IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 16 April 2023, accepted 7 May 2023, date of publication 19 May 2023, date of current version 30 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3277862

RESEARCH ARTICLE

Key Reduction in Multi-Key and Threshold Multi-Key Homomorphic Encryptions by Reusing Error

ZAHYUN KOO[1], JOON-WOO LEE[2], (Member, IEEE), JONG-SEON NO[1], (Fellow, IEEE), AND YOUNG-SIK KIM[3], (Member, IEEE)

[1]Department of Electrical and Computer Engineering, INMC, Seoul National University, Seoul 08826, South Korea
[2]School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, South Korea
[3]Department of Information and Communication Engineering, Chosun University, Gwangju 61452, South Korea

Corresponding authors: Joon-Woo Lee (jwlee2815@cau.ac.kr) and Jong-Seon No (jsno@snu.ac.kr)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korea Government[Ministry of Science and ICT (MSIT)]), Development of Highly Efficient Post-Quantum Cryptography (PQC) Security and Performance Verification for Constrained Devices under Grant 2021-0-00400; and in part by the Chung-Ang University Research Grants in 2022.

ABSTRACT As cloud computing and AI as a Service are provided, it is increasingly necessary to deal with privacy sensitive data. To deal with the sensitive data, there are two cases of outsourcing process: i) many clients participate dynamically ii) many clients are pre-determined. The solutions for protecting sensitive data in both cases are the multi-key homomorphic encryption (MKHE) scheme and the threshold multi-key homomorphic encryption (TMKHE) scheme. However, these schemes may be difficult for clients with limited resources to perform MKHE and TMKHE. In addition, due to the large size of the evaluation keys, in particular multiplication and rotation keys, the communication between the clients and server that provide outsourcing service increases. Also, the size of the evaluation keys that the server must hold is tremendous, in particular, for the multiplication and rotation keys, which are essential for bootstrapping operation. In this paper, we propose a variant of MKHE and TMKHE with reduced evaluation keys. To reduce the size of the evaluation keys, we propose a variant of ring learning with errors (RLWE), called RLWE reusing errors (ReRLWE). ReRLWE generates other components by reusing the error that is used when generating an RLWE sample. We prove that RLWE can be reduced to ReRLWE and propose modified evaluation keys under the ReRLWE assumption, which are the modified multiplication and rotation keys. For MKHE, multiplication and rotation keys are reduced by 66% and 25%, respectively. For TMKHE, a multiplication and rotation keys are reduced by 50% and 25%, respectively.

INDEX TERMS Homomorphic encryption (HE), multi-key homomorphic encryption (MKHE), ring learning with error (RLWE), threshold multi-key homomorphic encryption (TMKHE).

I. INTRODUCTION

Recently, cloud computing services that provide on-demand computing resources through a network are being actively used, and AIaaS (AI as a Service), which provides various AI-based functions to customers, has attracted much attention. However, when an outsourcing server uses the client's private data, there is a risk of leakage of the client's data. This means that it is difficult to use the client's raw data and hinder outsourcing companies from developing various related systems. To prevent this problem, a client that requires cloud services and AIaaS uses the cryptogrphic scheme. However, it was difficult to process encrypted data in an encrypted state in the previous public key encryption schemes. To process encrypted data, the homomorphic encryption (HE) scheme has been proposed. HE is the public key encryption scheme that enables the homomorphic operations on encrypted data and makes it possible to preserve

The associate editor coordinating the review of this manuscript and approving it for publication was Neetesh Saxena.

TABLE 1. Summary of the reduced ratio for modified evaluation keys for MK-CKKS and TMK-CKKS.

	Multiplication		Rotation key with two indcies	
	MK-CKKS	TMK-CKKS	MK-CKKS	TMK-CKKS
Previous work	4.92 MB	3.27 MB	6.55 MB	6.55 MB
This work	1.63 MB	1.63 MB	4.91 MB	4.91 MB
Reduced ratio	66%	50%	25%	25%

security in the data processing. Fully HE(FHE), which supports an arbitrary number of two homomorphic operations simultaneously, was first constructed by Gentry [1] by applying bootstrapping, but this was impractical for outsourcing services. Many HE schemes have been developed with various improvements and optimizations [2], [3], [4], [5]. Thus, a client using HE can protect sensitive personal data when receiving the cloud computing service and AIaaS [6], [7], [8].

However, HE is not always an appropriate solution when many clients participate in a single outsourcing server. For example, in HE for multiple clients and a single outsourcing server, the data should be encrypted under the same public key. If Bob generates a public key through his secret key and shares it with all clients in the outsourcing server, each client encrypts its own data using the shared public key. This means that Bob may access the data of other clients. In other words, the problem of concentration of authority arises. To solve this problem, the multi-key HE (MKHE) [9], [10], [11] allows each client to generate its own secret/public key pair, and an outsourcing server performs homomorphic operations using all clients' public and evaluation keys. Therefore, when many clients simultaneously participate in the outsourcing server, MKHE is more appropriate than HE.

However, there are several problems for MKHE. First, the ciphertext of MKHE is expanded during homomorphic operations and this expansion is proportional to the number of clients. Second, each client needs to generate the evaluation keys and the outsourcing server must have the evaluation keys of all clients to support homomorphic operations. Thus, due to the large size of evaluation keys, the computational and memory costs are higher than those of HE. Also the communication cost between all clients and the outsourcing server is increased.

A partial solution to these problems is the threshold MKHE (TMKHE). TMKHE is the variant of MKHE schemes with the pre-defined clients to generate a common public key and many evaluation keys. It also allows prior communication between clients to generate a common public key and evaluation keys [14]. Thus, the ciphertext expansion does not depend on the number of clients, and the size of the evaluation keys possessed in the outsourcing server can be reduced [12], [13]. However, in TMKHE, a new client is hard to participate in the ongoing process, while MKHE has the advantage that new client can easily participate. Therefore, it is necessary to choose MKHE or TMKHE according to the situation.

Although there are studies that practically implement HE [2], [3], [4], [5], MKHE and TMKHE, there is a problem to solve. When a client's computer resources are limited, it may be difficult to generate many evaluation keys. To support homomorphic operation, in particular bootstrapping, each client must generate evaluation keys, which may be difficult to generate due to a lack of computer memory. In addition, although the evaluation keys have been reduced due to TMKHE, it is still a burden for clients with limited resources. This problem becomes a bottleneck for clients with limited resources to be served the cloud computing services and AIaaS.

MKHE and TMKHE are constructed based on the ring-learning with errors problem (RLWE). The learning with errors problem (LWE) was introduced by Regev in 2005 [15],and then its ring variant was also proposed [16]. The hardness of LWE is first demonstrated when the secret distribution is uniform. However, in [17], LWE also satisfies the hardness when the secret distribution is the error distribution, and in the case of RLWE, it can be applied similarly. At this time, we consider the following question.

If the error defined in the RLWE problem is used once again as a secret, is the problem still difficult?

That is, the sample of the variant RLWE distribution is of the form

$$(a, a \cdot s + x, a \cdot x + e),$$

where $a \leftarrow R_q$, $s \leftarrow R_q$, and x, e are chosen from the error distribution, and it is asked if it is distinguishable from this sample and the sample of the uniform distribution.

A. OUR CONTRIBUTIONS

In this paper, we focus on the multi-key residue number system (RNS) variant Cheon-Kim-Kim-Song scheme (MK-RNS-CKKS), and the threshold MK-RNS-CKKS (TMK-RNS-CKKS). For convenience, we refer to MK-RNS-CKKS and TMK-RNS-CKKS as MK-CKKS and TMK-CKKS, respectively.

First, we propose a variant of RLWE by reusing the error used to define an RLWE problem, called the ReRLWE problem. To define the ReRLWE problem, we define the ReRLWE distribution, where the error for generating an RLWE sample is reused as a secret for the other RLWE sample. In other words, the sample of the ReRLWE distribution is of the form $(a, a \cdot s + x, a \cdot x + e)$, where $a \leftarrow R_q$, $s \leftarrow R_q$,

and $x, e \leftarrow \chi$ for polynomial ring R_q and error distribution χ. Second, we demonstrate that there exists a reduction from RLWE to ReRLWE. Since the RLWE problem is NP-hard, the ReRLWE problem is also NP-hard (Section III).

Third, under the ReRLWE assumption, we propose modified evaluation keys, called modified multiplication and rotation keys in MK-CKKS and TMK-CKKS. The modified multiplication key is generated by reusing the error used when generating the public key for each client. Also, the ReRLWE sample can be used per two rotation indices instead of using RLWE samples per one rotation index (Section IV).

Finally, we verify the correctness and security of the proposed schemes. Also, due to the modified evaluation keys, the homomorphic operation time increases slightly. However, the size of evaluation keys can be reduced. For MK-CKKS, the modified multiplication key and many rotation keys are reduced by 66% and 25%, respectively. Also, for TMK-CKKS, the modified multiplication and many rotation keys are reduced by 50% and 25%, respectively (Section V). Our contributions are briefly described in Table 1. Therefore, it is possible to perform the multi-key homomorphic operation even though client's computer resources are limited.

