Asymptotically Faster Multi-Key Homomorphic Encryption from Homomorphic Gadget Decomposition*

Taechan Kim[1], Hyesun Kawk[2], Dongwon Lee[2], Jinyeong Seo[2], and Yongsoo Song[2]

Samsung Research
tckim1458@samsung.com
Seoul National University
{hskwak, dongwonlee95, jinyeong.seo, y.song}@snu.ac.kr

Abstract. Homomorphic Encryption (HE) is a cryptosytem that allows us to perform an arbitrary computation on encrypted data. The standard HE, however, has a disadvantage in that the authority is concentrated in the secret key owner since computations can only be performed on ciphertexts encrypted under the same secret key. To resolve this issue, research is underway on Multi-Key Homomorphic Encryption (MKHE), which is a variant of HE supporting computations on ciphertexts possibly encrypted under different keys. Despite its ability to provide privacy for multiple parties, existing MKHE schemes suffer from poor performance due to the cost of multiplication which grows at least quadratically with the number of keys involved.
In this paper, we revisit the work of Chen et al. (ACM CCS 2019) on MKHE schemes from CKKS and BFV and significantly improve their performance. Specifically, we redesign the multi-key multiplication algorithm and achieve an asymptotically optimal complexity that grows linearly with the number of keys. Our construction relies on a new notion of gadget decomposition, which we call homomorphic gadget decomposition, where arithmetic operations can be performed over the decomposed vectors with guarantee of its functionality. Finally, we implement our MKHE schemes and demonstrate their benchmarks. For example, our multi-key CKKS multiplication takes only 0.5, 1.0, and 1.9 seconds compared to 1.6, 5.9, and 23.0 seconds of the previous work when 8, 16, and 32 keys are involved, respectively.

Keywords: Multi-Key Homomorphic Encryption, Ring Learning with Errors

1 Introduction

Homomorphic encryption (HE) is a cryptosystem that enables computation on encrypted data without decrypting them first. It has been a long-standing open problem to construct a fully HE (which supports arbitrary computations) until Gentry's breakthrough [19]. Since then, there have been made significant progresses on HE construction such as BFV [5, 17], GSW [21], BGV [6], TFHE [15], and CKKS [14]. HE inherently supports an on-the-fly secure computation, *i.e.,* no need for data owners to be online during the computation since the whole evaluation process can be done by a public server. Such characteristic is especially well-suited for the cases such as cloud-based environments.

Recently, there has been a growing demand for secure multi-party computation (MPC) protocols where participants collaboratively evaluate a circuit on their private inputs without revealing anything other than the result, with a wide range of applications such as federated learning [24]. However, the single-key HE is less amenable to this multi-party setting. For instance, when there are multiple data sources, the standard HE causes an authority concentration issue. If one considers directly applying standard single-key HE, data should be encrypted under the same encryption key. In this case, the person who has the corresponding secret key gains access to all data and thus the privacy of data owners may be exposed.

In the last decade, there have been several attempts to extend the functionality of HE to solve the aforementioned issues. Threshold HE [3, 4, 29, 32, 28] and Multi-Key HE (MKHE) [27, 16, 31, 33, 9, 10] are two representative methods that overcome the limitation of single-key HE by delegating decryption

* This work was supported by SAMSUNG Research, Samsung Electronics Co., Ltd.

authority to multiple parties so that no single party has access to the secret key. These primitives can be naturally extended to build secure multi-party computation (MPC) protocols that retain the advantages of HE, so NIST's recent call for multi-party threshold schemes [8] includes these primitives as promising candidates to be standardized.

In Threshold HE, a set of parties jointly generate a common public key while the corresponding secret key is shared among the participants. Threshold HE schemes have comparable performance as single-key HE and tend to be more efficient than MKHEs, but have a major limitation to have a static key access structure. In other words, all participants should be determined and fixed at the setup phase. In this paper, we focus on MKHE which enjoys considerable advantages in terms of interaction and flexibility. To be precise, an MKHE scheme allows each participant to generate its secret and public key pair without any knowledge of other parties. It supports advanced functionality of operating ciphertexts under different keys so that all computations can be done by a public cloud without building a common public key. Moreover, recent MKHE schemes are fully dynamic, *i.e.,* the computational task does not have to be pre-determined but an arbitrary circuit can be evaluated over any ciphertexts on the fly, and new users (ciphertexts) can be introduced into the computation anytime. Therefore, one can build a secure MPC protocol on top of MKHE which inherits this dynamic nature [31].

While MKHE enables flexible and dynamic setup, it is technically challenging, compared to other HE variants, to design an efficient MKHE scheme due to the strong requirement on the functionality. Since López-Alt et al. [27] presented the first MKHE scheme based on NTRU, there have been several studies [16, 31, 33, 7, 9, 12, 10] that convert the existing single-key HE schemes into multi-key versions, but the poor performance of MKHE still remains a major bottleneck. Earlier schemes were relatively impractical, but recent researches [9, 10] demonstrated viable instantiations with implementation results which are currently the best-performing MKHE schemes in terms of both asymptotic and concrete complexity.

This paper extends the previous work of Chen, Dai, Kim and Song (CDKS) [10] by presenting a multi-key variant of the RLWE-based BFV scheme that supports homomorphic operations in a SIMD manner. In CDKS, a multi-key ciphertext is a tuple $(c_0, c_1, \dots, c_n)$ where n is the number of associated keys and c_i's are elements of the base polynomial ring. It can be decrypted by the secret keys $s_1, \dots, s_n$ of n key owners, such that $c_0 + c_1 \cdot s_1 + \cdots + c_n \cdot s_n$ is a randomized encoding of the plaintext.

The most expensive operation is homomorphic multiplication, which consists of two steps: tensor product and subsequent relinearization. Given encryptions $(c_i)_{0 \le i \le n}$ and $(c'_i)_{0 \le j \le n}$ of m and m', respectively, it first computes their product $(c_{i,j} := c_i \cdot c'_j)_{0 \le i,j \le n}$, which can be viewed as a valid encryption of mm' that is decryptable by $s_i \cdot s_j$. Then, the relinearization procedure is followed, which converts $(c_{i,j})_{0 \le i,j \le n}$ back to the standard form with linear decryption structure.

For the relinearization procedure, CDKS uses a well-known technique in the construction of HE schemes called *gadget decomposition* [18], which is used to reduce the noise growth from homomorphic operations. Briefly speaking, gadget decomposition is a mapping h that transforms an ring element a modulo Q into a small-sized vector $h(a)$, such that $\langle h(a), \mathbf{g} \rangle = a \pmod{Q}$ holds for some fixed vector $\mathbf{g}$. By applying gadget decomposition technique on each $c_{i,j}$, it obtains a standard-form ciphertext removing quadratic structures. Therefore, the total complexity of relinearization grows quadratically with n since these procedures should be repeated on $c_{i,j}$ for all $1 \le i, j \le n$.

1.1 Our Contributions

In this paper, we propose new multi-key BFV and CKKS homomorphic multiplication algorithms with linear complexity by modifying the previous construction by CDKS [10]. To achieve linear complexity with respect to the number of associated keys, we need to reduce the number of gadget decompositions and ring multiplications, which induce quadratic complexity.

First, to avoid the expensive computation of $h(c_{i,j})$, we introduce a new notion called *homomorphic gadget decomposition.* We say that a gadget decomposition is homomorphic if it supports computation over decomposed vectors. In other words, we can perform arithmetic operations over the gadget decompositions $h(a), h(b)$ of any elements a, b, such that $h(a) + h(b)$ and $h(a) \circ h(b)$ satisfy $\langle h(a) + h(b), \mathbf{g} \rangle = a + b \pmod{Q}$ and $\langle h(a) \circ h(b), \mathbf{g} \rangle = ab \pmod{Q}$, where $\circ$ denotes the component-wise product of vectors. Hence, $h(a) + h(b)$ and $h(a) \circ h(b)$ can be considered valid decompositions of $a + b$ and ab, respectively. As a result, instead of repeating n^2 gadget decompositions for all pairs (i, j), we separately compute $h(c_i)$ and $h(c'_j)$ for $1 \leq i, j \leq n$ and combine them to represent a valid decomposition of $c_i \cdot c'_j$.

However, there still remain quadratic ring multiplication operations due to the tensor product procedure. To achieve linear complexity in ring multiplications, we depart from the conventional multiplication strategy based on tensor product and relinearization. In particular, we merge the two steps and refactor the whole multiplication algorithm so that it utilizes the homomorphic property of gadget decomposition. As a result, we reduce the overall complexity of n-key homomorphic multiplication from $O(n^2)$ down to $O(n)$, which we believe is asymptotically optimal.

While our idea can be directly applied to designing an efficient multi-key CKKS scheme, there still remains an issue for BFV. As the tensor product and relinearization procedures are performed over different algebraic spaces in BFV, such inconsistency inhibits applying homomorphic gadget decomposition. We resolve this issue by tweaking the public key structure so that the entire computation can be performed in the same ring. Additionally, we present another implementation-friendly variant of our multi-key BFV scheme that requires no multi-precision arithmetic.

Finally, we implement our MKHE schemes [1] and provide benchmark results to demonstrate its concrete performance in terms of complexity and noise growth. Our experiments show that our scheme rapidly outperforms the prior work as the number of keys increases. For example, our construction achieves about 12 and 7 times speed-up in 32-key CKKS and BFV multiplications, respectively, compared to CDKS while preserving the same level of noise growth.

1.2 Related Works

As mentioned above, there are several directions to extend the functionality of HE to multi-party settings. One such approach is Threshold HE [3, 4, 29, 32, 28], which distributes the authority and provides t-out-of-n access structure. In Threshold HEs, it is usually required to run a multi-round protocol to generate a shared public key that is used for encryption and evaluation.

MKHE is yet another scheme for multi-party scenarios, specifically designed to enable computation on ciphertexts that are encrypted under different keys. Early studies on MKHE [16, 31, 33] are mostly based on GSW [21], but they require huge space and time complexity. Brakerski and Perlman [7] designed an MKHE scheme from LWE with quasi-linear expansion rate, but its concrete performance was not clearly understood. A follow-up study was conducted by Chen, Chillotti and Song [9] who presented a multi-key variant of TFHE and demonstrated the first implementation result. On the other hand, there has been another line of work [12, 10] constructing multi-key variants of batch HE schemes such as BGV, BFV and CKKS. One common problem of the previous MKHE constructions is that they rely on the CRS assumption. Recently, Ananth et al. [2] constructed the first MKHE scheme in the plain model by combining the oblivious transfer protocol, MKHE with trusted setup, and MKHE in the plain model with interactive decryption.

2 Background

2.1 Notation

Let N be a power of two and Q be an integer. We denote by $R = \mathbb{Z}[X]/(X^N + 1)$ the ring of integers of the $(2N)$-th cyclotomic field and $R_Q = \mathbb{Z}_Q[X]/(X^N + 1)$ the residue ring of R modulo Q. We use $\mathbb{Z} \cap (-Q/2, Q/2]$ as a representative of $\mathbb{Z}_Q$, and denote by $[a]_Q$ the reduction of a modulo Q. For a polynomial a in R or R_Q, we define $\|a\|_\infty$ as the ℓ^∞-norm of its coefficient vector.

Throughout this paper, we write $x \leftarrow D$ to represent that x is sampled from the distribution D. We denote by $\mathcal{U}(S)$ the uniform distribution over a finite set S. For $\sigma > 0$, we denote by D_σ a distribution over R sampling N coefficients independently from the discrete Gaussian distribution of variance σ^2, and B_σ an (overwhelming probability) upper bound of D_σ with respect to the infinite norm.

[1] The source code is available at `https://github.com/SNUCP/MKHE-KKLSS`.

2.2 Ring Learning with Errors

Let χ be a distribution over R and $\sigma > 0$ a real. The ring learning with errors (RLWE) assumption with respect to the parameter (N, Q, χ, σ) is that given polynomially many samples of either (a, b) or $(a, as + e)$, where $a, b \leftarrow R_Q$, $s \leftarrow \chi$, $e \leftarrow D_\sigma$, it is computationally hard to distinguish which is the case. The security of lattice-based HE schemes, such as BFV [5, 17] and CKKS [14], rely on the RLWE assumption.

2.3 Multi-Key Homomorphic Encryption

A multi-key homomorphic encryption (MKHE) is an encryption scheme that allows for computation on encrypted data that may be encrypted under different secret keys. This is in contrast to plain homomorphic encryption (HE) schemes, which require data to be encrypted under the same secret key to perform homomorphic operations. Therefore, MKHE schemes can be considered as a generalization of HE that overcome this limitation and allow for more flexible computation on encrypted data.

An MKHE scheme consists of five probabilistic polynomial-time (PPT) algorithms: $\texttt{Setup}$, $\texttt{KeyGen}$, $\texttt{Enc}$, $\texttt{Eval}$, and $\texttt{Dec}$.

– **Setup:** $\mathsf{pp} \leftarrow \texttt{MKHE.Setup}(1^\lambda)$. Given the security parameter λ, it returns the public parameter set pp.
– **Key Generation:** $(\mathsf{sk}, \mathsf{pk}) \leftarrow \texttt{MKHE.KeyGen}(\mathsf{pp})$. Output a secret key sk and a public key pk.
– **Encryption:** $\mathsf{ct} \leftarrow \texttt{MKHE.Enc}(\mu; \mathsf{pk})$. Output a MKHE ciphertext which encrypts a plaintext μ under pk. The output ciphertext's reference set is $\{\mathsf{pk}\}$.
– **Evaluation:** $\mathsf{ct} \leftarrow \texttt{MKHE.Eval}(\mathcal{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_k; \{\mathsf{pk}_i\}_{i \in I})$. Given a circuit $\mathcal{C}$ and ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$ where the union of each ciphertexts' reference set is $\{\mathsf{pk}_i\}_{i \in I}$ for some index set I, it returns a ciphertext ct with a reference set $\{\mathsf{pk}_i\}_{i \in I}$.
– **Decryption:** $\mu \leftarrow \texttt{MKHE.Dec}(\mathsf{ct}; \{\mathsf{sk}_i\}_{i \in I})$. Given a ciphertext ct with a reference set $\{\mathsf{pk}_i\}_{i \in I}$ and the corresponding secret keys sk_i for each pk_i, it outputs a plaintext μ.

Each MKHE ciphertext implicitly maintains a set of references to the public keys associated with it. At the very beginning, a fresh ciphertext is associated with a single key that is used to generate it. As computations proceed with other ciphertexts encrypted under different keys, the reference set grows. Finally, when decrypting a ciphertext, all of its associated keys are required.

It is also possible for key owners to jointly decrypt a ciphertext without revealing their secret keys via a distributed decryption protocol. In this protocol, each key owner broadcasts a partial decryption of the ciphertext using its own secret, and then each key owners can recover the message by merging the partial decryptions of all key owners. More details about distributed decryption are described in [31, 10].

The correctness and semantic security of MKHE are defined as follows.

