Multi-key Fully Homomorphic Encryption with Non-Interactive Setup in the Plain Model

Seonhong Min[1], Jeongeun Park[2], and Yongsoo Song[1]

[1] Seoul National University
[2] Norwegian University of Science and Technology

Abstract. Multi-key fully homomorphic encryption (MKFHE) enables computation over encrypted data under multiple different keys. Constructing MKFHE without any trusted or interactive setup remains an open problem. In the context of MKFHE, a trusted setup is often assumed to mean the use of a common random string (CRS).

In this paper, we present the first MKFHE scheme in the plain model (i.e., without any trusted or interactive setup) based on the RLWE assumption. Specifically, we construct a multi-key somewhat homomorphic encryption based on the RLWE assumption and extend it to a multi-key variant of the Gentry-Sahai-Waters (GSW) scheme with a circular security assumption.

Our design yields a 2-round multi-party computation (MPC) in the plain model against semi-honest adversaries. Moreover, it can be applied to transform existing FHE schemes that rely on RGSW in their construction into a multi-key variant. We also provide concrete conversions for widely-used FHE schemes, including BGV, BFV, CKKS, FHEW, TFHE, and Carousel.

Finally, we implement our scheme and present experimental results for the expansion algorithm from a single-key ciphertext to a multi-key ciphertext and the multi-key homomorphic multiplication algorithm.

1 Introduction

Over the last decade, fully homomorphic encryption (FHE), which enables computation over encrypted data, has emerged as a promising solution for privacy-preserving computation. Its non-interactive nature makes it particularly well-suited for cloud-based applications, enabling secure processing of sensitive data without requiring interaction between parties [23, 43, 24, 31, 22]. Notably, recent advances have brought FHE to a level of practicality where it is not only adopted in research prototypes but also deployed in real-world systems, for example, Apple's FHE-based caller ID system [5].

Despite recent promising progress, most of the development has focused on the single-user setting, whereas real-world cloud-based applications often involve data from multiple owners. Examples include collaborative analysis, federated learning, joint financial risk assessments, and secure multi-party healthcare data sharing. Recognizing the importance of this field, the U.S. National Institute of

Standards and Technology (NIST) recently issued a call [47] to promote research on extending FHE to support multi-party settings.

Multi-key fully homomorphic encryption (MKFHE) [41] directly supports computation across multiple parties without requiring interaction between participants. In other words, MKFHE enables computation on inputs encrypted under different keys, allowing a function to be evaluated over multiple parties' data while ensuring that each party's input remains secret from the others.

Such a strong functionality of MKFHE can be applied to a wide range of real-world scenarios, including blockchain platforms. In fact, FHE has been applied to blockchain in works aimed at enabling privacy-preserving smart contracts. However, all of these constructions are designed for the single-user setting [52] or rely on a global secret key distributed to trusted MPC nodes [59]. Therefore, to support multi-user computation in a truly decentralized manner, it is essential to achieve MKFHE without a trusted setting.

Indeed, the first ever construction of MKFHE [41], namely the LTV scheme, requires no interaction among parties and does not rely on any shared randomness, such as a common random/reference string (CRS), which is generated from *a trusted setup*. This feature was mainly due to the employment of the DSPR (Decisional Small Polynomial Ratio) problem, which is a milder variant of the NTRU problem. On the other hand, its disadvantage lies in the multi-round decryption, which stems from the multiplicative joint key structure. This issue was later resolved by Ananth, Jain, Jin, and Malavolta, from a generic transformation for constructing MKFHE in the plain model (without a trusted setup), which supports a one-round decryption protocol [4]. The transformation requires the following three building blocks:

1. An MKFHE scheme with a trusted setup and one-round decryption,
2. An MKFHE scheme without a trusted setup but with arbitrary round decryption,
3. A two-round semi-malicious oblivious transfer protocol without a trusted setup.

Concretely, the first building block can be instantiated easily with existing RLWE-based MKFHE schemes in the CRS model [15, 16, 34, 36], as they inherently support one-round decryption. The LTV scheme can be used for the second building block, and finally, the third block can be constructed from LWE-based two-round OT protocols, for instance, [10, 8]. However, as the LTV scheme has exponential noise growth with respect to the number of parties, it is hard to concretely instantiate it in a secure parameter regime [18, 1].

Nevertheless, all secure constructions [17, 46, 34, 36, 15, 16] based on RLWE rely on a CRS to achieve practical efficiency and simplify key entanglement for correct decryption. While there have been a few attempts [11, 32] to remove reliance on a CRS by introducing an interactive setup among a fixed number of parties, this approach sacrifices the dynamic property, namely, the ability to support ad hoc participants on-the-fly, which was inherently supported by the original MKFHE construction. Several recent works [56, 49, 54] attempt to revive DSPR in the context of MKFHE with carefully chosen small parameters

to ensure security, but all of them ultimately rely on (R)LWE ciphertexts for correctness, thereby requiring a CRS. Furthermore, the scheme proposed in [56] was later shown to be insecure due to its use of a fixed Hamming weight, as demonstrated in [33].

Consequently, it leads us to the following fundamental research question:

Can we construct a secure multi-key HE scheme in the plain model without any pre-computation, and with one-round decryption?

Importantly, even under a relaxed version of the problem where one-round decryption is not required and it suffices to support an arbitrary number of decryption rounds, no existing *secure MKFHE scheme* without a trusted setup is known to fulfill this requirement [3].

Our Contribution. In this paper, we resolve the above research question by presenting two multi-key HE schemes. We first construct a multi-key somewhat HE (MKSHE) scheme based on the RLWE problem in the plain model with no interactive setup, in Sec. 3.1. At the core of our MKFHE, there lie new single-key encryptions called the *compact LEV* (cLEV) and *compact GSW* (cGSW) that rely on the shared-A RLWE assumption. These encryptions enable the evaluation of arbitrary polynomials of a-priori chosen degree without the CRS assumption.

We then achieve a multi-key variant of the Gentry-Sahai-Waters (GSW) fully homomorphic encryption scheme [27] by extending the cLEV and cGSW schemes in Sec. 3, assuming the circular security of the cGSW scheme. This new multi-key RGSW cryptosystem has a non-linear access structure for the final joint key, which depends on the multiplication order in the computation, unlike the CRS-based MKFHE schemes. In our solution, the server follows the joint key structure by keeping track of the order of multiplications and uses this information for key switching and bootstrapping. After the desired homomorphic computation, the decryption can be executed by evaluating the decryption circuit of the RGSW scheme using any existing MPC primitives.

We also demonstrate that we can construct an MKFHE scheme in the plain model with one-round decryption via the transformation by Ananth et al. [4]. While we do not explicitly construct a one-round decryption protocol in combination with the other components, our result provides the final missing ingredient required for its theoretical realization. Our work thus bridges a critical gap and opens the way for the future concrete instantiation of MKFHE in the plain model with one-round decryption. We note that this furthermore yields a two-round multi-party computation (MPC) in the plain model against semi-honest adversaries, as established in [4] [4]. We describe a protocol based on our MKFHE in Sec. 4.3.

Moreover, we show that our scheme can be applied to existing RLWE-based schemes such as BFV [9, 26], CKKS [19], TFHE [20], LMKCDEY [38], and

[3] Note that the transform from [4] directly implies a one-round MKFHE as a result.

[4] Since our design satisfies all the assumptions and conditions of their paper, we do not provide a separate security proof.

Carousel [30], as we discuss in Sec. 5. These RLWE-based FHE schemes employ the RGSW scheme to perform multiplicative operations, and our design almost directly implies the multi-key variant of these schemes. In other words, our multi-key RGSW ciphertext can replace any CRS-based RGSW ciphertext, thereby eliminating the use of a CRS in the multi-key variants of existing schemes.

Finally, we provide a proof-of-concept implementation of our MKFHE scheme using the HIENAA.jl library [29] with concrete parameter sets and demonstrate the performance of our scheme in Sec. 6.

Scheme	Security Model	Hardness Assumption
[41]	Plain	DSPR
[46, 21],[50, Scheme I]	CRS	(R)LWE
[13, 17, 39, 15, 16, 34, 36, 55] [50, Scheme II]	CRS	(R)LWE, Circular Security
[54, 49]	CRS	DSPR, (R)LWE, Circular Security
MKSHE (Sec. 3.1)	Plain	(R)LWE
MKFHE (Sec. 4)	Plain	(R)LWE, Circular Security

Table 1. Comparison of our scheme with related works.

In Table 1, the security model, hardness assumptions of each multi-key homomorphic encryption scheme are summarized. As shown in the table, our multi-key scheme is the only one that is constructed in the plain model and relies on the RLWE hardness assumption.

1.1 Technical Overview

For some positive integer $Q > 0$ and a power-of-two N (which is a fixed parameter throughout the paper), we write $R := \mathbb{Z}[X]/(X^N+1)$ and $R_Q := R/QR$. Also, $\vec{g} = (g_0, \ldots, g_{d-1}) \in \mathbb{Z}^d$ is a gadget vector, and $g^{-1} : R_Q \to R^d$ is the corresponding decomposition function, such that $\|g^{-1}(a)\|_\infty$ is small and $a = g^{-1}(a)^\top \cdot \vec{g} \pmod{Q}$ for any $a \in R_Q$. For a vector $\mathbf{a} = (a_i)_{1 \le i \le k} \in R_Q^k$, we write $g^{-1}(\mathbf{a})$ to denote $\left[g^{-1}(a_1)^\top \ \ldots \ g^{-1}(a_k)^\top\right]^\top$.

cLEV and cGSW. Consider a multi-key RLWE ciphertext $\mathbf{c} = (c_0, \ldots, c_{k-1}) \in R_Q^k$ of a plaintext $m \in R_Q$ under a joint key $\mathsf{sk} = (s_0, \ldots, s_{k-1}) \in R^k$ of parties $1, \ldots, k-1$. In other words,

$$\sum_{i=0}^{k-1} c_i s_i = \mathsf{sk}^\top \cdot \mathbf{c} \approx m \pmod{Q}.$$

Assume that the k-th party wants to multiply some plaintext $\mu \in R_Q$ with this ciphertext. In the conventional MLWE-RGSW multiplication, the linear

decryption formula is homomorphically computed, but the secret is multiplied by the plaintext. Specifically, we homomorphically evaluate

$$\sum_{i=0}^{k-1} c_i \cdot (\mu \cdot s_i) = (\mu \cdot \mathsf{sk})^\top \cdot \mathbf{c} \approx \mu \cdot m \pmod{Q}$$

given the encryptions of $g_j \cdot \mu \cdot s_i$ for $0 \leq i < k$ and $0 \leq j < d$. However, this is challenging to realize in the conventional multi-key scenario, as the parties do not have access to all s_i's and therefore cannot generate the encryptions of $g_j \cdot \mu \cdot s_i$'s directly. In prior works [46, 50, 15, 36], this challenge was resolved by using the CRS. Roughly speaking, the encryptions of $g_j \cdot \mu \cdot s_i$ can be homomorphically generated from the pre-computed arithmetic relation between the CRS and each party's key.

Instead, we homomorphically compute a vector $\mu \cdot \mathbf{c} \in R_Q^k$ packed into a single ciphertext. Our approach does not require any information about the other parties' secrets, removing the need for additional arithmetic relations using CRS. A conventional multi-key ciphertext can encrypt only a single ring element, not a vector. To overcome this challenge, we construct a new cryptosystem that can encrypt a plaintext vector into a single ciphertext, on top of the *shared-A RLWE*. Briefly speaking, the shared-A RLWE is a set of several RLWE ciphertexts that share the randomness part, encrypted under different keys. Concretely, given the secret key $\mathbf{t} = (t_1, \ldots, t_k) \in R^k$, a shared-A RLWE encryption of a vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_k) \in R_Q^k$ is given by $(b_1, \ldots, b_k, a) \in R_Q^{k+1}$ where

$$\forall 1 \leq i \leq k, \quad b_i + a \cdot t_i \approx \mu_i \pmod{Q}.$$

It is worth noting that the shared-A RLWE has been explicitly and implicitly considered in many works [51, 12, 6, 7], including MKFHE [15, 16, 34, 36].

On top of shared-A RLWE, we construct the cLEV and cGSW schemes. Let $k \in \mathbb{Z}_{>0}$ be the dimension of the encryption. Concretely, a cLEV encryption of μ under the secret $\mathbf{t}$ is a matrix $\mathbf{C} \in R_Q^{(k+1)\times dk}$ of ring elements such that $\left[\mathbf{I}_k \mid \mathbf{t}\right] \cdot \mathbf{C} \approx \mu \cdot \mathbf{I}_k \otimes \vec{g}^\top \bmod Q$, and cGSW encryption of μ is a matrix $\overline{\mathbf{C}} \in R_Q^{(k+1)\times d(k+1)}$ such that $\left[\mathbf{I}_k \mid \mathbf{t}\right] \cdot \overline{\mathbf{C}} \approx \mu \cdot \left[\mathbf{I}_k \mid \mathbf{t}\right] \otimes \vec{g}^\top \bmod Q$, where $\otimes$ denotes the tensor product of matrices and $\mathbf{I}_k$ denotes the identity matrix of size $k \times k$.

We also propose new multiplicative operations called the *vector gadget product*, which is defined between a vector of ring elements and a cLEV encryption, and the *vector external product*, which is defined between a shared-A RLWE encryption and a cGSW encryption. The vector gadget product is defined between a vector $\mathbf{v} \in R_Q^k$ and a cLEV ciphertext $\mathbf{C} \in R_Q^{(k+1)\times dk}$, and is denoted as $\mathbf{v} \odot_v \mathbf{C}$. If $\mathbf{C}$ is an encryption of $\mu \in R$, it outputs a shared-A encryption of $\mu \cdot \mathbf{v}$ (mod Q). The vector external product is defined between a shared-A encryption $\mathbf{c} \in R_Q^{k+1}$ and a cGSW ciphertext $\overline{\mathbf{C}} \in R_Q^{(k+1)\times d(k+1)}$, and is denoted as $\mathbf{c} \boxdot_v \overline{\mathbf{C}}$. If $\overline{\mathbf{C}}$ is an encryption of $\mu \in R$, it returns a shared-A encryption of $\mu \cdot \mathbf{m}$ for the underlying plaintext $\mathbf{m} \in R^k$ of $\mathbf{c}$. Consequently, they allow multiplicative operations over vectors.

Our main observation is that if $\mathbf{c}$ is a multi-key encryption of m under the joint secret sk, and $\mathbf{C}$ is a cLEV encryption of μ under the secret $\mathbf{t} \in R^k$, $\mathbf{c}' := \mathbf{c} \odot_v \mathbf{C}$ can be seen as an encryption of $\mu \cdot m$ under a new joint secret sk', obtained from sk and $\mathbf{t}$. More specifically, the following (approximate) equality implies that the vector gadget product outputs a ciphertext encrypted under a joint key $\mathsf{sk}' = \begin{bmatrix} \mathbf{I}_k \\ \mathbf{t}^\top \end{bmatrix} \cdot \mathsf{sk}$.

$$\mathsf{sk}^\top \cdot \begin{bmatrix} \mathbf{I}_k \mid \mathbf{t} \end{bmatrix} \cdot \mathbf{c}' = \mathsf{sk}^\top \cdot \begin{bmatrix} \mathbf{I}_k \mid \mathbf{t} \end{bmatrix} \cdot (\mathbf{c} \odot_v \mathbf{C}) \approx \mathsf{sk}^\top \cdot \mu \cdot \mathbf{c} \approx \mu \cdot m \pmod{Q}.$$

We also observe that we can perform additional multiplications over $\mathbf{c}'$, from the cGSW encryption and gadget external product. Let $\overline{\mathbf{C}}$ be a cGSW encryption of μ' under secret $\mathbf{t}$. Then, $\mathbf{c}'' := \mathbf{c}' \boxdot_v \overline{\mathbf{C}}$ is a multi-key encryption of $\mu' \cdot \mu \cdot m$ under sk'. Since $\mathbf{c}'$ is an encryption of $\mu \cdot \mathbf{c}$ under secret $\begin{bmatrix} \mathbf{I}_k \mid \mathbf{t} \end{bmatrix}$, $\mathbf{c}''$ decrypts to $\mu' \cdot (\mu \cdot \mathbf{c})$ by the functionality of the vector external product. Then, with the joint key sk', it decrypts to $\mu'\mu m$ by the discussion above.