B. RELATED WORKS

The CKKS scheme [2] was first proposed without bootstrapping as a somewhat homomorphic encryption scheme that supports only a finite number of multiplications. Cheon et al. [19] suggested a bootstrapping operation using a sine function with a Taylor approximation. Chen et al. [20] and Han and Ki [21] proposed improved bootstrapping methods. Because bit integers used to represent the ciphertexts in the CKKS scheme cannot be stored with the basic data type, the CKKS scheme had to use arbitrary precision data type libraries. However, Cheon et al. [3] applied the RNS system in the CKKS scheme to remove the external library. The RNS-CKKS scheme causes more approximation error in the homomorphic multiplication of the RNS-CKKS scheme than in that of the original CKKS scheme. To overcome this problem, Kim et al. [22] proposed a management method for the scaling factor in the RNS-CKKS scheme. In addition, Lee et al. [4] proposed two procedures to significantly increase the precision of the RNS-CKKS bootstrapping operation: an algorithm for deriving the optimal minimax approximation polynomial for modular reduction, and a composite-function procedure involving the inverse sine function. The multi-key version of CKKS (MK-CKKS) was proposed by Chen et al. [9]. The public key size of MK-CKKS increased linearly with the number of clients. To overcome this problem, Mouchet et al. [13] suggested a common public, multiplication, and rotation keys using communication between clients. However, owing to the common public keys of clients, the size of the public key for the server is reduced, while the communication cost is increased. To reduce the communication cost, Part [14] suggested a variant of TMK-HE (called compact MK-HE in [14]). In this scheme, communication between clients is used to generate a common public key. However, in order to generate common multiplication and rotation keys, the client's multiplication and rotation keys sent to the server are generated in the server.

TABLE 2. The notations for the RLWE problem and homomprhic encryption scheme.

$\mathbb{Z}$	the set of integer numbers
$\mathbb{Q}$	the set of rational numbers
$\mathbb{C}$	the set of complex numbers
M	the power of two
N	the power of two with $N = M/2$
R	$\mathbb{Z}[X]/\langle X^N + 1\rangle$, the ring over integers
R_q	R/qR, a cyclotomic ring

C. ORGANIZATION

The remainder of this paper is organized as follows: In Section II, the RLWE problem, MK-CKKS, and TMK-CKKS are introduced. In Section III, we propose a variant of RLWE and prove the hardness of this problem. Section IV proposes the variant of MK-CKKS and TMK-CKKS based on the variant of RLWE. Section V shows the correctness and security of the proposed schemes and compares them with previous schemes. Finally, the conclusion is provided in Section VI.

II. PRELIMINARIES

In this section, we introduce the RLWE problem, MK-CKKS, and TMK-CKKS. See Table 2 for related notations of the RLWE problem and homomorphic encryption schemes.

A. LATTICES AND LATTICE PROBLEM

An n-dimensional lattice is a discrete subgroup of $\mathbb{R}^n$. More specifically, for linearly independent vectors $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subseteq \mathbb{R}^n$, the set

$$\mathcal{L} = \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i \ : \ x_i \in \mathbb{Z} \right\}$$

is a lattice in $\mathbb{R}^n$ with basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$. A lattice is an ideal lattice if it is isomorphic to some ideal I of R. The i-th successive minimum $\lambda_i(\mathcal{L})$ is the smallest radius r such that $\mathcal{L}$ contains i linearly independent vectors of norm at most r.

Now, we introduces the shortest independent vector problem over the lattice $\mathcal{L}$.

Definition 1: The SIVP is defined as follows: Given a lattice $\mathcal{L}$ of dimension n, the SIVP is to find the n linearly independent vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathcal{L}$ such that $\max_i \|\mathbf{v}_i\| \leq \gamma \cdot \lambda_n(\mathcal{L})$, where $\gamma \geq 1$ is a function of dimension n.

This problem is known to be NP-hard for any approximation factor $\gamma \leq O(1)$ [18]. The SIVP problem can be extended to the polynomial ring R if the lattice $\mathcal{L}$ is the ideal lattice, denoted as Id-SIVP.

B. RING LEARNING WITH ERRORS

In this subsection, we introduce the RLWE problem.

Definition 2 (RLWE Distribution): For a secret $s \in R_q$ and a distribution χ over R_q, a sample from the RLWE distribution $A_{s,\chi}$ over R_q^2 is generated by choosing $a \leftarrow R_q$ uniformly at random, choosing $e \leftarrow \chi$, and outputting $(a, b = a \cdot s + e \mod qR)$.

Definition 3 (RLWE Problem, Average-Case Decision): The average-case decision version of the RLWE problem, denoted $\mathrm{RLWE}_{q,\chi}$, is to distinguish with non-negligible advantage between independent samples from $A_{q,\chi}$ and the same number of uniformly at random and independent samples from R_q^2, where $s \leftarrow R_q$ is uniformly at random.

In [16], there exists a reduction from SIVP to RLWE when the error distribution χ is the Gaussian distribution. This means that the RLWE problem is also NP-hard. In addition, if the error distribution χ is supported on R_q, then the secret s can also be chosen from χ without affecting the hardness of the RLWE problem [17].

C. MULTI-KEY HOMOMORPHIC ENCRYPTION SCHEME

The CKKS scheme is the homomorphic encryption for the arithmetic of approximate numbers based on the RLWE problem [2]. The main idea of CKKS is to consider an encryption error as part of a computational error that occurs during approximate computations. Since the CKKS scheme have to use multi-precision data type libraries, the authors of [3] introduce the residue number system variant of CKKS, called RNS-CKKS. RNS-CKKS uses the RNS form to represent ciphertexts and perform homomorphic operations efficiently. The product of the large primes is used for the ciphertext modulus in RNS-CKKS, and these large primes are chosen to be similar to the scaling factor, which is a parameter controlling the approximated error.

1) RESIDUE NUMBER SYSTEM

Let $\mathcal{C} = \{q_0, \ldots, q_{\ell-1}\}$ is the set of positive integers coprime each other and let $a \in \mathbb{Z}_Q$ where $Q = \prod_{i=0}^{\ell-1} q_i$. We denote by $[\cdot]_{\mathcal{C}}$ the map from $\mathbb{Z}_Q \to \mathbb{Z}_{q_0} \times \cdots \times \mathbb{Z}_{q_{\ell-1}}$, defined by

$$a \to [a]_{\mathcal{C}} = ([a]_{q_0}, \ldots, [a]_{q_{\ell-1}}) \in \mathbb{Z}_{q_0} \times \cdots \times \mathbb{Z}_{q_{\ell-1}}.$$

This map is a ring isomorphism from the Chinese Remainder Theorem (CRT) and $[a]_{\mathcal{C}}$ is called the RNS representation of $a \in \mathbb{Z}_Q$. This isomorphism over the integers can be extended to a ring isomorphism $[\cdot]_{\mathcal{C}} : R_q \to R_{q_0} \times \cdots \times R_{q_{\ell-1}}$ by applying it coefficient-wise over the cyclotomic rings.

2) CANONICAL EMBEDDING

Let $K = \mathbb{Q}[X]/\langle X^N + 1\rangle$. The canonical embeddings are the N ring homomorphisms $\sigma_j : K \to \mathbb{C}$ for all $j = 1, \ldots, N$, where $\mathbb{C}$ is the set of complex numbers. They are defined by $\sigma_j(X) = \zeta_M^j$, where ζ_M is the solution of $X^N + 1$ for any $j \in \mathbb{Z}_M^\times$ with $N = 2^r$ for some positive integer r, where $\mathbb{Z}_M^\times$ denotes the set of integer j module M such that $\gcd(j, M) = 1$. We define the canonical embedding vector as the ring homomorphism $\sigma : K \to \mathbb{C}^N$ as $\sigma(x) = (\sigma_j(x))_{j \in \mathbb{Z}_M^\times}$ under component-wise addition and multiplication. Let $\mathbb{H} = \{(z_j)_{j \in \mathbb{Z}_M^*} : z_j = \bar{z}_{-j}\}$, and π be a natural projection from $\mathbb{H}$ to $\mathbb{C}^{N/2}$. Then the range of σ is exactly $\mathbb{H}$.

3) FAST BASIS CONVERSION AND MODULUS SWITCHING

Let $\mathcal{D} = \{q_0, \ldots, q_{\ell-1}, p_0, \ldots, p_{k-1}\}$ be a basis and $\mathcal{B} = \{p_0, \ldots, p_{k-1}\}$ and $\mathcal{C} = \{q_0, \ldots, q_{\ell-1}\}$ be its subbases. Let $P = \prod_{i=0}^{k-1} p_i$ and $Q = \prod_{j=0}^{\ell-1} q_j$. Then the fast basis conversion converts the RNS bases from $\mathcal{C}$ to $\mathcal{B}$ without the merging process of CRT, which is defined as

$$\mathsf{Conv}_{\mathcal{C}\to\mathcal{B}}([a]_{\mathcal{C}}) = \left(\sum_{j=0}^{\ell-1} [a^{(j)} \cdot \hat{q}_j^{-1}] \cdot \hat{q}_j \mod p_i\right)_{0 \le i < k},$$

where $[a]_{\mathcal{C}} = (a^{(0)}, \ldots, a^{(\ell-1)}) \in \prod_{j=0}^{\ell-1} \mathbb{Z}_{q_j}$ and $\hat{q}_j = \prod_{i \neq j} q_i \in \mathbb{Z}$. This operation can be extended to the cyclotomic rings $\prod_{j=0}^{\ell-1} R_{q_j}$. Now, we define the modulus switching operations ModUp and $\mathsf{ModDown}$, which are necessary during homomorphic operations. ModUp operation is to add other moduli in $\mathcal{B}$ to the current RNS basis $\mathcal{C}$ to expand the modulus space without changing the value:

$$\mathsf{ModUp}_{\mathcal{C}\to\mathcal{D}}([a]_{\mathcal{C}}) = (\mathsf{Conv}_{\mathcal{C}\to\mathcal{B}}([a]_{\mathcal{C}}), [a]_{\mathcal{C}}).$$

