

Fig. 2. The time elapsed in NAND algorithms of ours and CCS19 for parameter set IV and IV′, respectively.

- A gadget decomposition h_{ksk} of key-switching key and the corresponding gadget vector $\mathbf{g}_{ksk}$ with base B_{ksk} and degree d_{ksk}.
- A gadget decomposition h_{gsw} of RGSW encryption and the corresponding gadget vector $\mathbf{g}_{gsw}$ with base B_{gsw} and degree d_{gsw}.
- A gadget decomposition h_{lev} of RLEV encryption and the corresponding gadget vector $\mathbf{g}_{lev}$ with base B_{lev} and degree d_{lev}.
- A gadget decomposition h_{uni} of uni-encryption and the corresponding gadget vector $\mathbf{g}_{uni}$ with base B_{uni} and degree d_{uni}.

• $\underline{\texttt{KeyGen}(i)}$: A party i generates its secret and public keys as follows.

– Sample LWE secret key $\mathbf{z}_i = (z_{i,0}, \ldots, z_{i,n-1}) \leftarrow \chi$.
– Sample RLWE secret key $s_i = s_{i,0} + s_{i,1}X + \cdots + s_{i,N-1}X^{N-1} \leftarrow \chi'$ and error $\mathbf{e} \leftarrow \psi^{d_{uni}}$. Compute $\mathbf{b}_i = -s_i \cdot \mathbf{a} + \mathbf{e} \pmod 1$ and set the public key as $\mathsf{pk}_i = \mathbf{b}_i$.

• $\underline{\texttt{BootKeyGen}(i)}$: A party i generates and publishes a blind rotation key brk_i, a relinearization key rlk_i and a key-switching key ksk_i as follows.

– Sample $t_i \leftarrow \chi'$ and generate $\mathsf{brk}_{i,j} \leftarrow \texttt{RGSW.Enc}_{h_{gsw}}(t_i; z_{i,j})$ for $0 \le j < n$. Set the blind rotation key $\mathsf{brk}_i = \{\mathsf{brk}_{i,j}\}_{0 \le j < n}$.
– Generate the relinearization key $\mathsf{rlk}_i \leftarrow \texttt{UniEnc}_{h_{uni}}(s_i; t_i)$.
– Let $(s^*_{i,0}, \ldots, s^*_{i,N-1}) = (s_{i,0}, -s_{i,N-1}, \ldots, -s_{i,1})$. Sample $\mathbf{A}_{i,j} \leftarrow \mathbb{T}^{d_{ksk} \times n}$ and $\mathbf{e}_{i,j} \leftarrow \psi'^{d_{ksk}}$ for $0 \le j < N$, and let $\mathsf{ksk}_{i,j} = [\mathbf{b}_{i,j} | \mathbf{A}_{i,j}]$ where $\mathbf{b}_{i,j} = -\mathbf{A}_{i,j} \cdot \mathbf{s}_i + \mathbf{e}_{i,j} + s^*_{i,j} \cdot \mathbf{g}_{ksk}$. Set the key-switching key $\mathsf{ksk}_i = \{\mathsf{ksk}_{i,j}\}_{0 < N}$.

• $\underline{\texttt{HomNAND}(\{(\mathsf{pk}_i, \mathsf{brk}_i, \mathsf{rlk}_i, \mathsf{ksk}_i)\}_{i \in [k]}; \overline{\mathsf{ct}}_1, \overline{\mathsf{ct}}_2)}$: Given two LWE ciphertexts $\overline{\mathsf{ct}}_1$, $\overline{\mathsf{ct}}_2$ and key-quadruple $\{(\mathsf{pk}_i, \mathsf{brk}_i, \mathsf{rlk}_i, \mathsf{ksk}_i)\}_{i \in [k]}$ of associated parties, perform the following steps:

1. Compute $\overline{\mathsf{ct}} \leftarrow (\frac{5}{8}, 0, \ldots, 0) - \overline{\mathsf{ct}}_1 - \overline{\mathsf{ct}}_2 \pmod 1$.
2. Compute $\overline{\mathsf{ct}} \leftarrow \mathtt{BlindRotate}(\{(\mathsf{pk}_i, \mathsf{brk}_i, \mathsf{rlk}_i)\}_{i\in[k]}; \overline{\mathsf{ct}})$ where $\mathtt{BlindRotate}(\cdot)$ is the blind rotation algorithm in Alg. 3.
3. Compute $\overline{\mathsf{ct}} \leftarrow (\frac{1}{8}, 0, \ldots, 0) + \overline{\mathsf{ct}} \pmod 1$ and return $\mathsf{ct} = (b, \mathbf{a}_1, \ldots, \mathbf{a}_k) \in \mathbb{T}^{kN+1}$ where b is the constant term of $\overline{\mathsf{ct}}_0$ and $\mathbf{a}_i$ is the coefficient vector of $\overline{\mathsf{ct}}_i$ for $i \in [k]$.
4. Perform the key-switching process: Compute $(b'_i, \mathbf{a}'_i) = \sum_{j=0}^{N-1} h'(a_{i,j}) \cdot \mathsf{ksk}_{i,j} \pmod 1$ for $i \in [k]$ and $b' = b + \sum_{i\in[k]} b'_i$. Return $\mathsf{ct}' = (b', \mathbf{a}'_1, \ldots, \mathbf{a}'_k) \in \mathbb{T}^{kn+1}$.

Algorithm 3 New Blind Rotation using Different Gadget Decompositions

Input: $\overline{\mathsf{ct}} = (b, \mathbf{a}_1, \ldots, \mathbf{a}_k), \{(\mathsf{pk}_i, \mathsf{brk}_i, \mathsf{rlk}_i)\}_{i\in[k]}$
Output: ACC

Let $\tilde{b} := \lfloor 2Nb \rceil$, $\tilde{a}_{i,j} := \lfloor 2Na_{i,j} \rceil$ for $1 \le i \le k$, $0 \le j < n$
  and $v := -\frac{1}{8} \cdot (1 + X + \cdots + X^{N/2-1} - X^{N/2} - \cdots - X^{N-1})$.
$\mathsf{ACC} \leftarrow (v \cdot X^{\tilde{b}}, 0, \ldots, 0)$
for $1 \le i \le k$ **do**
  $\mathsf{ACC}'_i \leftarrow (\mathbf{g}_{lev}, \mathbf{0})$
  for $0 \le j < n$ **do**
    $\mathsf{ACC}'_i \leftarrow \mathsf{ACC}'_i + \left[(X^{\tilde{a}_{i,j}} - 1) \cdot \mathsf{ACC}'_i\right] \circledast_{h_{lev}} \mathsf{brk}_{i,j}$
  end for
end for
for $1 \le i \le k$ **do**
  $\mathsf{ACC} \leftarrow \mathtt{ExtProd}_{h_{lev}, h_{uni}}(\{\mathsf{pk}_\ell\}_{\ell\in[k]}, \mathsf{rlk}_i; \mathsf{ACC}, \mathsf{ACC}'_i)$
end for

B Proofs for the Noise Analysis

First we define GdErr_{gsw}, GdErr_{lev} and GdErr_{uni}, the gadget decomposition error of h_{gsw}, h_{lev} and h_{uni} respectively.