Multi-key Somewhat HE. The cLEV and cGSW encryptions directly imply an MKSHE scheme. Concretely, each i-th party outputs a cLEV encryption of dimension i and performs the vector gadget product sequentially in order to compute the product of their messages. More specifically, let $\mathbf{C}_i \in R_Q^{(i+1)\times id}$ be a cLEV ciphertext of dimension i encrypting some message $\mu_i \in R_Q$ under $\mathbf{s}_i \in R^i$, the secret key of the i-th party for $1 \le i \le k$, where k denotes the number of participants. Then, for some plaintext $m \in R_Q$, $m \cdot \prod_{i=1}^k \mu_i \in R_Q$ can be homomorphically computed as $\mathbf{c} := (\dots((m \odot_v \mathbf{C}_1) \odot_v \mathbf{C}_2) \odot_v \dots) \odot_v \mathbf{C}_k$, encrypted under the secret $\overline{\mathsf{sk}} := \begin{bmatrix} \mathbf{I}_k \\ \mathbf{s}_k^\top \end{bmatrix} \cdot \begin{bmatrix} \mathbf{I}_{k-1} \\ \mathbf{s}_{k-1}^\top \end{bmatrix} \dots \begin{bmatrix} \mathbf{I}_1 \\ \mathbf{s}_1^\top \end{bmatrix}$. Note that the parties can perform additions thanks to the linearity of MLWE encryption and extra multiplications using the cGSW encryption and vector external product after their vector gadget product, thereby allowing the evaluation of multivariate polynomials.

RGSW Extension. The cLEV and cGSW encryptions enable multi-party computation of polynomials, however, their functionality is still limited as they do not support multi-hop circuit evaluation or full homomorphism, *i.e.,* an unlimited number of homomorphic multiplications and additions. We achieve this by extending a cGSW ciphertext into a multi-key RGSW ciphertext.

The RGSW encryption of a plaintext $\mu \in R_Q$ under a secret key $\mathsf{sk} = (\overline{s}_0 = 1, \overline{s}_1, \dots, \overline{s}_k)$ is a stack of MLWE encryptions of each entry of $\mu \cdot \vec{g} \otimes \mathsf{sk}$. Note that each entry of sk is essentially a multivariate polynomial of $s_{i,j}$'s, where $s_{i,j}$ denotes the j-th entry of the secret $\mathbf{s}_i$ of the i-th party, for $1 \le i \le k$ and $1 \le j \le i$. Our solution is to compute this multivariate polynomial based on the cLEV and cGSW encryptions, the vector gadget product, and the vector external product, given the cGSW encryption of μ and the cLEV encryption of $s_{i,j}$. Naïvely, each multivariate polynomial can be computed independently, however,

this naïve approach may require unnecessarily many multiplicative operations, leading to poor performance. Instead, we propose a circuit-like approach that can inductively generate each row of RGSW encryptions.

The core intuition of the circuit approach is based on the recurrence relation of the joint keys. Let $\overline{\mathsf{sk}}_i := \prod_{j=0}^{i-1} \begin{bmatrix} \mathbf{I}_{i-j} \\ \mathbf{s}_{i-j}^\top \end{bmatrix}$ be the joint key after multiplication by i-th party. Then, we obtain the relation $\overline{\mathsf{sk}}_{i+1} = \begin{bmatrix} \overline{\mathsf{sk}}_i \\ \mathbf{s}_{i+1}^\top \cdot \overline{\mathsf{sk}}_i \end{bmatrix}$. Using this relation, we can generate the encryptions of $\overline{\mathsf{sk}}_{i+1}$ by computing $\mathbf{s}_{i+1}^\top \cdot \overline{\mathsf{sk}}_i$ iteratively from the encryptions of each entry of $\mathbf{s}_{i+1}$.

To be specific, assume that μ belongs to the idx-th party. Our goal is to generate the MLWE encryptions of $g_j\mu$ and $g_j\mu \cdot \overline{s}_i$ under secret $\overline{\mathsf{sk}}$ for $1 \le i \le k$ and $0 \le j < d$. To allow this, we first compute all the encryptions of $g_j\overline{s}_i$ under the key $\mathsf{sk}_{\mathsf{idx}}$ for all $0 \le i \le \mathsf{idx}$ and $0 \le j < d$ iteratively. Specifically, given the MLWE encryptions of $g_j \cdot \overline{s}_{\ell-1}$ under sk_i and cGSW encryptions of $s_{i+1,\ell}$ for $0 \le j < d$ and $1 \le \ell \le i+1$, the vector gadget product allows us to obtain the encryption of $g_j\overline{s}_{\ell-1}s_{i+1,\ell}$ under the secret sk_{i+1}. Then, by summing them all up, we obtain the MLWE encryption of $g_j\overline{s}_{i+1}$ due to the linearity of MLWE ciphertexts. The mathematical induction implies that we can generate the encryptions of $g_j\overline{s}_i$ for $1 \le i \le \mathsf{idx}$ under the secret $\mathsf{sk}_{\mathsf{idx}}$. Then we compute the encryptions of $g_j\mu\overline{s}_i$ for $1 \le i \le \mathsf{idx}$ by using the cGSW encryptions of μ and the vector external product. Finally, given the encryptions of $g_j\mu\overline{s}_i$ for all $1 \le i \le \mathsf{idx}$ under secret $\mathsf{sk}_{\mathsf{idx}}$, we can easily compute the encryptions of $g_j\mu\overline{s}_i$ under secret sk_k by inductively performing a homomorphic evaluation of the equation $g_j\mu\overline{s}_{i+1} = \sum_{\ell=1}^{i+1} g_j\mu\overline{s}_{\ell-1} \cdot s_{i+1,\ell}$.

Multi-key Fully HE. Based on the RGSW extension algorithm, we formally describe a new multi-key RGSW scheme. Our scheme meets the multi-key syntax, including the key generation, encryption, evaluation, and decryption algorithms. Recall that the multiplication from cLEV and cGSW requires the parties' secrets to have different dimensions. Therefore, in a direct instantiation of our scheme, the key generation and encryption algorithms depend on the index of the ciphertext and, hence, are inconsistent among the parties. To resolve this issue, we set the maximum dimension K for ciphertexts and then switch the ciphertext size on-the-fly.

Let us describe our MKFHE scheme more precisely. The key generation algorithm sets the secret, public, and evaluation keys with respect to the maximum size K. The evaluation key for our multi-key scheme consists of cLEV encryptions of each entry of the secret keys. The encryption algorithm outputs the cGSW ciphertext with the dimension K, similar to the key generation algorithm. The evaluation algorithm can receive the ciphertexts from different parties, choose the evaluation order, and proceed with the evaluation in that order while converting the ciphertexts to the correct dimension. When there are multiple evaluators, their joint keys may differ as they can choose different multiplication orders. To allow computation between the multi-key RGSW ciphertexts under different

joint keys, we also describe how to generate key-switching keys. Briefly speaking, we first generate RGSW encryptions of each key component using the RGSW extension algorithm under the target key. Then, the recurrence relation for the original key can be homomorphically evaluated with the RGSW encryptions, resulting in the RGSW encryption of the original key, which is essentially the key-switching key from the original key to the target key. This result is implied by an additional circular security assumption. Specifically, we assume the circular security for the cGSW cryptosystem, which is equivalent to a combination of the shared-A assumption and the key-clique circular security assumption [45]. Finally, the distributed decryption can be instantiated with a generic MPC protocol.

We achieve one-round decryption relying on the transformation of Ananth et al. [4], which enables one-round decryption for MKHE schemes with any non-linear decryption formula.

Universal Key Generation. The RGSW is theoretically useful due to its structural design, which does not require any evaluation keys other than the ciphertexts themselves. However, its performance makes it difficult to use as a main cryptographic tool in real-world applications. Therefore, we also describe how to apply our construction of the multi-key RGSW scheme to the best-performing FHE schemes, including CKKS [19], BFV [9, 26], TFHE [20], Carousel [30] and LMKCDEY [38], thereby allowing us to compute with those schemes in the plain model.

The main challenge for MKFHE schemes lies in the evaluation key generation. Especially for the CKKS or BFV schemes, the generation of the relinearization and automorphism keys is the most crucial part. We generate the automorphism keys by computing the recurrence relations for the joint key homomorphically in a similar manner to the RGSW extension algorithm, but with cGSW encryptions of the secret key entries on which the automorphism operation is applied. On the other hand, note that the relinearization key can be seen as RGSW encryptions of the joint key entries; first we obtain the RGSW encryptions of the secret key entries from each party using the RGSW extension algorithm and then compute the recurrence relation with RGSW operations to obtain the RGSW encryptions of the joint key entries.

In contrast, the main challenge for the AP-like MKFHE schemes, namely TFHE [20], Carousel [30] and LMKCDEY [38], lies in the blind rotation key generation. Note that the blind rotation keys for these schemes are RGSW encryptions of some values that are dependent on each coefficient of the joint key. We show that one can generate these encryptions from the RGSW encryptions of some values that are dependent on each coefficient of the secret entries, which can be extended from the cGSW encryptions.

2 Preliminary

2.1 Notations

For a power-of-two N, $R := \mathbb{Z}[X]/(X^N + 1)$ be the $2N$-th cyclotomic ring of integers, and $R_Q := R/QR$. We do not specify N when N is consistent in the context. An element in R, or R_Q is represented with a polynomial with degree less than N, in other words, $a(X) = \sum_{i=0}^{N-1} a_i X^i$. $\delta_{a,b}$ denotes the delta function which outputs 1 if and only if $a = b$, and 0 otherwise. $\mathbf{I}_k$ denotes the identity matrix of the size $k \times k$. For a polynomial a (either in R or R_Q), $\|a\|_\infty$ denotes the infinity norm of the coefficient vector of a. $\mathbf{M}^\top$ denotes the transpose of a matrix $\mathbf{M}$.

2.2 Multi-key FHE

A multi-key fully homomorphic encryption (MKFHE) is an encryption scheme that can evaluate an arbitrary circuit over encrypted data. We follow the definition by Lopez-Alt et al. [41], where the algorithms are dependent on a parameter K, the number of distinct secret keys that the scheme can deal with.

Definition 1 (Multi-key Fully Homomorphic Encryption). *A multi-key fully homomorphic encryption (MKFHE) scheme is a family of probabilistic polynomial-time (PPT) algorithms* $\{\mathcal{E}^{(K)} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})\}_{K>0}$*, defined as follows.*

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^K)$: *On input a security parameter* λ *and the number of parties* K*, returns a public parameter* pp.
- $(\mathsf{sk}, \mathsf{ek}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$: *On input a public parameter* pp*, returns a secret key, (public) evaluation key, and public key pair* $(\mathsf{sk}, \mathsf{ek}, \mathsf{pk})$.
- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}; m)$: *On input a public key* pk *and a message* m*, returns a ciphertext* ct.
- $\mathsf{ct} \leftarrow \mathsf{Eval}(\mathcal{C}, \{(\mathsf{ek}_i, \mathsf{pk}_i, \mathsf{ct}_i)\}_{i\in[\ell]})$: *On input the circuit* $\mathcal{C}$ *and* ℓ *tuples* $(\mathsf{ek}_i, \mathsf{pk}_i, \mathsf{ct}_i)$ *where each consists of an (public) evaluation key* ek_i*, a public key* pk_i *and a ciphertext* ct_i*, returns a ciphertext* ct.
- $m \leftarrow \mathsf{Dec}(\{\mathsf{sk}_i\}_{i\in[K]}; \mathsf{ct})$: *On input secret keys* $(\mathsf{sk}_i)_{i\in[K]}$ *and a ciphertext* ct*, returns a message* m.

An MKFHE scheme needs to satisfy correctness and compactness. Formally, the following properties hold for all integers $K > 0$*, circuits* $\mathcal{C}$*, all sequences of* K *key tuples* $\{(\mathsf{ek}'_i, \mathsf{pk}'_i, \mathsf{sk}'_i)\}_{i\in[K]}$ *each of which is in the support of* $\mathsf{KeyGen}(\mathit{pp})$*, all sequences of* ℓ *key tuples* $\{(\mathsf{ek}_i, \mathsf{pk}_i, \mathsf{sk}_i)\}_{i\in[\ell]}$ *each of which is in* $\{(\mathsf{ek}'_i, \mathsf{pk}'_i, \mathsf{sk}'_i)\}_{i\in[K]}$*, and all messages* $(m_i)_{i\in[\ell]}$ *and ciphertexts* $(\mathsf{ct}_i)_{i\in[\ell]}$ *such that* ct_i *is in the support of* $\mathsf{Enc}(\mathsf{pk}_i; m_i)$.

- ***Correctness****: For* $\mathsf{ct} \leftarrow \mathsf{Eval}(\mathcal{C}, \{(\mathsf{ek}_i, \mathsf{pk}_i, \mathsf{ct}_i)\}_{i\in[\ell]})$, $\mathsf{Dec}(\{\mathsf{sk}_i\}_{i\in[K]}; \mathsf{ct}) = \mathcal{C}(m_1, \ldots, m_K)$.
- ***Compactness****: For* $\mathsf{ct} \leftarrow \mathsf{Eval}(\mathcal{C}, \{(\mathsf{ek}_i, \mathsf{pk}_i, \mathsf{ct}_i)\}_{i\in[\ell]})$*, there exists a bivariate polynomial* P *such that the size of* ct *is bounded by* $P(\lambda, K)$*, hence independent from the depth of the evaluated circuit* $\mathcal{C}$.

2.3 Ring Learning With Errors

We describe the ring variants of the decisional Learning With Errors (LWE) problem, namely the Ring-LWE (RLWE) and Module-LWE (MLWE) problems. The RLWE [53, 42] and MLWE [37] problems are defined over the ideal lattices and module lattices, respectively, especially over the ring of integers of a cyclotomic field, below.

Definition 2 (Decisional RLWE). *Let Q be a positive integer, N be a power-of-two, and ψ be an error distribution over $R := \mathbb{Z}[X]/(X^N + 1)$. The Ring Learning with Errors (RLWE) distribution with parameters N, Q, ψ is defined as follows: sample $a \xleftarrow{\$} R_Q$ and $e \leftarrow \psi$, and output (b, a) where $b = -as + e \pmod{Q}$ for the secret $s \in R$. For a key distribution χ over R, the decisional RLWE problem, denoted by $\mathsf{RLWE}(N, Q, \psi, \chi)$ calls for distinguishing the RLWE distribution for a secret $s \leftarrow \chi$ from the uniform distribution over R_Q^2.*

Definition 3 (Decisional MLWE). *Let k, Q be positive integers, N be a power-of-two, and ψ be an error distribution over $R := \mathbb{Z}[X]/(X^N + 1)$. The Module Learning With Errors (MLWE) distribution with parameters k, N, Q, ψ is defined as follows: sample $\mathbf{a} \xleftarrow{\$} R_Q^k$ and $e \leftarrow \psi$, and output $(b, \mathbf{a}) \in R_Q^{k+1}$ where $b = -\langle \mathbf{a}, \mathbf{s} \rangle + e \pmod{Q}$ for the secret $\mathbf{s} \in R^k$. For a key distribution χ over R, the decisional MLWE problem, denoted by $\mathsf{MLWE}(k, N, Q, \psi, \chi)$, calls for distinguishing the MLWE distribution for a secret $\mathbf{s} \leftarrow \chi^k$ from the uniform distribution over R_Q^{k+1}.*

We also remark that the RLWE problem is simply an extreme case of the MLWE problem, with $k = 1$.

In many works [51, 12, 28, 6, 7], a variant of RLWE called the shared-A RLWE, which considers multiple (R)LWE samples with shared (public) randomness, has been studied for better efficiency. We describe the shared-A RLWE as follows.