$\mathsf{ModDown}$ is to remove the moduli in $\mathcal{B}$ from the current RNS basis $\mathcal{D}$ by dividing the value by P:

$$\begin{aligned}&\mathsf{ModDown}_{\mathcal{D}\to\mathcal{C}}([a]_{\mathcal{B}}, [b]_{\mathcal{C}})\\ &= ([b]_{\mathcal{C}} - \mathsf{Conv}_{\mathcal{B}\to\mathcal{C}}([a]_{\mathcal{B}})) \cdot [P^{-1}]_{\mathcal{C}}\end{aligned}$$

4) MULTI-KEY RNS-CKKS

MK-CKKS follows the same pipeline as RNS-CKKS. The difference between RNS-CKKS and MK-CKKS is that all clients use the public parameter randomly generated polynomial $a \in \prod_{j=0}^{\ell-1} R_{q_j}$. This assumption is called the common reference string (CRS) assumption. Let d be the number of different clients. Then, a ciphertext related to d different clients is of the form

$$\mathsf{ct} = (c_0, \ldots, c_d) \in \left(\prod_{j=0}^{\ell-1} R_{q_j}\right)^{d+1},$$

which is decryptable using the concatenated secret key $\overline{\mathsf{sk}} = (s_0, \ldots, s_{d-1}, 1)$ as

$$\mu = \langle \mathsf{ct}, \overline{\mathsf{sk}} \rangle = \sum_{i=0}^{d-1} c_i \cdot s_i + c_d.$$

The detailed procedures in the MK-CKKS scheme are given as follows:

- $\mathsf{MK\text{-}CKKS.Setup}(1^\lambda)$: Given a security parameter λ, set the RLWE dimension N, ciphertext modulus $q_0, \ldots, q_L$ and special modulus $p_0, \ldots, p_{k-1}$ satisfying $q_j \equiv p_i \equiv 1 \mod 2N$ for all $j = 0, \ldots, L$ and for all

$i = 0, \ldots k-1$, secret key distribution χ_{sec}, and error distribution χ_{err} over R. Generate a random polynomial

$$a = (a^{(0)}, \ldots, a^{(k+L)}) \leftarrow \prod_{i=0}^{k-1} R_{p_i} \times \prod_{j=0}^{L} R_{q_j}.$$

Return the public parameter

$$pp = (N, \chi_{\mathsf{sec}}, \chi_{\mathsf{err}}, a, \{q_j\}, \{p_i\}),$$

for all $j = 0, \ldots, L$ and $i = 0, \ldots, k-1$.

- MK-CKKS.UniEnc(μ; s): For $\mu \in R$, generate

$$mk = (mk_0, mk_1, mk_2) \in \left(\prod_{i=0}^{k-1} R_{p_i} \times \prod_{j=0}^{L} R_{q_j} \right)^3$$

as follows:

1) Sample $r \leftarrow \chi_{\mathsf{sec}}$.
2) Sample

$$mk_0 = (mk_0^{(0)}, \ldots, mk_0^{(k+L)}) \leftarrow \prod_{i=0}^{k-1} R_{p_i} \times \prod_{j=0}^{L} R_{q_j}.$$

3) Sample $e_1 \leftarrow \chi_{\mathsf{err}}$ and set

$$mk_1^{(j)} = -s \cdot mk_0^{(j)} + e_1^{(j)} \mod p_j$$

for $0 \le j < k$ and

$$mk_1^{(k+j)} = -s \cdot mk_0^{(k+j)} + e_1^{(k+j)} + [P]_{q_j} \cdot r \mod q_j$$

for $0 \le j \le L$.

4) Sample $e_2 \leftarrow \chi_{\mathsf{err}}$ and set

$$mk_2^{(j)} = r \cdot a^{(j)} + e_2^{(j)} \mod p_j$$

for $0 \le j < k$ and

$$mk_2^{(k+j)} = r \cdot a^{(k+j)} + e_2^{(k+j)} + [P]_{q_j} \cdot s \mod q_j$$

for $0 \le j \le L$.

- MK-CKKS.KeyGen(pp): Each i-th client samples the secret key $s_i \leftarrow \chi_{\mathsf{sec}}$, and an error $e_i \leftarrow \chi_{\mathsf{err}}$. Set the public key as

$$pk_i = \left(pk_i^{(j)} = (a^{(j)}, b_i^{(j)}) \right)_{0 \le j \le k+L},$$

where

$$b_i^{(j)} \leftarrow -a^{(j)} \cdot s_i + e_i \mod p_j$$

for $0 \le j < k$ and

$$b_i^{(k+j)} \leftarrow -a^{(k+j)} \cdot s_i + e_i \mod q_j$$

for $0 \le j \le L$. The multiplication key is set by

$$mk_i \leftarrow \text{MK-CKKS.UniEnc}(s_i; s_i).$$

- MK-CKKS.Ecd($\mathbf{z}$; Δ): For a vector $\mathbf{z} \in \mathbb{C}^{N/2}$ and the scaling factor Δ, return

$$m(X) = \sigma^{-1}\left(\lfloor \Delta \cdot \pi^{-1}(\mathbf{z}) \rceil_{\sigma(R)} \right) \in R,$$

where $\lfloor \pi^{-1}(\mathbf{z}) \rceil_{\sigma(R)}$ denotes the rounding of $\pi^{-1}(\mathbf{z})$ into an element of $\sigma(R)$.

- MK-CKKS.Dcd($m(X)$; Δ): For a polynomial $m(X) \in R$, return a vector $\mathbf{z} \in \mathbb{C}^{N/2}$ whose entry of index j is $z_j = \left\lfloor \Delta^{-1} \cdot m(\zeta_M^{5^j}) \right\rceil$ for $j \in \{0, 1, \ldots, N/2-1\}$, where ζ_M is the M-th root of unity.

- MK-CKKS.Enc($\mathbf{z}$; pk_i): For a message $\mathbf{z} \in \mathbb{C}^{N/2}$ and the scaling factor Δ, generate the message polynomial by

$$m(X) = \text{MK-CKKS.Ecd}(\mathbf{z}; \Delta).$$

Then, sample $v \leftarrow \chi_{\mathsf{sec}}$ and $e_0, e_1 \leftarrow \chi_{\mathsf{err}}$ and generate the ciphertext

$$\mathsf{ct} = \left(\mathsf{ct}^{(j)} = \lfloor P^{-1} \cdot (c_0^{(j)}, c_1^{(j)}) \rceil + (0, m) \mod q_j \right),$$

where $(c_0^{(j)}, c_1^{(j)}) = v \cdot pk_i^{(j)} + (e_0, e_1)$ for $0 \le j \le L$.

As in [9], a ciphertext $ct_i = (c_0, \ldots, c_{k_i})$ corresponding to the tuple of the parties $(id_0, \ldots, id_{k_i-1}) \in \{0, \ldots, d-1\}^{k_i+1}$ is converted into the ciphertext

$$\mathsf{ct}^* = (c_0^*, \ldots, c_{\mathsf{d}}^*) \in \left(\prod_{i=0}^{L} R_{q_i} \right)^{d+1}$$

that is defined as

$$c_d^* = c_{k_i} \text{ and } c_i^* = \begin{cases} c_i & \text{if } i = id_j \text{ for some } 0 \le j \le k_i - 1 \\ 0 & \text{otherwise} \end{cases}$$

for $0 \le i \le d-1$. Then we obtain

$$\langle \mathsf{ct}_i, (s_{id_1}, \ldots, s_{id_{k_i}}, 1) \rangle = \langle \mathsf{ct}^*, (s_0, \ldots, s_{d-1}, 1) \rangle.$$

Hereafter, we will assume that this pre-processing is always performed before homomorphic operations such that two input ciphertexts are related to the same set of d parties.

- MK-CKKS.Add($\mathsf{ct}_1, \mathsf{ct}_2$): Given two ciphertexts $\mathsf{ct}_1, \mathsf{ct}_2 \in \left(\prod_{i=0}^{\ell} R_{q_i} \right)^{d+1}$ at level ℓ, return the ciphertext

$$\mathsf{ct}' = \mathsf{ct}_1 + \mathsf{ct}_2 \in \left(\prod_{i=0}^{\ell} R_{q_i} \right)^{d+1}.$$

- MK-CKKS.Mult($\mathsf{ct}_1, \mathsf{ct}_2$; $\{(mk_i, b_i)\}_{0 \le i \le d-1}$): Given two ciphertexts $\mathsf{ct}_i \in \left(\prod_{i=0}^{\ell} R_{q_i} \right)^{d+1}$ at level ℓ, compute

$$\hat{\mathsf{ct}} = \mathsf{ct}_1 \otimes \mathsf{ct}_2 \in \left(\prod_{i=0}^{\ell} R_{q_i} \right)^{(d+1)^2}$$

and return the ciphertext

$$\overline{\mathsf{ct}} \leftarrow \text{MK-CKKS.Relin}(\hat{\mathsf{ct}}; \{(mk_i, b_i)\}_{0 \le i \le d-1})$$

as described in Algorithm 1.