Proof of Lemma 1 (T-RLEV Multiplication)

Proof. By definition, we have

$$\begin{aligned}\varphi_s(c \odot \mathbf{C}) &= \langle h_{lev}(c), \varphi_s(\mathbf{C})\rangle \\ &= \langle h_{lev}(c), \mu \cdot \mathbf{g}_{lev} + \mathsf{Err}(\mathbf{C})\rangle \\ &= \mu \cdot c + \mu \cdot \mathsf{GdErr}_{lev}(c) + \langle h_{lev}(c), \mathsf{Err}(\mathbf{C})\rangle \pmod 1.\end{aligned}$$

Therefore $e = \mu \cdot \mathsf{GdErr}_{lev}(c) + \langle h_{lev}(c), \mathsf{Err}(\mathbf{C})\rangle$. Since $h_{lev}(c)$ and $\mathsf{err}(\mathbf{C})$ are vectors of length d_{lev}, we get

$$\mathsf{Var}(e) = \|\mu\|_2^2 \epsilon_{lev}^2 + d_{lev} N V_{lev} \mathsf{VarErr}(\mathbf{C})$$

□

Proof of Lemma 2 (RLWE-RGSW Multiplication)

Proof. Let $\mathbf{c} = (b, a)$ and $\mathsf{Err}(\mathbf{C}) = \begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_1 \end{bmatrix}$ where $\mathbf{e}_0, \mathbf{e}_1 \in T^{d_{gsw}}$. Then we can obtain

$$\begin{aligned}\varphi_s(\mathbf{c} \otimes \overline{\mathbf{C}}) &= \left\langle h_{gsw}(\mathbf{c}), \varphi_s(\overline{\mathbf{C}}) \right\rangle \\ &= \langle h_{gsw}(b), \mu \cdot \mathbf{g}_{gsw} + \mathbf{e}_0 \rangle + \langle h_{gsw}(a), \mu \cdot s \cdot \mathbf{g}_{gsw} + \mathbf{e}_1 \rangle \\ &= \mu \cdot \varphi_s(\mathbf{c}) + \mu \cdot (\mathsf{GdErr}_{gsw}(b) + \mathsf{GdErr}_{gsw}(a)s) + \left\langle h_{gsw}(\mathbf{c}), \mathsf{Err}(\overline{\mathbf{C}}) \right\rangle \pmod 1.\end{aligned}$$

Therefore, we can get the following error variance.

$$\mathsf{Var}(e) = (1 + N/2)\|\mu\|_2^2 \epsilon_{gsw}^2 + 2d_{gsw} N V_{gsw} \mathsf{VarErr}(\overline{\mathbf{C}}).$$

□

Proof of Corollary 1 (RLEV-RGSW Multiplication)

Proof. An RLEV ciphertext can be seen as a column vector of RLWE ciphertexts. Therefore, this corollary can be shown directly from the previous lemma. □

For efficiency, we prove Lemma 4 first, and then prove Lemma 3.

Proof of Lemma 4 (New Hybrid Product)

Proof. We shall use $\mathbf{e}_0 = \mathbf{0}$ and the same temporary variables as in the algorithm description for the easier notation. Let $\mathbf{u} = (u_0, \ldots, u_k)$ and $\mathbf{w} = (w_0, w_1)$. Then we have $\varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}') = \varphi_{\overline{\mathbf{s}}}(\mathbf{u}) + \varphi_{s_i}(\mathbf{w}) \pmod 1$. The first term is

$$\begin{aligned}\varphi_{\overline{\mathbf{s}}}(u) &= \sum_{j=0}^{k} \langle h_{uni}(c_j), r \cdot \mathbf{a} + \mu \cdot \mathbf{g}_{uni} + \mathbf{e}_{i,1} \rangle \cdot s_j \\ &= \mu \cdot \varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}) + \mu \cdot \sum_{j=0}^{k} \mathsf{GdErr}_{uni}(c_j) \cdot s_j + r \cdot \sum_{j=0}^{k} \langle h_{uni}(c_j), s_j \cdot \mathbf{a} \rangle \\ &\quad + \sum_{j=0}^{k} \langle h_{uni}(c_j), \mathbf{e}_{i,1} \rangle \cdot s_j \pmod 1,\end{aligned}$$

and the second term is

$$\begin{aligned}\varphi_{s_i}(\mathbf{w}) &= \langle h_{uni}(v), \mathbf{F}_{i,0} \rangle + \langle h_{uni}(v), \mathbf{F}_{i,1} \rangle \cdot s_i \\ &= \langle h_{uni}(v), r \cdot \mathbf{g}_{uni} + \mathbf{e}_{i,2} \rangle \\ &= r \cdot v + r \cdot \mathsf{GdErr}_{uni}(v) + \langle h_{uni}(v), \mathbf{e}_{i,2} \rangle \pmod 1.\end{aligned}$$