Definition 4 (Decisional shared-A RLWE). *Let k, Q be positive integers, N be a power-of-two, and ψ be an error distribution over $R := \mathbb{Z}[X]/(X^N + 1)$. The shared-A Ring Learning With Errors (shared-A RLWE) distribution with parameters k, N, Q, ψ is defined as follows: sample $a \xleftarrow{\$} R_Q$ and $e_i \leftarrow \psi$ for $1 \leq i \leq k$, and output $(b_1, \ldots, b_k, a) \in R_Q^{k+1}$ where $b_i = -a \cdot s_i + e_i \pmod{Q}$ for $1 \leq i \leq k$, for the secret $\mathbf{s} = (s_1, \ldots, s_k) \in R^k$. For a key distribution χ over R, the decisional shared-A RLWE problem, denoted by $\mathsf{sARLWE}(k, N, Q, \psi, \chi)$, calls for distinguishing the shared-A RLWE distribution for a secret $\mathbf{s} \leftarrow \chi^k$ from the uniform distribution over R_Q^{k+1}.*

Based on these hardness problems, we can define the MLWE and shared-A encryption schemes.[5] An MLWE encryption of a plaintext $\mu \in R_Q$ under the

[5] We exclude the RLWE encryption, as it is essentially an extreme case of the MLWE encryption.

secret $\mathbf{s} \in R^k$ is given by a tuple $(b, \mathbf{a}) \in R_Q^{k+1}$ such that $b + \langle \mathbf{s}, \mathbf{a} \rangle \approx \mu \pmod{Q}$. For convenience, we also say that the secret of this ciphertext is $\mathsf{sk} = \begin{bmatrix} 1 \\ \mathbf{s} \end{bmatrix} \in R^{k+1}$, so that $\mathsf{sk}^\top \cdot (b, \mathbf{a}) = b + \langle \mathbf{s}, \mathbf{a} \rangle \approx \mu \pmod{Q}$. On the other hand, a shared-A encryption of a plaintext vector $\boldsymbol{\mu} \in R_Q^k$ under the secret $\mathbf{s} \in R^k$ is given by a tuple $(b_1, \ldots, b_k, a) \in R_Q^{k+1}$ such that $b_i + a s_i \approx \mu_i \pmod{Q}$ for $1 \leq i \leq k$. We remark that these relations can be written in a single linear transformation, as follows.

$$\left[\mathbf{I}_k \mid \mathbf{s}\right] \cdot \begin{bmatrix} b_1 \\ \vdots \\ b_k \\ a \end{bmatrix} \approx \boldsymbol{\mu} \pmod{Q}.$$

2.4 Gadget Decomposition and GSW

The gadget decomposition technique is a widely used noise management technique in lattice-based homomorphic cryptography. Let $R := \mathbb{Z}[X]/(X^N + 1)$ for some power-of-two N. Roughly speaking, the gadget decomposition is an injective function $g^{-1} : R_Q \to R^d$ that takes an arbitrary ring element modulo Q as input and returns a small vector over R such that the input can be recovered with a dot product between the output vector and some vector $\vec{g} \in R^d$. Here, the vector $\vec{g} \in \mathbb{Z}^d$ is called the *gadget vector*, and parameter $d > 0$ is called the *gadget dimension*. Let us formally describe the gadget decomposition below.

Definition 5 (Gadget Decomposition). *Let Q and d be positive integers. Then, an injective function $g^{-1} : R_Q \to R^d$ is called the gadget decomposition with respect to the gadget vector $\vec{g} \in \mathbb{Z}^d$ if it satisfies the following two conditions for all $a \in R_Q$ and some $B > 0$.*

- $\|g^{-1}(a)\|_\infty \leq B$.
- $\langle g^{-1}(a), \vec{g} \rangle = a \pmod{Q}$.

Typically, we choose the gadget decomposition g^{-1} and the gadget vector $\vec{g}$ so that the bound $B \approx Q^{1/d}$. In the rest of the draft, we abuse the notation of the gadget decomposition g^{-1} for the point-wise decomposition of matrices as well.

Building upon the gadget decomposition, we also describe the ring variant of the Gentry-Sahai-Waters (GSW) cryptosystem, referred to as *RGSW*, built on top of the gadget encryption. We fix the gadget decomposition $g^{-1} : R_Q \to R^d$, the corresponding gadget vector $\vec{g} \in \mathbb{Z}^d$, and the key and noise distributions χ and ψ. Let us first describe the encryption and decryption algorithms.

• $\underline{\mathsf{RGSW.Enc}(\mathbf{s}; \mu)}$: Given the secret key $\mathbf{s} \in R^k$ and the plaintext μ, sample $\mathbf{A} \xleftarrow{\$} R_Q^{k \times d(k+1)}$ and $\mathbf{e} \leftarrow \psi^{d(k+1)}$. Output

$$\overline{\mathbf{C}} = \begin{bmatrix} -\mathbf{s}^\top \cdot \mathbf{A} + \mathbf{e}^\top \\ \mathbf{A} \end{bmatrix} + \mu \cdot \mathbf{I}_{k+1} \otimes \vec{g}^\top \in R_Q^{(k+1) \times d(k+1)}.$$

• $\mathsf{RGSW.Dec}(\mathbf{s}; \overline{\mathbf{C}})$: Given the secret $\mathbf{s} \in R^k$ and the RGSW encryption $\overline{\mathbf{C}} \in R_Q^{(k+1)\times d(k+1)}$, find and output μ that minimizes

$$\mathbf{e} := \left[1 \mid \mathbf{s}^\top\right] \cdot \overline{\mathbf{C}} - \mu \cdot \left[1 \mid \mathbf{s}^\top\right] \otimes \vec{g}^\top \pmod{Q}.$$

The RGSW cryptosystem offers two multiplicative operations in addition to the additive and subtractive operations that are naturally supported by the linearity of MLWE, called the *internal product* [27] and *external product* [20]. Roughly speaking, the internal product takes two RGSW ciphertexts and outputs an RGSW ciphertext encrypting the multiplication of the messages of the input ciphertexts. On the other hand, the external product takes MLWE and RGSW ciphertexts for each and outputs an MLWE ciphertext encrypting the multiplication of the messages of the input ciphertexts. We describe them formally below.

Definition 6 (Internal Product). *Given RGSW ciphertexts $\overline{\mathbf{C}}_1, \overline{\mathbf{C}}_2 \in R_Q^{(k+1)\times d(k+1)}$, the internal product $\boxtimes : R_Q^{(k+1)\times d(k+1)} \times R_Q^{(k+1)\times d(k+1)} \to R_Q^{(k+1)\times d(k+1)}$ is defined as follows:*

$$\overline{\mathbf{C}}_1 \boxtimes \overline{\mathbf{C}}_2 = \overline{\mathbf{C}}_2 \cdot g^{-1}(\overline{\mathbf{C}}_1) \pmod{Q}.$$

Definition 7 (External Product). *Given an MLWE ciphertext $\mathbf{c} \in R_Q^{k+1}$ and an RGSW ciphertext $\overline{\mathbf{C}} \in R_Q^{(k+1)\times d(k+1)}$, the external product $\boxdot : R_Q^{k+1} \times R_Q^{(k+1)\times d(k+1)} \to R_Q^{k+1}$ is defined as follows:*

$$\mathbf{c} \boxdot \overline{\mathbf{C}} = \overline{\mathbf{C}} \cdot g^{-1}(\mathbf{c}) \pmod{Q}.$$

We roughly show the correctness of the internal and external products. Let $\mathbf{s} \in R^k$ be the secret key, and m, μ_1, μ_2 be the underlying plaintexts of $\mathbf{c}, \overline{\mathbf{C}}_1, \overline{\mathbf{C}}_2$, respectively. Then,

$$\begin{aligned}\left[1 \mid \mathbf{s}^\top\right] \cdot (\overline{\mathbf{C}}_1 \boxtimes \overline{\mathbf{C}}_2) &\approx \mu_2 \cdot \left[1 \mid \mathbf{s}^\top\right] \otimes \vec{g}^\top \cdot g^{-1}(\overline{\mathbf{C}}_1) = \mu_2 \cdot \left[1 \mid \mathbf{s}^\top\right] \cdot \overline{\mathbf{C}}_1 \\ &\approx \mu_1\mu_2 \cdot \left[1 \mid \mathbf{s}^\top\right] \otimes \vec{g}^\top \pmod{Q}.\end{aligned}$$

Therefore, $\overline{\mathbf{C}}_1 \boxtimes \overline{\mathbf{C}}_2$ is a valid RGSW encryption of the message $\mu_1\mu_2$. On the other hand,

$$\begin{aligned}\left[1 \mid \mathbf{s}^\top\right] \cdot \mathbf{c} \boxdot \overline{\mathbf{C}}_1 &\approx \mu_1 \left[1 \mid \mathbf{s}^\top\right] \otimes \vec{g}^\top \cdot g^{-1}(\mathbf{c}) \\ &= \mu_1 \cdot \left[1 \mid \mathbf{s}^\top\right] \cdot \mathbf{c} \approx \mu_1 m \pmod{Q}.\end{aligned}$$

Hence, $\mathbf{c} \boxdot \overline{\mathbf{C}}_1$ is an MLWE encryption of the message $\mu_1 m$.

3 Multi-key RGSW without CRS

In this section, we introduce new multi-key cryptosystems called the compact leveled encryption (cLEV) and the compact GSW (cGSW), which allow the parties to evaluate a multivariate polynomial with a-priori degree without CRS or

circular security assumption. Building upon the cLEV and cGSW cryptosystems, we construct a new multi-key variant of the RGSW cryptosystem, which allows an unlimited number of homomorphic multiplications without CRS, under a new circular security assumption.

3.1 Multi-Key SHE from cLEV and cGSW

cLEV. We begin by formally describing the encryption algorithm of the compact leveled encryption (cLEV) and the vector gadget product, which is a multiplicative operation between a vector of ring elements and a cLEV encryption. Below, ψ denotes the error distribution defined over R. $\vec{g}$ and $g^{-1} : R_Q \to R^d$ are the gadget vector and the corresponding decomposition function. Finally, k denotes the dimension of cLEV encryption.

• $\underline{\mathsf{cLEV.Enc}(\mathbf{s}; \mu)}$: Given the message $\mu \in R_Q$ and secret key $\mathbf{s} \in R^k$, sample $\mathbf{a} \xleftarrow{\$} R_Q^{kd}$, and $\mathbf{E} \leftarrow \psi^{k \times kd}$. Output

$$\begin{bmatrix} -\mathbf{s} \cdot \mathbf{a}^\top + \mu \cdot \mathbf{I}_k \otimes \vec{g}^\top + \mathbf{E} \\ \mathbf{a}^\top \end{bmatrix} \in R_Q^{(k+1) \times kd}.$$

Definition 8 (Vector Gadget Product). *Given a vector $\mathbf{v} \in R_Q^k$ and a cLEV ciphertext $\mathbf{C} \in R_Q^{(k+1) \times dk}$ of dimension k, the binary operation $\odot_v : R_Q^k \times R_Q^{(k+1) \times dk}$ is defined as $\mathbf{v} \odot_v \mathbf{C} = \mathbf{C} \cdot g^{-1}(\mathbf{v}) \pmod{Q}$.*

The vector gadget product between a vector $\mathbf{v} \in R_Q^k$ and a cLEV encryption $\mathbf{C}$ of message $\mu \in R$ outputs a shared-A encryption of $\mu \cdot \mathbf{v} \in R_Q^k$. Observe that $\left[\mathbf{I}_k \mid \mathbf{s}\right] \cdot \mathbf{C} \approx \mu \cdot \mathbf{I}_k \otimes \vec{g}^\top \bmod Q$ holds for the cLEV encryption $\mathbf{C} \in R_Q^{(k+1) \times kd}$ of $\mu \in R$ under $\mathbf{s} \in R^k$. This implies that, given $\mathbf{C} \in R_Q^{(k+1) \times dk}$, a cLEV encryption of μ under the secret $\mathbf{s} = (s_1, \ldots, s_k)$, the following holds for any $\mathbf{v} \in R_Q^k$.

$$\begin{aligned} \left[\mathbf{I}_k \mid \mathbf{s}\right] \cdot (\mathbf{v} \odot_v \mathbf{C}) &= \left[\mathbf{I}_k \mid \mathbf{s}\right] \cdot \mathbf{C} \cdot g^{-1}(\mathbf{v}) \\ &\approx \mu \cdot \mathbf{I}_k \otimes \vec{g}^\top \cdot g^{-1}(\mathbf{v}) = \mu \cdot \mathbf{v} \pmod{Q}. \end{aligned} \tag{1}$$

For convenience, let us write a cLEV encryption of a plaintext μ under the secret $\mathbf{s}$ as $\mathsf{cLEV}_{\mathbf{s}}(\mu)$. Note that the dimension is implicit from the dimension of the secret.

cGSW. Additionally, we introduce the compact GSW scheme and the vector external product, which can be seen as the analogues of the GSW and external product for shared-A ciphertexts.

• $\underline{\mathsf{cGSW.Enc}(\mathbf{s}; \mu)}$: Given the message $\mu \in R_Q$ and secret key $\mathbf{s} \in R^k$, sample $\mathbf{a} \xleftarrow{\$} R_Q^{d(k+1)}$, and $\mathbf{E} \leftarrow \psi^{k \times d(k+1)}$. Output

$$\begin{bmatrix} -\mathbf{s} \cdot \mathbf{a}^\top + \mathbf{E} \\ \mathbf{a}^\top \end{bmatrix} + \mu \cdot \mathbf{I}_{k+1} \otimes \vec{g}^\top \in R_Q^{(k+1) \times d(k+1)}.$$

Definition 9 (Vector External Product). *Given a shared-A ciphertext* $\mathbf{c} \in R_Q^{k+1}$ *and a cGSW ciphertext* $\overline{\mathbf{C}} \in R_Q^{(k+1)\times d(k+1)}$ *of dimension* k*, the binary operation* $\boxdot_v : R_Q^{k+1} \times R_Q^{(k+1)\times d(k+1)}$ *is defined as* $\mathbf{c} \boxdot_v \overline{\mathbf{C}} = \overline{\mathbf{C}} \cdot g^{-1}(\mathbf{c}) \pmod{Q}$.

If the input ciphertext $\mathbf{c}$ is a shared-A encryption of a vector $\mathbf{v}$ under the secret $\mathbf{s}$, the vector external product between $\mathbf{c}$ and a cGSW encryption $\overline{\mathbf{C}}$ of message $\mu \in R$ outputs a shared-A encryption of $\mu \cdot \mathbf{v} \in R_Q^k$. Note that $\left[\mathbf{I}_k \mid \mathbf{s}\right] \cdot \overline{\mathbf{C}} \approx \mu \cdot \left[\mathbf{I}_k \mid \mathbf{s}\right] \otimes \vec{g}^\top \bmod Q$ by definition, the following holds:

$$\begin{aligned} &\left[\mathbf{I}_k \mid \mathbf{s}\right] \cdot (\mathbf{c} \boxdot_v \overline{\mathbf{C}}) = \left[\mathbf{I}_k \mid \mathbf{s}\right] \cdot \overline{\mathbf{C}} \cdot g^{-1}(\mathbf{c}) \\ &\approx \mu \cdot \left[\mathbf{I}_k \mid \mathbf{s}\right] \otimes \vec{g}^\top \cdot g^{-1}(\mathbf{c}) = \mu \cdot \left[\mathbf{I}_k \mid \mathbf{s}\right] \cdot \mathbf{c} \approx \mu \cdot \mathbf{v} \pmod{Q}. \end{aligned} \tag{2}$$

It is worth noting that the vector gadget product can be seen as a special instance of the vector external product, where the input ciphertext is a trivial encryption without masking. To be precise, it is equivalent to the case where the ciphertext $\mathbf{c} = \left[\mathbf{v} \mid 0\right] \in R_Q^{k+1}$ for some vector $\mathbf{v} \in R_Q^k$. Then the rightmost d columns of cGSW are multiplied by zeros in the vector external product, and thus we can drop the rightmost d columns to obtain a cLEV encryption.

Similar to the cLEV encryption, we denote a cGSW encryption of μ under secret $\mathbf{s}$ as $\mathsf{cGSW}_{\mathbf{s}}(\mu)$ for convenience.

Homomorphic Operation. Now we discuss how to perform a homomorphic multiplication between cLEV and cGSW ciphertexts encrypted under different keys, leveraging the vector gadget product and vector external product. Specifically, with k parties participating in the computation, where each i-th party discloses a cLEV encryption of plaintext $\mu_{i,0} \in R$ and cGSW encryptions of plaintexts $\mu_{i,j} \in R$ for some $1 \le j \le n_i$ with dimension i under the secret $\mathbf{s}_i \in R^i$. We describe the algorithm in Algorithm 1.