- MK-CKKS.Rescale(ct): For a ciphertext

$$\mathsf{ct} = (c_0 = (c_0^{(j)}), \ldots, c_d = (c_d^{(j)})) \in \left(\prod_{j=0}^{\ell} R_{q_j} \right)^{d+1}$$

Algorithm 1 Relinearization for MK-CKKS (adapted From [9])

Input : $\hat{\mathsf{ct}} = (\hat{c}_{i,j})_{0\le i,j\le d}, \{(mk_i, b_i)\}_{0\le i\le d-1}$.
Output : $\overline{\mathsf{ct}} = (\overline{c}_i)_{0\le i\le d} \in \left(\prod_{i=0}^{\ell} R_{q_i}\right)^{d+1}$
$(c_i'')_{0\le i\le d} \leftarrow 0$
for $0 \le i, j \le d-1$ **do**
  $c''_{i,j} \leftarrow \mathsf{ModUp}(\hat{c}_{i,j}) \cdot b_j$
  $c'_{i,j} \leftarrow \mathsf{ModDown}(c''_{i,j})$
  $c'_{i,j} \leftarrow \mathsf{ModUp}(c'_{i,j})$
  $(c''_i, c''_d) \leftarrow (c''_i, c''_d) + c'_{i,j} \cdot (mk_{i,0}, mk_{i,1})$
  $c''_j \leftarrow c''_j + \mathsf{ModUp}(\hat{c}_{i,j}) \cdot mk_{i,2}$
end for
$\overline{c}_d \leftarrow \hat{c}_{d,d} + \mathsf{ModDown}(c''_d)$
for $0 \le i \le d-1$ **do**
  $\overline{c}_i \leftarrow \hat{c}_{i,d} + \hat{c}_{d,i} + \mathsf{ModDown}(c''_i)$
end for

at level ℓ, compute

$$c'^{(j)}_i = q_\ell^{-1} \cdot \left(c_i^{(j)} - c_i^{(\ell)}\right) \mod q_j$$

for $i = 0, \ldots d$ and $0 \le j \le \ell - 1$ and return

$$\mathsf{ct}' = (c'_0, \ldots, c'_d) \in \left(\prod_{j=0}^{\ell-1} R_{q_j}\right)^{d+1}.$$

- $\mathsf{MK\text{-}CKKS.RotKeyGen}(t; s_i)$: For each i-th client and given rotation index $t \in \mathbb{Z}_{2N}^*$, generate a random polynomial $rk_{i,0} \leftarrow \prod_{j=0}^{k-1} R_{p_j} \times \prod_{j=0}^{L} R_{q_j}$ and an error $e \leftarrow \chi_{\mathsf{err}}$. Set the rotation key as

$$rk_i = (rk_i^{(j)} = (rk_{i,0}^{(j)}, rk_{i,1}^{(j)}))_{0\le j\le k+L},$$

where

$$rk_{i,1}^{(j)} \leftarrow -rk_{i,0}^{(j)} \cdot s_i + e^{(j)} \mod p_j$$

for $0 \le j < k$ and

$$rk_{i,1}^{(k+j)} \leftarrow -rk_{i,0}^{(k+j)} \cdot s_i + e^{(k+j)} + [P]_{q_j} \cdot \tau_t(s_i) \mod q_j$$

for $0 \le j \le L$.

- $\mathsf{MK\text{-}CKKS.Rot}(\mathsf{ct}, \{rk_i\}_{0\le i\le d-1})$: Given a ciphertext $\mathsf{ct} = (c_0, \ldots, c_d)$ at level q_ℓ, compute $\tau_t(\mathsf{ct}) = (\tau_t(c_i))_{0\le i\le d}$ and return the ciphertext $\overline{\mathsf{ct}}$ as described in Algorithm 2.

To decrypt the ciphertext related to multiple clients, authors in [9] use the distributed decryption. The distributed decryption consists of two algorithms: partial decryption and merge. First, each i-th client receives the i-th entry of a ciphertext and decrypts it with noise. Subsequently, the partially decrypted results are merged with c_d to recover the message.

- $\mathsf{MK\text{-}CKKS.ParDec}(\mathsf{ct}, s_i)$: For each i-th client, given a ciphertext $\mathsf{ct} = (c_0, \ldots, c_d)$, and a secret s_i, sample an error $e_i \leftarrow \chi_{\mathsf{err}}$ and return

$$\mu_i = c_i \cdot s_i + e_i \mod q_0.$$

Algorithm 2 Rotation for MK-CKKS (adapted From [9])

Input: $\tau_t(\mathsf{ct}) = (\tau_t(c_i))_{0\le i\le d}, \{rk_i\}_{0\le i\le d-1}$
Output: $\overline{\mathsf{ct}} = (\overline{c}_i)_{0\le i\le d}$
for $0 \le i \le d-1$ **do**
  $c'_i \leftarrow \mathsf{ModUp}(\tau_t(c_i))$
  $\overline{c}_d \leftarrow \overline{c}_d + \mathsf{ModDown}(c'_i \cdot rk_{i,1})$
  $c'_i \leftarrow c'_i \cdot rk_{i,0}$
  $\overline{c}_i \leftarrow \mathsf{ModDown}(c'_i)$
end for
$\overline{c}_d \leftarrow \overline{c}_d + \tau_t(c_d)$

- $\mathsf{MK\text{-}CKKS.Merge}(\mathsf{ct}, \{\mu_i\}_{0\le i\le d-1})$: Compute and return

$$\mu = \sum_{i=0}^{d-1} \mu_i + c_d \mod q_0.$$

D. THRESHOLD MULTI-KEY RNS-CKKS

In MK-HE, ciphertext expansion occurs as homomorphic operation proceeds. This expansion is proportional to the number of clients. Also, MK-HE is possible only when all clients' public keys are possessed in the outsourcing server. A partial solution to overcome the ciphertext expansion is for a pre-defined number of clients to generate a common public key. MK-HE that achieves multi-key security and no ciphertext expansion is called a threshold MK-HE (TMK-HE). This subsection introduces TMK-CKKS adapted from [14]. In [14], TMK-CKKS generates a common public key using the prior communication between clients. A common public key is generated as follows.

- $\mathsf{TMK\text{-}CKKS.ComPK}(pk_0, \ldots, pk_{d-1})$: Given all clients' public keys

$$pk_i = (pk_i^{(j)} = (a^{(j)}, b_i^{(j)}))_{0\le j\le k+L},$$

return a common public key

$$\hat{pk} = (\hat{pk}^{(j)} = (a^{(j)}, b^{(j)}))_{0\le j\le k+L},$$

where $b^{(j)} := \sum_{i=0}^{d-1} b_i^{(j)}$.

After generating a common public key through the prior communication between clients, encryption is performed through a common public key. In addition, each client generates the evaluation keys by using the common public key. The following algorithms are for the encryption and the evaluation key generation.

- $\mathsf{TMK\text{-}CKKS.Enc}(\mathbf{z}; \hat{pk} = (\hat{pk}^{(j)} = (a^{(j)}, b^{(j)})))$: For each client and a message $\mathbf{z} \in \mathbb{C}^{N/2}$ and the scaling factor Δ, generate the message polynomial by

$$m(X) = \mathsf{MK\text{-}CKKS.Ecd}(\mathbf{z}; \Delta).$$

Then, sample $v \leftarrow \chi_{\mathsf{sec}}$ and $e_0, e_1 \leftarrow \chi_{\mathsf{err}}$ and generate the ciphertext

$$\mathsf{ct} = \left(\mathsf{ct}^{(j)} = \lfloor P^{-1} \cdot (c_0^{(j)}, c_1^{(j)}) \rceil + (0, m) \mod q_j\right),$$

where $(c_0^{(j)}, c_1^{(j)}) = v \cdot pk^{(j)} + (e_0, e_1)$ for $0 \le j \le L$.

- TMK-CKKS.MultKeyGen($\hat{pk} = (a, b); s_i$): For each i-th client, generate the multiplication key $mk_i = (mk_{i,0}, mk_{i,1})$ as follows:
  1) Sample $r_i \leftarrow \chi_{\text{sec}}$
  2) Sample $e_i \leftarrow \chi_{\text{err}}$ and
$$mk_{i,0}^{(j)} = a^{(j)} \cdot r_i^{(j)} + e_i^{(j)} \mod p_j$$
for $0 \le j < k$ and
$$mk_{i,0}^{(k+j)} = a^{(k+j)} \cdot r_i^{(k+j)} + [P]_{q_j} \cdot s_i + e_i^{(k+j)} \mod q_j$$
for $0 \le j \le L$.
  3) Sample $e_i' \leftarrow \chi_{\text{err}}$ and
$$mk_{i,1}^{(j)} = b^{(j)} \cdot r_i^{(j)} + e'^{(j)}_i \mod p_j$$
for $0 \le j < k$ and
$$mk_{i,1}^{(k+j)} = b^{(k+j)} \cdot r_i^{(k+j)} + e'^{(k+j)}_i \mod q_j$$
for $0 \le j \le L$.
- TMK-CKKS.RotKeyGen($\hat{pk} = (a, b); t, s_i$): For each i-th client and rotation index t, generate the rotation key $rk_i = (rk_{i,0}, rk_{i,1})$ as follows:
  1) Sample $r_i \leftarrow \chi_{\text{sec}}$
  2) Sample $e_i \leftarrow \chi_{\text{err}}$ and
$$rk_{i,0}^{(j)} = a^{(j)} \cdot r_i^{(j)} + e_i^{(j)} \mod p_j$$
for $0 \le j < k$ and
$$rk_{i,0}^{(k+j)} = a^{(k+j)} \cdot r_i^{(k+j)} + e_i^{(k+j)} \mod q_j$$
for $0 \le j \le L$.
  3) Sample $e_i' \leftarrow \chi_{\text{err}}$ and
$$rk_{i,1}^{(j)} = b^{(j)} \cdot r_i^{(j)} + e'^{(j)}_i \mod p_j$$
for $0 \le j < k$ and
$$rk_{i,1}^{(k+j)} = b^{(k+j)} \cdot r_i^{(k+j)} + e'^{(k+j)}_i + [P]_{q_j} \cdot \tau_t(s_i) \mod q_j$$
for $0 \le j \le L$.