Definition 1 (Correctness). *Given MKHE ciphertexts* $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$ *that encrypt* $\mu_1, \ldots, \mu_k$*, respectively, let* $\{\mathsf{pk}_i\}_{i \in I}$ *be the union of their reference sets, and* $\mathsf{ct} \leftarrow \texttt{MKHE.Eval}(\mathcal{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_k; \{\mathsf{pk}_i\}_{i \in I})$*. Then,* $\texttt{MKHE.Dec}(\mathsf{ct}; \{\mathsf{sk}_i\}_{i \in I}) = \mathcal{C}(\mu_1, \ldots, \mu_k)$ *with an overwhelming probability.*

In the case of approximate HE such as CKKS [14], the correctness condition is weakened and substituted with an approximate equality: $\texttt{MKHE.Dec}(\mathsf{ct}; \{\mathsf{sk}_i\}_{i \in I}) \approx \mathcal{C}(\mu_1, \ldots, \mu_k)$.

Definition 2 (Security). *An MKHE scheme is called secure if it is semantically secure. More precisely, for a security parameter* λ*,* $\mathsf{pp} \leftarrow \texttt{MKHE.Setup}(1^\lambda)$*, and* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \texttt{MKHE.KeyGen}(\mathsf{pp})$*, the distributions of* $\texttt{MKHE.Enc}(\mathsf{pk}, \mu_0)$ *and* $\texttt{MKHE.Enc}(\mathsf{pk}, \mu_1)$ *are computationally indistinguishable for any messages* μ_0 *and* μ_1*.*

2.4 Gadget Decomposition

A gadget decomposition technique is frequently used in the construction of lattice-based HE schemes [14, 5, 17, 15] for reducing the noise growth of homomorphic operations. Informally, it is used to represent a ciphertext component as a linear combination of some fixed elements with small coefficients. Below, we review some basic terminology related to gadget decomposition.

Definition 3 (Gadget Decomposition). *Let Q and k be positive integers. A function $h : R_Q \to R^k$ is called a gadget decomposition if there exists a fixed vector $\mathbf{g} = (g_0, g_1, \ldots, g_{k-1}) \in R_Q^k$ and a bound $B_h > 0$ such that $\langle h(a), \mathbf{g}\rangle = a \pmod Q$ and $\|h(a)\|_\infty \le B_h$ for all $a \in R_Q$.*

The fixed vector $\mathbf{g}$ in the above is called the *gadget vector* and B_h a bound of h. Let us denote by $g : R^k \to R_Q$ the inner product function $g(\mathbf{u}) = \langle \mathbf{u}, \mathbf{g}\rangle \pmod Q$. We remark that h is a right inverse of g, *i.e.,* $g(h(a)) = a$ for all $a \in R_Q$. This is why the gadget decomposition is often denoted by g^{-1} in the literature, although it is an abuse of notation. The bound $B_h > 0$ is usually much smaller than the modulus Q, so a gadget decomposition can be viewed as a mapping to a short vector in the inverse image $g^{-1}(a) = \{\mathbf{u} \in R^k : \langle \mathbf{u}, \mathbf{g}\rangle = a \pmod Q\}$ of the input $a \in R_Q$.

Definition 4 (Gadget Encryption). *For an RLWE secret $s \in R$ and a message $\mu \in R$, we call $\mathbf{U} = (\mathbf{u}_0, \mathbf{u}_1) \in R_Q^{k\times 2}$ a gadget encryption of μ under s if $\mathbf{u}_0 + s \cdot \mathbf{u}_1 \approx \mu \cdot \mathbf{g} \pmod Q$.*

Definition 5 (External Product). *Let $a \in R_Q$ and $\mathbf{u} \in R_Q^k$. The external product of a and $\mathbf{u}$ is denoted and defined by $a \boxdot \mathbf{u} = \langle h(a), \mathbf{u}\rangle \pmod Q$. We also write $a \boxdot \mathbf{U} = (a \boxdot \mathbf{u}_0, a \boxdot \mathbf{u}_1)$ for $\mathbf{U} = (\mathbf{u}_0, \mathbf{u}_1) \in R_Q^{k\times 2}$.*

Using the gadget decomposition technique, one can homomorphically multiply arbitrary ring elements without introducing huge noise. To be precise, let $\mathbf{U} = (\mathbf{u}_0, \mathbf{u}_1) \in R_Q^{k\times 2}$ be a gadget encryption of $\mu \in R$ under s, *i.e.,* $\mathbf{u}_0 + s \cdot \mathbf{u}_1 = \mu \cdot \mathbf{g} + \mathbf{e} \pmod Q$ for some small $\mathbf{e} \in R^k$. Then, the external product $(c_0, c_1) \leftarrow a \boxdot \mathbf{U}$ satisfies

$$\begin{aligned} c_0 + s \cdot c_1 &= \langle h(a), \mathbf{u}_0 + s \cdot \mathbf{u}_1\rangle \\ &= \langle h(a), \mu \cdot \mathbf{g} + \mathbf{e}\rangle = a \cdot \mu + e \pmod Q \end{aligned} \tag{1}$$

for the error $e = \langle h(a), \mathbf{e}\rangle \in R$. Since e is small, (c_0, c_1) can be considered as noisy encryption of $a \cdot \mu$ as desired.

3 Overview of Prior Work

This section provides an overview of the most relevant research by Chen, Kim, Dai, and Song [10], which designs multi-key variants of CKKS and BFV. These MKHE schemes by Chen et al. (which we will call CDKS) are based on the common random string (CRS) model, where all key owners have access to the same random polynomials generated during the setup phase. Technically, a fresh ciphertext of CDKS has the same form as usual (single-key) HEs, but the ciphertext structure changes when we perform homomorphic computation on ciphertexts under different keys. More generally, an n-key ciphertext is the form of $\mathsf{ct} = (c_0, c_1, \ldots, c_n) \in R_Q^{n+1}$ whose decryption is defined as $c_0 + \sum_{i=1}^n c_i \cdot s_i$ where $s_1, \ldots, s_n$ are associated secret keys.

Homomorphic operations of CDKS are defined in a similar way to the single-key HE schemes. Suppose that $\mathsf{ct} = (c_i)_{0\le i\le n}$ and $\mathsf{ct}' = (c_i')_{0\le i\le n}$ are two multi-key ciphertexts under secrets $s_1, \ldots, s_n$. Then, their homomorphic addition is defined as $\mathsf{ct}_{add} = \mathsf{ct} + \mathsf{ct}' \pmod Q$. For homomorphic multiplication, it first computes the tensor product $\mathsf{ct}_{mul} = (c_{i,j} = c_i \cdot c_j')_{0\le i,j\le n}$, which satisfies that $c_{0,0} + \sum_{1\le i,j\le n} c_{i,j} \cdot s_i s_j = (c_0 + c_1 s_1 + \cdots + c_n s_n)(c_0' + c_1' s_1 + \cdots + c_n' s_n) \pmod Q$. Then, it is transformed into a standard MKHE ciphertext $\mathsf{ct}^* = (c_0^*, \ldots, c_n^*)$ encrypting the same plaintext as ct_{mul}. This procedure is called *relinearization* since the quadratic decryption structure of ct_{mul} is converted back to the linear form.

Lastly, when performing homomorphic operations on MKHE ciphertexts encrypted under different keys, it is required to synchronize their key structure by padding zeros or permuting some ciphertext components. For simplicity, we assume that this pre-processing is implicit (even if it is not described formally) so that input ciphertexts have the same key structure.

3.1 CDKS Relinearization

As we mentioned above, the homomorphic multiplication in CDKS requires relinearization procedure. In Alg. 1, we present the relinearization algorithm by CDKS. The relinearization algorithm takes a tensored

Algorithm 1 Relinearization of CDKS

```
Input: A MKHE ciphertext $\mathsf{ct}_{mul} = (c_{i,j})_{0\le i,j\le n} \in R_{Q_\ell}^{(n+1)\times(n+1)}$, and its associated public keys $\{\mathsf{pk}_i = (\mathbf{b}_i, \mathbf{d}_i, \mathbf{u}_i, \mathbf{v}_i)\}_{1\le i\le n}$
Output: $\mathsf{ct}^* = (c_i^*)_{0\le i\le n} \in R_{Q_\ell}^{n+1}$
1: $c_0^* \leftarrow c_{0,0}$
2: for $1 \le i \le n$ do
3:     $c_i^* \leftarrow c_{0,i} + c_{i,0} \pmod{Q_\ell}$
4: end for
5: for $1 \le i, j \le n$ do
6:     $c_j^* \leftarrow c_j^* + c_{i,j} \boxdot \mathbf{d}_i \pmod{Q_\ell}$
7:     $c'_{i,j} \leftarrow c_{i,j} \boxdot \mathbf{b}_j \pmod{Q_\ell}$
8:     $(c_0^*, c_i^*) \leftarrow (c_0^*, c_i^*) + c'_{i,j} \boxdot (\mathbf{v}_i, \mathbf{u}_i) \pmod{Q_\ell}$
9: end for
```

ciphertext $(c_{i,j})_{0\le i,j\le n}$ and public keys $(\mathbf{b}_i, \mathbf{d}_i, \mathbf{u}_i, \mathbf{v}_i)_{1\le i\le n}$ as inputs where the components of public keys satisfy the relations $\mathbf{b}_i \approx -s_i \cdot \mathbf{a} \pmod{Q}$, $\mathbf{d}_i \approx r_i \cdot \mathbf{a} + s_i \cdot \mathbf{g} \pmod{Q}$, and $\mathbf{v}_i \approx -s_i \cdot \mathbf{u}_i - r_i \cdot \mathbf{g}$ $\pmod{Q}$ for a CRS $\mathbf{a} \in R_Q^k$, a gadget vector $\mathbf{g} \in R_Q^k$, and a secret key $s_i \in R$. In Line 5–9 of Alg. 1, each entry $c_{i,j}$ is relinearized using $\mathbf{d}_i, \mathbf{u}_i, \mathbf{v}_i$ and $\mathbf{b}_j$ for $1 \le i, j \le n$. To be precise, it computes $c_{i,j} \boxdot \mathbf{d}_i$ and $c'_{i,j} \boxdot (\mathbf{v}_i, \mathbf{u}_i)$ for $c'_{i,j} = c_{i,j} \boxdot \mathbf{b}_j$, and arranges them so that

$$\begin{aligned}
&(c_{i,j} \boxdot \mathbf{d}_i) \cdot s_j + c'_{i,j} \boxdot (\mathbf{v}_i + s_i \cdot \mathbf{u}_i) \\
\approx &(c_{i,j} \boxdot \mathbf{d}_i) \cdot s_j - r_i \cdot c'_{i,j} = c_{i,j} \boxdot (s_j \cdot \mathbf{d}_i - r_i \cdot \mathbf{b}_j) \\
\approx &c_{i,j} \boxdot (r_i \cdot \mathbf{a} + \mathbf{d}_i) \cdot s_j \approx c_{i,j} \cdot s_i s_j \pmod{Q_\ell}.
\end{aligned}$$

Therefore, the output ciphertext $\mathsf{ct}^* = (c_0^*, c_1^*, \ldots, c_n^*)$ of Alg. 1 satisfies that $c_0^* + c_1^* s_1 + \cdots + c_n^* s_n \approx \sum_{0\le i,j\le n} c_{i,j} \cdot s_i s_j \pmod{Q_\ell}$ as desired.

We now briefly analyze the performance of the CDKS relinearization algorithm in terms of computational complexity and noise size. For the computational complexity, we count the number of ring multiplications and gadget decompositions. In Lines 5–7, the algorithm performs external products with $c_{i,j}$ and public key components $\mathbf{d}_i$ and $\mathbf{b}_j$. This step takes $2kn^2$ multiplications over R_{Q_ℓ} and n^2 gadget decompositions over R_{Q_ℓ}, since the decomposed value $h(c_{i,j})$ is reusable. Similarly, Line 7 also takes $2kn^2$ multiplications over R_{Q_ℓ} and n^2 gadget decompositions as it computes external products between $c'_{i,j}$ and public key components $\mathbf{v}_i$ and $\mathbf{u}_i$. Hence, Alg. 1 takes $4kn^2$ multiplications and $2n^2$ gadget decompositions over R_{Q_ℓ} in total. It is worth noting that there is an optimization technique proposed in [10] where $c_{i,j}$ is replaced by $c_i c'_j + c'_i c_j$. Then, it suffices to perform computation for $1 \le i \le j \le n$, which reduces the computational cost of relinearization almost by half.

For the noise analysis, we remark that the relinearization of $c_{i,j}$ introduces an error $e_{i,j} = c'_{i,j} \boxdot \mathbf{e}_{2,i} + c_{i,j} \boxdot (s_j \cdot \mathbf{e}_{1,i} - r_i \cdot \mathbf{e}_{0,j})$ which is bounded by$\|e_{i,j}\|_\infty \le kN \cdot B_h B_\sigma + 2kN^2 \cdot B_h B_\sigma \approx 2kN^2 \cdot B_h B_\sigma$. Therefore, the total relinearization noise has an upper bound

$$\left\| \sum_{1\le i,j\le n} e_{i,j} \right\|_\infty \lessapprox 2kn^2N^2 \cdot B_h B_\sigma. \tag{2}$$

In the following, we present multi-key variants of CKKS and BFV built upon the CDKS relinearization algorithm.

3.2 Multi-key CKKS

• $\underline{\texttt{MK-CKKS.Setup}(1^\lambda)}$: Let the RLWE dimension be N, the ciphertext modulus be $Q = \prod_{i=0}^{L} q_i$ for some integers q_i, the key distribution be χ over R and the error parameter be $\sigma > 0$. Let $h : R_Q \to R^k$ be a gadget decomposition corresponding to a gadget vector $\mathbf{g} \in R_Q^k$. Sample a CRS $\mathbf{a} \leftarrow \mathcal{U}(R_Q^k)$, and output a public parameter $\mathsf{pp} = (N, Q, \chi, \sigma, \mathbf{a}, h, \mathbf{g})$.

We write $Q_\ell = \prod_{i=0}^{\ell} q_i$ for $0 \le \ell \le L$.

- <u>`MK-CKKS.KeyGen(pp)`</u>: Return a secret key $\mathsf{sk} = s$ and a public key $\mathsf{pk} = (\mathbf{b}, \mathbf{d}, \mathbf{u}, \mathbf{v})$ generated as follows:
  - Sample $s \leftarrow \chi$.
  - Sample $\mathbf{e}_0 \leftarrow D_\sigma^k$ and let $\mathbf{b} = -s \cdot \mathbf{a} + \mathbf{e}_0 \pmod{Q}$.
  - Sample $r \leftarrow \chi$ and $\mathbf{e}_1 \leftarrow D_\sigma^k$. Let $\mathbf{d} = -r \cdot \mathbf{a} + s \cdot \mathbf{g} + \mathbf{e}_1 \pmod{Q}$.
  - Sample $\mathbf{u} \leftarrow \mathcal{U}(R_Q^k)$ and $\mathbf{e}_2 \leftarrow D_\sigma^k$. Let $\mathbf{v} = -s \cdot \mathbf{u} - r \cdot \mathbf{g} + \mathbf{e}_2 \pmod{Q}$.

We denote the encryption key as $\mathsf{ek} = (\mathbf{b}[0], \mathbf{a}[0]) \in R_Q^2$, which are the first components of $\mathbf{b}$ and $\mathbf{a}$, respectively. Additionally, we often use integer subscripts to identify keys from different key owners.