Algorithm 1 Multi-key Multiplication with cLEV and cGSW

Input: $\mathsf{cLEV}_{\mathbf{s}_i}(\mu_{i,0})$ and $\mathsf{cGSW}_{\mathbf{s}_i}(\mu_{i,j})$ for $1 \le i \le k$ and $1 \le j \le n_i$
Output: $\mathbf{c}_k$

```
c_0 ← m
for 1 ≤ i ≤ k do
    c_i ← c_{i−1} ⊙_v cLEV_{s_i}(μ_{i,0})
    for 1 ≤ j ≤ n_i do
        c_i ← c_i ⊡_v cGSW_{s_i}(μ_{i,j})
    end for
end for
```

The output of Algorithm 1 is a (noisy) encryption of $m \cdot \prod_{i=1}^{k} \left(\prod_{j=0}^{n_i} \mu_{i,j}\right)$ for any $m \in R_Q$ under the secret $\begin{bmatrix}\mathbf{I}_k \\ \mathbf{s}_k^\top\end{bmatrix} \cdot \begin{bmatrix}\mathbf{I}_{k-1} \\ \mathbf{s}_{k-1}^\top\end{bmatrix} \cdots \begin{bmatrix}\mathbf{I}_1 \\ \mathbf{s}_1^\top\end{bmatrix}$. This can be easily shown

by applying Eq. (1) and Eq. (2) multiple times. Note that

$$\mathbf{c}_i \leftarrow ((\mathbf{c}_{i-1} \odot_v \mathsf{cLEV}(\mu_{i,0})) \boxdot_v \mathsf{cGSW}_{\mathbf{s}_i}(\mu_{i,1}) \dots) \boxdot_v \mathsf{cGSW}_{\mathbf{s}_i}(\mu_{i,n_i}).$$

Therefore,

$$\begin{aligned}\big[\mathbf{I}_i \mid \mathbf{s}_i\big] \cdot \mathbf{c}_i &\approx \Bigg(\prod_{1\le j\le n_i} \mu_{i,j}\Bigg) \cdot \big[\mathbf{I}_i \mid \mathbf{s}_i\big] \cdot (\mathbf{c}_{i-1} \odot_v \mathsf{cLEV}(\mu_{i,0})) \\ &\approx \prod_{0\le j\le n_i} \mu_{i,j} \cdot \mathbf{c}_{i-1},\end{aligned}$$

and it implies that

$$\begin{aligned}&\big[\mathbf{I}_1 \mid \mathbf{s}_1\big] \dots \big[\mathbf{I}_k \mid \mathbf{s}_k\big] \cdot \mathbf{c}_k \\ &\approx \big[\mathbf{I}_1 \mid \mathbf{s}_1\big] \dots \big[\mathbf{I}_{k-1} \mid \mathbf{s}_{k-1}\big] \cdot \Big(\prod_{0\le j\le n_k} \mu_{k,j}\Big) \cdot \mathbf{c}_{k-1} \\ &\approx \dots \approx \Bigg(\prod_{0\le j\le n_k} \mu_{k,j}\Bigg) \dots \Bigg(\prod_{0\le j\le n_1} \mu_{1,j}\Bigg) \cdot m \pmod{Q}, \end{aligned} \tag{3}$$

thereby proving the claim.

We note that the generation of cLEV and cGSW ciphertexts does not require any additional assumptions, such as CRS or circular security, thus naturally enabling multi-key multiplication in the plain model. Observe that the resulting ciphertext of our multi-key multiplication algorithm has a different joint key structure compared to the existing solutions. Previously, the secret of a multi-key ciphertext was given by $(1, s_1, \dots, s_k) \in R^{k+1}$ where each s_i belongs to the i-th party. In contrast, the joint secret $\begin{bmatrix}\mathbf{I}_k \\ \mathbf{s}_k^\top\end{bmatrix} \cdot \begin{bmatrix}\mathbf{I}_{k-1} \\ \mathbf{s}_{k-1}^\top\end{bmatrix} \dots \begin{bmatrix}\mathbf{I}_1 \\ \mathbf{s}_1^\top\end{bmatrix}$ is a multiplication of each party's secret matrix with the vector gadget product.

We also remark that this multi-key multiplication allows us to evaluate arbitrary multivariate polynomials in a multi-key manner, since MLWE inherently supports additive operations. Concretely, given a multivariate polynomial, we can simply compute each term using the vector gadget product and then compute a linear sum of them, with cLEV encryptions of each party's message as input. While this gives us functionality for computing arbitrary multivariate polynomials, this approach does not offer multi-hop evaluation, thereby limiting its applicability to a certain family of circuits. We describe how to achieve multi-hop circuit evaluation by constructing a multi-key RGSW from the vector gadget product and vector external product.

Security. The security of cLEV and cGSW encryption schemes relies on the shared-A RLWE problem, with parameters k, N, Q, ψ and χ where $\mathbf{s} \leftarrow \chi^k$ for the secret key distribution χ.

3.2 Multi-key RGSW

We construct a new multi-key variant of the RGSW scheme [27] on top of the cLEV and cGSW cryptosystems. Let $\overline{\mathsf{sk}} = (\overline{s}_0, \overline{s}_1, \ldots, \overline{s}_k)$ be the joint secret for k parties, obtained from the gadget vector products. Recall that the RGSW encryption of a plaintext $\mu \in R$ under this joint secret is composed of the MLWE encryptions of $g_j \mu \overline{s}_i$ for $0 \leq i \leq k$ and $0 \leq j < d$, and $\overline{s}_i$ is a multivariate polynomial of each party's secret key components.

Therefore, we can generate each row of the RGSW encryption using multi-key multiplication and ciphertext addition. To allow this, we require the cLEV encryptions of each party's secret key components and the cGSW encryption of μ. Let us first investigate the joint key structure of our multi-key scheme. Let $\mathbf{s}_i = (s_{i,1}, \ldots, s_{i,i}) \in R^i$ be the secret under which the cLEV and cGSW ciphertexts of the i-th party are encrypted, and let $\overline{\mathsf{sk}}_i \in R^{i+1}$ be the joint secret after multiplication by the cLEV ciphertext of the i-th party. Then, Eq. (3) implies that

$$\overline{\mathsf{sk}}_i = \begin{bmatrix} \mathbf{I}_i \\ \mathbf{s}_i^\top \end{bmatrix} \cdot \overline{\mathsf{sk}}_{i-1} = \begin{bmatrix} \overline{\mathsf{sk}}_{i-1} \\ \mathbf{s}_i^\top \cdot \overline{\mathsf{sk}}_{i-1} \end{bmatrix} \tag{4}$$

for all $i > 1$, where $\overline{\mathsf{sk}}_1 = \begin{bmatrix} \mathbf{I}_1 \\ \mathbf{s}_1^\top \end{bmatrix} = (1, s_{1,1})$. Therefore, for $\overline{\mathsf{sk}} = \overline{\mathsf{sk}}_k = (\overline{s}_0 = 1, \overline{s}_1, \ldots, \overline{s}_k)$, we have the recurrence relation

$$\overline{s}_i = \sum_{j=1}^{i} \overline{s}_{j-1} s_{i,j}. \tag{5}$$

It is easy to see that the number of terms of a multivariate polynomial $\overline{s}_i$ is the sum of the number of terms of each $\overline{s}_0, \ldots \overline{s}_{i-1}$. This is essentially a geometric sequence with a common ratio of two, making the number of terms of $\overline{s}_i$ grow exponentially, 2^{i-1} terms to be precise. Therefore, a naïve evaluation of $g_j \mu \overline{s}_i$ requires 2^{i-1} multi-key multiplications, making the RGSW extension inefficient with a time complexity exponential to the number of parties k. Instead, we leverage the recurrence relation to reduce the time complexity for the RGSW extension from $O(2^k)$ to $O(k^2)$.

The main idea is to generate the encryption of $g_j \mu \overline{s}_i$ from the encryptions of $g_j \mu \overline{s}_0, \ldots g_j \mu \overline{s}_{i-1}$. Since $g_j \mu \overline{s}_i = \sum_{\ell=1}^{i} g_j \mu \overline{s}_{\ell-1} \cdot s_{i,\ell}$, we can iteratively generate the encryptions of $g_j \mu \overline{s}_\ell$ and then multiply $s_{i,\ell}$. We provide the full algorithm in Algorithm 2 that generates an RGSW encryption of μ, which is secret data from the idx-th party under the joint key $\overline{\mathsf{sk}}$.

Correctness. We show the correctness of Algorithm 2.

Lemma 1. *Given cLEV encryptions* $\mathsf{cLEV}_{\mathbf{s}_i}(s_{i,j})$ *of* $s_{i,j}$ *for* $1 \leq i \leq k$ *and* $1 \leq j \leq i$*, and a cGSW encryption* $\mathsf{cGSW}_{\mathbf{s}_{\mathsf{idx}}}(\mu)$ *of the* idx*-th party, Algorithm 2 outputs an RGSW encryption of* μ *under the joint secret key* $\overline{\mathsf{sk}} = (\overline{s}_i)_{1 \leq i \leq k}$*, obtained from the recurrence relation from Eq.* (5)*.*

Algorithm 2 RGSW extension

Input: $\mathsf{cLEV}_{\mathbf{s}_i}(s_{i,j})$ for $1 \le i \le k$ and $1 \le j \le i$, and $\mathsf{cGSW}_{\mathbf{s}_{\mathsf{idx}}}(\mu)$.

for $0 \le i < d$ **do**
  $\mathsf{acc}_i^0 \leftarrow (g_i)$
end for
for $1 \le i \le k$, $0 \le j < d$ **do**
  for $0 \le \ell < i$ **do**
    $\mathsf{acc}_{d\ell+j}^i \leftarrow \begin{bmatrix} \mathsf{acc}_{d\ell+j}^{i-1} \\ 0 \end{bmatrix}$
  end for
  $\mathsf{acc}_{di+j}^i \leftarrow \sum_{\ell=1}^{i} \mathsf{acc}_{d(\ell-1)+j}^{i-1} \odot_v \mathsf{cLEV}_{\mathbf{s}_i}(s_{i,\ell})$
  if $i = \mathsf{idx}$ **then**
    for $0 \le \ell \le i$ **do**
      $\mathsf{acc}_{d\ell+j}^i \leftarrow \mathsf{acc}_{d\ell+j}^i \boxdot_v \mathsf{cGSW}_{\mathbf{s}_i}(\mu)$
    end for
  end if
end for

Output: $\overline{\mathbf{C}} = \left[\mathsf{acc}_0^k \mid \cdots \mid \mathsf{acc}_{d(k+1)-1}^k\right]$

Proof. Now we show the correctness of our RGSW extension algorithm from Algorithm 2, using mathematical induction. In the following, we let $\mathsf{sk}_0 = (1)$ for convenience. Firstly, we claim that $\mathsf{acc}_{d\ell+j}^i$ is an encryption of $\overline{s}_\ell g_j$ if $i < \mathsf{idx}$ and an encryption of $\mu\overline{s}_\ell g_j$ if $i \ge \mathsf{idx}$, under the secret $\overline{\mathsf{sk}}_i$ for any $0 \le j < d$ and $0 \le \ell \le i$. Let us first prove for the cases where $i < \mathsf{idx}$. This is trivially true for $i = 0$, since $\overline{s}_0 = 1$ and $\mathsf{acc}_j^0 = (g_j)$ (which trivially decrypts to g_j) for $0 \le j < d$ by the definition. Now, assume that the claim holds for some $i = \iota$ where $0 \le \iota < \mathsf{idx} - 1$, *i.e.,* $\mathsf{acc}_{d\ell+j}^\iota$ is an MLWE encryption of $\overline{s}_\ell g_j$ under the secret $\overline{\mathsf{sk}}_\iota$ for all $0 \le \ell \le \iota$. Then, in the $\iota + 1$, j-th iterations of the outer loop, we do the following two things:

1. $\mathsf{acc}_{d\ell+j}^{\iota+1} \leftarrow (\mathsf{acc}_{d\ell+j}^\iota \mid 0)$ for $0 \le \ell \le \iota$
2. $\mathsf{acc}_{d(\iota+1)+j}^{\iota+1} \leftarrow \sum_{\ell=1}^{\iota+1} \mathsf{acc}_{d(\ell-1)+j}^\iota \odot_v \mathsf{cLEV}_{\mathbf{s}_{\iota+1}}(s_{\iota+1,\ell})$

Thanks to the recurrence relation on the secret keys $\overline{\mathsf{sk}}_\iota$ and $\overline{\mathsf{sk}}_{\iota+1}$ from Eq. (4), we have

$$\begin{aligned}
\overline{\mathsf{sk}}_{\iota+1}^\top \cdot \mathsf{acc}_{d\ell+j}^{\iota+1} &= \overline{\mathsf{sk}}_{\iota+1}^\top \cdot \begin{bmatrix} \mathsf{acc}_{d\ell+j}^\iota \\ 0 \end{bmatrix} \\
&= \left[\overline{\mathsf{sk}}_\iota^\top \mid \overline{\mathsf{sk}}_\iota^\top \cdot \mathbf{s}_{\iota+1}\right] \cdot \begin{bmatrix} \mathsf{acc}_{d\ell+j}^\iota \\ 0 \end{bmatrix} \\
&= \overline{\mathsf{sk}}_\iota^\top \cdot \mathsf{acc}_{d\ell+j}^\iota \approx \overline{s}_\ell g_j \pmod{Q} \qquad (6)
\end{aligned}$$

for any $0 \le j < d$ and $0 \le \ell \le \iota$. On the other hand, we also have

$$\overline{\mathsf{sk}}_{\iota+1}^\top \cdot \mathsf{acc}_{d(\iota+1)+j}^{\iota+1} = \sum_{\ell=1}^{\iota+1} \overline{\mathsf{sk}}_{\iota+1}^\top \cdot \mathsf{acc}_{d(\ell-1)+j}^\iota \odot_v \mathsf{cLEV}_{\mathbf{s}_{\iota+1}}(s_{\iota+1,\ell})$$

$$\approx \sum_{\ell=1}^{\iota+1} s_{\iota+1,\ell} \cdot \overline{\mathsf{sk}}_{\iota}^{\top} \cdot \mathsf{acc}^{\iota}_{d(\ell-1)+j}$$
$$\approx \sum_{\ell=1}^{\iota+1} s_{\iota+1,\ell} \cdot \overline{s}_{\ell-1} g_j = \overline{s}_{\iota+1} g_j \pmod{Q}. \tag{7}$$

Here, the second line follows from Eq. (4) and Eq. (1), and the final equality follows from Eq. (5). Now, Eq. (6) and Eq. (7) directly imply that the claim is true for $i = \iota + 1$ as well. Consequently, by mathematical induction, we can deduce that $\mathsf{acc}^{i}_{d\ell+j}$ is an encryption of $\overline{s}_\ell g_j$ for all $0 \le i < \mathsf{idx}$, $0 \le j < d$, and $0 \le \ell \le i$. Next, if $i = \mathsf{idx}$, we additionally compute the following after computing the relations above:

$$\mathsf{acc}^{\mathsf{idx}}_{d\ell+j} \leftarrow \mathsf{acc}^{\mathsf{idx}}_{d\ell+j} \boxdot_v \mathsf{cGSW}_{\mathbf{s}_{\mathsf{idx}}}(\mu) \quad \text{for} \quad 0 \le \ell \le \mathsf{idx}.$$

By Eq. (6), Eq. (7) and Eq. (2), it naturally follows that

$$\overline{\mathsf{sk}}_{\mathsf{idx}}^{\top} \cdot \mathsf{acc}^{\mathsf{idx}}_{d\ell+j} \approx \mu \overline{s}_\ell g_j$$

for any $0 \le \ell \le \mathsf{idx}$, at the end of the loop.

Finally, we show that the claim is true for $i > \mathsf{idx}$, using mathematical induction again. Suppose that the claim is true for some $i = \iota$ where $\mathsf{idx} \le \iota$. Then, in the $\iota + 1, j$-th iteration of the outer loop, we compute the following:

1. $\mathsf{acc}^{\iota+1}_{d\ell+j} \leftarrow (\mathsf{acc}^{\iota}_{d\ell+j} \mid 0)$ for $0 \le \ell \le \iota$,
2. $\mathsf{acc}^{\iota+1}_{d(\iota+1)+j} \leftarrow \sum_{\ell=1}^{\iota+1} \mathsf{acc}^{\iota}_{d(\ell-1)+j} \odot_v \mathsf{cLEV}_{\mathbf{s}_{\iota+1}}(s_{\iota+1,\ell})$

Then, similar to the previous discussions, we obtain the following:

$$\overline{\mathsf{sk}}_{\iota+1}^{\top} \cdot \mathsf{acc}^{\iota+1}_{d\ell+j} = \left[\overline{\mathsf{sk}}_{\iota}^{\top} \mid \overline{\mathsf{sk}}_{\iota}^{\top} \cdot \mathbf{s}_{\iota+1}\right] \cdot \begin{bmatrix} \mathsf{acc}^{\iota}_{d\ell+j} \\ 0 \end{bmatrix} = \overline{\mathsf{sk}}_{\iota}^{\top} \cdot \mathsf{acc}^{\iota}_{d\ell+j} \approx \mu \overline{s}_\ell g_j \pmod{Q} \tag{8}$$

for all $0 \le \ell \le \iota$, and

$$\overline{\mathsf{sk}}_{\iota+1}^{\top} \cdot \mathsf{acc}^{\iota+1}_{d(\iota+1)+j} \approx \sum_{\ell=1}^{\iota+1} \mu \overline{s}_{\ell-1} g_j \cdot s_{\iota+1,\ell} = \mu \overline{s}_{\iota+1} g_j \pmod{Q}. \tag{9}$$

From these two equations Eq. (8) and Eq. (9), it follows that the claim holds for $i = \iota + 1$ if it holds when $i = \iota$. We can finally deduce that the claim holds when $i = k$ as well, thanks to mathematical induction. Therefore, the output ciphertext is indeed an RGSW encryption of μ under $\overline{\mathsf{sk}}_k$.