After the evaluation keys generated by each client are sent to the outsourcing server, the common evaluation keys can be generated in the outsourcing server as follows:

$$mk = \sum_{i=0}^{d-1} mk_i \quad \text{and} \quad rk = \sum_{i=0}^{d-1} rk_i.$$

Thus, we can perform the homomorphic operation so that ciphertext expansion does not occur as follows:

- TMK-CKKS.Add($\mathsf{ct}_1, \mathsf{ct}_2$): Given two ciphertexts $\mathsf{ct}_1, \mathsf{ct}_2 \in \left(\prod_{i=0}^{\ell} R_{q_i}\right)^2$ at level ℓ, return the ciphertext
$$\mathsf{ct}' = \mathsf{ct}_1 + \mathsf{ct}_2 \mod q_\ell.$$
- TMK-CKKS.Mult($\mathsf{ct}_1, \mathsf{ct}_2; mk$): Given two ciphertexts $\mathsf{ct}_1, \mathsf{ct}_2 \in \left(\prod_{i=0}^{\ell} R_{q_i}\right)^2$ at level ℓ, compute
$$\hat{\mathsf{ct}} = \mathsf{ct}_1 \otimes \mathsf{ct}_2 \in \left(\prod_{i=0}^{\ell} R_{q_i}\right)^4$$
and return the ciphertext
$$\overline{\mathsf{ct}} \leftarrow \text{TMK-CKKS.Relin}(\hat{\mathsf{ct}}; mk)$$
as described in Algorithm 3.

Algorithm 3 Relinearization for TMK-CKKS (adapted From [14])

Input: $\hat{\mathsf{ct}} = (\hat{c}_0, \hat{c}_1, \hat{c}_2, \hat{c}_3), mk = (mk_0, mk_1)$
Output: $\overline{\mathsf{ct}} = (\bar{c}_0, \bar{c}_1) \in \left(\prod_{i=0}^{\ell} R_{q_i}\right)^2$
$\bar{c}_0 \leftarrow \bar{c}_0 + \text{ModDown}(\text{ModUp}(\hat{c}_0) \cdot mk_0)$
$\bar{c}_1 \leftarrow \bar{c}_1 + \text{ModDown}(\text{ModUp}(\hat{c}_0) \cdot mk_1)$
$\bar{c}_0 \leftarrow \bar{c}_0 + \hat{c}_1 + \hat{c}_2$
$\bar{c}_1 \leftarrow \bar{c}_1 + \hat{c}_3$

- TMK-CKKS.Rot($\mathsf{ct}, rk = (rk_0, rk_1)$): Given a ciphertext $\mathsf{ct} = (c_0, c_1)$ at level q_ℓ, compute
$$\tau_t(\mathsf{ct}) = (\tau_t(c_0), \tau_t(c_1))$$
and return the ciphertext $\overline{\mathsf{ct}}$ as described in Algorithm 4.

Algorithm 4 Rotation for TMK-CKKS (adapted From [14])

Input: $\tau_t(\mathsf{ct}) = (\tau_t(c_0), \tau_t(c_1)), rk = (rk_0, rk_1)$
Output: $\overline{\mathsf{ct}} = (\bar{c}_0, \bar{c}_1) \in \left(\prod_{i=0}^{\ell} R_{q_i}\right)^2$
$\bar{c}_0 \leftarrow \text{ModDown}(\text{ModUp}(\tau_t(c_0)) \cdot rk_0)$
$\bar{c}_1 \leftarrow \text{ModDown}(\text{ModUp}(\tau_t(c_0)) \cdot rk_1)$
$\bar{c}_1 \leftarrow \bar{c}_1 + \tau_t(c_1)$

III. HARDNESS OF VARIANT OF RLWE

In this section, we propose a variant of RLWE, called ReRLWE. The hardness of RLWE is first demonstrated when the secret distribution is uniform [16]. However, RLWE also satisfies the hardness when the secret distribution is the error distribution. Through this fact, we can consider the RLWE sample by reusing the error as follows.

$$(a, b, c), \ b = a \cdot s + x, \ c = a \cdot x + e,$$

where $a \leftarrow R_q$, $s \leftarrow R_q$ and x, e are chosen from the error distribution. To prove the hardness of ReRLWE, we formally define this distribution as follows.

Definition 4 (ReRLWE Distribution): For a secret $s \leftarrow R_q$ from the uniform distribution over R_q and an error distribution ψ over R_q, a sample from ReRLWE distribution $\overline{A}_{s,\psi}$ over R_q^3 is generated by choosing $a \leftarrow R_q$ uniformly at random, choosing $x, e \leftarrow \psi$, and outputting (a, b, c), where

$$b = a \cdot s + x \mod qR$$
$$c = a \cdot x + e \mod qR.$$

Now, we define the ReRLWE problem. Similar to the RLWE problem, this problem asks to distinguish from the uniform distribution over R_q^3.

Definition 5 (ReRLWE Problem): The average-case decision version of the ReRLWE problem, denoted $\text{ReRLWE}_{q,\psi}$, is to distinguish with non-negligible advantage between the sample from $\overline{A}_{s,\psi}$ and the sample from the uniform distribution over R_q^3.

The following theorem is that the RLWE problem can be reduced to the ReRLWE problem. This means that ReRLWE is also an NP-hard problem, which is as hard as the RLWE problem.

Theorem 1: Let q be a prime and ψ be an error distribution. Assume that there exists an algorithm $\mathcal{A}$ to distinguish the $\text{ReRLWE}_{q,\psi}$ distribution from the uniform distribution over R_q^3. Then there exists an algorithm $\mathcal{B}$ to distinguish the $\text{RLWE}_{q,\psi}$ distribution from the uniform distribution over R_q^2.

Proof: Assume that $\mathcal{A}$ is a distinguisher of $\text{ReRLWE}_{q,\psi}$ with a non-negligible advantage. Then we can construct a distinguisher $\mathcal{B}$ against $\text{RLWE}_{q,\psi}$ as follows. $\mathcal{B}$ gets as inputs $a \in R_q$ and $b \in R_q$. Then $\mathcal{B}$ proceeds as follows.

(i) If a has no inverse, abort $\mathcal{B}$ and output reject.
(ii) $u \leftarrow R_q$
(iii) $c \leftarrow a^{-1} \cdot b + a \cdot u$
(iv) Output $\mathcal{A}(a, c, b)$.

If the input of $\mathcal{B}$ is distributed according to the uniform distribution over R_q^2, then c is also uniformly at random. If the input of $\mathcal{B}$ is distributed according to the RLWE distribution $A_{s,\psi}$ of the form $(a, b) = (a, a \cdot s + x)$, where $s, x \leftarrow \psi$, we have

$$\begin{aligned} c &= a^{-1} \cdot b + a \cdot u \\ &= a^{-1}(a \cdot s + x) + a \cdot u \\ &= s + a^{-1} \cdot x + a \cdot u \\ &= s + a \cdot (a^{-2} \cdot x + u). \end{aligned}$$

Denote $s' = a^{-2} \cdot x + u$. Then s' is uniformly at random and independent of x since s' and $a^{-2} \cdot x$ are independent from Remark 2. Then $c = a \cdot s' + s$ and $(a, c, b) = (a, a \cdot s' + s, a \cdot s + x)$, which has the $\text{ReRLWE}_{q,\psi}$ distribution. Thus, we conclude that $\mathcal{B}$ has the same advantage as $\mathcal{A}$, which contradicts the hardness of $\text{RLWE}_{q,\psi}$. ∎

Remark 1: In MK-CKKS and TMK-CKKS, we use the prime modulus q satisfying $q \equiv 1 \mod 2N$ and $q \gg 2N$. Then we obtain $R_q \simeq \mathbb{Z}_q^N$ by the number-theoretic transformation [24]. For $(c_0, \ldots, c_N) \in \mathbb{Z}_q^N$, if $c_i \neq 0$ for all $i = 0, \ldots, N-1$, then $(c_0, \ldots, c_{N-1})$ has an inverse for element-wise product. This means that

$$\begin{aligned} \Pr[a \in R_q \text{ has an inverse in } R_q] &= \left(1 - \frac{1}{q}\right)^N \\ &\gg \left(1 - \frac{1}{2N}\right)^N \\ &\geq e^{-\frac{1}{2}}, \end{aligned}$$

which e is the Euler's constant. The last inequality can be obtained as $N \to \infty$. Thus, the probability $a \in R_q$ has an inverse is non-negligible.

Remark 2: Let U be a uniform distribution over R_q. Now, we will prove that $a^{-2} \cdot x + u$ is uniformly at random and satisfies the perfect secrecy. Let $k = a^{-2} \cdot x$ and $c = k + u$. Then, we obtain

$$\begin{aligned} \Pr[C = c] &= \sum_u \Pr[C = c \wedge U = u] \\ &= \sum_u \Pr[K = c - u \wedge U = u] \\ &= \sum_u \Pr[K = c - u] \Pr[U = u]. \end{aligned}$$

The third equality holds because the distribution K and U are independent. Since u runs through all possible elements in R_q, $c - u$ also runs through all possible elements in R_q. This means that

$$\sum_u \Pr[K = c - u] = 1.$$

Therefore, we obtain

$$\begin{aligned} \Pr[C = c] &= \sum_u \Pr[K = c - u] \Pr[U = u] \\ &= \sum_u \Pr[K = c - u] \left(\frac{1}{q^N}\right) \\ &= \frac{1}{q^N}. \end{aligned}$$

Therefore, $a^{-2} \cdot x + u$ is uniformly at random. Also, the independence of $a^{-2} \cdot x$ and u yields

$$\begin{aligned} \Pr[K = k | C = c] &= \frac{\Pr[C = c | K = k] \Pr[K = k]}{\Pr[C = c]} \\ &= \frac{\Pr[U = u] \Pr[K = k]}{\Pr[C = c]} \\ &= \frac{\frac{1}{q^N} \cdot \Pr[K = k]}{\frac{1}{q^N}} \\ &= \Pr[K = k]. \end{aligned}$$

This means that $a^{-2} \cdot x + u$ satisfies the perfect secrecy. Therefore, $a^{-2} \cdot x + u$ and $a^{-2} \cdot x$ are independent. Since a^{-2} is known, $a^{-2} \cdot x + u$ and x are independent.