Now, from the fact that

$$\begin{aligned}v + \sum_{j=0}^{k} \langle h_{uni}(c_j), s_j \cdot \mathbf{a} \rangle &= \sum_{j=0}^{k} (\langle h_{uni}(c_j), \mathbf{b}_j \rangle + \langle h_{uni}(c_j), s_j \cdot \mathbf{a} \rangle) \\ &= \sum_{j=1}^{k} \langle h_{uni}(c_j), \mathbf{e}_j \rangle \pmod 1,\end{aligned}$$

it follows that $e = \varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}') - \mu \cdot \varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}})$ is

$$\mu \cdot \sum_{j=0}^{k} \mathsf{GdErr}_{uni}(c_j) \cdot s_j + \sum_{j=0}^{k} \langle h_{uni}(c_j), \mathbf{e}_{i,1} \rangle \cdot s_j +$$

$$r \cdot \mathsf{GdErr}_{uni}(v) + \langle h_{uni}(v), \mathbf{e}_{i,2} \rangle + r \cdot \sum_{j=1}^{k} \langle h_{uni}(c_j), \mathbf{e}_j \rangle .$$

Therefore, we have

$$\begin{aligned}\mathsf{Var}(e) &= \|\mu\|_2^2 (1 + kN/2) N \epsilon_{uni}^2 + (1 + kN/2) d_{uni} N V_{uni} \beta^2 \\ &\quad + (N/2)\epsilon_{uni}^2 + d_{uni} N V_{uni} \beta^2 + k d_{uni} (N^2/2) V_{uni} \beta^2 \\ &\approx \frac{k}{2} \|\mu\|_2^2 N^2 \epsilon_{uni}^2 + k d_{uni} N^2 V_{uni} \beta^2 .\end{aligned}$$

□

Proof of Lemma 3 (Hybrid Product)

Proof. The only difference of the error variance of `HbProd` to the error variance `NewHbProd` is the error from $w_{j,0}$ and $w_{j,1}$'s, which is $k+1$ times bigger than `NewHbProd`. Therefore, we get the error variance of

$$\begin{aligned}\mathsf{Var}(e) &= \|\mu\|_2^2 (1 + kN/2) N \epsilon_{uni}^2 + (1 + kN/2) d_{uni} N V_{uni} \beta^2 \\ &\quad + (k+1)(N/2)\epsilon_{uni}^2 + (k+1) d_{uni} N V_{uni} \beta^2 + k d_{uni} (N^2/2) V_{uni} \beta^2 \\ &\approx \frac{k}{2} \|\mu\|_2^2 N^2 \epsilon_{uni}^2 + k d_{uni} N^2 V_{uni} \beta^2 .\end{aligned}$$

□

Proof of Corollary 2 (Blind Rotation of CCS19)

Proof. We first analyze the line 6. Let $\overline{\mathbf{c}}_{i,j} = \mathtt{HbProd}(\{\mathsf{pk}_j\}_{j \in [k]}, (X^{\tilde{a}_{i,j}} - 1) \cdot \mathsf{ACC}, \mathsf{BK}_{i,j})$ and $\mathbf{e}_{i,j}$ be the error solely from the `HbProd`. Then,

$$\begin{aligned}\varphi_{\overline{\mathbf{s}}}(\mathsf{ACC} + \overline{\mathbf{c}}_{i,j}) &= \varphi_{\overline{\mathbf{s}}}(\mathsf{ACC}) + \varphi_{\overline{\mathbf{s}}}((X^{\tilde{a}_{i,j}} - 1) \cdot \mathsf{ACC}) + \mathbf{e}_{i,j} \\ &= \varphi_{\overline{\mathbf{s}}}(X^{\tilde{a}_{i,j} z_{i,j}} \cdot \mathsf{ACC}) + \mathbf{e}_{i,j} \pmod 1.\end{aligned}$$