3.3 Discussion

Full Homomorphism. The RGSW cryptosystem allows us to evaluate arbitrary circuits with bounded depth. However, as homomorphic operations incur

noise growth, the bootstrapping operation is required to refresh the ciphertext and consequently obtain full homomorphism. Roughly speaking, the bootstrapping takes a ciphertext and the bootstrapping key and homomorphically computes the decryption circuit. It is known that the bootstrapping for GSW only requires a polynomial overhead (consuming only one level) with the branching program-based approach [14, 3, 28, 25]. We remark that this approach can be adopted in our scheme, with each party disclosing encryptions of each bit of their secret key, since essentially the decryption circuit of our scheme can be written as a branching program. Therefore, as long as the ciphertext modulus can tolerate the decryption circuit (with some margin for homomorphic computation and plaintext), our scheme is fully homomorphic[6].

LWE-based Construction. We emphasize that our construction can be naturally adapted in the LWE setting as well, thereby achieving our result under a weaker security assumption. Let n be the LWE dimension. Then, unlike the MLWE-based approach, the LWE variant of cGSW encryption takes a short matrix $S \in \mathbb{Z}^{kn \times n}$ as a secret. Note that this can be seen as a juxtaposition of kn LWE keys. Now let us describe the LWE-based cGSW encryption. First, we use the same notation for the gadget vector $\vec{g} \in \mathbb{Z}^d$ and the corresponding gadget decomposition function $g^{-1} : \mathbb{Z}_Q \to \mathbb{Z}^d$ as in the ring setting. In the scalar setting, the gadget vector remains the same, and the gadget decomposition is applied to a single scalar rather than to every coefficient of a ring element. Then, an LWE-based cGSW encryption $\overline{C} \in \mathbb{Z}_Q^{(k+1)n \times (k+1)nd}$ of a plaintext $\mu \in \mathbb{Z}_Q$ under a secret $S \in \mathbb{Z}^{kn \times n}$ satisfies that

$$\left[I_{kn} \mid S\right] \cdot \overline{C} \approx \mu \cdot \left[I_{kn} \mid S\right] \otimes \vec{g}^{\top} \pmod{Q},$$

for the identity matrix $I_{kn} \in \mathbb{Z}^{kn \times kn}$, the vector external product is defined as

$$\vec{c} \boxdot_v \overline{C} = \overline{C} \cdot g^{-1}(\vec{c}) \pmod{Q}$$

for any LWE ciphertext $\vec{c} \in \mathbb{Z}_Q^{(k+1)n}$. We can also define the cLEV encryption by taking only the leftmost knd columns of the cGSW encryption, and the vector gadget product $\odot$ is also naturally defined accordingly.

This LWE-based construction can generate a GSW encryption from cGSW by homomorphically evaluating a recurrence relation obtained from the multiplication of keys, using an analogous algorithm to Algorithm 2. Like the ring-based approach, this would require cGSW encryption of μ.

Public Key Encryption. Previously, our multi-key homomorphic encryption schemes were portrayed as symmetric key encryption schemes. We note that they can, in fact, be easily converted into public key variants as well. As the

[6] It is worth noting that this family of bootstrapping is highly optimized in recent works [20, 38]. We provide a more detailed description of generating the bootstrapping keys for these recent schemes in Sec. 5.2.

cLEV ciphertext can be trivially obtained from the cGSW encryption, we only cover the cGSW encryption scheme. The public key for the cGSW encryption is given by a shared-A zero encryption, *i.e.,* $\mathsf{pk} \in R_Q^{k+1}$ where

$$\big[I_k \mid \mathbf{s}\big] \cdot \mathsf{pk} \approx 0 \pmod{Q}$$

for the secret $\mathbf{s} \in R^k$. Given a public key $\mathsf{pk} \in R_Q^{k+1}$, the public key encryption algorithm for cGSW is given as follows, for the key distribution χ and the noise distribution ψ over R.

• $\underline{\mathsf{cGSW.Enc}(\mathsf{pk}; \mu)}$: Given the message $\mu \in R_Q$ and a public key $\mathsf{pk} \in R^{k+1}$, sample $\mathbf{r} \leftarrow \chi^{d(k+1)}$, and $\mathbf{E} \leftarrow \psi^{(k+1)\times d(k+1)}$. Output

$$\mathsf{pk} \cdot \mathbf{r}^\top + \mathbf{E} + \mu \cdot \mathbf{I}_{k+1} \otimes \vec{g}^\top \in R_Q^{(k+1)\times d(k+1)}.$$

Observe this is indeed a cGSW encryption of μ, since

$$\big[I_k \mid \mathbf{s}\big] \cdot (\mathsf{pk} \cdot \mathbf{r}^\top + \mathbf{E} + \mu \cdot \mathbf{I}_{k+1} \otimes \vec{g}^\top) \approx \mu \cdot \big[I_k \mid \mathbf{s}\big] \otimes \vec{g}^\top. \pmod{Q}$$

We also note that the security of the public key variant of cGSW encryption relies on the vanilla RLWE assumption and shared-A RLWE assumption with secret distribution χ and the noise distribution ψ.

3.4 Noise Analysis

In this section, we provide a noise analysis for the vector gadget product and the RGSW extension algorithm. We first begin by specifying the key and noise distributions. In typical settings for FHE, the key distribution is chosen to be a ternary distribution with a fixed Hamming weight H, and the noise distribution is chosen to be some discrete Gaussian distribution with a small width σ. In the following analysis, we follow this tradition and assume that the parties sample their keys and noises from these distributions. The bound for the gadget decomposition g^{-1} is denoted by B_g, and the probabilistic bound for the noise distribution is denoted by B_σ. We use the same notation as the algorithm descriptions from the previous subsections for simplicity.

Lemma 2 (Vector Gadget Product). *Given a vector $\mathbf{v} \in R_Q^k$ and a (fresh) cLEV encryption $\mathbf{C} \in R_Q^{(k+1)\times dk}$ of μ under the secret $\mathbf{s}$ for the dimension k, let $\mathbf{c} \leftarrow \mathbf{v} \odot_v \mathbf{C}$. Then, for the noise vector $\mathbf{e} := \big[\mathbf{I}_k \mid \mathbf{s}\big] \cdot \mathbf{c} - \mu \cdot \mathbf{v} \pmod{Q}$, it holds that $\|\mathbf{e}\|_\infty \le kdB_\sigma B_g N$.*

Proof. By definition, $\mathbf{C} = \begin{bmatrix} -\mathbf{s} \cdot \mathbf{a}^\top + \mu \cdot \mathbf{I}_k \otimes \vec{g}^\top + \mathbf{E} \\ \mathbf{a}^\top \end{bmatrix}$ for the randomness $\mathbf{a} \in R_Q^{dk}$ and the noise $\mathbf{E} \in R^{k\times dk}$. Then, we have the following:

$$\mathbf{e} = \big[\mathbf{I}_k \mid \mathbf{s}\big] \cdot \mathbf{c} - \mu \cdot \mathbf{v} = \big[\mathbf{I}_k \mid \mathbf{s}\big] \cdot \mathbf{C} \cdot g^{-1}(\mathbf{v}) - \mu \cdot \mathbf{v}$$

$$
\begin{aligned}
&= \left[\mathbf{I}_k \mid \mathbf{s}\right] \cdot \begin{bmatrix} -\mathbf{s}\cdot\mathbf{a}^\top + \mu\cdot\mathbf{I}_k \otimes \vec{g}^\top + \mathbf{E} \\ \mathbf{a}^\top \end{bmatrix} \cdot g^{-1}(\mathbf{v}) - \mu\cdot\mathbf{v} \\
&= \left(\mu\cdot\mathbf{I}_k \otimes \vec{g}^\top + \mathbf{E}\right)\cdot g^{-1}(\mathbf{v}) - \mu\cdot\mathbf{v} \\
&= \mathbf{E}\cdot g^{-1}(\mathbf{v}) \pmod{Q}.
\end{aligned}
$$

Since $\|g^{-1}(\mathbf{v})\|_\infty \leq B_g$ and $\|\mathbf{E}\|_\infty \leq B_\sigma$, it directly follows that $\|\mathbf{e}\|_\infty \leq kdNB_\sigma B_g$.

Lemma 3 (Vector External Product). *Given a shared-A RLWE encryption $\mathbf{c} \in R_Q^{k+1}$ and a (fresh) cGSW encryption $\overline{\mathbf{C}} \in R_Q^{(k+1)\times d(k+1)}$ of μ under the secret $\mathbf{s}$ for the dimension k, let $\mathbf{c}' \leftarrow \mathbf{c} \boxdot_v \overline{\mathbf{C}}$. Then, for the noise vector $\mathbf{e} := \left[\mathbf{I}_k \mid \mathbf{s}\right]\cdot\mathbf{c}' - \mu\cdot\left[\mathbf{I}_k \mid \mathbf{s}\right]\cdot\mathbf{c} \pmod{Q}$, it holds that $\|\mathbf{e}\|_\infty \leq (k+1)dB_\sigma B_g N$.*

Proof. By definition, $\overline{\mathbf{C}} = \begin{bmatrix} -\mathbf{s}\cdot\mathbf{a}^\top + \mathbf{E} \\ \mathbf{a}^\top \end{bmatrix} + \mu\cdot\mathbf{I}_{k+1}\otimes\vec{g}^\top$ for the randomness $\mathbf{a} \in R_Q^{d(k+1)}$ and the noise $\mathbf{E} \in R^{k\times d(k+1)}$. Then, we have the following:

$$
\begin{aligned}
\mathbf{e} &= \left[\mathbf{I}_k \mid \mathbf{s}\right]\cdot\mathbf{c}' - \mu\cdot\left[\mathbf{I}_k \mid \mathbf{s}\right]\cdot\mathbf{c} = \left[\mathbf{I}_k \mid \mathbf{s}\right]\cdot\overline{\mathbf{C}}\cdot g^{-1}(\mathbf{c}) - \mu\cdot\left[\mathbf{I}_k \mid \mathbf{s}\right]\cdot\mathbf{c} \\
&= \left[\mathbf{I}_k \mid \mathbf{s}\right]\cdot\left(\begin{bmatrix} -\mathbf{s}\cdot\mathbf{a}^\top + \mathbf{E} \\ \mathbf{a}^\top \end{bmatrix} + \mu\cdot\mathbf{I}_{k+1}\otimes\vec{g}^\top\right)\cdot g^{-1}(\mathbf{c}) - \mu\cdot\left[\mathbf{I}_k \mid \mathbf{s}\right]\cdot\mathbf{c} \\
&= \left(\mu\cdot\left[\mathbf{I}_k \mid \mathbf{s}\right]\otimes\vec{g}^\top + \mathbf{E}\right)\cdot g^{-1}(\mathbf{c}) - \mu\cdot\left[\mathbf{I}_k \mid \mathbf{s}\right]\cdot\mathbf{c} \\
&= \mathbf{E}\cdot g^{-1}(\mathbf{c}) \pmod{Q}.
\end{aligned}
$$

Since $\|g^{-1}(\mathbf{c})\|_\infty \leq B_g$ and $\|\mathbf{E}\|_\infty \leq B_\sigma$, it directly follows that $\|\mathbf{e}\|_\infty \leq (k+1)dNB_\sigma B_g$.

Proposition 1. *For the joint secret key $\overline{\mathsf{sk}} = (\overline{s}_0 = 1, \overline{s}_1, \ldots, \overline{s}_k)$, it holds that $\|\overline{s}_i\|_\infty \leq (H+1)^{i-1}$ for $1 \leq i \leq k$.*

Proof. Due to the recurrence relation from Eq. (5), we have the following:

$$
\|\overline{s}_i\|_\infty = \|\sum_{j=0}^{i-1} s_{i,j}\overline{s}_j\|_\infty \leq \sum_{j=0}^{i-1}\|s_{i,j}\overline{s}_j\|_\infty \leq 1 + H\cdot\sum_{j=1}^{i-1}\|\overline{s}_j\|_\infty.
$$

With $\|\overline{s}_1\|_\infty = 1$, we can inductively obtain the upper norm bound $(H+1)^{i-1}$ for $\overline{s}_i$.

Lemma 4 (RGSW Extension). *Given (fresh) cLEV encryptions $\mathsf{cLEV}_{\mathbf{s}_i}(s_{i,j})$ for $1 \leq i \leq k$ and $1 \leq j \leq i$, and a (fresh) cGSW encryption $\mathsf{cGSW}_{\mathbf{s}_{\mathsf{idx}}}(\mu)$ of the idx-th party, let $\overline{\mathbf{C}}$ be the output of Algorithm 2. Then, for the noise vector $\mathbf{e} = (e_0, \ldots, e_{d(k+1)-1}) := \overline{\mathsf{sk}}^\top\cdot\overline{\mathbf{C}} - \mu\cdot\overline{\mathsf{sk}}^\top\otimes\vec{g}^\top \pmod{Q}$, the following inequality holds,*

$$
\|e_{di+j}\|_\infty \lessapprox \begin{cases} B_\sigma & \text{if} \quad i = 0 \\ dNB_\sigma B_g\left(\frac{1}{3}\|\mu\|_1 N i^3 (H+1)^{i-2} + (\mathsf{idx}+1)\right) & \text{if} \quad 1 \leq i \leq \mathsf{idx} \\ \frac{1}{3}dN^2 B_\sigma B_g (H+1)^{i-2}(i^3 + \|\mu\|_1 \mathsf{idx}^4) & \text{otherwise} \end{cases}
$$

for all $0 \le i \le k$ *and* $0 \le j < d$.

Proof. Similar to the correctness proof, we prove the claim using mathematical induction on the noise bound of $\mathsf{acc}^i_{d\ell+j}$'s. Let us define the error function $\mathsf{err}_{\overline{\mathsf{sk}}_i}(\cdot) : R_Q^{i+1} \to R$ for the MLWE encryptions $\mathsf{acc}^i_{d\ell+j}$ as follows:

$$\mathsf{err}_{\overline{\mathsf{sk}}_i}(\mathsf{acc}^i_{d\ell+j}) = \begin{cases} \overline{\mathsf{sk}}_i^\top \cdot \mathsf{acc}^i_{d\ell+j} - \overline{s}_\ell g_j & \text{if} \quad i < \mathsf{idx} \\ \overline{\mathsf{sk}}_i^\top \cdot \mathsf{acc}^i_{d\ell+j} - \mu\overline{s}_\ell g_j & \text{otherwise} \end{cases}$$

Let us begin by measuring the noise bound for $i < \mathsf{idx}$ cases. Note that $\mathsf{err}_{\overline{\mathsf{sk}}_i}(\mathsf{acc}^i_{d\ell+j}) = \mathsf{err}_{\overline{\mathsf{sk}}_\ell}(\mathsf{acc}^\ell_{d\ell+j})$, from the correctness proof of the algorithm. Therefore, we can deduce the following relation from the algorithm description and Lem. 2.

$$\begin{aligned} \mathsf{err}_{\overline{\mathsf{sk}}_i}(\mathsf{acc}^i_{di+j}) &= \overline{\mathsf{sk}}_i^\top \cdot \mathsf{acc}^i_{di+j} - \overline{s}_i g_j \\ &= \sum_{\ell=0}^{i-1} \left(\overline{\mathsf{sk}}_i^\top \cdot \left(\mathsf{acc}^{i-1}_{d\ell+j} \odot_v \mathsf{cLEV}_{\mathbf{s}_i}(s_{i,\ell+1}) \right) - \overline{s}_\ell s_{i,\ell+1} g_j \right) \\ &= \sum_{\ell=0}^{i-1} \left(\overline{\mathsf{sk}}_{i-1}^\top \cdot \left(s_{i,\ell+1} \cdot \mathsf{acc}^{i-1}_{d\ell+j} + \mathbf{f}_{i,\ell} \right) - \overline{s}_\ell s_{i,\ell+1} g_j \right) \\ &= \sum_{\ell=0}^{i-1} \left(s_{i,\ell+1} \cdot \mathsf{err}_{\overline{\mathsf{sk}}_{i-1}}(\mathsf{acc}^{i-1}_{d\ell+j}) + \overline{\mathsf{sk}}_{i-1}^\top \cdot \mathbf{f}_{i,\ell} \right), \end{aligned}$$

where $\mathbf{f}_{i,\ell}$ is the error vector from the vector gadget product $\mathsf{acc}^{i-1}_{d\ell+j} \odot_v \mathsf{cLEV}_{\mathbf{s}_i}(s_{i,\ell+1})$, *i.e.,*

$$\mathbf{f}_{i,\ell} = \left[\mathbf{I}_i \mid \mathbf{s}_i \right] \cdot \left(\mathsf{acc}^{i-1}_{d\ell+j} \odot_v \mathsf{cLEV}_{\mathbf{s}_i}(s_{i,\ell+1}) \right) - s_{i,\ell+1} \cdot \mathsf{acc}^{i-1}_{d\ell+j} \pmod{Q}.$$