- <u>`MK-CKKS.Enc`$(\mu; \mathsf{ek})$</u>: Sample $w \leftarrow \chi$ and $e_0, e_1 \leftarrow D_\sigma$. Given a plaintext $\mu \in R$, output the ciphertext $\mathsf{ct} = w \cdot \mathsf{ek} + (\mu + e_0, e_1) \pmod{Q}$.

- <u>`MK-CKKS.Dec`$(\mathsf{ct}; \{\mathsf{sk}_i\}_{1 \le i \le n})$</u>: Given a ciphertext $\mathsf{ct} = (c_0, c_1, \ldots, c_n) \in R_{Q_\ell}^{n+1}$ and its associated secret keys $\mathsf{sk}_i = s_i$, return $\mu = c_0 + \sum_{1 \le i \le n} c_i \cdot s_i \pmod{Q_\ell}$.

- <u>`MK-CKKS.Add`$(\mathsf{ct}, \mathsf{ct}')$</u>: Given two ciphertexts $\mathsf{ct}, \mathsf{ct}' \in R_{Q_\ell}^{n+1}$, output $\mathsf{ct}_{add} = \mathsf{ct} + \mathsf{ct}' \pmod{Q_\ell}$.

- <u>`MK-CKKS.Mult`$(\mathsf{ct}, \mathsf{ct}'; \{\mathsf{pk}_i\}_{1 \le i \le n})$</u>: Given two input ciphertexts $\mathsf{ct} = (c_i)_{0 \le i \le n}$, $\mathsf{ct}' = (c'_i)_{0 \le i \le n} \in R_{Q_\ell}^{n+1}$ and their associated public keys $\{\mathsf{pk}_i\}_{1 \le i \le n}$, compute $\mathsf{ct}_{mul} = (c_{i,j})_{0 \le i,j \le n}$ where $c_{i,j} = c_i c'_j \pmod{Q_\ell}$. Run Alg. 1 with $(\mathsf{ct}_{mul}, \{\mathsf{pk}_i\}_{1 \le i \le n})$, and output the result ciphertext ct^*.

- <u>`MK-CKKS.Rescale`(ct)</u>: Given a ciphertext $\mathsf{ct} = (c_0, c_1, \ldots, c_n) \in R_{Q_\ell}^{n+1}$, output $\mathsf{ct}' = (c'_0, c'_1, \ldots, c'_n) \in R_{Q_{\ell-1}}^{n+1}$ where $c'_i = \lfloor q_\ell^{-1} \cdot c_i \rceil \pmod{Q_{\ell-1}}$ for $0 \le i \le n$.

Similar to the original CKKS scheme, a tensored ciphertexts ct_{mul} is obtained by tensor product of two input ciphertexts so that it satisfies a quadratic equation $\sum_{0 \le i \le j \le n} c_{i,j} \cdot s_i s_j = (c_0 + c_1 s_1 + \cdots + c_n s_n)(c'_0 + c'_1 s_1 + \cdots + c'_n s_n) \pmod{Q_\ell}$. Then, the relinearization procedure (Alg. 1) converts ct_{mul} into the ciphertext ct^* in standard form, while almost preserving the underlying plaintext.

3.3 Multi-Key BFV

- <u>`MK-BFV.Setup`(1^λ)</u>: Let the RLWE dimension be N, the ciphertext modulus be $Q = \prod_{i=0}^{L} q_i$ for some integers q_i, the key distribution be χ over R and the error parameter be $\sigma > 0$. Let $h : R_Q \to R^k$ be a gadget decomposition corresponding to a gadget vector $\mathbf{g} \in R_Q^k$. Let $t \in \mathbb{Z}$ be the plaintext modulus. Sample a CRS $\mathbf{a} \leftarrow \mathcal{U}(R_Q^k)$, and output a public parameter $\mathsf{pp} = (N, t, Q, \chi, \sigma, \mathbf{a}, h, \mathbf{g})$.

- <u>`MK-BFV.KeyGen(pp)`</u>: Return a secret key $\mathsf{sk} = s$ and a public key $\mathsf{pk} = (\mathbf{b}, \mathbf{d}, \mathbf{u}, \mathbf{v})$ generated as follows:
  - Sample $s \leftarrow \chi$.
  - Sample $\mathbf{e}_0 \leftarrow D_\sigma^k$ and let $\mathbf{b} = -s \cdot \mathbf{a} + \mathbf{e}_0 \pmod{Q}$.
  - Sample $r \leftarrow \chi$ and $\mathbf{e}_1 \leftarrow D_\sigma^k$. Let $\mathbf{d} = -r \cdot \mathbf{a} + s \cdot \mathbf{g} + \mathbf{e}_1 \pmod{Q}$.
  - Sample $\mathbf{u} \leftarrow \mathcal{U}(R_Q^k)$ and $\mathbf{e}_2 \leftarrow D_\sigma^k$. Let $\mathbf{v} = -s \cdot \mathbf{u} - r \cdot \mathbf{g} + \mathbf{e}_2 \pmod{Q}$.

- <u>`MK-BFV.Enc`$(m; \mathsf{ek})$</u>: Sample $w \leftarrow \chi$ and $e_0, e_1 \leftarrow D_\sigma$. Given a message $m \in R_t$, output the ciphertext $\mathsf{ct} = w \cdot \mathsf{ek} + (\lfloor (Q/t) \cdot m \rceil + e_0, e_1) \pmod{Q}$.

- <u>`MK-BFV.Dec`$(\mathsf{ct}; \{\mathsf{sk}_i\}_{1 \le i \le n})$</u>: Given a ciphertext $\mathsf{ct} = (c_0, c_1, \ldots, c_n) \in R_Q^{n+1}$ and its associated secret keys $\{\mathsf{sk}_i\}_{1 \le i \le n}$, return the plaintext $m = \left\lfloor (t/Q) \cdot (c_0 + \sum_{1 \le i \le n} c_j \cdot s_j) \right\rceil \pmod{t}$.

- <u>`MK-BFV.Add`$(\mathsf{ct}, \mathsf{ct}')$</u>: Given two ciphertexts $\mathsf{ct}, \mathsf{ct}' \in R_Q^{n+1}$, output $\mathsf{ct}_{add} = \mathsf{ct} + \mathsf{ct}' \pmod{Q}$.

Algorithm 2 Simplified relinearization [26]

```
Input: A MKHE ciphertext $\mathsf{ct}_{mul} = (c_{i,j})_{0 \le i \le j \le n}$ with each $c_{i,j} \in R_{Q_\ell}$, and its associated public keys
    $\{\mathsf{pk}_i = (\mathbf{b}_i, \mathbf{d}_i, \mathbf{u}_i, \mathbf{v}_i)\}_{1 \le i \le n}$
Output: $\mathsf{ct}^* = (c_i^*)_{0 \le i \le n} \in R_{Q_\ell}^{n+1}$
 1: $c_0^* \leftarrow c_{0,0}$
 2: for $1 \le i \le n$ do
 3:     $c_i^* \leftarrow c_{0,i} + c_{i,0} \pmod{Q_\ell}$
 4: end for
 5: for $1 \le j \le n$ do
 6:     $c_j^* \leftarrow c_j^* + \sum_{1 \le i \le n} c_{i,j} \boxdot \mathbf{d}_i \pmod{Q_\ell}$
 7: end for
 8: for $1 \le i \le n$ do
 9:     $x_i \leftarrow \sum_{1 \le j \le n} c_{i,j} \boxdot \mathbf{b}_j \pmod{Q_\ell}$
10:     $(c_0^*, c_i^*) \leftarrow (c_0^*, c_i^*) + x_i \boxdot (\mathbf{v}_i, \mathbf{u}_i) \pmod{Q_\ell}$
11: end for
```

• $\underline{\texttt{MK-BFV.Mult}(\mathsf{ct}, \mathsf{ct}'; \{\mathsf{pk}_i\}_{1 \le i \le n})}$: Given two ciphertexts $\mathsf{ct} = (c_i)_{0 \le i \le n}$, $\mathsf{ct}' = (c'_i)_{0 \le i \le n} \in R_Q^{n+1}$ and their associated public keys $\{\mathsf{pk}_i\}_{1 \le i \le n}$, compute $\mathsf{ct}_{mul} = (c_{i,j})_{0 \le i,j \le n} \pmod{Q}$ where $c_{i,j} = \lfloor (t/Q) \cdot c_i c'_j \rceil \pmod{Q}$. Run Alg. 1 with $(\mathsf{ct}_{mul}, \{\mathsf{pk}_i\}_{1 \le i \le n})$, and output the result ct^*.

For the multi-key BFV scheme, a tensored ciphertext ct_{mul} is obtained by tensor product of two input ciphertexts, scaled by (t/Q). We remark that if two input ciphertexts $\mathsf{ct} = (c_i)_{0 \le i \le n}$ and $\mathsf{ct}' = (c'_i)_{0 \le i \le n}$ satisfy $c_0 + c_1 s_1 + \cdots + c_n s_n \approx (Q/t) \cdot \mu \pmod{Q}$ and $c'_0 + c'_1 s_1 + \cdots + c'_n s_n \approx (Q/t) \cdot \mu' \pmod{Q}$, then $\mathsf{ct}_{mul} = (c_{i,j})_{0 \le i,j \le n}$ satisfies $\sum_{0 \le i,j \le n} c_{i,j} \cdot s_i s_j \approx (t/Q) \cdot (c_0 + c_1 s_1 + \cdots + c_n s_n)(c'_0 + c'_1 s_1 + \cdots + c'_n s_n) \approx (Q/t) \cdot \mu\mu' \pmod{Q}$. The same relinearization procedure (Alg. 1) is followed to transform it into the standard form.

3.4 Simplified Relinearization

Recently, Kwak et al. [26] proposed an improved relinearization procedure that reduces the number of external products. They observed that for a fixed i, the external products $c'_{i,j} \boxdot (\mathbf{v}_i, \mathbf{u}_i)$ are used to update the same components c_0^* and c_i^* (Line 7 in Alg. 1), so it is possible to reduce the number of external products by computing $x_i = \sum_{1 \le j \le n} c_{i,j} \boxdot \mathbf{b}_j$ and $x_i \boxdot (\mathbf{v}_i, \mathbf{u}_i)$ instead of $(c_{i,j} \boxdot \mathbf{b}_j) \boxdot (\mathbf{v}_i, \mathbf{u}_i)$ for all $1 \le i, j \le n$. Their algorithm is described in Alg. 2. Nevertheless, the asymptotic complexity for gadget decompositions and ring multiplications still remains at $O(n^2)$.

4 Homomorphic Gadget Decomposition

To linearize the computational complexity of homomorphic multiplication algorithms for the multi-key CKKS and BFV schemes, we need to linearize the required number of gadget decompositions and ring multiplications with respect to the number of associated keys.

In this section, we show how to linearize the complexity of gadget decompositions by introducing a new concept, which we call the *homomorphic gadget decomposition*. Roughly speaking, we consider the homomorphic property that enables us to generate a valid decomposition of $c_{i,j}$ from the decompositions of c_i and c'_j. This property allows us to derive decompositions of $(c_{i,j})_{1 \le i,j \le n}$ with decompositions of $(c_i)_{1 \le i \le n}$ and $(c'_i)_{1 \le i \le n}$, reducing the number of required gadget decompositions from $O(n^2)$ to $O(n)$.

4.1 Definition

As discussed above, gadget decompositions induce quadratic complexity in multi-key homomorphic multiplication, but it seems unlikely that we can get $h(c_{i,j})$ directly without computing $c_{i,j} = c_i \cdot c'_j$ first since the gadget decomposition has no algebraic structure in general. Our key observation is that we do not always have to compute the exact gadget decomposition but it suffices to find any *valid decomposition* satisfying certain conditions. In other words, the correctness of external product (1) still holds even if we replace $h(a)$ by another vector in the inverse image $g^{-1}(a)$ with a reasonably small size.

In this context, we propose a new concept, called *homomorphic gadget decomposition*, where some operations can be performed over the decomposed polynomials. Briefly speaking, if a gadget decomposition h is homomorphic, then we can operate between $h(a)$ and $h(b)$ for any $a, b \in R_Q$ to obtain reasonably short vectors which are not equal to but can substitute $h(a + b)$ and $h(ab)$.

Definition 6. *A homomorphic gadget decomposition $h : R_Q \to R^k$ is a gadget decomposition which satisfies that*

$$\langle h(a) + h(b), \mathbf{g} \rangle = a + b \pmod{Q},$$
$$\langle h(a) \circ h(b), \mathbf{g} \rangle = ab \pmod{Q}$$

for all $a, b \in R_Q$ where $\circ$ denotes the component-wise product of two vectors.

If $h : R_Q \to R^k$ is a homomorphic gadget decomposition, then $h(a) + h(b)$ and $h(a) \circ h(b)$ are elements of $g^{-1}(a + b)$ and $g^{-1}(ab)$ which are bounded by $\|h(a) + h(b)\|_\infty \le 2B_h$ and $\|h(a) \circ h(b)\|_\infty \le N \cdot B_h^2$, respectively. The first additive condition is always true for any gadget decompositions, but the multiplicative property is not in general. Fortunately, the most widely used gadget decomposition technique in state-of-the-art HE libraries [11, 30] is homomorphic, which we will later introduce. We also point out that a homomorphic gadget decomposition is not necessarily a ring homomorphism in a mathematical manner, but we nevertheless call it 'homomorphic' to represent its property (similarly to the terminology of HE).

4.2 Implications

We now briefly describe how homomorphic gadget decomposition can reduce the number of gadget decompositions required in the relinearization step of multi-key CKKS. In the simplified version of CDKS relinearization (Alg. 2), a quadratic number of gadget decompositions is required due to the computation of $h(c_{i,j})$ in Line 6 and 9. This is where homomorphic gadget decomposition can be applied. Note that $c_{i,j} = c_i c'_j \pmod{Q_\ell}$ holds in multiplication of multi-key CKKS ciphertexts. If h is a homomorphic gadget decomposition, the decomposition $h(c_{i,j})$ can be substituted with $h(c_i) \circ h(c'_j)$ under this assumption. Thus, if we precompute $h(c_i)$ and $h(c'_j)$ for $1 \le i, j \le n$, we can reduce the number of gadget decompositions from $O(n^2)$ to $O(n)$ since it only requires a component-wise product of $h(c_i)$ and $h(c'_j)$ to derive $h(c_i) \circ h(c'_j)$.

Even with the aforementioned optimization technique, achieving linear computational complexity for homomorphic multiplication in both CKKS and BFV schemes is still challenging. The main issue is that, while homomorphic gadget decomposition can significantly reduce the number of required gadget decompositions, the number of ring multiplications remains $O(n^2)$ in both schemes. This is because we need to compute $O(n^2)$ ring multiplications for a tensored ciphertext $(c_{i,j})_{0 \le i,j \le n}$, as well as external products $c_{i,j} \boxdot \mathbf{d}_i$ and $c_{i,j} \boxdot \mathbf{b}_j$ for all $1 \le i, j \le n$. Moreover, applying homomorphic gadget decomposition to multi-key BFV multiplication is not straightforward as the components $c_{i,j}$ of a tensored ciphertext are obtained by $\lfloor (t/Q) c_i c'_j \rceil \pmod{Q}$, and thus the property of homomorphic gadget decomposition cannot be directly applied to these components.