Note that during the i-th iteration, ACC should be regarded as a multi-key RLWE ciphertext with i parties since $i+1, \ldots, k$-th indices remains zero. Therefore,

$$\begin{aligned}\mathsf{Var}(\mathbf{e}_{i,j}) &\approx \frac{i}{2} \|z_{i,j}\|_2^2 N^2 \epsilon_{uni}^2 + i d_{uni} N^2 V_{uni} \beta^2 \\ &= \frac{i}{4} N^2 \epsilon_{uni}^2 + i d_{uni} N^2 V_{uni} \beta^2\end{aligned}$$

Since $X^{\tilde{a}_{i,j} z_{i,j}}$ is a monomial, the variance adds up every iteration in the inner loop (line 5-7) and therefore an error variance of $\frac{i}{4} n N^2 \epsilon_{uni}^2 + i d_{uni} n N^2 V_{uni} \beta^2$ is added for every iteration in the outer loop (line 4-8). Therefore we get the error variace of

$$\frac{k(k+1)}{8} n N^2 (\epsilon_{uni}^2 + 4 d_{uni} V_{uni} \beta^2).$$

□

Proof of Lemma 5 (Generalized External Product)

Proof. Let us follow the notations from the algorithm description. First, by Lemma 1 we obtain

$$\begin{aligned}\varphi_{\overline{\mathbf{s}}}(\overline{\mathbf{x}} + \overline{\mathbf{y}} \cdot t) &= \sum_{j=0}^{k} (x_j + y_j \cdot t) \cdot s_j \\ &= \sum_{j=0}^{k} \varphi_t(c_j \odot \mathbf{C}_i) \cdot s_j \\ &= \mu \cdot \varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}) + \sum_{j=0}^{k} e_j \cdot s_j \pmod 1.\end{aligned}$$

where $e_j = \varphi_t(c_j \odot \mathbf{C}_i) - \mu \cdot c_j (0 \le j \le k) \pmod 1 \in T$ with variance

$$\mathsf{Var}(e_j) = \|\mu\|_2^2 \epsilon_{lev}^2 + d_{lev} N V_{lev} \mathsf{VarErr}(\mathbf{C}_i).$$

Let $\overline{\mathbf{y}}' = \mathtt{NewHbProd}(\{\mathbf{b}_j\}_{j\in[k]}, \mathsf{ct}, \mathsf{rlk}_i)$. By Lemma 4, $\varphi_{\overline{\mathbf{s}}}(\overline{\mathbf{y}}') = t \cdot \varphi_{\overline{\mathbf{s}}}(\overline{\mathbf{y}}) + e'$ for some $e' \in T$ with variance

$$\begin{aligned}\mathsf{Var}(e') &\approx \frac{k}{2} \mathsf{Var}(\|t\|_2^2) N^2 \epsilon_{uni}^2 + k d_{uni} N^2 V_{uni} \beta^2 \\ &= \frac{k}{4} N^3 \epsilon_{uni}^2 + k d_{uni} N^2 V_{uni} \beta^2.\end{aligned}$$

Therefore, we get

$$\begin{aligned}\varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}') &= \varphi_{\overline{\mathbf{s}}}(\overline{\mathbf{x}}) + \varphi_{\overline{\mathbf{s}}}(\overline{\mathbf{y}}') \\ &= \varphi_{\overline{\mathbf{s}}}(\overline{\mathbf{x}}) + \varphi_{\overline{\mathbf{s}}}(\overline{\mathbf{y}}) \cdot t + e' \\ &= \mu \cdot \varphi_{\overline{\mathbf{s}}}(\overline{\mathsf{ct}}) + \sum_{j=0}^{k} e_j \cdot s_j + e' \pmod 1.\end{aligned}$$