Then, by Lem. 2 and Proposition 1, we have

$$\begin{aligned} &\|\mathsf{err}_{\overline{\mathsf{sk}}_i}(\mathsf{acc}^i_{di+j})\|_\infty \\ &\le \sum_{\ell=0}^{i-1} \| \left(s_{i,\ell+1} \cdot \mathsf{err}_{\overline{\mathsf{sk}}_{i-1}}(\mathsf{acc}^{i-1}_{d\ell+j}) + \overline{\mathsf{sk}}_{i-1}^\top \cdot \mathbf{f}_{i,\ell} \right) \|_\infty \\ &\le \sum_{\ell=0}^{i-1} \left(H \cdot \|\mathsf{err}_{\overline{\mathsf{sk}}_\ell}(\mathsf{acc}^\ell_{d\ell+j})\|_\infty + \frac{idN^2 B_\sigma B_g}{H} \left((H+1)^{i-1} + H - 1 \right) \right) \\ &= \frac{i^2 dN^2 B_\sigma B_g}{H} \left((H+1)^{i-1} + H - 1 \right) + H \sum_{\ell=0}^{i-1} \|\mathsf{err}_{\overline{\mathsf{sk}}_\ell}(\mathsf{acc}^\ell_{d\ell+j})\|_\infty \end{aligned}$$

By mathematical induction, we obtain the norm bound of $\mathsf{err}_{\overline{\mathsf{sk}}_i}(\mathsf{acc}^i_{di+j})$ for $1 \le i < \mathsf{idx}$, as follows.

$$\frac{dN^2 B_\sigma B_g}{H} \cdot \left(\frac{2H-2}{H} i + \frac{H^2+H-2}{H^2} + \right.$$

$$\left(\frac{H}{3}i^3+\frac{H+2}{2}i^2+\frac{H}{6}i+\frac{H^4+3H^3+H^2-3H-2}{H^2}\right)(H+1)^{i-2}\Big)$$

For simplicity, we use the approximate bound of $\approx \frac{1}{3}dN^2B_\sigma B_g i^3(H+1)^{i-2}$. Now, let us focus on the case where $i=\mathsf{idx}$. Firstly, observe that $\mathsf{acc}_j^{\mathsf{idx}}$ before the if loop (lines 9 to 13) is a trivial encryption of g_j, the j-th entry of the gadget vector. This implies that the gadget decomposition of this vector is a one-hot vector, making the vector external product 'choosing' a single row from the cGSW encryption. Consequently, we have $\|\mathsf{err}_{\overline{\mathsf{sk}}_{\mathsf{idx}}}(\mathsf{acc}_j^{\mathsf{idx}})\|_\infty \le B_\sigma$. Also, the noise bound above and Lem. 3 imply that

$$\|\mathsf{err}_{\overline{\mathsf{sk}}_{\mathsf{idx}}}(\mathsf{acc}_{d\mathsf{idx}+j}^{\mathsf{idx}})\|_\infty \lessapprox \frac{1}{3}\|\mu\|_1 dN^2B_\sigma B_g i^3(H+1)^{i-2}+(\mathsf{idx}+1)\cdot dNB_\sigma B_g$$

for $1\le i\le \mathsf{idx}$.

Finally, we cover the case where $i>\mathsf{idx}$. By the discussion from the correctness proof, it is easy to see that $\mathsf{err}_{\overline{\mathsf{sk}}_i}(\mathsf{acc}_{dk+j}^i)=\mathsf{err}_{\overline{\mathsf{sk}}_{\mathsf{idx}}}(\mathsf{acc}_{dk+j}^{\mathsf{idx}})$. Therefore, the algorithm description and the above noise bounds imply that

$$\begin{aligned}
&\|\mathsf{err}_{\overline{\mathsf{sk}}_i}(\mathsf{acc}_{di+j}^i)\|_\infty = \|\sum_{\ell=0}^{i-1}\left(s_{i,\ell+1}\mathsf{err}_{\overline{\mathsf{sk}}_{i-1}}(\mathsf{acc}_{d\ell+j}^{i-1})+\overline{\mathsf{sk}}_{i-1}^\top\cdot\mathbf{f}_{i,\ell}\right)\|_\infty\\
&\le \sum_{\ell=0}^{\mathsf{idx}} H\cdot\left(\frac{1}{3}\|\mu\|_1 dN^2B_\sigma B_g\ell^3(H+1)^{\ell-2}+(\mathsf{idx}+1)\cdot dNB_\sigma B_g\right)\\
&+\frac{i^2dN^2B_\sigma B_g}{H}\left((H+1)^{i-1}+H-1\right)+\sum_{\ell=\mathsf{idx}+1}^{i-1}\left(s_{i,\ell+1}\mathsf{err}_{\overline{\mathsf{sk}}_{i-1}}(\mathsf{acc}_{d\ell+j}^{i-1})\right)\\
&\lessapprox \frac{1}{3}\|\mu\|_1 dN^2B_\sigma B_g(\mathsf{idx}+1)\mathsf{idx}^3(H+1)^{\mathsf{idx}-1}+(\mathsf{idx}+1)^2dNB_\sigma B_g\\
&+\frac{idN^2B_\sigma B_g}{H}((H+1)^{i-1}+H-1)+H\sum_{\ell=\mathsf{idx}+1}^{i-1}\|\mathsf{err}_{\overline{\mathsf{sk}}_{\mathsf{idx}}}(\mathsf{acc}_{d\ell+j}^{\mathsf{idx}})\|_\infty.
\end{aligned}$$

By mathematical induction, we obtain the approximate noise bound

$$\frac{1}{3}dN^2B_\sigma B_g i^3(H+1)^{i-2}+\frac{1}{3}\|\mu\|_1 dN^2B_\sigma B_g\mathsf{idx}^4(H+1)^{i-2}.$$

for $\mathsf{idx}<i\le k$.

4 MKFHE in the Plain Model

In this section, we formalize our multi-key RGSW scheme and discuss how it can be used as a baseline solution for a two-round multi-party computation (MPC) protocol.

4.1 Building Blocks

In this section, we introduce some important building blocks required to realize the traditional syntax of the multi-key homomorphic encryption scheme with our multi-key RGSW scheme. Specifically, we devise a ciphertext shrinking method for the cGSW encryption, a key-switching key generation algorithm for our RGSW scheme, and finally formalize the evaluation algorithm with our multi-key RGSW scheme.

Ciphertext Shrinking. Note that our multi-key scheme from Sec. 3.2 requires each party to know the exact order in which its ciphertexts are multiplied beforehand since the sizes of the input cGSW ciphertexts differ with respect to the multiplication order. However, this contradicts the definition of MKFHE, as the key generation algorithm only takes the public parameters, not the index of the party. We can handle this issue by having each party output the cGSW encryptions of the maximum size and shrink the size of the cGSW encryption upon receiving them. Let us elaborate in detail. Recall that the cGSW encryption of $\mu \in R$ under the secret $\mathbf{s} = (s_1, \ldots, s_K) \in R^K$ with respect to the gadget vector $\vec{g} \in \mathbb{Z}^d$ and the corresponding decomposition function $g^{-1} : R_Q \to R^d$ is given by $\overline{\mathbf{C}} = (c_{i,j})_{1 \le i \le K+1, 1 \le j \le (K+1)d} \in R_Q^{(K+1)\times(K+1)d}$, such that $\left[\mathbf{I}_K \mid \mathbf{s}\right] \cdot \overline{\mathbf{C}} \approx \mu \cdot \left[\mathbf{I}_K \mid \mathbf{s}\right] \otimes \vec{g}^\top \pmod{Q}$. From this relation, we can easily show that

$$\overline{\mathbf{C}}' = \begin{bmatrix} c_{1,1} & \cdots & c_{1,kd} & c_{1,Kd+1} & \cdots & c_{1,(K+1)d} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ c_{k,1} & \cdots & c_{k,kd} & c_{k,Kd+1} & \cdots & c_{k,(K+1)d} \\ c_{K+1,1} & \cdots & c_{K+1,kd} & c_{K+1,Kd+1} & \cdots & c_{K+1,(K+1)d} \end{bmatrix} \in R_Q^{(k+1)\times(k+1)d}$$

is a cGSW encryption of μ under the secret $\mathbf{s}' = (s_1, \ldots, s_k)$ for any $1 \le k \le K$. We note that ciphertext shrinking can be directly applied to the cLEV encryption as well.

Key-Switching Key Generation. In real-world protocols using homomorphic encryption schemes with multiple parties, there can be more than two evaluators, each of whom evaluates their circuit and then potentially evaluates another circuit on their outputs. However, their choices of indices for parties may differ in the asymmetric setting, making the joint secret of each evaluator completely different. We remark that these secrets cannot be interchanged with a simple rearrangement, unlike the prior works in the CRS model. This raises the need for key-switching keys between these ciphertexts.

Suppose that we want to switch the joint secret from $\overline{\mathsf{sk}} = (\overline{s}_0 = 1, \overline{s}_1, \ldots, \overline{s}_k)$ to $\overline{\mathsf{sk}}' = (\overline{s}'_0 = 1, \overline{s}'_1, \ldots, \overline{s}'_k)$. Then, we require the key-switching key $\mathsf{ksk}_{\overline{\mathsf{sk}} \to \overline{\mathsf{sk}}'} \in R_Q^{(k+1)\times kd}$ from $\overline{\mathsf{sk}}$ to $\overline{\mathsf{sk}}'$, which satisfies the following relation.

$$\overline{\mathsf{sk}}'^\top \cdot \mathsf{ksk}_{\overline{\mathsf{sk}} \to \overline{\mathsf{sk}}'} \approx \left[\overline{s}_1 \ldots \overline{s}_k\right] \otimes \vec{g}^\top \pmod{Q}$$

Given an MLWE ciphertext $\mathbf{c} = (c_0, \ldots, c_k) \in R_Q^{k+1}$, its key can be switched from $\overline{\mathsf{sk}}$ to $\overline{\mathsf{sk}}'$ as follows:

$$\mathbf{c}' = \left[c_0\ 0 \ldots 0\right]^\top + \mathsf{ksk}_{\overline{\mathsf{sk}} \to \overline{\mathsf{sk}}'} \cdot g^{-1}(\left[c_1 \ldots c_k\right]^\top) \pmod{Q}.$$

It is easy to show that this algorithm is valid since

$$\overline{\mathsf{sk}}'^\top \cdot \mathbf{c}' \approx c_0 + \left[\overline{s}_1 \ldots \overline{s}_k\right] \otimes \vec{g}^\top \cdot g^{-1}(\left[c_1 \ldots c_k\right]^\top) = \overline{\mathsf{sk}}^\top \cdot \mathbf{c} \pmod{Q}.$$

Now, observe that the key-switching key $\mathsf{ksk}_{\overline{\mathsf{sk}} \to \overline{\mathsf{sk}}'}$ from $\overline{\mathsf{sk}}$ to $\overline{\mathsf{sk}}'$ is essentially a stack of MLWE encryptions of $g_j \overline{s}_i$ for $1 \leq i \leq k$, $0 \leq j < d$ under $\overline{\mathsf{sk}}'$. We remark that this can be computed homomorphically. We briefly outline two potential methods for this. Observe that $g_j \overline{s}_i$ is a multivariate polynomial of the entries of each party's secret. Therefore, this can be generated from the cLEV encryptions of $s_{i,j}$'s using the homomorphic encryption scheme from Sec. 3.1. However, this can be computationally expensive since we can no longer utilize the recurrence relation for the joint secret, as the index set is not compatible. Instead, we can also utilize the internal product of RGSW.

Roughly speaking, we first generate the RGSW encryptions of the entries of each party's secret under $\overline{\mathsf{sk}}'$ given the cGSW encryptions of $s_{i,j}$'s. Then, we utilize the internal product to generate MLWE encryptions of $\overline{s}_i$ from them. With this approach, we can construct an iterative circuit using the recurrence relations for the joint key, thereby making the evaluation much conceptually simpler.

Remark 1. We note that the input index set idx and output index set idx' for the joint secrets $\overline{\mathsf{sk}}$ and $\overline{\mathsf{sk}}'$, may contain different parties. However, it is obvious that idx should be a subset of idx', and then we can simply append the missing parties to the end of the index set idx and obtain a new index set $\overline{idx}$. Subsequently, as observed in Sec. 3.2, we can easily convert the ciphertext under the joint key with respect to idx into one under the joint key with respect to $\overline{idx}$ by appending some zeros to the ciphertext. Therefore, we can easily perform key-switching with the key switching key from multiplicative index $\overline{idx}$ to idx'.

Circuit Evaluation. Finally, we describe the circuit evaluation with our multi-key RGSW scheme. Assume a circuit $\mathcal{C}$ that takes the messages $\mu_1, \ldots, \mu_m$ as input. Also, let K be the maximum possible number of parties. Then, each party outputs the cGSW encryption of their messages and the cLEV encryptions of each entry of the secret key, of dimension K. Upon receiving these encryptions, the evaluator first picks the multiplication order. Next, it uses ciphertext shrinking to match the dimension of the input ciphertexts with the multiplication order. Next, it applies the RGSW extension algorithm to the input ciphertexts to generate the RGSW encryptions of $\mu_1, \ldots, \mu_m$. Then, the evaluator directly uses the circuit evaluation algorithm of the RGSW scheme.

The input to the circuit may not be a fresh ciphertext but rather the outputs of another circuit evaluated by different evaluators. In this case, the evaluators

may have chosen different multiplication orders. Then, the key-switching key generation algorithm can be run to generate key-switching keys, and the secrets of the input ciphertexts can be switched to one universal secret. Then, the RGSW circuit evaluation can be applied in a black-box manner again.

4.2 Scheme Description

Now we formally describe our scheme.

•$\underline{\mathsf{MKFHE.Setup}}(1^\lambda, 1^K)$: On input a security parameter λ and the maximum number of parties K, choose the ciphertext dimension N, the ciphertext modulus Q, the Hamming weight H of the secret, the noise distribution ψ over $R := \mathbb{Z}[X]/(X^N + 1)$, the gadget vector $\vec{g} \in \mathbb{Z}^d$, and the corresponding decomposition function $g^{-1} : R_Q \to R^d$. Return the public parameter $\mathsf{pp} = (K, N, Q, H, \psi, \vec{g}, g^{-1})$.

•$\underline{\mathsf{MKFHE.KeyGen}(\mathsf{pp})}$: On input a public parameter pp, sample s_i uniformly from the set of ternary polynomials with Hamming weight H for $1 \le i \le K$, and set $\mathsf{sk} = (s_1, \ldots, s_K) \in R^K$. Let $\mathsf{ek}_i \leftarrow \mathsf{cGSW.Enc}(\mathsf{sk}; s_i)$ for $1 \le i \le K$ and set the evaluation key $\mathsf{ek} = \{\mathsf{ek}_i\}_{1 \le i \le K}$ [7]. Sample $a \xleftarrow{\$} R_Q$ and $\mathbf{e} \leftarrow \psi^K$, and set $\mathsf{pk} = \begin{bmatrix} -a \cdot \mathsf{sk} + \mathbf{e} \\ a \end{bmatrix} \in R_Q^{K+1}$. Return $(\mathsf{sk}, \mathsf{ek}, \mathsf{pk})$.

•$\underline{\mathsf{MKFHE.Enc}(\mathsf{pk}; \mu)}$: On input a public key pk and a message $\mu \in R$, set $\mathsf{ct} \leftarrow \underline{\mathsf{cGSW.Enc}(\mathsf{pk}; \mu)}$ and return ct.