IV. MULTI-KEY AND THRESHOLD MULTI-KEY RNS-CKKS UNDER ReRLWE ASSUMPTION

In this section, we introduce the modified evaluation keys, called modified multiplication and rotation keys, for MK-CKKS and TMK-CKKS schemes under the ReRLWE assumption. The error used for generating the public key can be reused when generating the modified multiplication key. Also, we need many rotation keys to operate the bootstrapping for MK-CKKS and TMK-CKKS. For two rotation indices $t_1, t_2 \in \mathbb{Z}_{2N}^*$, two RLWE samples are required. However, by reusing error, a modified rotation key can be generated through on ReRLWE sample for two rotation indices t_1

and t_2. And thus, we can generate fewer keys while reusing the error of the RLWE sample.

A. MULTI-KEY RNS-CKKS UNDER ReRLWE ASSUMPTION

In this subsection, we introduce the modified evaluation keys for the MK-CKKS scheme under the ReRLWE assumption, called ReMK-CKKS. Most of the algorithms in Subsection II-D are the same, but the KeyGen, Relin, RotKeyGen, and Rot algorithms are modified as follows.

- ReMK-CKKS.KeyGen: Each i-th client samples the secret key $s_i \leftarrow \chi_{\text{sec}}$, an error $x_i, e_i \leftarrow \chi_{\text{err}}$ and sets the public key as

$$pk_i = \left(pk_i^{(j)} = (a^{(j)}, b_i^{(j)})\right)_{0\le j\le k+L},$$

where

$$b_i^{(j)} \leftarrow -a^{(j)} \cdot s_i + x_i^{(j)} \mod p_j$$

for $0 \le j < k$, and

$$b_i^{(k+j)} \leftarrow -a^{(k+j)} \cdot s_i + x_i^{(k+j)} \mod q_j$$

for $0 \le j \le L$. Set the modified multiplication key as

$$mk_i^{(j)} = a^{(j)} \cdot x_i^{(j)} + e_i^{(j)} \mod p_j$$

for $0 \le j < k$ and

$$mk_i^{(k+j)} = a^{(k+j)} \cdot x_i^{(k+j)} + e_i^{(k+j)} + [P]_{q_j} \cdot s_i \mod q_j$$

for $0 \le j \le L$.

- ReMK-CKKS.Mult($\mathsf{ct}_1, \mathsf{ct}_2; \{(mk_i, pk_i)\}_{0\le i<d}$): Given two ciphertexts $\mathsf{ct}_1, \mathsf{ct}_2 \in \left(\prod_{i=0}^{\ell} R_{q_i}\right)^{d+1}$ at level ℓ, compute

$$\hat{\mathsf{ct}} = \mathsf{ct}_1 \otimes \mathsf{ct}_2 \in \left(\prod_{i=0}^{\ell} R_{q_i}\right)^{(d+1)^2}$$

and return the ciphertext

$$\overline{\mathsf{ct}} \leftarrow \text{ReMK-CKKS.Relin}(\hat{\mathsf{ct}}; \{(mk_i, pk_i)\}_{0\le i<d})$$

as described in Algorithm 5.
We generate three components

$$mk_i = (mk_{i,0}, mk_{i,1}, mk_{i,2})$$

to perform the homomorphic multiplication in MK-CKKS, but in ReMK-CKKS, only one component mk_i needs to be generated by reusing the error used in the public key.

- ReMK-CKKS.RotKeyGen($t_1, t_2; s_i$): For each i-th client and given rotation indices $t_1, t_2 \in \mathbb{Z}_{2N}^*$, generate a random polynomial $rk_{i,0} \leftarrow \prod_{i=0}^{k-1} R_{p_i} \times \prod_{i=0}^{L} R_{q_i}$ and an error $x_i, e_i \leftarrow \chi_{\text{err}}$. Sets the rotation key as

$$rk_i = (rk_i^{(j)} = (rk_{i,0}^{(j)}, rk_{i,1}^{(j)}, rk_{i,2}^{(j)}))_{0\le j\le k+L},$$

where

$$rk_{i,1}^{(j)} \leftarrow -rk_{i,0}^{(j)} \cdot s_i + x_i^{(j)} \mod p_j$$

for $0 \le j < k$ and

$$rk_{i,1}^{(k+j)} \leftarrow -rk_{i,0}^{(k+j)} \cdot s_i + x_i^{(k+j)} + [P]_{q_j} \cdot \tau_{t_1}(s_i) \mod q_j$$

for $0 \le j \le L$, and

$$rk_{i,2}^{(j)} \leftarrow -rk_{i,0}^{(j)} \cdot x_i^{(j)} + e_i^{(j)} \mod p_j$$

for $0 \le j < k$ and

$$rk_{i,2}^{(k+j)} \leftarrow -rk_{i,0}^{(k+j)} \cdot x_i^{(k+j)} + e_i^{(k+j)} + [P]_{q_j} \cdot \left(\tau_{t_2}(s_i) - \tau_{t_1}(s_i) \cdot rk_{i,0}^{(k+j)}\right) \mod q_j$$

for $0 \le j \le L$.

- ReMK-CKKS.Rot($\mathsf{ct}, \{rk_i\}_{0\le i<d}$): Given a ciphertext $\mathsf{ct} = (c_0, \ldots, c_d)$ at level q_ℓ, compute $\tau_t(\mathsf{ct}) = (\tau_t(c_i))_{0\le i\le d}$ and return the ciphertext $\overline{\mathsf{ct}}$ as described in Algorithm 6.

Algorithm 5 Modified Relinearization for ReMK-CKKS

Input : $\hat{\mathsf{ct}} = (\hat{c}_{i,j})_{0\le i,j\le d}, a, \{(mk_i, pk_i = (a, b_i))\}_{0\le i\le d-1}$.
Output : $\overline{\mathsf{ct}} = (\bar{c}_i)_{0\le i\le d} \in \left(\prod_{i=0}^{\ell} R_{q_i}\right)^{d+1}$
for $0 \le i, j \le d-1$ **do**
  $\hat{c}_{i,j} \leftarrow \text{ModUp}(\hat{c}_{i,j})$
  $\hat{c}'_{i,j} \leftarrow \hat{c}_{i,j} \cdot b_j$
  $\bar{c}_i \leftarrow \bar{c}_i + \hat{c}'_{i,j} \cdot a$
  $\bar{c}_j \leftarrow \bar{c}_j + \hat{c}_{i,j} \cdot mk_i$
  $\bar{c}_d \leftarrow \bar{c}_d + \hat{c}'_{i,j} \cdot b_i$
end for
for $0 \le i \le d$ **do**
  $\bar{c}_i \leftarrow \text{ModDown}(\bar{c}_i)$
  if $i \le d-1$ **then**
    $\bar{c}_i \leftarrow \bar{c}_i + \hat{c}_{d,i} + \hat{c}_{i,d}$
  else
    $\bar{c}_d \leftarrow \bar{c}_d + \hat{c}_{d,d}$
  end if
end for

We generate $rk_i = (rk_{i,0}, rk_{i,1})$ and $rk'_i = (rk'_{i,0}, rk'_{i,1})$ for two rotation indices $t_1, t_2 \in \mathbb{Z}_{2N}^*$ in MK-CKKS. However, in ReMK-CKKS, we generate $rk_i = (rk_{i,0}, rk_{i,1}, rk_{i,2})$ for two rotation indices $t_1, t_2 \in \mathbb{Z}_{2N}^*$. Also, unlike the rotation operation of MK-CKKS, ReMK-CKKS must generate $f_1 := rk_{i,0} \cdot rk_{i,0}$ and $f_2 := rk_{i,0} \cdot rk_{i,1} + rk_{i,2}$ for the rotation index $t_2 \in \mathbb{Z}_{2N}^*$. And thus, the rotation operation time of ReMK-CKKS increases slightly.