In the next section, we explain how we address these issues and redesign homomorphic multiplication algorithm so that it achieves linear complexity with respect to the number of keys.

5 New Multi-Key Variants of CKKS and BFV

In this section, we present new multiplication algorithms for multi-key CKKS and BFV that achieve linear complexity with respect to the number of associated keys via homomorphic gadget decomposition. As mentioned before, homomorphic gadget decomposition can reduce the number of gadget decompositions to $O(n)$ for the relinearization of tensored multi-key CKKS ciphertexts. However, several issues still remain. First, the number of ring multiplications remains $O(n^2)$ due to the tensor-then-relinearize pipeline of the previous multiplication algorithm, which requires computing a tensored ciphertext as a middle-product. Additionally, it is unclear how to apply homomorphic gadget decomposition for tensored multi-key BFV ciphertexts.

To address these issues, we first redesign the entire multiplication algorithm so that it fully utilizes the homomorphic properties of homomorphic gadget decomposition. Specifically, we merge the two steps of tensoring and relinearization and rearrange each operation using homomorphic properties to achieve $O(n)$ complexity for both gadget decompositions and ring multiplications. For multi-key BFV, we observe that we can still utilize homomorphic gadget decompositions for tensored BFV ciphertexts if we modify the public key structure. As a result, we can also achieve linear complexity for multi-key BFV multiplication.

5.1 Improved Multi-Key CKKS

We first describe how we modified the previous multiplication algorithm to achieve $O(n)$ complexity for the multi-key CKKS scheme using homomorphic gadget decomposition. Recall that each component of a tensored ciphertext is obtained by $c_{i,j} = c_i c'_j \pmod{Q_\ell}$. Then, $h(c_{i,j})$ is replaced by $h(c_i) \circ h(c'_j)$ if h is a homomorphic gadget decomposition. As a result, we can replace the terms $c_{i,j} \boxdot \mathbf{d}_i = \langle h(c_{i,j}), \mathbf{d}_i \rangle$ and $c_{i,j} \boxdot \mathbf{b}_j = \langle h(c_{i,j}), \mathbf{b}_j \rangle$ with $\langle h(c_i) \circ h(c_j), \mathbf{d}_i \rangle$ and $\langle h(c_i) \circ h(c_j), \mathbf{b}_j \rangle$, respectively. Since the order of component-wise product can be switched without affecting the correctness of the inner product, we can also replace them with $\langle h(c_i) \circ h(c_j), \mathbf{d}_i \rangle$ and $\langle h(c_i) \circ h(c_j), \mathbf{b}_j \rangle$ along with $\langle h(c_j), h(c_i) \circ \mathbf{d}_i \rangle$ and $\langle h(c_i), h(c_j) \circ \mathbf{b}_j \rangle$, respectively. Therefore, we can substitute $\sum_{1 \le i \le n} c_{i,j} \boxdot \mathbf{d}_i$ and $\sum_{1 \le j \le n} c_{i,j} \boxdot \mathbf{b}_j$ in Line 6 and 9 of Alg 2 as follows:

$$\sum_{1 \le i \le n} \langle h(c_i) \circ h(c'_j), \mathbf{d}_i \rangle = \left\langle h(c'_j), \sum_{1 \le i \le n} h(c_i) \circ \mathbf{d}_i \right\rangle = c'_j \boxdot \left(\sum_{1 \le i \le n} h(c_i) \circ \mathbf{d}_i \right) \tag{3}$$

$$\sum_{1 \le j \le n} \langle h(c_i) \circ h(c'_j), \mathbf{b}_j \rangle = \left\langle h(c_i), \sum_{1 \le j \le n} h(c'_j) \circ \mathbf{b}_j \right\rangle = c_i \boxdot \left(\sum_{1 \le j \le n} h(c'_j) \circ \mathbf{b}_j \right) \tag{4}$$

Note that $\sum_{1 \le i \le n} h(c_i) \circ \mathbf{d}_i$ is independent of the index j. Thus, once it is precomputed, we can reuse it for computing the external products $c'_j \boxdot \left(\sum_{1 \le i \le n} h(c_i) \circ \mathbf{d}_i \right)$ for different $1 \le j \le n$. Similarly, we can reuse the precomputed $\sum_{1 \le j \le n} h(c'_j) \circ \mathbf{b}_j$ for computing $c_i \boxdot \left(\sum_{1 \le j \le n} h(c'_j) \circ \mathbf{b}_j \right)$ for different $1 \le i \le n$. This optimization not only reduces the number of gadget decompositions but also the number of ring multiplications from $O(n^2)$ to $O(n)$. A homomorphic multiplication algorithm based on this optimization is presented in Alg. 3. We now provide a formal description of our multi-key CKKS construction based on this optimization. We mainly modify the setup and multiplication algorithms while keeping the other algorithms the same as before.

• <u>`MK-CKKS.Setup`(1^λ)</u>: Let the RLWE dimension be N, the ciphertext modulus be $Q = \prod_{i=0}^{L} q_i$ for some integers q_i, the key distribution be χ over R and the error parameter be $\sigma > 0$. Let $h : R_Q \to R^k$ be a *homomorphic* gadget decomposition corresponding to a gadget vector $\mathbf{g} \in R_Q^k$. Sample a CRS $\mathbf{a} \leftarrow \mathcal{U}(R_Q^k)$, and output a public parameter $\mathsf{pp} = (N, Q, \chi, \sigma, \mathbf{a}, h, \mathbf{g})$.

• <u>`MK-CKKS.KeyGen(pp)`</u>: Return a secret key $\mathsf{sk} = s$ and a public key $\mathsf{pk} = (\mathbf{b}, \mathbf{d}, \mathbf{u}, \mathbf{v})$ generated as follows:

– Sample $s \leftarrow \chi$.
– Sample $\mathbf{e}_0 \leftarrow D_\sigma^k$ and let $\mathbf{b} = -s \cdot \mathbf{a} + \mathbf{e}_0 \pmod{Q}$.
– Sample $r \leftarrow \chi$ and $\mathbf{e}_1 \leftarrow D_\sigma^k$. Let $\mathbf{d} = -r \cdot \mathbf{a} + s \cdot \mathbf{g} + \mathbf{e}_1 \pmod{Q}$.
– Sample $\mathbf{u} \leftarrow \mathcal{U}(R_Q^k)$ and $\mathbf{e}_2 \leftarrow D_\sigma^k$. Let $\mathbf{v} = -s \cdot \mathbf{u} - r \cdot \mathbf{g} + \mathbf{e}_2 \pmod{Q}$.

Algorithm 3 New multi-key CKKS multiplication algorithm

```
Input: $\mathsf{ct} = (c_i)_{0\le i\le n} \in R_{Q_\ell}^{n+1}$, $\mathsf{ct}' = (c'_i)_{0\le i\le n} \in R_{Q_\ell}^{n+1}$ , $\{\mathsf{pk}_i = (\mathbf{b}_i, \mathbf{d}_i, \mathbf{u}_i, \mathbf{v}_i)\}_{1\le i\le n}$
Output: $\mathsf{ct}^* = (c^*_i)_{0\le i\le n} \in R_{Q_\ell}^{n+1}$
 1: $c^*_0 \leftarrow c_0 \cdot c'_0 \pmod{Q_\ell}$
 2: for $1 \le i \le n$ do
 3:     $c^*_i \leftarrow c_0 \cdot c'_i + c_i \cdot c'_0 \pmod{Q_\ell}$
 4: end for
 5: $\mathbf{z} \leftarrow \sum_{1\le i\le n} h(c_i) \circ \mathbf{d}_i \pmod{Q_\ell}$
 6: $\mathbf{w} \leftarrow \sum_{1\le j\le n} h(c'_j) \circ \mathbf{b}_j \pmod{Q_\ell}$
 7: for $1 \le j \le n$ do
 8:     $c^*_j \leftarrow c^*_j + c'_j \boxdot \mathbf{z} \pmod{Q_\ell}$
 9: end for
10: for $1 \le i \le n$ do
11:     $(c^*_0, c^*_i) \leftarrow (c^*_0, c^*_i) + (c_i \boxdot \mathbf{w}) \boxdot (\mathbf{v}_i, \mathbf{u}_i) \pmod{Q_\ell}$
12: end for
```

We denote the encryption key by $\mathsf{ek} = (\mathbf{b}[0], \mathbf{a}[0]) \in R_Q^2$.

• $\underline{\texttt{MK-CKKS.Enc}(\mu; \mathsf{ek})}$: Sample $w \leftarrow \chi$ and $e_0, e_1 \leftarrow D_\sigma$. Given a plaintext $\mu \in R$, output the ciphertext $\mathsf{ct} = w \cdot \mathsf{ek} + (\mu + e_0, e_1) \pmod{Q}$.

• $\underline{\texttt{MK-CKKS.Dec}(\mathsf{ct}; \{\mathsf{sk}_i\}_{1\le i\le n})}$: Given a ciphertext $\mathsf{ct} = (c_0, c_1, \ldots, c_n) \in R_{Q_\ell}^{n+1}$ and its associated secret keys $\mathsf{sk}_i = s_i$, return $\mu = c_0 + \sum_{1\le i\le n} c_i \cdot s_i \pmod{Q_\ell}$.

• $\underline{\texttt{MK-CKKS.Add}(\mathsf{ct}, \mathsf{ct}')}$: Given two ciphertexts $\mathsf{ct}, \mathsf{ct}' \in R_{Q_\ell}^{n+1}$, output $\mathsf{ct}_{add} = \mathsf{ct} + \mathsf{ct}' \pmod{Q_\ell}$.

• $\underline{\texttt{MK-CKKS.Mult}(\mathsf{ct}, \mathsf{ct}'; \{\mathsf{pk}_i\}_{1\le i\le n})}$: Given two ciphertexts $\mathsf{ct} = (c_i)_{0\le i\le n}$, $\mathsf{ct}' = (c'_i)_{0\le i\le n} \in R_{Q_\ell}^{n+1}$ and their associated public keys $\{\mathsf{pk}_i\}_{1\le i\le n}$, execute Alg. 3 and return the output ciphertext ct^*.

• $\underline{\texttt{MK-CKKS.Rescale}(\mathsf{ct})}$: Given a ciphertext $\mathsf{ct} = (c_0, c_1, \ldots, c_n) \in R_{Q_\ell}^{n+1}$, output $\mathsf{ct}' = (c'_0, c'_1, \ldots, c'_n) \in R_{Q_{\ell-1}}^{n+1}$ where $c'_i = \lfloor q_\ell^{-1} \cdot c_i \rceil \pmod{Q_{\ell-1}}$ for $0 \le i \le n$.

Note that our multiplication algorithm no longer follows the previous pipeline where the tensor product and relinearization are performed sequentially; instead, it performs both operations simultaneously. Below, we present an analysis of the security, correctness, and noise growth of our proposed scheme.

Security. The security proof is quite similar to that of CDKS since our scheme shares almost the same key generation algorithm but just requires to take a gadget decomposition with homomorphic property.

First of all, the proposed cryptosystem is semantically secure under the RLWE assumption of parameter (N, Q, χ, σ) since $(\mathbf{b}, \mathbf{a})$ is an RLWE instance and the standard RLWE encryption is used in our construction. However, this security proof relies on an implicit assumption that the proposed scheme remains secure even if the public key is given to the adversary.

To complete the security proof, we claim that the distribution of a public key $\mathsf{pk}_i = (\mathbf{b}_i, \mathbf{d}_i, \mathbf{u}_i, \mathbf{v}_i)$ is indistinguishable from a uniform distribution over $R_Q^{k\times 4}$. It can be shown under the same RLWE assumption of parameter (N, Q, χ, σ) since $(\mathbf{v}_i, \mathbf{u}_i)$ and $(\mathbf{d}_i, \mathbf{a})$ are gadget encryptions of $-r_i$ and s_i under secrets s_i and r_i, respectively. We also make a circular security assumption (which is standard in HE construction) since $(\mathbf{d}_i, \mathbf{a})$ and $(\mathbf{v}_i, \mathbf{u}_i)$ form a chain of gadget encryptions related to secrets s_i and r_i.

Correctness. We focus on the correctness of our new multiplication algorithm. Suppose that $\mathsf{ct} = (c_0, c_1, \ldots, c_n)$ and $\mathsf{ct}' = (c'_0, c'_1, \ldots, c'_n)$ are multi-key ciphertexts under a tuple of secrets $(s_1, \ldots, s_n)$. Our goal is to show that the result of multiplication $\mathsf{ct}^* = (c^*_i)_{0\le i\le n} \leftarrow \texttt{MK-CKKS.Mult}(\{\mathsf{pk}_i\}_{1\le i\le n}; \mathsf{ct}, \mathsf{ct}')$ satisfies $c^*_0 + \sum_{i=1}^n c^*_i s_i \approx (c_0 + \sum_{i=1}^n c_i s_i)(c'_0 + \sum_{i=1}^n c'_i s_i) \pmod{Q_\ell}$.

First of all, we have

$$c_0^* + \sum_{1\le i\le n} c_i^* \cdot s_i = c_0 \cdot c_0' + \sum_{1\le i\le n} (c_0 \cdot c_i' + c_i \cdot c_0') \cdot s_i$$
$$+ \sum_{1\le j\le n} (c_j' \boxdot \mathbf{z}) \cdot s_j + \sum_{1\le i\le n} (c_i \boxdot \mathbf{w}) \boxdot (\mathbf{v}_i + s_i \cdot \mathbf{u}_i) \pmod{Q_\ell}.$$

from Alg. 3.

Using the facts that $s_j \cdot \mathbf{d}_i \approx -r_i s_j \cdot \mathbf{a} + s_i s_j \cdot \mathbf{g} \approx r_i \cdot \mathbf{b}_j + s_i s_j \cdot \mathbf{g} \pmod{Q}$ and h is homomorphic, the third term can be written as follows:

$$\begin{aligned}\sum_{1\le j\le n} (c_j' \boxdot \mathbf{z}) \cdot s_j &= \sum_{1\le j\le n} \left(c_j' \boxdot \sum_{1\le i\le n} (h(c_i) \circ \mathbf{d}_i) \right) \cdot s_j \\ &= \sum_{1\le i,j\le n} \left\langle h(c_j'), h(c_i) \circ \mathbf{d}_i \right\rangle \cdot s_j = \sum_{1\le i,j\le n} \left\langle h(c_i) \circ h(c_j'), \mathbf{d}_i \right\rangle \cdot s_j \\ &\approx \sum_{1\le i,j\le n} r_i \cdot \left\langle h(c_i) \circ h(c_j'), \mathbf{b}_j \right\rangle + \sum_{1\le i,j\le n} c_i c_j' \cdot s_i s_j \pmod{Q_\ell}. \end{aligned} \tag{5}$$

From $\mathbf{v}_i + s_i \cdot \mathbf{u}_i \approx r_i \cdot \mathbf{g} \pmod{Q}$, the fourth term is simplified as

$$\begin{aligned}\sum_{1\le i\le n} (c_i \boxdot \mathbf{w}) \boxdot (\mathbf{v}_i + s_i \cdot \mathbf{u}_i) &\approx - \sum_{1\le i\le n} r_i \cdot (c_i \boxdot \mathbf{w}) \\ &= - \sum_{1\le i\le n} r_i \cdot \left(c_i \boxdot \sum_{1\le j\le n} (h(c_j') \circ \mathbf{b}_j) \right) \\ &= - \sum_{1\le i,j\le n} r_i \cdot \left\langle h(c_i), h(c_j') \circ \mathbf{b}_j \right\rangle \\ &= - \sum_{1\le i,j\le n} r_i \cdot \left\langle h(c_i) \circ h(c_j'), \mathbf{b}_j \right\rangle \pmod{Q_\ell}. \end{aligned} \tag{6}$$

Putting it all together, we obtain $c_0^* + \sum_{1\le i\le n} c_i^* \cdot s_i \approx c_0 c_0' + \sum_{1\le i\le n} (c_0 c_i' + c_i c_0') \cdot s_i + \sum_{1\le i,j\le n} c_i c_j' \cdot s_i s_j = (c_0 + \sum_{i=1}^n c_i s_i)(c_0' + \sum_{i=1}^n c_i' s_i) \pmod{Q_\ell}$ which completes the correctness proof of our multiplication algorithm.