•$\underline{\mathsf{MKFHE.Eval}(\mathcal{C}, \{(\mathsf{ek}_i, \mathsf{pk}_i, \mathsf{ct}_i)\}_{i \in [k]})}$: On input the circuit $\mathcal{C}$ and k tuples $(\mathsf{ek}_i, \mathsf{pk}_i, \mathsf{ct}_i)$ for some $1 \le k \le K$ where each consists of an evaluation key ek_i, and cGSW ciphertexts ct_i, do as described in Sec. 4.1. More specifically,

1. Choose a multiplication order as a list $idx \in \mathbb{Z}^k$.
2. Perform the ciphertext shrinking from Sec. 4.1 on the evaluation keys $\{\mathsf{ek}_i\}_{i \in [k]}$ and cGSW ciphertexts $\{\mathsf{ct}_i\}_{i \in [k]}$ with respect to the multiplication order idx [8].
3. Perform the RGSW extension algorithm from Sec. 3.2 on the shrunk cGSW ciphertexts using the evaluation keys $\{\mathsf{ek}_i\}_{i \in [k]}$ to obtain multi-key RGSW ciphertexts $\{\mathsf{CT}_i\}_{i \in [k]}$, where each ciphertext contains information about the multiplication order idx.
4. Compute the circuit $\mathcal{C}$ from $\{\mathsf{CT}_i\}_{i \in [k]}$ using the internal product algorithm and ciphertext addition.
5. Return the output multi-key RGSW ciphertext CT.

[7] Note that we provide cGSW encryptions of s_i, from which the cLEV encryption of s_i can be trivially obtained as discussed in Sec. 3.1.

[8] There can be several ways to choose the index set. For example, we can choose the order based on the alphabetical order of the ciphertexts to ensure that the order remains the same for all evaluators.

•$\mathsf{MKFHE.Dec}(\{\mathsf{sk}_i\}_{i\in[k]}; \mathsf{CT})$: On input all participants' secret keys and a multi-key RGSW ciphertext CT, do the following with a secure MPC protocol.

1. Compute the joint key

$$\overline{\mathsf{sk}} = \prod_{i=0}^{k-1} \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \\ s_{idx[k-i],1} & \cdots & s_{idx[k-i],k-i} \end{bmatrix}$$

   with respect to the multiplicative order of CT.
2. Output $\mathsf{RGSW.Dec}(\overline{\mathsf{sk}}; \mathsf{CT})$.

We also show the correctness and compactness, and we address the security of our MKFHE scheme below.

Correctness. By Lem. 4, each party's cGSW encryptions can be switched to the multi-key RGSW ciphertexts that encrypt the same underlying message. Therefore, due to the correctness of the circuit evaluation from the RGSW scheme, the correctness of our scheme holds.

Compactness. The output RGSW ciphertext from Algorithm 2 has a size of $d(k+1)^2N$. In the typical RLWE setting, we set $\log Q, N \in O(\lambda)$ for the security parameter λ. Since $d = O(\log Q)$, our ciphertext has an asymptotic size of $O(k^2\lambda^2)$, which is a bivariate polynomial of k and λ. Since the RGSW scheme preserves the size after circuit evaluation, our scheme also satisfies the compactness from the definition of MKFHE.

Security. The security of our scheme relies on the shared-A RLWE hardness problem and the following circular security assumption: For the secret distribution χ and the dimension k, let $\mathbf{s} = (s_i)_{1\le i\le k} \leftarrow \chi^k$ be the secret key. Then, the shared-A RLWE samples for the secret $\mathbf{s}$ is indistinguishable from the uniform distribution, even given cGSW encryptions of s_i under the secret $\mathbf{s}$ for all $1 \le i \le k$.

We remark that our circular security fundamentally differs from the conventional circular security used in the literature. The most widely-used FHE schemes, including GSW, BFV, BGV, and CKKS, rely on the *quadratic circular LWE* problem, which calls for whether it is secure to disclose encryptions of a quadratic function of the secret key along with LWE samples [45]. In contrast, the circular security assumption of our scheme is necessary for analyzing the security of providing the evaluation key, which is basically a set of cGSW encryptions of s_i. We note that this is captured by the *clique-key assumption*, where all s_i's are encrypted under s_j's for all (i, j) pairs. While the conventional clique-key assumption for LWE is reduced from the basic quadratic circular security of LWE [45], the circular security of our scheme does not follow from the quadratic circular security as it is combined with the shared-A RLWE problem.

4.3 Two-round MPC from MKFHE

Finally, we provide a brief overview of how to build a two-round MPC protocol in the plain model, based on the transformation from [4]. Roughly speaking, a circuit can be computed with our multi-key FHE using each party's ciphertexts, and then the output ciphertexts can be decrypted via a one-round distributed decryption protocol.

As described in prior works [41, 35, 36, 4], the distributed decryption process can be instantiated with another MPC protocol, including multi-party (threshold) FHE or a garbled circuit. However, the use of a general MPC would substantially increase the round complexity of the decryption protocol. Therefore, we leverage the transformation from Ananth et al. [4] to achieve one-round decryption for our MKFHE scheme. In [4], the authors built an MKFHE scheme in the plain model with one-round decryption from three ingredients, as we describe in Thm. 1.

Theorem 1 ([4, Thm 1.1]). *Assuming,*

1. *Multi-key FHE with trusted setup and one-round decryption,*
2. *Multi-key FHE in the plain model but with arbitrary round decryption,*
3. *Two-round semi-malicious oblivious transfer in the plain model*

there exists multi-key FHE in the plain model with one-round decryption.

Briefly speaking, they evaluate the 'complex' decryption circuit of an MKFHE scheme in the plain model with arbitrary round decryption, using a non-compact multi-key homomorphic encryption scheme made from the OT and an MKFHE scheme with trusted setup (this is referred to as MHE in [4]). Therefore, this construction can be naturally extended to our MKFHE scheme by realizing the decryption algorithm of our MKFHE scheme with this MHE. Note that the authors provided a concrete instantiation of MHE solely based on the hardness of the (R)LWE problem, from existing LWE-based MKFHE schemes including [46, 50, 15, 36, 16] and LWE-based OT protocols such as [10, 8, 44, 40, 57, 58]. Consequently, we can build an MKFHE in the plain model by combining these works and our MKFHE, due to Thm. 1. For a more detailed construction, we refer to Section 8 of [4].

5 Multi-key FHE Key Generation

In this section, we explain how our multi-key RGSW scheme can be used as a baseline for the construction of a multi-key variant of the existing RLWE-based FHE schemes, such as BFV [9, 26], CKKS [19], TFHE [20], LMKCDEY [38], and Carousel [30]. The main difficulty in extending a single-key scheme to a multi-key scheme lies in the evaluation key generation. This is mainly because a single-key shared-A ciphertext can be easily converted into a multi-key ciphertext with simple zero padding, as discussed in Sec. 3.2, whereas the evaluation keys are essentially encryptions of some multivariate polynomial of each party's secret. Therefore, in this section, we only discuss how to generate the evaluation keys for a variety of homomorphic encryption schemes.

5.1 BFV/CKKS

We consider the key generation for the multi-key BFV and CKKS schemes [9]. These two schemes have the same evaluation key structure, although they have different functionalities; one supports finite field arithmetic, whereas the other supports fixed-point arithmetic. Roughly speaking, their evaluation key consists of two types of key-switching keys, namely the automorphism key and the relinearization key. Below, we describe how to generate them one by one.

Automorphism Keys. The automorphism key with respect to the automorphism Ψ is given by a key-switching key from $\Psi(\overline{\mathsf{sk}}) = (\Psi(s_0) = 1, \Psi(s_1), \ldots, \Psi(s_k))$ to $\overline{\mathsf{sk}}$. Given an MLWE encryption $\mathbf{c} = (c_0, \ldots, c_k) \in R_Q^{k+1}$ of the plaintext m under the secret $\overline{\mathsf{sk}}$, *i.e.,* $\overline{\mathsf{sk}}^\top \cdot \mathbf{c} = \sum_{i=0}^{k} c_i \overline{s}_i \approx m \pmod{Q}$, it is easy to see that $\Psi(\mathbf{c}) = (\Psi(c_0), \ldots, \Psi(c_k)) \in R_Q^{k+1}$ is an encryption of $\Psi(m)$ under the secret $\Psi(\overline{\mathsf{sk}})$ since

$$\Psi(\overline{\mathsf{sk}})^\top \cdot \Psi(\mathbf{c}) = \sum_{i=0}^{k} \Psi(c_i) \cdot \Psi(\overline{s}_i) \approx \Psi(m) \pmod{Q}.$$

Therefore, we can perform the homomorphic automorphism operation by simply switching the secret key of $\Psi(\mathbf{c})$ from $\Psi(\overline{\mathsf{sk}})$ to $\overline{\mathsf{sk}}$.

To generate the key-switching key, we utilize a similar induction from Algorithm 2. Note that the secret key $\Psi(\overline{\mathsf{sk}})$ can be iteratively generated with the recurrence relation Eq. (5) in the same manner as $\overline{\mathsf{sk}}$. We describe the full algorithm in Algorithm 3.

Algorithm 3 Automorphism Key Generation

Input: cLEV encryptions $\mathsf{cLEV}_{\mathbf{s}_i}(\Psi(s_{i,j}))$ for $1 \le i \le k$ and $1 \le j \le i$.

for $0 \le i < d$ **do**
  $\mathsf{acc}_i^0 \leftarrow (g_i)$
end for
for $1 \le i \le k$, $0 \le j < d$ **do**
  for $0 \le \ell < i$ **do**
    $\mathsf{acc}_{d\ell+j}^i \leftarrow \begin{bmatrix} \mathsf{acc}_{d\ell+j}^{i-1} \\ 0 \end{bmatrix}$
  end for
  $\mathsf{acc}_{di+j}^i \leftarrow \sum_{\ell=1}^{i} \mathsf{acc}_{d(\ell-1)+j}^{i-1} \odot_v \mathsf{cLEV}_{\mathbf{s}_i}(\Psi(s_{i,\ell}))$
end for

Output: $\mathsf{ksk}_{\Psi(\overline{\mathsf{sk}}) \to \overline{\mathsf{sk}}} = \left[\mathsf{acc}_d^k \ldots \mathsf{acc}_{d(k+1)-1}^k\right]$

[9] We remark that the key generation for the BGV scheme is also feasible. We opted it out of the paper since it has a slightly different key structure compared to BFV and CKKS.

Relinarization Key. The relinearization key is a special form of key-switching key that is necessary to perform homomorphic multiplication with the BFV or CKKS scheme. Let us briefly explain. Suppose that we are given two MLWE encryptions $\mathbf{c} = (c_0, \ldots, c_k) \in R_Q^{k+1}$ and $\mathbf{c}' = (c'_0, \ldots, c'_k) \in R_Q^{k+1}$ of plaintexts m and m' under the secret $\overline{\mathsf{sk}}$, *i.e.,* $\mathsf{sk}^\top \cdot \mathbf{c} = \sum_{i=0}^{k} c_i \overline{s}_i \approx m \pmod{Q}$ and $\mathsf{sk}^\top \cdot \mathbf{c}' = \sum_{i=0}^{k} c'_i \overline{s}_i \approx m' \pmod{Q}$. Then, it is easy to see that the tensor $\mathbf{c} \otimes \mathbf{c}'$ of $\mathbf{c}$ and $\mathbf{c}'$ is an MLWE encryption of $m \cdot m'$ under the tensored key $\overline{\mathsf{sk}} \otimes \overline{\mathsf{sk}}$, since

$$(\overline{\mathsf{sk}} \otimes \overline{\mathsf{sk}})^\top \cdot (\mathbf{c} \otimes \mathbf{c}') = (\overline{\mathsf{sk}}^\top \cdot \mathbf{c}) \cdot (\overline{\mathsf{sk}}^\top \cdot \mathbf{c}') \approx m \cdot m' \pmod{Q}.$$ [10]

Therefore, the homomorphic multiplication can be performed by switching the secret of the ciphertext $\mathbf{c} \otimes \mathbf{c}'$ from the tensored key $\overline{\mathsf{sk}} \otimes \overline{\mathsf{sk}}$ to the original key $\overline{\mathsf{sk}}$. In the prior multi-key schemes [16, 34], this key-switching was done with k external products, given the RGSW encryptions of $\overline{s}_1, \ldots, \overline{s}_k$. Specifically, for two input ciphertexts $\mathbf{c} = (c_0, \ldots, c_k) \in R_Q^{k+1}$ and $\mathbf{c}' = (c'_0, \ldots, c'_k) \in R_Q^{k+1}$, (roughly) we compute the following [11].

$$c_0 \cdot \mathbf{c}' + \sum_{i=1}^{k} (c_i \cdot \mathbf{c}') \boxdot \mathsf{RGSW}(\overline{s}_i) \pmod{Q}.$$

Following this approach, we only need to generate the RGSW encryptions of $\overline{s}_i$, $1 \le i \le k$. To achieve this, we first generate RGSW encryptions of $s_{i,j}$ using Algorithm 2. Then, we homomorphically compute Eq. (5) with the RGSW cryptosystem to obtain the RGSW encryptions of $\overline{s}_i$ iteratively. We describe the full algorithm in Algorithm 4.

Algorithm 4 Relinearization Key Generation

Input: cGSW encryptions $\mathsf{cGSW}_{\mathbf{s}_i}(s_{i,j})$ for $1 \le i \le k$ and $1 \le j \le i$.

1: **for** $1 \le i \le k, 1 \le j \le i$ **do**
2: Run Algorithm 2 to obtain the RGSW encryption $\overline{\mathbf{C}}_{i,j}$ of $s_{i,j}$.
3: **end for**
4: $\mathsf{rlk}_0 \leftarrow \mathbf{I}_{k+1} \otimes \vec{g}^\top$
5: **for** $1 \le i \le k$ **do**
6: $\mathsf{rlk}_i \leftarrow \sum_{\ell=1}^{i} \mathsf{rlk}_{\ell-1} \boxtimes \overline{\mathbf{C}}_{i,\ell}$
7: **end for**

Output: $\{\mathsf{rlk}_i\}_{1 \le i \le k}$

[10] We remark that the BFV scheme has a different tensoring process due to its MSB encoding strategy. Nevertheless, the structure of the relinearization key remains the same.

[11] The performance can be improved with re-ordering of entries, however, the asymptotic complexity remains the same.

5.2 TFHE/LMKCDEY/Carousel

The homomorphic encryption schemes such as TFHE, LMKCDEY, and Carousel (often called the AP[3] family) require a special bootstrapping key, called the blind rotation key, in addition to the optional automorphism keys. Suppose that each coefficient $\overline{s}_{i,j}$ of the secret $\overline{\mathsf{sk}}$ is confined to some interval $[-B, B]$ for some integer B, for $1 \leq i \leq k$ and $1 \leq j \leq N$. Then, TFHE and Carousel require the RGSW encryptions of a binary value that indicates whether $\overline{s}_{i,j} = k$ or not, for all $-B \leq k \leq B$. On the other hand, the LMKCDEY scheme requires the RGSW encryptions of $X^{\overline{s}_{i,j}}$.

We remark that the blind rotation key can be easily obtained with a homomorphic evaluation of Eq. (5), but with a coefficient-to-coefficient naïve multiplication. Firstly, we describe the generation of the blind rotation key for TFHE/Carousel. Assume that we are given two numbers $-B_x \leq x \leq B_x, -B_y \leq y \leq B_y$ for bounds $B_x, B_y > 0$, given in the indicator form

$$x_i = \begin{cases} 1 & \text{if } x = i \\ 0 & \text{otherwise} \end{cases} \text{ for } -B_x \leq i \leq B_x,$$

$$y_i = \begin{cases} 1 & \text{if } y = i \\ 0 & \text{otherwise} \end{cases} \text{ for } -B_y \leq i \leq B_y.$$

Then, we can obtain the indicator form of $z^+ = x + y$ and $z^\times = xy$ as follows:

$$z_i^+ = \sum_{\substack{-B_x \leq j \leq B_x, \\ -B_y \leq k \leq B_y, \\ j+k=i}} x_j \cdot y_k \text{ for } -B_x - B_y \leq i \leq B_x + B_y, \tag{10}$$

$$z_i^\times = \sum_{\substack{-B_x \leq j \leq B_x, \\ -B_y \leq k \leq B_y, \\ j \cdot k=i}} x_j \cdot y_k \text{ for } -B_x B_y \leq i \leq B_x B_y. \tag{11}$$

Therefore, we can iteratively compute the indicator form of each coefficient of $\overline{s}_i$'s, given the indicator form of each coefficient of $s_{i,j}$ for all $1 \leq i \leq k$ and $1 \leq j \leq i$ [12]. The generation of the blind rotation key for LMKCDEY can be directly followed from this, since there is a trivial transformation from the indicator form to the exponent form. To be specific, for the indicator form $y_{-B}, \ldots, y_B$ of $-B \leq y \leq B$ for some bound B, we have the following equality:

$$X^y = \sum_{i=-B}^{B} y_i \cdot X^i.$$

So, we can simply run the blind rotation key generation algorithm for the TFHE/Carousel scheme, and then homomorphically evaluate the equation above, to obtain the blind rotation key for the LMKCDEY scheme. We provide the full algorithm for TFHE/Carousel blind rotation key algorithm in Algorithm 5.