B. THRESHOLD MULTI-KEY RNS-CKKS UNDER ReRLWE ASSUMPTION

In this section, we introduce the modified evaluation keys for the TMK-CKKS scheme under the ReRLWE assumption, called ReTMK-CKKS. In [14], common public key was generated through communication between clients to resolve the expansion of ciphertext. In addition, by sending

Algorithm 6 Modified Rotation for ReMK-CKKS

Input: $\tau_t(\mathsf{ct}) = (\tau_t(c_i))_{0\le i\le d}, \{rk_i = (rk_{i,0}, rk_{i,1}, rk_{i,2})\}_{0\le i\le d-1}$
Output: $\overline{\mathsf{ct}} = (\bar{c}_i)_{0\le i\le d}$
$\tilde{c} \leftarrow 0$
if $t = t_1$ **then**
  for $0 \le i \le d-1$ **do**
    $\tau_t(c_i) \leftarrow \mathsf{ModUp}(\tau_t(c_i))$
    $\bar{c}_i \leftarrow \tau_t(c_i) \cdot rk_{i,0}$
    $\tilde{c} \leftarrow \tilde{c} + \tau_t(c_i) \cdot rk_{i,1}$
    $\bar{c}_i \leftarrow \mathsf{ModDown}(\bar{c}_i)$
  end for
else if $t = t_2$ **then**
  for $0 \le i \le d-1$ **do**
    $\tau_t(c_i) \leftarrow \mathsf{ModUp}(\tau_t(c_i))$
    $\bar{c}_i \leftarrow \tau_t(c_i) \cdot rk_{i,0}^2$
    $\tilde{c} \leftarrow \tilde{c} + \tau_t(c_i) \cdot (rk_{i,0} \cdot rk_{i,1} + rk_{i,2})$
    $\bar{c}_i \leftarrow \mathsf{ModDown}(\bar{c}_i)$
  end for
end if
$\tilde{c} \leftarrow \mathsf{ModDown}(\tilde{c})$
$\bar{c}_d \leftarrow \bar{c}_d + \tilde{c}$

the evaluation keys generated by users to the server, a common evaluation keys are generated to lower the communication cost between clients. Although many operations in the proposed scheme are similar to those in [14], the algorithms for generating the modified evaluation keys are different. This generating method is used in Subsection IV-A. Also, to reduce the rotation keys, we modify the setup algorithm for ReTMK-CKKS, that is, we consider a more common reference string. In this way, the size of the evaluation keys can be reduced compared to that in [14]. The following operation is the modified setup.

- $\mathsf{ReTMK\text{-}CKKS.Setup}(1^\lambda)$: Given a security parameter λ, set

$$pp \leftarrow \mathsf{MK\text{-}CKKS.Setup}(1^\lambda)$$

for all $0 \le j \le L$ and $0 \le i < k$. Let $w = |\mathbb{Z}_{2N}^*|/2$. Generate random polynomials

$$a_1, a_2 \ldots, a_{w-1}, a_w \leftarrow \prod_{i=0}^{k-1} R_{p_i} \times \prod_{i=0}^{L} R_{q_i}.$$

Return the public parameter $\overline{pp} = (pp, \{a_i\}_{1\le i\le w})$.

The ReTMK-CKKS scheme also generates modified common evaluation keys. At this time, to generate common rotation keys, we use one of the common reference strings, a_i. That is, in $\mathsf{ReMK\text{-}CKKS.RotKeyGen}$, each i-th client uses the common reference strings a_j as $rk_{i,0}$ for some $j \in \{1, 2, \ldots, w\}$. The generation method of modified common evaluation keys is as follows.

- $\mathsf{ReTMK\text{-}CKKS.ComMultKey}(mk_0, \ldots, mk_{d-1})$: Given all client's modified multiplication keys mk_i, the server generates a common modified multiplication key

$$mk = \sum_{i=0}^{d-1} mk_i.$$

- $\mathsf{ReTMK\text{-}CKKS.ComRotKey}(rk_0, \ldots, rk_{k-1})$: Given all clients' modified rotation keys $rk_i = (rk_{i,0} = a_j, rk_{i,1}, rk_{i,2})$, the server generates a common modified rotation key $rk = (rk_0 = a_j, rk_1, rk_2)$ as

$$rk_1 = \sum_{i=0}^{d-1} rk_{i,1} \text{ and } rk_2 = \sum_{i=0}^{d-1} rk_{i,2}.$$

We perform homomorphic multiplication and rotation algorithms using modified common multiplication and rotation keys as follows.

- $\mathsf{ReTMK\text{-}CKKS.Mult}(\mathsf{ct}_1, \mathsf{ct}_2; pk = (a, b), mk)$: Given two ciphertexts $\mathsf{ct}_1, \mathsf{ct}_2 \in \left(\prod_{i=0}^{\ell} R_{q_i}\right)^2$ at level ℓ, compute

$$\hat{\mathsf{ct}} = \mathsf{ct}_1 \otimes \mathsf{ct}_2 \in \left(\prod_{i=0}^{\ell} R_{q_i}\right)^4$$

and return the ciphertext

$$\overline{\mathsf{ct}} \leftarrow \mathsf{MK\text{-}CKKS.Relin}(\hat{\mathsf{ct}}; mk) \in \left(\prod_{i=0}^{\ell} R_{q_i}\right)^2$$

as described in Algorithm 7.

Algorithm 7 Modified Relinearization for ReTMK-CKKS

Input: $\hat{\mathsf{ct}} = \hat{c}_0, \hat{c}_1, \hat{c}_2, \hat{c}_3), \{pk = (a, b), mk\}$
Output: $\overline{\mathsf{ct}} = (\bar{c}_0, \bar{c}_1) \in \left(\prod_{i=0}^{\ell} R_{q_i}\right)^2$
$\hat{c}_0 \leftarrow \mathsf{ModUp}(\hat{c}_0)$
$\bar{c}_0 \leftarrow \hat{c}_1 + \hat{c}_2 + \mathsf{ModDown}(\hat{c}_0 \cdot (ab + mk))$
$\bar{c}_1 \leftarrow \hat{c}_3 + \mathsf{ModDown}(\hat{c}_0 \cdot b^2)$

- $\mathsf{ReTMK\text{-}CKKS.Rot}(\mathsf{ct}, rk = (rk_0, rk_1, rk_2))$: Given a ciphertext $\mathsf{ct} = (c_0, c_1)$ at level ℓ, compute

$$\tau_t(\mathsf{ct}) = (\tau_t(c_0), \tau_t(c_1))$$

and return the ciphertext $\bar{\mathsf{ct}}$ as described in Algorithm 8.

In the TMK-CKKS, the multiplication key is generated as $mk = (mk_0, mk_1)$ and two rotation keys are generated as $rk = (rk_0, rk_1)$ and $rk' = (rk'_0, rk'_1)$ for two rotation indices $t_1, t_2 \in \mathbb{Z}_{2N}^*$. However, in ReTMK-CKKS, mk is generated as only one component and $rk = (rk_0, rk_1, rk_2)$ is generated for two rotations $t_1, t_2 \in \mathbb{Z}_{2N}^*$ under ReRLWE assumption (we can think $(pk = (a, b), mk)$ as a ReRLWE sample). Unlike Subsection IV-A, the multiplication operation time increases slightly because $f_1 := ab + mk$ and $f_2 := b^2$ have to be generated. In the case of rotation operation, the operation time increases as in Subsection IV-A.

Algorithm 8 Modified Rotation for ReTMK-CKKS

Input : $\tau_t(\mathsf{ct}) = (\tau_t(c_0), \tau_t(c_1)), rk = (rk_0 = a_j, rk_1, rk_2)$
Output : $\overline{\mathsf{ct}} = (\bar{c}_0, \bar{c}_1) \in \left(\prod_{i=0}^{\ell} R_{q_i}\right)^2$
if $t = t_1$ **then**
  $\tau_t(c_0) \leftarrow \mathsf{ModUp}(\tau_t(c_0))$
  $\bar{c}_0 \leftarrow \mathsf{ModDown}(\tau_t(c_0) \cdot rk_0)$.
  $\bar{c}_1 \leftarrow \hat{c}_1 + \mathsf{ModDown}(\tau_t(c_0) \cdot rk_1)$.
else if $t = t_2$ **then**
  $\tau_t(c_0) \leftarrow \mathsf{ModUp}(\tau_t(c_0))$
  $\bar{c}_0 \leftarrow \mathsf{ModDown}(\tau_t(c_0) \cdot a_j^2)$.
  $\bar{c}_1 \leftarrow \hat{c}_1 + \mathsf{ModDown}(\tau_t(c_0) \cdot (rk_2 + rk_0 \cdot rk_1))$.
end if

V. CORRECTNESS, SECURITY, AND COMPARISON

A. CORRECTNESS

In this subsection, we will show that the multiplication and rotation in the proposed ReMK-CKKS and ReTMK-CKKS schemes satisfy the correctness.

Theorem 2: ReMK-CKKS.Mult and ReMK-CKKS.Rot are correct.

Proof: First, we will show the correctness of ReMK-CKKS.Mult. Let $\mathsf{sk} = (s_0, \ldots, s_{d-1}, 1)$ be a secret key and let ct_1 and ct_2 be ciphertexts for messages m_1 and m_2, respectively. Let $\overline{\mathsf{ct}}_{\mathsf{mult}}$ be a ciphertext after ReMK-CKKS.Mult between ct_1 and ct_2. In lines 2–9 of Algorithm 5, $\mathsf{ModDown}(\hat{c}_{i,j} \cdot a \cdot b_j)$ is added to the i-th component of $\overline{\mathsf{ct}}_{\mathsf{mult}}$, and $\mathsf{ModDown}(\hat{c}_{i,j} \cdot mk_i)$ is added to the j-th component of $\overline{\mathsf{ct}}_{\mathsf{mult}}$. In addition, $\mathsf{ModDown}(\hat{c}_{i,j} b_i b_j)$ is added to c_d of $\overline{\mathsf{ct}}_{\mathsf{mult}}$. And thus, when decryption is performed, we first calculate the following equation.

$$\begin{aligned}
&\hat{c}_{i,j} \cdot a \cdot b_j \cdot s_i + \hat{c}_{i,j} \cdot mk_i \cdot s_j \\
&= \hat{c}_{i,j} \cdot (a \cdot (-a \cdot s_j + x_j) \cdot s_i + (a \cdot x_i + e_i + P \cdot s_i) \cdot s_j) \\
&= \hat{c}_{i,j} \cdot (-a^2 \cdot s_i \cdot s_j + a \cdot s_i \cdot x_j + a \cdot s_j \cdot x_i \\
&\quad + e_i \cdot s_j + P \cdot s_i \cdot s_j),
\end{aligned}$$

and

$$\hat{c}_{i,j} \cdot b_i \cdot b_i = \hat{c}_{i,j} \cdot (-a \cdot s_i + x_i) \cdot (-a \cdot s_j + x_j)$$