Noise Analysis. For the noise analysis, we provide a worst-case bound of the multiplication noise of our scheme. We refer the reader to App. A.1 for a tighter average-case analysis based on the noise variance. Note that an output ciphertext ct^* satisfies

$$c_0^* + \sum_{1\le i\le n} c_i^* \cdot s_i = \left(c_0 + \sum_{i=1}^n c_i s_i \right) \left(c_0' + \sum_{i=1}^n c_i' s_i \right) + e_1 + e_2 \pmod{Q_\ell}$$

where e_1 and e_2 are the errors from approximate equalities in (5) and (6), respectively. To be precise, these error terms can be written as

$$e_1 = \sum_{1\le i,j\le n} \left\langle h(c_i) \circ h(c_j'), s_j \cdot \mathbf{e}_{1,i} - r_i \cdot \mathbf{e}_{0,j} \right\rangle,$$
$$e_2 = \sum_{1\le i\le n} (c_i \boxdot \mathbf{w}) \boxdot \mathbf{e}_{2,i}$$

which are bounded by $\|e_1\|_\infty \le 2kn^2N^3 \cdot B_h^2 B_\sigma$ and $\|e_2\|_\infty \le knN \cdot B_h B_\sigma$. As a result, we get a worst-case bound $2kn^2N^3 \cdot B_h^2 B_\sigma + knN \cdot B_h B_\sigma \approx 2kn^2N^3 \cdot B_h^2 B_\sigma$ of the multiplication noise.

We note that our new multiplication algorithm yields slightly larger noise whose bound is about $N \cdot B_h$ times larger than that of the previous method. This extra factor comes from the component-wise product of gadget decompositions $h(c_i) \circ h(c_j')$ that substitutes $h(c_{i,j})$. However, this issue can be easily addressed using the *special modulus* method (see App. B for details).

Algorithm 4 New multi-key BFV multiplication algorithm

```
Input: $\mathsf{ct} = (c_i)_{0 \le i \le n} \in R_Q^{n+1}$, $\mathsf{ct}' = (c'_i)_{0 \le i \le n} \in R_Q^{n+1}$, $\{\mathsf{pk}_i = (\mathbf{b}_i, \mathbf{d}_i, \mathbf{u}_i, \mathbf{v}_i)\}_{1 \le i \le n}$
Output: $\mathsf{ct}^* = (c^*_i)_{0 \le i \le n} \in R_Q^{n+1}$
 1: $c^*_0 \leftarrow \lfloor (t/Q) \cdot (c_0 c'_0) \rceil \pmod Q$
 2: for $1 \le i \le n$ do
 3:     $c^*_i \leftarrow \lfloor (t/Q) \cdot (c_0 c'_i + c_i c'_0) \rceil \pmod Q$
 4: end for
 5: $\mathbf{z} \leftarrow \sum_{1 \le i \le n} \tilde{h}(c_i) \circ \mathbf{d}_i \pmod Q$
 6: $\mathbf{w} \leftarrow \sum_{1 \le j \le n} \tilde{h}(c'_j) \circ \mathbf{b}_j \pmod Q$
 7: for $1 \le j \le n$ do
 8:     $c^*_j \leftarrow c^*_j + c'_j \,\tilde{\boxdot}\, \mathbf{z} \pmod Q$
 9: end for
10: for $1 \le i \le n$ do
11:     $(c^*_0, c^*_i) \leftarrow (c^*_0, c^*_i) + (c_i \,\tilde{\boxdot}\, \mathbf{w}) \boxdot (\mathbf{v}_i, \mathbf{u}_i) \pmod Q$
12: end for
```

5.2 Improved Multi-Key BFV

Compared to the multi-key CKKS scheme, homomorphic gadget decomposition is not directly applicable to the multi-key BFV case. This is due to the structure of tensored ciphertexts $(c_{i,j})_{1 \le i,j \le n}$, where $c_{i,j} = \lfloor (t/Q) \cdot c_i c'_j \rceil \pmod Q$, and the product of c_i and c'_j is performed in R instead of R_Q. To cope with this issue, we first instantiate R as $R_{\tilde{Q}}$ where $\tilde{Q} := Q^2$. Then, we observe the following property for a homomorphic gadget decomposition $\tilde{h} : R_{\tilde{Q}} \to R^{\tilde{k}}$ with a gadget vector $\tilde{\mathbf{g}} \in R_{\tilde{Q}}^{\tilde{k}}$

$$\begin{aligned} \left\langle \tilde{h}(c_i) \circ \tilde{h}(c_j), t \cdot \tilde{\mathbf{g}} \right\rangle &= t \cdot c_i c'_j \approx Q \cdot c_{i,j} \pmod{\tilde{Q}} \\ \Longrightarrow \left\langle \tilde{h}(c_i) \circ \tilde{h}(c_j), \lfloor (t/Q) \cdot \tilde{\mathbf{g}} \rceil \right\rangle &\approx c_{i,j} \pmod Q \end{aligned} \tag{7}$$

where c_i and c'_j in the above equations are regarded as elements of $R_{\tilde{Q}}$ via the natural embedding $R_Q \hookrightarrow R_{\tilde{Q}}$. Using the above property, we can rewrite the external products $c_{i,j} \boxdot \mathbf{d}_i$ and $c_{i,j} \boxdot \mathbf{b}_j$, which are associated with $c_{i,j}$, in a way that exploits homomorphic gadget decomposition, as presented in (3). By doing so, we can achieve linear complexity for multi-key BFV multiplication. Below, we provide a formal description of our new multi-key BFV scheme.

• MK-BFV.Setup(1^λ): Let the RLWE dimension be N, the ciphertext modulus be $Q = \prod_{i=0}^{L} q_i$ for some integers q_i, the key distribution be χ over R and the error parameter be $\sigma > 0$. Let $h : R_Q \to R^k$ be a gadget decomposition corresponding to a gadget vector $\mathbf{g} \in R_Q^k$, and let $\tilde{h} : R_{\tilde{Q}} \to R^{\tilde{k}}$ be a *homomorphic* gadget decomposition corresponding to a gadget vector $\tilde{\mathbf{g}} \in R_{\tilde{Q}}^{\tilde{k}}$. Let $t \in \mathbb{Z}$ be the plaintext modulus. Sample a CRS $\mathbf{a} \leftarrow \mathcal{U}(R_Q^{\tilde{k}})$, and output a public parameter $\mathsf{pp} = (N, t, Q, \chi, \sigma, \mathbf{a}, h, \mathbf{g}, \tilde{h}, \tilde{\mathbf{g}})$.

We denote the external product with respect to the gadget decomposition $\tilde{h}$ by $\tilde{\boxdot}$.

• MK-BFV.KeyGen(pp): Return a secret key $\mathsf{sk} = s$ and a public key $\mathsf{pk} = (\mathbf{b}, \mathbf{d}, \mathbf{u}, \mathbf{v})$ generated as follows:

– Sample $s \leftarrow \chi$.
– Sample $\mathbf{e}_0 \leftarrow D_\sigma^{\tilde{k}}$ and let $\mathbf{b} = -s \cdot \mathbf{a} + \mathbf{e}_0 \pmod Q$.
– Sample $r \leftarrow \chi$ and $\mathbf{e}_1 \leftarrow D_\sigma^{\tilde{k}}$. Let $\mathbf{d} = -r \cdot \mathbf{a} + s \cdot \lfloor (t/Q) \cdot \tilde{\mathbf{g}} \rceil + \mathbf{e}_1 \pmod Q$.
– Sample $\mathbf{u} \leftarrow \mathcal{U}(R_Q^k)$ and $\mathbf{e}_2 \leftarrow D_\sigma^k$. Let $\mathbf{v} = -s \cdot \mathbf{u} - r \cdot \mathbf{g} + \mathbf{e}_2 \pmod Q$.

We denote the encryption key by $\mathsf{ek} = (\mathbf{b}[0], \mathbf{a}[0]) \in R_Q^2$.

• $\underline{\texttt{MK-BFV.Enc}(m;\mathsf{ek})}$: Sample $w \leftarrow \chi$ and $e_0, e_1 \leftarrow D_\sigma$. Given a message $m \in R_t$, output the ciphertext $\mathsf{ct} = w \cdot \mathsf{ek} + (\lfloor (Q/t) \cdot m \rceil + e_0, e_1) \pmod Q$.

• $\underline{\texttt{MK-BFV.Dec}(\mathsf{ct};\{\mathsf{sk}_i\}_{1\le i\le n})}$: Given a ciphertext $\mathsf{ct} = (c_0, c_1, \ldots, c_n) \in R_Q^{n+1}$ and its associated secret keys $\{\mathsf{sk}_i\}_{1\le i\le n}$, return the plaintext $m = \left\lfloor (t/Q) \cdot (c_0 + \sum_{1\le i\le n} c_j \cdot s_j) \right\rceil \pmod t$.

• $\underline{\texttt{MK-BFV.Add}(\mathsf{ct},\mathsf{ct}')}$: Given two ciphertexts $\mathsf{ct}, \mathsf{ct}' \in R_Q^{n+1}$, output $\mathsf{ct}_{add} = \mathsf{ct} + \mathsf{ct}' \pmod Q$.

• $\underline{\texttt{MK-BFV.Mult}(\mathsf{ct},\mathsf{ct}';\{\mathsf{pk}_i\}_{1\le i\le n})}$: Given two ciphertexts $\mathsf{ct} = (c_i)_{0\le i\le n}$, $\mathsf{ct}' = (c'_i)_{0\le i\le n} \in R_Q^{n+1}$ and associated public keys $\{\mathsf{pk}_i\}_{1\le i\le n}$, run Alg. 4 and return the ciphertext $\mathsf{ct}^* = (c^*_i)_{0\le i\le n} \in R_Q^{n+1}$.

We assume that the entries of the input ciphertexts ct and ct' are embedded into $R_{\tilde{Q}}$ in Lines 5–6 of Alg. 4 so that they can be taken as input for the gadget decomposition $\tilde{h}$, even if it is not explicitly mentioned. We also note that the public key structure is modified so that we can utilize the property presented in (7). Below, we provide security, correctness, and noise analysis of the proposed scheme.

Security. Similar to the case of CKKS, our multi-key BFV scheme is IND-CPA secure under the RLWE assumption of parameter (N, Q, χ, σ) since it uses the standard BFV encryption algorithm. It also requires a circular security assumption since $(\mathbf{d}, \mathbf{a})$ and $(\mathbf{v}_i, \mathbf{u}_i)$ form a chain of encryptions of $s_i \cdot \lfloor (t/Q) \cdot \tilde{\mathbf{g}} \rceil$ and $-r_i \cdot \mathbf{g}$ under r_i and s_i, respectively.

Correctness. Suppose $\mathsf{ct}^* \leftarrow \texttt{MK-BFV.Mult}(\mathsf{ct},\mathsf{ct}';\{\mathsf{pk}_i\}_{1\le i\le n})$ for some multi-key ciphertexts ct and ct'. Our goal is to show that

$$\left\langle \mathsf{ct}^*, (1, \overline{\mathsf{sk}}) \right\rangle \approx (t/Q) \cdot \sum_{0\le i,j\le n} c_i c'_j \cdot s_i s_j \approx (Q/t) \cdot mm' \pmod Q$$

whenever $\left\langle \mathsf{ct}, (1, \overline{\mathsf{sk}}) \right\rangle \approx (Q/t) \cdot m$ and $\left\langle \mathsf{ct}', (1, \overline{\mathsf{sk}}) \right\rangle \approx (Q/t) \cdot m'$. From Alg. 4, we have

$$\begin{aligned} \left\langle \mathsf{ct}^*, (1, \overline{\mathsf{sk}}) \right\rangle &= c^*_0 + \sum_{1\le i\le n} c^*_i \cdot s_i \\ &= \lfloor (t/Q) \cdot (c_0 c'_0) \rceil + \sum_{1\le i\le n} \lfloor (t/Q) \cdot (c_0 c'_i + c_i c'_0) \rceil \cdot s_i \\ &\quad + \sum_{1\le j\le n} (c'_j \,\tilde{\boxdot}\, \mathbf{z}) \cdot s_j + \sum_{1\le i\le n} (c_i \,\tilde{\boxdot}\, \mathbf{w}) \boxdot (\mathbf{v}_i + s_i \cdot \mathbf{u}_i). \end{aligned}$$

The last two terms satisfy that

$$\begin{aligned} \sum_{1\le j\le n} (c'_j \,\tilde{\boxdot}\, \mathbf{z}) \cdot s_j &= \sum_{1\le j\le n} \left(c'_j \,\tilde{\boxdot} \sum_{1\le i\le n} (\tilde{h}(c_i) \circ \mathbf{d}_i) \right) \cdot s_j \\ = \sum_{1\le i,j\le n} \left\langle \tilde{h}(c'_j), \tilde{h}(c_i) \circ \mathbf{d}_i \right\rangle \cdot s_j &= \sum_{1\le i,j\le n} \left\langle \tilde{h}(c_i) \circ \tilde{h}(c'_j), \mathbf{d}_i \right\rangle \cdot s_j \\ \approx \sum_{1\le i,j\le n} &\left\langle \tilde{h}(c_i) \circ \tilde{h}(c'_j), -r_i \cdot \mathbf{a} + s_i \cdot \lfloor (t/Q) \cdot \tilde{\mathbf{g}} \rceil \right\rangle \cdot s_j \\ \approx \sum_{1\le i,j\le n} &r_i \cdot \left\langle \tilde{h}(c_i) \circ \tilde{h}(c'_j), \mathbf{b}_j \right\rangle + c_{i,j} \cdot s_i s_j \pmod Q, \end{aligned} \tag{8}$$

and

$$\begin{aligned}
\sum_{1\le i\le n} (c_i \,\tilde{\boxdot}\, \mathbf{w}) \boxdot (\mathbf{v}_i + s_i\cdot \mathbf{u}_i) &\approx - \sum_{1\le i\le n} r_i \cdot (c_i \,\tilde{\boxdot}\, \mathbf{w})\\
&= - \sum_{1\le i\le n} r_i \cdot \left\langle \tilde{h}(c_i), \sum_{1\le j\le n} \tilde{h}(c'_j)\circ \mathbf{b}_j \right\rangle\\
&= - \sum_{1\le i\le n} r_i \cdot \left\langle \tilde{h}(c_i), \sum_{1\le j\le n} \tilde{h}(c'_j)\circ \mathbf{b}_j \right\rangle\\
&= - \sum_{1\le i,j\le n} r_i \cdot \left\langle \tilde{h}(c_i)\circ \tilde{h}(c'_j), \mathbf{b}_j \right\rangle \pmod{Q} \qquad (9)
\end{aligned}$$

Therefore, we obtain

$$\begin{aligned}
\left\langle \mathsf{ct}^*, (1, \overline{\mathsf{sk}})\right\rangle \approx &(t/Q)\cdot (c_0c'_0) + \sum_{1\le i\le n} (t/Q)(c_0c'_i + c_ic'_0)\cdot s_i\\
&+ (t/Q)\cdot \sum_{1\le i,j\le n} c_ic'_j \cdot s_is_j \approx (Q/t)\cdot mm' \pmod{Q}
\end{aligned}$$

as desired.