[12] We remark that this approach can be seen as a generalization of the n-out-of-n threshold TFHE construction by Park and Rovira [48]. In their work, they only considered the addition between the indicator form.

Algorithm 5 Blind Rotation Key Generation

Input: cGSW encryptions of the indicator form of each coefficient of $s_{i,j}$ for $1 \leq i \leq k$ and $1 \leq j \leq i$.

1: Run Algorithm 2 to obtain the RGSW encryptions of the indicator form of each coefficient of $s_{i,j}$ for all $1 \leq i \leq k$ and $1 \leq j \leq i$.
2: **for** $1 \leq i \leq k$ **do**
3: Use Eq. (10) and Eq. (11) to iteratively obtain the RGSW encryptions of the indicator form of each coefficient of $\overline{\mathsf{sk}}_i$, using Eq. (4).
4: **end for**

Output: Output the RGSW encryptions of the indicator form of each coefficient of $\overline{\mathsf{sk}}_i$ for $1 \leq i \leq k$.

6 Implementation

To show the correctness and to demonstrate the concrete performance of our MKFHE scheme, we provide a proof-of-concept implementation of it. We implemented our scheme on top of HIENAA.jl [29] library, written in Julia. The source code is available at https://github.com/SNUCP/crsless.

Our recommended parameter sets allow the parties to evaluate circuits with a depth of at most two. We fix the Hamming weight of the secret keys by $H = 192$, and choose the ring degree N and the bit-length of modulus $\log Q$ to achieve 128-bit security. The lattice-estimator [2] was used to measure the security of our parameter sets. In the experiments, we used the Residue Number System (RNS) gadget decomposition technique. Roughly speaking, the ciphertext modulus Q is a product of d primes $Q_1, \ldots, Q_d$ of the size $\log Q/d$, and the gadget decomposition function outputs the ring element modulo Q_i for all $1 \leq i \leq d$. Therefore, B_g naturally becomes $\max(Q_1, \ldots, Q_d) \approx Q^{1/d}$. We choose the discrete Gaussian distribution with $\sigma = 3.2$ as the noise distribution, which has the probabilistic bound of $3.2 \times 13.11 \approx 42$ with an overwhelming probability of $1 - 2^{-128}$. By substituting these values into the noise formula for RGSW generation Lem. 4, we can guarantee that the decryption failure probability with our parameters is smaller than 2^{-128}.

We present the recommended parameter sets for our MKFHE in Fig. 1a, and provide the performance of our algorithms under these parameters. All experiments were performed on a machine equipped with an Intel(R) Xeon(R) Platinum 8268 CPU @ 2.90GHz and 376GB RAM, running Ubuntu 20.04.4 LTS. Note that the runtime of the internal product is independent of the underlying multi-key scheme, but purely depends on the dimension of the ciphertext.

Acknowledgement

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. RS-2023-00211649) and the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. RS-2024-00399491,

K	N	$\log Q$	d
2	2^{12}	100	6
4	2^{13}	214	5
8	2^{13}	214	6

(a)

K	Vector Gadget Product	RGSW Extension	Internal Product	Ciphertext Size	Public Key Size
2	11ms	177ms	178ms	10.6MB	21.3MB
4	66ms	962ms	756ms	41.0MB	164.0MB
8	425ms	10897ms	5718ms	191.28MB	1.530GB

(b)

Fig. 1. (a) Recommended parameter sets for our MKFHE scheme. K denotes the maximum number of the participants, N denotes the ring dimension, $\log Q$ denotes the bit-length of the ciphertext modulus, and d denotes the gadget length. (b) Performance of vector gadget product, RGSW extension, and internal product algorithms, respectively, and the size of the ciphertexts and public key, with respect to the number of the parties K.

Development of Privacy-Preserving Multiparty Computation Techniques for Secure Multiparty Data Integration).

References

1. Albrecht, M.R., Bai, S., Ducas, L.: A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 153–178. Springer, Berlin, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53018-4_6
2. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology **9**(3), 169–203 (2015)
3. Alperin-Sheriff, J., Peikert, C.: Faster bootstrapping with polynomial error. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 297–314. Springer, Berlin, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44371-2_17
4. Ananth, P., Jain, A., Jin, Z., Malavolta, G.: Multi-key fully-homomorphic encryption in the plain model. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part I. LNCS, vol. 12550, pp. 28–57. Springer, Cham (Nov 2020). https://doi.org/10.1007/978-3-030-64375-1_2
5. Apple Machine Learning Research: Combining machine learning and homomorphic encryption in the apple ecosystem (2024), https://machinelearning.apple.com/research/homomorphic-encryption
6. Bae, Y., Cheon, J.H., Hanrot, G., Park, J.H., Stehlé, D.: Plaintext-ciphertext matrix multiplication and FHE bootstrapping: Fast and fused. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part III. LNCS, vol. 14922, pp. 387–421. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68382-4_12
7. Bergerat, L., Bonte, C., Curtis, B.R., Orfila, J.B., Paillier, P., Tap, S.: Sharing the mask: Tfhe bootstrapping on packed messages. IACR Transactions on Cryptographic Hardware and Embedded Systems **2025**(4), 925–971 (2025)

8. Blazy, O., Chevalier, C., Vu, Q.H.: Post-quantum uc-secure oblivious transfer in the standard model with adaptive corruptions. In: Proceedings of the 14th International Conference on Availability, Reliability and Security. ARES '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3339252.3339280, https://doi.org/10.1145/3339252.3339280
9. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Berlin, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-32009-5_50
10. Brakerski, Z., Döttling, N.: Two-message statistically sender-private OT from LWE. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part II. LNCS, vol. 11240, pp. 370–390. Springer, Cham (Nov 2018). https://doi.org/10.1007/978-3-030-03810-6_14
11. Brakerski, Z., Halevi, S., Polychroniadou, A.: Four round secure computation without setup. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 645–677. Springer, Cham (Nov 2017). https://doi.org/10.1007/978-3-319-70500-2_22
12. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. pp. 575–584. ACM Press (Jun 2013). https://doi.org/10.1145/2488608.2488680
13. Brakerski, Z., Perlman, R.: Lattice-based fully dynamic multi-key FHE with short ciphertexts. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 190–213. Springer, Berlin, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53018-4_8
14. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) lwe. SIAM Journal on computing **43**(2), 831–871 (2014)
15. Chen, H., Chillotti, I., Song, Y.: Multi-key homomorphic encryption from TFHE. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part II. LNCS, vol. 11922, pp. 446–472. Springer, Cham (Dec 2019). https://doi.org/10.1007/978-3-030-34621-8_16
16. Chen, H., Dai, W., Kim, M., Song, Y.: Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 395–412. ACM Press (Nov 2019). https://doi.org/10.1145/3319535.3363207
17. Chen, L., Zhang, Z., Wang, X.: Batched multi-hop multi-key FHE from ring-LWE with compact ciphertext extension. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part II. LNCS, vol. 10678, pp. 597–627. Springer, Cham (Nov 2017). https://doi.org/10.1007/978-3-319-70503-3_20
18. Cheon, J.H., Jeong, J., Lee, C.: An algorithm for ntru problems and cryptanalysis of the ggh multilinear map without a low-level encoding of zero. LMS Journal of Computation and Mathematics **19**(A), 255–266 (2016). https://doi.org/10.1112/S1461157016000371
19. Cheon, J.H., Kim, A., Kim, M., Song, Y.S.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 409–437. Springer, Cham (Dec 2017). https://doi.org/10.1007/978-3-319-70694-8_15
20. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 3–33. Springer, Berlin, Heidelberg (Dec 2016). https://doi.org/10.1007/978-3-662-53887-6_1

21. Clear, M., McGoldrick, C.: Multi-identity and multi-key leveled FHE from learning with errors. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 630–656. Springer, Berlin, Heidelberg (Aug 2015). https://doi.org/10.1007/978-3-662-48000-7_31
22. Colombo, L., Falcetta, A., Roveri, M.: Training encrypted neural networks on encrypted data with fully homomorphic encryption. In: Proceedings of the 12th Workshop on Encrypted Computing & Applied Homomorphic Cryptography. p. 64–75. WAHC '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3689945.3694802, https://doi.org/10.1145/3689945.3694802
23. Cong, K., Das, D., Park, J., Pereira, H.V.L.: SortingHat: Efficient private decision tree evaluation via homomorphic encryption and transciphering. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 563–577. ACM Press (Nov 2022). https://doi.org/10.1145/3548606.3560702
24. Cong, K., Kang, J., Nicolas, G., Park, J.: Faster private decision tree evaluation for batched input from homomorphic encryption. In: Galdi, C., Phan, D.H. (eds.) SCN 24, Part II. LNCS, vol. 14974, pp. 3–23. Springer, Cham (Sep 2024). https://doi.org/10.1007/978-3-031-71073-5_1
25. Ducas, L., Micciancio, D.: FHEW: Bootstrapping homomorphic encryption in less than a second. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 617–640. Springer, Berlin, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46800-5_24
26. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012), https://eprint.iacr.org/2012/144
27. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 75–92. Springer, Berlin, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40041-4_5
28. Hiromasa, R., Abe, M., Okamoto, T.: Packing messages and optimizing bootstrapping in GSW-FHE. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 699–715. Springer, Berlin, Heidelberg (Mar / Apr 2015). https://doi.org/10.1007/978-3-662-46447-2_31
29. Hwang, I., Min, S.: HIENAA.jl. Online: https://github.com/snu-lukemin/HIENAA.jl (2025)
30. Hwang, I., Min, S., Song, Y.: Carousel: Fully homomorphic encryption with bootstrapping over automorphism group. In: ASIACRYPT 2025, Part VII. pp. 131–164. LNCS, Springer, Singapore (Dec 2025). https://doi.org/10.1007/978-981-95-5122-4_5
31. Ju, J.H., Park, J., Kim, J., Kang, M., Kim, D., Cheon, J.H., Ahn, J.H.: NeuJeans: Private neural network inference with joint optimization of convolution and FHE bootstrapping. In: Luo, B., Liao, X., Xu, J., Kirda, E., Lie, D. (eds.) ACM CCS 2024. pp. 4361–4375. ACM Press (Oct 2024). https://doi.org/10.1145/3658644.3690375
32. Kim, E., Lee, H.S., Park, J.: Towards round-optimal secure multiparty computations: Multikey FHE without a CRS. In: Susilo, W., Yang, G. (eds.) ACISP 18. LNCS, vol. 10946, pp. 101–113. Springer, Cham (Jul 2018). https://doi.org/10.1007/978-3-319-93638-3_7
33. Kim, J., Lee, C.: A polynomial time algorithm for breaking NTRU encryption with multiple keys. DCC **91**(8), 2779–2789 (2023). https://doi.org/10.1007/s10623-023-01233-5

34. Kim, T., Kwak, H., Lee, D., Seo, J., Song, Y.: Asymptotically faster multi-key homomorphic encryption from homomorphic gadget decomposition. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) ACM CCS 2023. pp. 726–740. ACM Press (Nov 2023). https://doi.org/10.1145/3576915.3623176
35. 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: Jang-Jaccard, J., Guo, F. (eds.) ACISP 19. LNCS, vol. 11547, pp. 192–210. Springer, Cham (Jul 2019). https://doi.org/10.1007/978-3-030-21548-4_11
36. Kwak, H., Min, S., Song, Y.: Towards practical multi-key TFHE: Parallelizable, key-compatible, quasi-linear complexity. In: Tang, Q., Teague, V. (eds.) PKC 2024, Part II. LNCS, vol. 14604, pp. 354–385. Springer, Cham (Apr 2024). https://doi.org/10.1007/978-3-031-57728-4_12
37. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. DCC **75**(3), 565–599 (2015). https://doi.org/10.1007/s10623-014-9938-4
38. Lee, Y., Micciancio, D., Kim, A., Choi, R., Deryabin, M., Eom, J., Yoo, D.: Efficient FHEW bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part III. LNCS, vol. 14006, pp. 227–256. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30620-4_8
39. Li, N., Zhou, T., Yang, X., Han, Y., Liu, W., Tu, G.: Efficient multi-key fhe with short extended ciphertexts and directed decryption protocol. IEEE Access **7**, 56724–56732 (2019)
40. Liu, M., Hu, Y.: Universally composable oblivious transfer from ideal lattice. Frontiers of Computer Science **13**(4), 879–906 (2019)
41. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Karloff, H.J., Pitassi, T. (eds.) 44th ACM STOC. pp. 1219–1234. ACM Press (May 2012). https://doi.org/10.1145/2213977.2214086
42. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Berlin, Heidelberg (May / Jun 2010). https://doi.org/10.1007/978-3-642-13190-5_1
43. Mahdavi, R.A., Ni, H., Linkov, D., Kerschbaum, F.: Level up: Private non-interactive decision tree evaluation using levelled homomorphic encryption. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) ACM CCS 2023. pp. 2945–2958. ACM Press (Nov 2023). https://doi.org/10.1145/3576915.3623095
44. Micciancio, D., Sorrell, J.: Simpler statistically sender private oblivious transfer from ideals of cyclotomic integers. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 381–407. Springer (2020)
45. Micciancio, D., Vaikuntanathan, V.: SoK: Learning with errors, circular security, and fully homomorphic encryption. In: Tang, Q., Teague, V. (eds.) PKC 2024, Part II. LNCS, vol. 14604, pp. 291–321. Springer, Cham (Apr 2024). https://doi.org/10.1007/978-3-031-57728-4_10
46. Mukherjee, P., Wichs, D.: Two round multiparty computation via multi-key FHE. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 735–763. Springer, Berlin, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_26
47. National Institute of Standards and Technology (NIST): Multi-party threshold cryptography (2025), https://csrc.nist.gov/projects/threshold-cryptography

48. Park, J., Rovira, S.: Efficient tfhe bootstrapping in the multiparty setting. Ieee Access **11**, 118625–118638 (2023)
49. Park, J., Van Leeuwen, B., Zajonc, O.: FINALLY: A multi-key FHE scheme based on NTRU and LWE. CiC **1**(3), 15 (2024). https://doi.org/10.62056/aebn-4c2h
50. Peikert, C., Shiehian, S.: Multi-key FHE from LWE, revisited. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 217–238. Springer, Berlin, Heidelberg (Oct / Nov 2016). https://doi.org/10.1007/978-3-662-53644-5_9
51. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 187–196. ACM Press (May 2008). https://doi.org/10.1145/1374376.1374406
52. Solomon, R., Weber, R., Almashaqbeh, G.: smartFHE: Privacy-preserving smart contracts from fully homomorphic encryption. In: 2023 IEEE European Symposium on Security and Privacy. pp. 309–331. IEEE Computer Society Press (Jul 2023). https://doi.org/10.1109/EuroSP57164.2023.00027
53. Stehlé, D., Steinfeld, R., Tanaka, K., Xagawa, K.: Efficient public key encryption based on ideal lattices. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 617–635. Springer, Berlin, Heidelberg (Dec 2009). https://doi.org/10.1007/978-3-642-10366-7_36
54. Xiang, B., Zhang, J., Wang, K., Deng, Y., Feng, D.: NTRU-based bootstrapping for MK-FHEs without using overstretched parameters. In: Chung, K.M., Sasaki, Y. (eds.) ASIACRYPT 2024, Part I. LNCS, vol. 15484, pp. 241–270. Springer, Singapore (Dec 2024). https://doi.org/10.1007/978-981-96-0875-1_8
55. Xu, K., Huang, R.: Accelerated multi-key homomorphic encryption via automorphism-based circuit bootstrapping. IEEE Access (2024)
56. Xu, K., Tan, B.H.M., Wang, L.P., Aung, K.M.M., Wang, H.: Multi-key fully homomorphic encryption from ntru and (r)lwe with faster bootstrapping. Theoretical Computer Science **968**, 114026 (2023). https://doi.org/https://doi.org/10.1016/j.tcs.2023.114026, https://www.sciencedirect.com/science/article/pii/S0304397523003390
57. Yadav, V.K., Verma, S., Venkatesan, S.: Efficient and secure location-based services scheme in vanet. IEEE Transactions on Vehicular Technology **69**(11), 13567–13578 (2020)
58. Yadav, V.K., Verma, S., Venkatesan, S.: Linkable privacy-preserving scheme for location-based services. IEEE Transactions on Intelligent Transportation Systems **23**(7), 7998–8012 (2021)
59. Zama: Zama confidential blockchain protocol litepaper (2025), https://docs.zama.ai/protocol/zama-protocol-litepaper