Therefore the variance of error $e = \sum_{j=0}^{k} e_j \cdot s_j + e'$ be

$$\begin{aligned}\mathsf{Var}(e) \approx &(1 + kN/2) \left[\|\mu\|_2^2 \epsilon_{lev}^2 + d_{lev} N V_{lev} \mathsf{VarErr}(\mathbf{C}_i) \right] \\ &+ \frac{k}{4} N^3 \epsilon_{uni}^2 + k d_{uni} N^2 V_{uni} \beta^2.\end{aligned}$$

□

Proof of Theorem 1 (Our Blind Rotation)

Proof. We start from analyzing line 7 of the algorithm. By Corollary 1,

$$\begin{aligned}\varphi_{t_i}(\mathsf{ACC}_i' + &\left[(X^{\tilde{a}_{i,j}} - 1) \cdot \mathsf{ACC}_i' \right] \circledast \mathsf{BK}_{i,j}) \\ &= \varphi_{t_i}(\mathsf{ACC}_i') + \varphi_{t_i}(\left[(X^{\tilde{a}_{i,j}} - 1) \cdot \mathsf{ACC}_i' \right] \circledast \mathsf{BK}_{i,j}) \\ &= \varphi_{t_i}(X^{\tilde{a}_{i,j} z_{i,j}} \cdot \mathsf{ACC}_i') + \mathbf{e}_{i,j} \pmod 1\end{aligned}$$

for some $\mathbf{e}_{i,j} \in T^{d_{lev}}$ with the common variance of rows $(1+N/2)\|\mu z_{i,j}\|_2^2\epsilon_{gsw}^2 + 2d_{gsw}NV_{gsw}\mathsf{VarErr}(\mathsf{BK}_{i,j})$. Therefore, with each iteration within the for loop 6-8, the error variance increases by $V_{ACC} = (1+N)\epsilon_{gsw}^2 + 2d_{gsw}NV_{gsw}\beta^2$. Hence, the error variance of the resulting RLEV ciphertext ACC'_i of the for loop is $n \cdot ((1+N)\epsilon_{gsw}^2 + 2d_{gsw}NV_{gsw}\beta^2)$ with message $X^{\sum_{j=1}^{n}\tilde{a}_{i,j}z_{i,j}}$.

Note that only $0, \ldots, i$-th indices of the MK-RLWE ciphertext ACC are non-zero after the $i-th$ iteration of the for loop 4-10, hence it can be regarded as an MK-RLWE encryption of i parties. Now we let ACC_i be ACC after the `ExtProd`, then by the Lemma 5, we have

$$\varphi_{\bar{\mathbf{s}}}(\mathsf{ACC}_i) = X^{\langle \tilde{\mathbf{a}}_i, \mathbf{z}_i \rangle} \cdot \varphi_{\bar{\mathbf{s}}}(\mathsf{ACC}_{i-1}) + e_i (1 \le i \le k)$$

where

$$\mathsf{Var}(e_i) \approx (1+iN/2)\left[\epsilon_{lev}^2 + d_{lev}NV_{lev}V_{ACC}\right] + \frac{i}{4}N^3\epsilon_{uni}^2 + id_{uni}N^2V_{uni}\beta^2.$$

Since $X^{\tilde{\mathbf{a}}_i, \mathbf{z}_i}$ is a monomial, the error variance adds up every iteration and thus the error variance of the final output of our new blind rotation algorithm is

$$\frac{k(k+1)}{8}\left[2d_{lev}nN^3V_{lev} \cdot \left(2d_{gsw}V_{gsw}\beta^2 + \epsilon_{gsw}^2\right) + N^3\epsilon_{uni}^2 + 4d_{uni}N^2V_{uni}\beta^2\right].$$