Noise Analysis. For the noise analysis, we provide a worst-case bound for the multiplication noise. For the average-case analysis, we refer the reader to App. A.2. Our focus is on the dominant noise terms that result from external products, while disregarding noise from rounding. The error terms from equations (8) and (9) can be expressed as follows:

$$\begin{aligned}
e_1 &= \sum_{1\le i,j\le n} \left\langle \tilde{h}(c_i)\circ\tilde{h}(c'_j), s_j\cdot \mathbf{e}_{1,i} - r_i\cdot \mathbf{e}_{0,j}\right\rangle\\
e_2 &= \sum_{1\le i\le n} (c_i \,\tilde{\boxdot}\, \mathbf{w}) \boxdot \mathbf{e}_{2,i}.
\end{aligned}$$

Therefore, the total multiplication noise is bounded by $\|e_1\|_\infty + \|e_2\|_\infty \le 2\tilde{k}n^2N^3\cdot B_{\tilde{h}}^2 B_\sigma + knN\cdot B_hB_\sigma$.

5.3 Complexity Analysis

We provide a complexity analysis of our schemes by counting the number of gadget decompositions and ring multiplications in Alg. 3 and 4, as they dominate the overall performance of homomorphic multiplication in both asymptotic and practical manners.

We first analyze the multi-key CKKS multiplication algorithm (Alg. 3). In Line 3, the algorithm performs $2n$ ring multiplications. In Lines 5 and 6, it performs $2kn$ ring multiplications and $2n$ gadget decompositions. In Line 8, only kn ring multiplications are performed since $h(c'_j)$ can be reused from Line 6. Finally, in Line 11, it performs $3kn$ ring multiplications and n gadget decompositions since $h(c_i)$ can be reused from Line 5. In total, Alg. 3 requires $3n$ gadget decompositions over R_{Q_ℓ} and $(6k+2)n \approx 6kn$ multiplications over R_{Q_ℓ}.

Next, we analyze the complexity of multi-key BFV multiplication (Alg. 4). In Line 3, the algorithm performs $2n$ ring multiplications over $R_{\tilde{Q}}$. In Lines 5 and 6, it performs $2\tilde{k}n$ ring multiplications over R_Q and $2n$ gadget decompositions over $R_{\tilde{Q}}$. In Line 8, only $\tilde{k}n$ ring multiplications over R_Q are performed since $\tilde{h}(c'_j)$ can be reused from Line 6. Finally, in Line 11, it performs $(2k+\tilde{k})n$ ring multiplications over R_Q and n gadget decompositions over R_Q since $\tilde{h}(c_i)$ can be reused from Line 5. In total, Alg. 4 requires n gadget decompositions over R_Q, $2n$ gadget decompositions over $R_{\tilde{Q}}$, $2n$ multiplications over $R_{\tilde{Q}}$ and $(2k+4\tilde{k})n$ multiplications over R_Q.

	Scheme	Ring multiplication	Gadget decomposition	
			h	$\tilde{h}$
CDKS [10]	CKKS	$2kn^2$	n^2	-
	BFV	$2kn^2$	n^2	-
Ours	CKKS	$6kn$	$3n$	-
	BFV	$(2k+4\tilde{k})n$	n	$2n$

Table 1. Complexity estimation of CDKS and our multi-key homomorphic multiplications in terms of polynomial multiplication (over R_{Q_ℓ} or R_Q) and gadget decomposition (h or $\tilde{h}$). n denotes the number of associated keys and k and $\tilde{k}$ denotes the dimension of the gadget decompositions h and $\tilde{h}$, respectively.

We summarize the results in Table 1 along with the complexity of the previous multiplication algorithm by CDKS. For CDKS, we applied the aforementioned optimization technique proposed in [10] that reduces the overall computational cost by almost half. For the multi-CKKS scheme, the number of ring multiplications is reduced from $2kn^2$ to $6kn$, and the number of gadget decompositions is reduced from n^2 to $3n$. Thus, it achieves linear complexity with respect to the number of associated keys. For the multi-key BFV scheme, our algorithm introduces a new gadget decomposition $\tilde{h}$ of dimension $\tilde{k}$. Since $\tilde{h}$ is determined by $R_{\tilde{Q}}$ and is independent of the number of associated keys, our multi-key BFV multiplication algorithm also achieves linear complexity with respect to the number of associated keys.

6 Implementation

We provide a proof-of-concept implementation of our MKHE schemes and demonstrate their concrete performance. Our implementation is based on Lattigo [30] which is one of the state-of-the-art HE libraries written in the Go language. The source code is available at `https://github.com/SNUCP/MKHE-KKLSS`.

In Sec. 6.1 and 6.2, we explain in details about ring operations and gadget decompositions. Specifically, we show a concrete instantiation of homomorphic gadget decomposition which meets our requirements. Then, we present several optimization techniques in Sec. 6.3 and 6.4 to solve the issues regarding noise growth and implementation efficiency. Finally, we propose secure parameter sets in Sec. 6.5 and provide experimental results in Sec. 6.6.

6.1 RNS and NTT

We briefly review some techniques for efficient implementation of polynomial arithmetic. First, the Residue Number System (RNS) is widely used to represent a polynomial with a large modulus as a tuple of polynomials with small coefficients. Specifically, if the ciphertext modulus Q is a product of pairwise coprime integers $q_0, q_1, \ldots, q_{\ell-1}$, then we get an isomorphism from R_Q to $R_{q_0} \times \cdots \times R_{q_{\ell-1}}$ defined by $a \mapsto ([a]_{q_0}, \ldots, [a]_{q_{\ell-1}})$ from the Chinese Remainder Theorem. We call $([a]_{q_0}, \ldots, [a]_{q_{\ell-1}})$ the RNS representation of $a \in R_Q$. Its major advantage is that arithmetic operations over R_Q can be instantiated using R_{q_i} without inefficient high-precision arithmetic.

Moreover, to accelerate polynomial multiplications in R_{q_i}'s, we use the discrete Fourier transformation over $\mathbb{Z}_{q_i}$, also known as Number Theoretic Transform (NTT). If $q_i = 1 \pmod{2N}$, there exists a $(2N)$-th root of unity $\rho_i \in \mathbb{Z}_{q_i}$. Then, we obtain a ring isomorphism from R_{q_i} to $\mathbb{Z}_{q_i}^N$ defined by $a \mapsto (a(\rho_i), a(\rho_i^3), \ldots, a(\rho_i^{2N-1}))$. This isomorphism is called the NTT conversion modulo q_i. We also note that the NTT conversion (or its inverse) can be computed in $O(N \log N)$ arithmetic operations over $\mathbb{Z}_{q_i}$.

As a result, an element $a = a_0 + a_1 X + \cdots + a_{N-1} X^{N-1}$ of R_{q_i} can be either represented as its coefficient vector $(a_0, a_1, \ldots, a_{N-1})$ or in the NTT form $(a(\rho_i), a(\rho_i^3), \ldots, a(\rho_i^{2N-1}))$. In our implementation, we use the NTT representation by default since it enables fast polynomial multiplication which is simply written as component-wise product over $\mathbb{Z}_{q_i}^N$.

6.2 Homomorphic Gadget Decomposition

There have been several studies (e.g. [13, 22, 23]) to design and implement HE schemes in a full-RNS manner without high-precision arithmetic. Below we describe an RNS-friendly gadget decomposition method commonly used by most HE libraries [11, 30] today.

Let $0 = j_0 < j_1 < \cdots < j_k = \ell$ be a partition of $\{0, 1, \ldots, \ell\}$, and $D_i = \prod_{j_i \le j < j_{i+1}} q_j$ for $0 \le i < k$. Then, D_i's are pairwise coprime integers with $\prod_{0 \le i < k} D_i = Q$, and the RNS-based gadget decomposition is defined as follows:

$$h : R_Q \to R^k, \quad h(a) = ([a]_{D_0}, \ldots, [a]_{D_{k-1}}).$$

We can show that h is a gadget decomposition with a bound $\|h(a)\|_\infty \le \frac{1}{2} \max_i D_i$, corresponding to the gadget vector $\mathbf{g} = (g_0, \ldots, g_{k-1}) \in R_Q^k$ satisfying $g_i = 1 \pmod{D_i}$ and $g_i = 0 \pmod{D_{i'}}$ for $i' \ne i$ from the Chinese Remainder Theorem.

If a ring element is stored in the NTT form, we first transform it into the coefficient form via inverse NTT before computing its gadget decomposition. Moreover, we need to perform further NTT conversion on each component $[a]_{D_i}$ to convert it back to the NTT form. Therefore, the RNS-based gadget decomposition is in fact the most expensive operation in HE implementation since it implicitly involves about k NTT conversions over R_Q. We refer the reader to [13, 22, 23] for more details.

Interestingly, the RNS-based gadget decomposition satisfies the homomorphic property that

$$\langle h(a) \circ h(b), \mathbf{g} \rangle = \sum_{0 \le i < k} [a]_{D_i} [b]_{D_i} \cdot g_i = ab \pmod{Q}$$

since $[a]_{D_i} \cdot [b]_{D_i} = ab \pmod{D_i}$ for all $a, b \in R_Q$. Hence, our implementation also uses the same RNS-based gadget decomposition since it meets our requirements as a concrete instantiation of homomorphic gadget decomposition. To the best of our knowledge, this decomposition method is originally introduced to support efficient implementation of HE schemes. However, we have a new definition of the homomorphic gadget decomposition and this is the first work which takes advantage of its homomorphic property.

6.3 Special Modulus Method

The special modulus method [20] is a well-known optimization technique in HE to reduce the noise growth from homomorphic operations. In this variant, public keys are generated in R_{QP} for an integer P called the special modulus. In the relinearization procedure, the external product operation is mostly executed over R_{QP} between the decomposed ring elements and public key, then the result is scaled down by P to recover the ciphertext modulus Q. Roughly speaking, this technique reduces the error bound by a factor of about P, which provides a fine trade-off between maximal ciphertext level and noise growth. For further details, we refer to App. B.

Recall that our construction has a larger multiplication error which grows quadratically with the bound of gadget decomposition, compared to linear growth of CDKS. Hence, we choose a larger special modulus to offset this disadvantage and obtain the same level of noise growth. Instead, our maximal ciphertext level is reduced by one to preserve the same security level as before.

6.4 Optimization of Multi-key BFV

Our multi-key BFV scheme described in Sec. 5.2 requires a gadget decomposition over $R_{\tilde{Q}}$ for $\tilde{Q} = Q^2$. However, the gadget decomposition over $R_{\tilde{Q}}$ is not RNS-friendly since $Q^2 = q_0^2 \ldots q_{\ell-1}^2$ does not factorize into distinct primes. Thus, it requires performing multi-precision arithmetic over the moduli q_i^2's to compute operations over R_{Q^2}, which contradicts the main purpose of RNS-based implementation.

To overcome this issue, we develop an optimization technique inspired by Kim et al. [25]. We introduce a new modulus Q', which is a product of distinct primes $q_0', \ldots, q_{\ell-1}'$ such that $Q' \approx Q$, then substitute the modulus Q^2 with QQ'. In multi-key homomorphic multiplication, we add a pre-processing of switching the modulus of an input ciphertext from Q to Q' so that the tensor product can be performed in $R_{QQ'}$ instead of R_{Q^2}. With this optimization method, we can implement our multi-key BFV scheme in an RNS-friendly manner using homomorphic gadget decomposition over $R_{QQ'}$. A full description of the modified multi-key BFV is given in App. B.

$\log N$	Ours			CDKS [10]		
	$\#q_i$	$\#p_i$	$\lceil \log QP \rceil$	$\#q_i$	$\#p_i$	$\lceil \log QP \rceil$
14	6	2	439	7	1	439
15	14	2	880	15	1	880

Table 2. Parameter sets. $\#q_i$ and $\#p_i$ indicate the number of primes used for ciphertext modulus $Q = \prod_i q_i$ and special modulus $P = \prod_i p_i$, respectively.

6.5 Parameter Setting

We describe how we set the parameters for the key distributions, RNS moduli chain, and homomorphic gadget decomposition in our implementation. We basically follow the parameter sets in [10] for a fair comparison. For the key distributions, we set the secret key distribution χ to sample each coefficient from $0, \pm 1$ with a probability of 0.25 for each of -1 and 1, and with a probability of 0.5 for 0. We also use the error distribution D_σ with $\sigma = 3.2$.

Regarding homomorphic gadget decomposition, we simply use $h(a) = ([a]_{q_0}, \ldots, [a]_{q_\ell - 1})$ for R_{Q_ℓ} where each digit D_i is a wordsize prime. In the case of multi-key BFV, we use another gadget decomposition $\tilde{h}(a) = ([a]_{q_0}, \ldots, [a]_{q_{L-1}}, [a]_{q'_0}, \ldots, [a]_{q'_{L-1}})$ with a larger modulus $\tilde{Q} = QQ'$. We note that $\tilde{h}$ takes about twice the computational costs of h since it needs to perform $2L$ NTT operations over R_Q, whereas h takes L NTT operations over R_Q.

Table 2 presents two parameter sets used in our implementation, achieving at least 128-bit security level according to [1]. For the moduli chain for RNS representation, we set each prime factor q_i for the ciphertext modulus Q to be 52–55 bits in size. In addition, 60-bit primes are used to form a special modulus P. As noted in Sec. 6.3, we assign two primes to the special modulus P (compared to a single prime in CDKS) to obtain a comparable noise bound. As a result, our scheme supports at most 6 or 14 levels when $\log N = 14$ or 15, respectively, compared to 7 or 15 of CDKS.

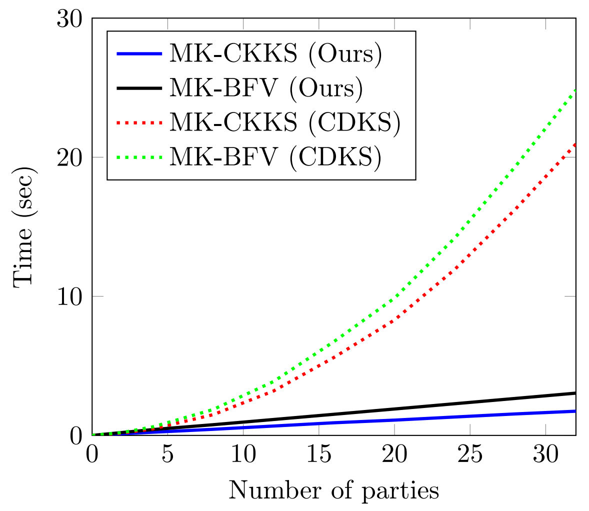
6.6 Benchmark Results

We provide benchmark results for our implementation of new multi-key CKKS and BFV multiplication algorithms. All experiments were performed with a single thread on a server machine with Intel(R) Xeon(R) Platinum 8268 @ 2.90GHz CPU and 192GB RAM running Ubuntu 20.04.3 LTS.

$\log N$	n	Ours		CDKS [10]	
		CKKS	BFV	CKKS	BFV
14	2	0.12 s	0.21 s	0.14 s	0.21 s
	4	0.23 s	0.42 s	0.45 s	0.58 s
	8	0.46 s	0.79 s	1.57 s	1.98 s
	16	0.95 s	1.52 s	5.91 s	7.31 s
	32	1.87 s	2.99 s	22.97 s	27.87 s
15	2	1.12 s	1.79 s	1.14 s	1.67 s
	4	2.02 s	3.52 s	3.65 s	4.35 s
	8	3.91 s	6.87 s	12.96 s	15.14 s
	16	7.81 s	13.18 s	48.21 s	55.37 s
	32	15.43 s	27.21 s	184.72 s	198.74 s

Table 3. Performance of multiplication algorithms of CDKS and our MKHE schemes. N and n denote the RLWE dimension and the number of associated keys, respectively.

In Table 3, we provide execution times of our multiplication algorithms and CDKS [10] for the number of associate keys $n = 2, 4, \ldots, 32$ and RLWE dimension $N = 2^{14}, 2^{15}$. Since the source code of CDKS



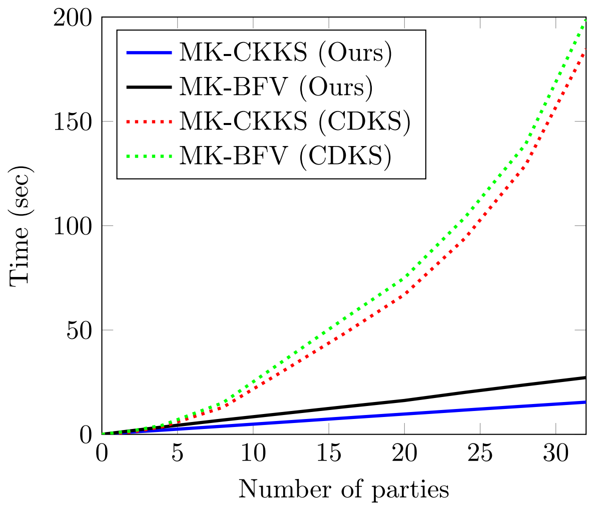


Fig. 1. Complexity of multiplication algorithms for $\log N = 14$ (left) or 15 (right).

is not publicly available, we implement both schemes using the same library for fair comparison. As expected from the complexity analysis, the time complexity grows linearly with n, which rapidly outperforms CDKS with quadratic complexity, as demonstrated in Fig. 1. To be precise, the execution time of homomorphic multiplication highly depends on the number of gadget decompositions, since gadget decomposition includes expensive NTT operations, which are the main bottleneck of the multiplication procedure. For the multi-key CKKS, we can expect about $n/3$ times speed-up since our method reduces the number of required gadget decompositions from n^2 to $3n$ according to Table 1. Meanwhile, for the multi-key BFV, we can expect about $n/5$ times speed-up since $\tilde{h}$ takes about twice the cost compared to h. We verify that our theoretic analysis aligns with benchmark results: for example, our method achieves about 11.9 and 7.3 times speed-up in multi-key CKKS and BFV, respectively, compared to the previous work when $n = 32$ and $N = 2^{15}$.

To compare the noise growth in our method with the previous method, we compute the difference between the decryption result of the multiplied ciphertext and the plaintext multiplication. We performed the experiment with the parameter set in Table 2, and the results are summarized in Table 4. As shown by experiments, our method has almost the same noise growth as the previous work in the recommended parameter sets.

$\log N$	n	Ours		CDKS [10]	
		CKKS	BFV	CKKS	BFV
14	2	5.84	43.74	5.84	42.01
	4	6.33	44.54	6.32	43.12
	8	6.83	45.53	6.81	45.65
15	2	6.33	45.23	6.33	44.83
	4	6.82	45.96	6.81	45.69
	8	7.31	46.81	7.32	46.72

Table 4. Noise growth of CDKS and our multiplication algorithms (in bits). N and n denote the RLWE dimension and the number of associated keys, respectively.

References

1. Albrecht, M., Chase, M., Chen, H., Ding, J., Goldwasser, S., Gorbunov, S., Halevi, S., Hoffstein, J., Laine, K., Lauter, K., Lokam, S., Micciancio, D., Moody, D., Morrison, T., Sahai, A., Vaikuntanathan, V.: Homomorphic encryption security standard. Tech. rep., HomomorphicEncryption.org, Toronto, Canada (November 2018)
2. Ananth, P., Jain, A., Jin, Z., Malavolta, G.: Multi-key fully-homomorphic encryption in the plain model. In: Theory of Cryptography Conference. pp. 28–57. Springer (2020)
3. Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., Wichs, D.: Multiparty computation with low communication, computation and interaction via threshold fhe. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 483–501. Springer (2012)
4. Boneh, D., Gennaro, R., Goldfeder, S., Jain, A., Kim, S., Rasmussen, P.M., Sahai, A.: Threshold cryptosystems from threshold fully homomorphic encryption. In: Annual International Cryptology Conference. pp. 565–596. Springer (2018)
5. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical gapsvp. In: Annual Cryptology Conference. pp. 868–886. Springer (2012)
6. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) **6**(3), 1–36 (2014)
7. Brakerski, Z., Perlman, R.: Lattice-based fully dynamic multi-key FHE with short ciphertexts. In: Annual Cryptology Conference. pp. 190–213. Springer (2016)
8. Brandao, L., Peralta, R.: Nist first call for multi-party threshold schemes (2023)
9. Chen, H., Chillotti, I., Song, Y.: Multi-key homomorphic encryption from TFHE. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 446–472. Springer (2019)
10. Chen, H., Dai, W., Kim, M., Song, Y.: Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 395–412 (2019)
11. Chen, H., Laine, K., Player, R.: Simple encrypted arithmetic library-seal v2.1. In: Financial Cryptography and Data Security: FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers 21. pp. 3–18. Springer (2017)
12. Chen, L., Zhang, Z., Wang, X.: Batched multi-hop multi-key FHE from Ring-LWE with compact ciphertext extension. In: Theory of Cryptography Conference. pp. 597–627. Springer (2017)
13. Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: A full rns variant of approximate homomorphic encryption. In: International Conference on Selected Areas in Cryptography. pp. 347–368. Springer (2018)
14. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 409–437. Springer (2017)
15. Chillotti, I., Gama, N., Georgieva, M., Izabachene, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: international conference on the theory and application of cryptology and information security. pp. 3–33. Springer (2016)
16. Clear, M., McGoldrick, C.: Multi-identity and multi-key leveled fhe from learning with errors. In: Annual Cryptology Conference. pp. 630–656. Springer (2015)
17. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive (2012)
18. Genise, N., Micciancio, D., Polyakov, Y.: Building an efficient lattice gadget toolkit: Subgaussian sampling and more. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 655–684. Springer (2019)

19. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing. pp. 169–178. ACM (2009)
20. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the aes circuit. In: Annual Cryptology Conference. pp. 850–867. Springer (2012)
21. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Annual Cryptology Conference. pp. 75–92. Springer (2013)
22. Halevi, S., Polyakov, Y., Shoup, V.: An improved rns variant of the bfv homomorphic encryption scheme. In: Cryptographers' Track at the RSA Conference. pp. 83–105. Springer (2019)
23. Han, K., Ki, D.: Better bootstrapping for approximate homomorphic encryption. In: Cryptographers' Track at the RSA Conference. pp. 364–390. Springer (2020)
24. Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al.: Advances and open problems in federated learning. Foundations and Trends® in Machine Learning **14**(1–2), 1–210 (2021)
25. Kim, A., Polyakov, Y., Zucca, V.: Revisiting homomorphic encryption schemes for finite fields. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 608–639. Springer (2021)
26. Kwak, H., Lee, D., Song, Y., Wagh, S.: A unified framework of homomorphic encryption for multiple parties with non-interactive setup. Cryptology ePrint Archive, Report 2021/1412 (2021), `https://ia.cr/2021/1412`
27. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Proceedings of the forty-fourth annual ACM symposium on Theory of computing. pp. 1219–1234. ACM (2012)
28. Mouchet, C., Bertrand, E., Hubaux, J.P.: An efficient threshold access-structure for rlwe-based multiparty homomorphic encryption. Journal of Cryptology **36**(2), 10 (2023)
29. Mouchet, C., Troncoso-Pastoriza, J., Bossuat, J.P., Hubaux, J.P.: Multiparty homomorphic encryption from ring-learning-with-errors. Proceedings on Privacy Enhancing Technologies **2021**(4), 291–311 (2021)
30. Mouchet, C.V., Bossuat, J.P., Troncoso-Pastoriza, J.R., Hubaux, J.P.: Lattigo: A multiparty homomorphic encryption library in go. In: Proceedings of the 8th Workshop on Encrypted Computing and Applied Homomorphic Cryptography. pp. 64–70. No. CONF (2020)
31. Mukherjee, P., Wichs, D.: Two round multiparty computation via multi-key FHE. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 735–763. Springer (2016)
32. Park, J.: Homomorphic encryption for multiple users with less communications. IEEE Access **9**, 135915–135926 (2021)
33. Peikert, C., Shiehian, S.: Multi-key fhe from lwe, revisited. In: Theory of Cryptography Conference. pp. 217–238. Springer (2016)

A Average-Case Noise Analysis

We conduct average-case noise analysis to estimate the noise growth of our multi-key multiplication algorithms. In this section, all polynomials will be treated as if they are random variables over the ring of integers R or its residue ring R_Q for some integer Q. In our analysis, all coefficients of each error polynomial $e \in R$ are zero-mean and independent, and have the same variance. Hence, we simply denote by $\mathsf{Var}(e)$ the common variance of its coefficients. Then, if a, b are independent polynomial samples in R, we have $\mathsf{Var}(ab) = N \cdot \mathsf{Var}(a) \cdot \mathsf{Var}(b)$. Finally, we suppose that ciphertext entries are uniformly distributed over R_Q.

For the homomorphic gadget decomposition h, we denote V_h as an upper bound for the variances of the components of $h(c)$ for a uniformly random element c. For example, $V_h = \frac{1}{12}D_{\max}^2$ when $h(a) = ([a]_{D_0}, \ldots, [a]_{D_{k-1}})$ is an RNS-based gadget decomposition with basis $D_0, \ldots, D_{k-1}$ and $D_{\max} = \max_{0 \leq i < k} D_i$. Finally, we set the variance of the key distribution and error distribution to $1/2$ and σ^2 as presented in the parameter setting (Sec. 6.5). The final error bound can be obtained from the standard deviation of an error polynomial by multiplying a certain constant to achieve a desired probability (e.g. $6\sqrt{\mathsf{Var}(e)}$).

A.1 Multi-key CKKS

We showed in Sec. 5 that the output ciphertext ct^* of our multi-key CKKS multiplication algorithm satisfies that

$$\langle \mathsf{ct}^*, (1, \overline{\mathsf{sk}}) \rangle = \langle \mathsf{ct}, (1, \overline{\mathsf{sk}}) \rangle \cdot \langle \mathsf{ct}', (1, \overline{\mathsf{sk}}) \rangle + e_1 + e_2$$

where

$$e_1 = \sum_{1\le i,j\le n} \left\langle h(c_i) \circ h(c_j'), s_j \cdot \mathbf{e}_{1,i} - r_i \cdot \mathbf{e}_{0,j} \right\rangle,$$
$$e_2 = \sum_{1\le i\le n} (c_i \boxdot \mathbf{w}) \boxdot \mathbf{e}_{2,i}.$$

We make a heuristic assumption that each summand of e_1 and e_2 are independent. Then, it holds that $\mathsf{Var}(e_1+e_2) = \mathsf{Var}(e_1) + \mathsf{Var}(e_2)$ holds, and

$$\mathsf{Var}(e_1) \le kn^2N^3\sigma^2V_h^2, \quad \mathsf{Var}(e_2) \le knN\sigma^2V_h$$

Hence, the variance of the total noise is bounded by

$$\mathsf{Var}(e_1+e_2) \le kn^2N^3\sigma^2V_h^2 + knN\sigma^2V_h.$$

A.2 Multi-key BFV

The multiplication result $\mathsf{ct}^* \leftarrow \texttt{MK-BFV.Mult}(\{\mathsf{pk}_i\}_{1\le i\le n}; \mathsf{ct}, \mathsf{ct}')$ of our multi-key BFV introduced in Sec. 5.2, we focus on the noise term $e_1 + e_2$ where

$$e_1 = \sum_{1\le i,j\le n} \left\langle \tilde{h}(c_i) \circ \tilde{h}(c_j'), s_j \cdot \mathbf{e}_{1,i} - r_i \cdot \mathbf{e}_{0,j} \right\rangle,$$
$$e_2 = \sum_{1\le i\le n} (c_i \,\tilde{\boxdot}\, \mathbf{w}) \boxdot \mathbf{e}_{2,i}.$$

Under the same heuristic assumptions as the above, the variance of total noise is bounded by

$$\mathsf{Var}(e_1+e_2) \le \tilde{k}n^2N^3\sigma^2V_{\tilde{h}}^2 + knN\sigma^2V_h.$$

A.3 Noise from Rescaling

For the multi-key CKKS scheme, noise from the rescaling operation often acts as a dominant factor. To be precise, let $\mathsf{ct} = (c_i)_{0\le i\le n} \in R_{Q_\ell}^{n+1}$ be a multi-key CKKS ciphertext such that $c_0 + \sum_{i=1}^n c_i \cdot s_i = \mu \pmod{Q_\ell}$. Then, the rescaled ciphertext $\mathsf{ct}' = (c_i')_{0\le i\le n} \leftarrow \texttt{MK-CKKS.Rescale}(\mathsf{ct})$ satisfies that:

$$\begin{aligned} c_0' + \sum_{i=1}^n c_i' s_i &= \lfloor q_\ell^{-1} \cdot c_0 \rceil + \sum_{i=1}^n \lfloor q_\ell^{-1} \cdot c_i \rceil \cdot s_i \\ &= q_\ell^{-1} \left(c_0 + \sum_{i=1}^n c_i \cdot s_i \right) + e_{rs} \\ &= q_\ell^{-1} \cdot \mu + e_{rs} \pmod{Q_{\ell-1}}. \end{aligned}$$

where $e_{rs} = \left(\lfloor q_\ell^{-1} \cdot c_0 \rceil - q_\ell^{-1} \cdot c_0 \right) + \sum_{i=1}^n \left(\lfloor q_\ell^{-1} \cdot c_i \rceil - q_\ell^{-1} \cdot c_i \right) \cdot s_i$ denotes the rescaling error. Hence, assuming that the rounding errors follow the uniform distribution over $[-1/2, 1/2)$, we obtain the following error bounds:

$$\|e_{rs}\|_\infty \le \frac{nN+1}{2},$$
$$\mathsf{Var}(e_{rs}) = \frac{nN+2}{24}.$$

B Special Modulus Variants

In this section, we describe the special modulus method [20] and apply it to our MKHE schemes. We introduce a new constant P, called a *special modulus*, and redefine the gadget encryption and external product as follows. We also present the optimized version of multi-key BFV, which we discussed in Sec. 6.4.