□

References

1. Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., Wichs, D.: Multiparty computation with low communication, computation and interaction via threshold fhe. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 483–501. Springer (2012)
2. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical gapsvp. In: Annual Cryptology Conference. pp. 868–886. Springer (2012)
3. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) **6**(3), 1–36 (2014)
4. Brakerski, Z., Perlman, R.: Lattice-based fully dynamic multi-key fhe with short ciphertexts. In: Annual International Cryptology Conference. pp. 190–213. Springer (2016)
5. Chen, H., Chillotti, I., Song, Y.: Multi-key homomorphic encryption from tfhe. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 446–472. Springer (2019)
6. Chen, H., Dai, W., Kim, M., Song, Y.: Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 395–412 (2019)
7. Chen, L., Zhang, Z., Wang, X.: Batched multi-hop multi-key FHE from Ring-LWE with compact ciphertext extension. In: Theory of Cryptography Conference. pp. 597–627. Springer (2017)

8. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 409–437. Springer (2017)
9. Chillotti, I., Gama, N., Georgieva, M., Izabachene, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: international conference on the theory and application of cryptology and information security. pp. 3–33. Springer (2016)
10. Chillotti, I., Ligier, D., Orfila, J.B., Tap, S.: Improved programmable bootstrapping with larger precision and efficient arithmetic circuits for tfhe. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 670–699. Springer (2021)
11. Clear, M., McGoldrick, C.: Multi-identity and multi-key leveled fhe from learning with errors. In: Annual Cryptology Conference. pp. 630–656. Springer (2015)
12. Ducas, L., Micciancio, D.: Fhew: bootstrapping homomorphic encryption in less than a second. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 617–640. Springer (2015)
13. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptol. ePrint Arch. **2012**, 144 (2012)
14. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the forty-first annual ACM symposium on Theory of computing. pp. 169–178 (2009)
15. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Annual Cryptology Conference. pp. 75–92. Springer (2013)
16. Kim, A., Deryabin, M., Eom, J., Choi, R., Lee, Y., Ghang, W., Yoo, D.: General bootstrapping approach for rlwe-based homomorphic encryption. Cryptology ePrint Archive (2021)
17. Kim, T., Kwak, H., Lee, D., Seo, J., Song, Y.: Asymptotically faster multi-key homomorphic encryption from homomorphic gadget decomposition. Cryptology ePrint Archive (2022)
18. Klemsa, J., Önen, M., Akın, Y.: A practical tfhe-based multi-key homomorphic encryption with linear complexity and low noise growth. Cryptology ePrint Archive (2023)
19. Kraitsberg, M., Lindell, Y., Osheter, V., Smart, N.P., Talibi Alaoui, Y.: Adding distributed decryption and key generation to a ring-lwe based cca encryption scheme. In: Information Security and Privacy: 24th Australasian Conference, ACISP 2019, Christchurch, New Zealand, July 3–5, 2019, Proceedings 24. pp. 192–210. Springer (2019)
20. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Proceedings of the forty-fourth annual ACM symposium on Theory of computing. pp. 1219–1234 (2012)
21. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. Journal of the ACM (JACM) **60**(6), 1–35 (2013)
22. malb: lattice-estimator. `https://github.com/malb/lattice-estimator` (2022)
23. Mouchet, C., Troncoso-Pastoriza, J., Bossuat, J.P., Hubaux, J.P.: Multiparty homomorphic encryption from ring-learning-with-errors. Proceedings on Privacy Enhancing Technologies **2021**(4), 291–311 (2021)
24. Mukherjee, P., Wichs, D.: Two round multiparty computation via multi-key fhe. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 735–763. Springer (2016)

25. Park, J.: Homomorphic encryption for multiple users with less communications. IEEE Access **9**, 135915–135926 (2021)
26. Peikert, C., Shiehian, S.: Multi-key fhe from lwe, revisited. In: Theory of cryptography conference. pp. 217–238. Springer (2016)
27. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM) **56**(6), 1–40 (2009)