Definition 7. *Let s be an RLWE secret. We call $\mathbf{U} = (\mathbf{u}_0, \mathbf{u}_1) \in R_{QP}^{k\times 2}$ a gadget encryption of $\mu \in R$ under s if* $\mathbf{u}_0 + s \cdot \mathbf{u}_1 \approx P\mu \cdot \mathbf{g} \pmod{QP}$.

Definition 8. *Let $h : R_Q \to R^k$ be a gadget decomposition. For $a \in R_Q$ and $\mathbf{u} \in R_{QP}^k$, the external product of a and $\mathbf{u}$ is denoted and defined as follows.*

$$a \boxdot \mathbf{u} := \left\lfloor P^{-1} \cdot \langle h(a), \mathbf{u}\rangle \right\rceil \pmod{Q}$$

From the above definitions, it is satisfied that $a \boxdot (P\mu \cdot \mathbf{g}) = a \cdot \mu \pmod{Q}$ for all $a \in R_Q$ and any $\mu \in R$. If $\mathbf{U} = (\mathbf{u}_0, \mathbf{u}_1) \in R_Q^{k\times 2}$ is a gadget encryption of $\mu \in R$ under s so that $\mathbf{u}_0 + s \cdot \mathbf{u}_1 = P\mu \cdot \mathbf{g} + \mathbf{e}$ $\pmod{QP}$ for some small $\mathbf{e} \in R^k$, then the external product $a \boxdot \mathbf{U} = (c_0, c_1)$ of a polynomial a and $\mathbf{U}$ holds that

$$\begin{aligned} c_0 + c_1 \cdot s &= \left\lfloor P^{-1} \cdot \langle h(a), \mathbf{u}_0\rangle \right\rceil + \left\lfloor P^{-1} \cdot \langle h(a), \mathbf{u}_1\rangle \right\rceil \cdot s \\ &= P^{-1} \langle h(a), P\mu \cdot \mathbf{g} + \mathbf{e}\rangle + e_{rd} \\ &= a \cdot \mu + e \pmod{Q} \end{aligned}$$

for the rounding noise e_{rd} and $e = P^{-1} \cdot \langle h(a), \mathbf{e}\rangle + e_{rd}$, which is bounded by $\|e\|_\infty \le P^{-1}kN \cdot B_h\|\mathbf{e}\|_\infty + \frac{1}{2}(N+1)$. For the average case, we have $\mathsf{Var}(e) \le P^{-2}k\sigma^2 NV_h + \frac{1}{24}(N+2)$, assuming each component of $\mathbf{e}$ is sampled from the error distribution, and each rounding error follows the uniform distribution over $[-1/2, 1/2)$.

Note that the noise of external product is approximately reduced by a factor of P compared to the original external product. Therefore, we can choose a special modulus P properly to control the noise growth.

B.1 Multi-Key CKKS

• `MK-CKKS.Setup`(1^λ): Let the RLWE dimension be N, the ciphertext modulus be $Q = \prod_{i=0}^{L} q_i$ for some integers q_i, the key distribution be χ over R, the error parameter be $\sigma > 0$, and the special modulus P. Let $h : R_Q \to R^k$ be a *homomorphic* gadget decomposition corresponding to a gadget vector $\mathbf{g} \in R_Q^k$. Sample a CRS $\mathbf{a} \leftarrow \mathcal{U}(R_{QP}^k)$, and output a public parameter $\mathsf{pp} = (N, Q, P, \chi, \sigma, \mathbf{a}, h, \mathbf{g})$.

• `MK-CKKS.KeyGen`(pp): Return a secret key $\mathsf{sk} = s$ and a public key $\mathsf{pk} = (\mathbf{b}, \mathbf{d}, \mathbf{u}, \mathbf{v})$ generated as follows:

- Sample $s \leftarrow \chi$.
- Sample $\mathbf{e}_0 \leftarrow D_\sigma^k$ and let $\mathbf{b} = -s \cdot \mathbf{a} + \mathbf{e}_0 \pmod{QP}$.
- Sample $r \leftarrow \chi$ and $\mathbf{e}_1 \leftarrow D_\sigma^k$. Let $\mathbf{d} = -r \cdot \mathbf{a} + Ps \cdot \mathbf{g} + \mathbf{e}_1 \pmod{QP}$.
- Sample $\mathbf{u} \leftarrow \mathcal{U}(R_{QP}^k)$ and $\mathbf{e}_2 \leftarrow D_\sigma^k$. Let $\mathbf{v} = -s \cdot \mathbf{u} - Pr \cdot \mathbf{g} + \mathbf{e}_2 \pmod{QP}$.

We denote the encryption key by $\mathsf{ek} = (\mathbf{b}[0], \mathbf{a}[0]) \in R_{QP}^2$.

• `MK-CKKS.Enc`$(\mu; \mathsf{ek})$: Sample $w \leftarrow \chi$ and $e_0, e_1 \leftarrow D_\sigma$. Given a plaintext $\mu \in R$, output the ciphertext $\mathsf{ct} = \left\lfloor P^{-1} \cdot (w \cdot \mathsf{ek} + (e_0, e_1)) \right\rceil + (\mu, 0) \pmod{Q}$.

• `MK-CKKS.Mult`$(\mathsf{ct}, \mathsf{ct}'; \{\mathsf{pk}_i\}_{1\le i\le n})$: Given two ciphertexts $\mathsf{ct} = (c_i)_{0\le i\le n}$, $\mathsf{ct}' = (c_i')_{0\le i\le n} \in R_{Q_\ell}^{n+1}$ and their associated public keys $\{\mathsf{pk}_i\}_{1\le i\le n}$, execute Alg. 5 and return the output ciphertext ct^*.

In the case of special modulus variants, the worst-case and average-case upper bounds for noise change as follows.

$$\|e_1 + e_2\|_\infty \le \frac{1}{P}(2kn^2N^3 \cdot B_h^2 B_\sigma + knN \cdot B_h B_\sigma) + \frac{n(N+1)}{2}$$
$$\mathsf{Var}(e_1 + e_2) \le \frac{1}{P^2}(kn^2N^3\sigma^2 V_h^2 + knN\sigma^2 V_h) + \frac{n(N+2)}{24}$$

Algorithm 5 Multi-key CKKS multiplication algorithm with special modulus

```
Input: $\mathsf{ct} = (c_i)_{0\le i\le n} \in R_{Q_\ell}^{n+1}$, $\mathsf{ct}' = (c'_i)_{0\le i\le n} \in R_{Q_\ell}^{n+1}$ , $\{\mathsf{pk}_i = (\mathbf{b}_i, \mathbf{d}_i, \mathbf{u}_i, \mathbf{v}_i)\}_{1\le i\le n}$
Output: $\mathsf{ct}^* = (c_j^*)_{0\le j\le n} \in R_{Q_\ell}^{n+1}$
 1: $c_0^* \leftarrow c_0 c'_0 \pmod{Q_\ell}$
 2: for $1 \le i \le n$ do
 3:     $c_i^* \leftarrow c_0 c'_i + c_i c'_0 \pmod{Q_\ell}$
 4: end for
 5: $\mathbf{z} \leftarrow \sum_{1\le i\le n} h(c_i) \circ \mathbf{d}_i \pmod{Q_\ell P}$
 6: $\mathbf{w} \leftarrow \sum_{1\le j\le n} h(c'_j) \circ \mathbf{b}_j \pmod{Q_\ell P}$
 7: for $1 \le j \le n$ do
 8:     $c_j^* \leftarrow c_j^* + c'_j \boxdot \mathbf{z} \pmod{Q_\ell}$
 9: end for
10: for $1 \le i \le n$ do
11:     $(c_0^*, c_i^*) \leftarrow (c_0^*, c_i^*) + (c_i \boxdot \mathbf{w}) \boxdot (\mathbf{v}_i, \mathbf{u}_i) \pmod{Q_\ell}$
12: end for
```

B.2 Multi-Key BFV

• $\underline{\texttt{MK-BFV.Setup}(1^\lambda)}$: Let the RLWE dimension be N, the ciphertext modulus be $Q = \prod_{i=0}^{L} q_i$ for some integers q_i, the key distribution be χ over R, the error parameter be $\sigma > 0$, the special modulus P and $\tilde{Q} = QQ'$. Let $h : R_Q \to R^k$ be a gadget decomposition corresponding to a gadget vector $\mathbf{g} \in R_Q^k$, and let $\tilde{h} : R_{\tilde{Q}} \to R^{\tilde{k}}$ be a *homomorphic* gadget decomposition corresponding to a gadget vector $\tilde{\mathbf{g}} \in R_{\tilde{Q}}^{\tilde{k}}$. Let $t \in \mathbb{Z}$ be the plaintext modulus. Sample a CRS $\mathbf{a} \leftarrow \mathcal{U}(R_{QP}^{\tilde{k}})$, and output a public parameter $\mathsf{pp} = (N, t, Q, Q', P, \chi, \sigma, \mathbf{a}, h, \mathbf{g}, \tilde{h}, \tilde{\mathbf{g}})$.

We denote the external product with respect to the gadget decomposition $\tilde{h}$ by $\tilde{\boxdot}$.

• $\underline{\texttt{MK-BFV.KeyGen}(\mathsf{pp})}$: Return a secret key $\mathsf{sk} = s$ and a public key $\mathsf{pk} = (\mathbf{b}, \mathbf{d}, \mathbf{u}, \mathbf{v})$ generated as follows:

– Sample $s \leftarrow \chi$.
– Sample $\mathbf{e}_0 \leftarrow D_\sigma^{\tilde{k}}$ and let $\mathbf{b} = -s \cdot \mathbf{a} + \mathbf{e}_0 \pmod{Q}$.
– Sample $r \leftarrow \chi$ and $\mathbf{e}_1 \leftarrow D_\sigma^{\tilde{k}}$. Let $\mathbf{d} = -r \cdot \mathbf{a} + Ps \cdot \lfloor (t/Q) \cdot \tilde{\mathbf{g}} \rceil + \mathbf{e}_1 \pmod{QP}$.
– Sample $\mathbf{u} \leftarrow \mathcal{U}(R_{QP}^k)$ and $\mathbf{e}_2 \leftarrow D_\sigma^k$. Let $\mathbf{v} = -s \cdot \mathbf{u} - Pr \cdot \mathbf{g} + \mathbf{e}_2 \pmod{QP}$.

• $\underline{\texttt{MK-BFV.Enc}(m; \mathsf{ek})}$: Sample $w \leftarrow \chi$ and $e_0, e_1 \leftarrow D_\sigma$. Given a message $m \in R_t$, output the ciphertext $\mathsf{ct} = \lfloor P^{-1} \cdot (w \cdot \mathsf{ek} + (e_0, e_1)) \rceil + (\lfloor (Q/t) \cdot m \rceil, 0) \pmod{Q}$.

We denote the encryption key by $\mathsf{ek} = (\mathbf{b}[0], \mathbf{a}[0]) \in R_{QP}^2$.

• $\underline{\texttt{MK-BFV.Mult}(\mathsf{ct}, \mathsf{ct}'; \{\mathsf{pk}_i\}_{1\le i\le n})}$: Given two ciphertexts $\mathsf{ct} = (c_i)_{0\le i\le n}$, $\mathsf{ct}' = (c'_i)_{0\le i\le n} \in R_Q^{n+1}$ and associated public keys $\{\mathsf{pk}_i\}_{1\le i\le n}$, run Alg. 6 and return the ciphertext $\mathsf{ct}^* = (c_i^*)_{0\le i\le n} \in R_Q^{n+1}$.

In the case of special modulus variants, the worst-case and average-case upper bounds for noise change as follows.

$$\begin{aligned}\|e_1+e_2\|_\infty &\le \frac{1}{P}(2\tilde{k}n^2N^3\cdot B_{\tilde{h}}^2B_\sigma + knN\cdot B_hB_\sigma) + \frac{n(N+1)}{2}\\ \mathsf{Var}(e_1+e_2) &\le \frac{1}{P^2}(\tilde{k}n^2N^3\sigma^2V_{\tilde{h}}^2 + knN\sigma^2V_h) + \frac{n(N+2)}{24}\end{aligned}$$

Algorithm 6 Multi-key BFV multiplication algorithm with special modulus

```
Input: $\mathsf{ct} = (c_i)_{0\le i\le n} \in R_Q^{n+1}$, $\mathsf{ct}' = (c'_i)_{0\le i\le n} \in R_Q^{n+1}$, $\{\mathsf{pk}_i = (\mathbf{b}_i, \mathbf{d}_i, \mathbf{u}_i, \mathbf{v}_i)\}_{1\le i\le n}$
Output: $\mathsf{ct}^* = (c^*_j)_{0\le j\le n} \in R_Q^{n+1}$
 1: for $0 \le j \le n$ do
 2:     $c''_j \leftarrow \lfloor (Q'/Q)\cdot c'_j \rceil \pmod{Q'}$
 3: end for
 4: $c^*_0 \leftarrow \lfloor (t/Q')\cdot c_0c''_0 \rceil \pmod{Q}$
 5: for $1 \le j \le n$ do
 6:     $c^*_j \leftarrow \lfloor (t/Q')\cdot c_0c''_j \rceil + \lfloor (t/Q')\cdot c_jc''_0 \rceil \pmod{Q}$
 7: end for
 8: $\mathbf{z} \leftarrow \sum_{1\le i\le n} \tilde{h}(c_i) \circ \mathbf{d}_i \pmod{QP}$
 9: $\mathbf{w} \leftarrow \sum_{1\le j\le n} \tilde{h}(c''_j) \circ \mathbf{b}_j \pmod{QP}$
10: for $1 \le j \le n$ do
11:     $c^*_j \leftarrow c^*_j + c''_j \,\tilde{\boxdot}\, \mathbf{z} \pmod{Q}$
12: end for
13: for $1 \le i \le n$ do
14:     $(c^*_0, c^*_i) \leftarrow (c^*_0, c^*_i) + (c_i \,\tilde{\boxdot}\, \mathbf{w}) \boxdot (\mathbf{v}_i, \mathbf{u}_i) \pmod{Q}$
15: end for
```