Thus, we obtain

$$\begin{aligned}
&\hat{c}_{i,j} \cdot a \cdot b_j \cdot s_i + \hat{c}_{i,j} \cdot mk_i \cdot s_j + \hat{c}_{i,j} \cdot b_i \cdot b_i \\
&= \hat{c}_{i,j} \cdot (P \cdot s_i \cdot s_j + e_i \cdot s_j + x_i \cdot x_j)
\end{aligned}$$

and through the ModDown operations, we obtain

$$\hat{c}_{i,j} \cdot s_i \cdot s_j + P^{-1} \cdot \hat{c}_{i,j} \cdot (e_i \cdot s_j + x_i \cdot x_j).$$

Therefore, we obtain

$$\begin{aligned}
\langle \overline{\mathsf{ct}}_{\mathsf{mult}}, \mathsf{sk} \rangle &\approx \langle \mathsf{ct}_1 \otimes \mathsf{ct}_2, \mathsf{sk} \otimes \mathsf{sk} \rangle \\
&= \langle \mathsf{ct}_1, \mathsf{sk} \rangle \cdot \langle \mathsf{ct}_2, \mathsf{sk} \rangle \\
&\approx m_1 \cdot m_2.
\end{aligned}$$

Now, we will show the correctness of ReMK-CKKS.Rot. Let $\overline{\mathsf{ct}}_{\mathsf{rot}}$ be a ciphertext after ReMK-CKKS.Rot for $\tau_t(\mathsf{ct})$. If $t = t_1$, in the lines 3–9 of Algorithm 6, $\mathsf{ModDown}(\tau_t(c_i) \cdot rk_{i,0})$ is added to the i-th component of $\overline{\mathsf{ct}}_{\mathsf{rot}}$ and $\mathsf{ModDown}(\tau_t(c_i) \cdot rk_{i,1})$ is added to the d-th component of $\overline{\mathsf{ct}}_{\mathsf{rot}}$. Thus, we obtain

$$\tau_t(c_i) \cdot \tau_t(s_i) + P^{-1} \cdot \tau_t(c_i) \cdot x_i$$

through the ModDown operation. Before we observe the case of $t = t_2$, we compute $rk_{i,2} + rk_{i,0} \cdot rk_{i,1}$ and obtain

$$rk_{i,2} + rk_{i,0} \cdot rk_{i,1} = -rk_{i,0}^2 \cdot s_i + e_i + P \cdot \tau_{t_2}(s_i).$$

Now, we observe the case of $t = t_2$. In the lines 11–19 of Algorithm 6, $\mathsf{ModDown}(\tau_t(c_i) \cdot rk_{i,0}^2)$ is added to the i-th component of $\overline{\mathsf{ct}}_{\mathsf{rot}}$ and $\mathsf{ModDown}(\tau_t(c_t) \cdot (rk_{i,2} + rk_{i,0} \cdot rk_{i,1}))$ is added to the d-th component of $\overline{\mathsf{ct}}_{\mathsf{rot}}$. Thus, we obtain

$$\tau_t(c_i) \cdot \tau_t(s_i) + P^{-1} \cdot \tau_t(c_i) \cdot e_i.$$

Therefore, we obtain

$$\langle \overline{\mathsf{ct}}_{\mathsf{rot}}, \mathsf{sk} \rangle \approx \langle \tau_t(\mathsf{ct}), \tau_t(\mathsf{sk}) \rangle \approx \tau_t(m).$$

■

Theorem 3: ReTMK-CKKS.Mult and ReTMK-CKKS.Rot are correct.

Proof: The correctness of ReTMK-CKKS.Rot is similar to the proof of the correctness of ReMK-CKKS.Rot. Therefore, we will only show the correctness of ThrReMK-CKKS.Mult.

Let $s = \sum_{i=0}^{d-1} s_i$ be the sum of secret keys for each client and $sk = (s, 1)$. Let $mk = \sum_{i=0}^{d-1} mk_i$ be the sum of the modified multiplication keys for each client. Let $\mathsf{ct} = (c_0, c_1)$ and $\mathsf{ct}' = (c_0', c_1')$ be ciphertexts corresponding to the messages m and m' with secret key sk, respectively. Let $\overline{\mathsf{ct}}^{\times} = (\bar{c}_0^{\times}, \bar{c}_1^{\times})$ be a ciphertext after ReTMK-CKKS.Mult between ct and ct'. Note that $\mathsf{ct} \otimes \mathsf{ct}' = (c_0 c_0', c_1 c_0', c_0 c_1', c_1 c_1')$. We first compute

$$\begin{aligned}
\langle (ab + mk, b^2), sk \rangle &= (ab + mk) \cdot s + b^2 \\
&= a \cdot s \cdot b + mk \cdot s + b^2 \\
&= (x - b) \cdot b + mk \cdot s + b^2 \\
&= x^2 + e \cdot s + P \cdot s^2.
\end{aligned}$$

Then, we obtain

$$\begin{aligned}
&c_0 c_0' \cdot (ab + mk) \cdot s + c_0 c_0' \cdot b^2 \\
&= c_0 c_0' \cdot \left((ab + mk) \cdot s + b^2\right) \\
&= c_0 c_0' \cdot \left(P \cdot s^2 + x^2 + e \cdot s\right),
\end{aligned}$$

and through the ModDown operation, we obtain

$$c_0 c_0' \cdot s^2 + P^{-1} \cdot c_0 c_0' \cdot (x^2 + e \cdot s).$$

Thus, the ciphertext $\overline{\mathsf{ct}}^{\times}$ satisfies that

$$\begin{aligned}
\langle \overline{\mathsf{ct}}^{\times}, sk \rangle &= \bar{c}_0^{\times} \cdot s + \bar{c}_1^{\times} \\
&\approx c_1 c_1' + (c_1 c_0' + c_0 c_1') \cdot s + c_0 c_0' \cdot s^2 \\
&\approx m \cdot m' \mod q_0.
\end{aligned}$$

■

B. SECURITY

In this subsection, we prove that the proposed scheme satisfies the indistinguishability under chosen-plaintext attack (IND-CPA) security. For convenience, we prove the theorem without considering the special modulus technique. We first show that the public key with the modified multiplication key is computationally indistinguishable from a uniform distribution over $\left(\prod_{i=0}^{L} R_{q_i}\right)^3$. Since it is similar to the case of the modified rotation keys, we omit this case.

Theorem 4: The distribution of public keys with the modified multiplication keys is computationally indistinguishable from a uniform distribution over $\left(\prod_{i=0}^{L} R_{q_i}\right)^3$ under the assumption of ReRLWE and circular security.

Proof: Let pp be the $\text{ReRLWE}_{q,\chi_{\text{err}}}$ parameters generated in ReMK-HE.Setup (This algorithm is the same as the MK-HE.Setup). We define the distribution $D_0 = \{a, b, mk\}$ over $\left(\prod_{i=0}^{L} R_{q_i}\right)^3$ as follows:

(i) $a \leftarrow \prod_{i=0}^{L} R_{q_i}$, $s \leftarrow \chi_{\text{sec}}$, $x \leftarrow \chi_{\text{err}}$, and $b = -a \cdot s + x$
(ii) $e \leftarrow \chi_{\text{err}}$ and $mk = a \cdot x + e + s$.

Now, we consider the distribution D_1 over $\left(\prod_{i=0}^{L} R_{q_i}\right)^3$, which is obtained from D_0 by modifying its definitions (i) and (ii) into

(i)' $a \leftarrow \prod_{i=0}^{L} R_{q_i}$ and $b \leftarrow \prod_{i=0}^{L} R_{q_i}$
(ii)' $mk \leftarrow \prod_{i=0}^{L} R_{q_i}$.

From Theorem 1 and the circular security, we obtain that D_0 and D_1 are computationally indistinguishable. ∎

Now, we will show that the ReMK-CKKS is IND-CPA secure under the ReRLWE assumption with parameter $pp \leftarrow$ ReMK-HE.Setup.

Theorem 5: Let $pp \leftarrow$ ReMK-HE.Setup be the ReRLWE parameter generated in the setup phase. Then the ReMK-CKKS is IND-CPA secure under the RLWE and ReRLWE assumptions with parameter pp.

Proof: Let $\mathcal{A}$ be an IND-CPA adversary for the ReMK-CKKS. We consider a series of hybrids, where $\mathsf{Adv}_H[\mathcal{A}]$ denotes the success probability of $\mathcal{A}$ in hybrid H.

- **Hybrid** H_0: This is identical to the IND-CPA game, where the adversary gets a distributed public key with the modified multiplication key generated by MK-CKKS.KeyGen. Also, the adversary gets encryption ct_0 and ct_1 of m_0 and m_1, respectively, computed using MK-CKKS.Enc. Note that the public key with the modified multiplication key consists of

$$(a, b, mk) := (a, -a \cdot s + x, a \cdot x + e + s),$$

where $a \leftarrow \prod_{i=0}^{L} R_{q_i}$, $s \leftarrow \chi_{\text{sec}}$, and $x, e \leftarrow \chi_{\text{err}}$. Assume that there is a polynomial $t(\cdot)$ such that

$$\begin{aligned}\mathsf{Adv}_{H_0}[\mathcal{A}] := |&\Pr[\mathcal{A}((a, b, mk), \mathsf{ct}_0) = 1] \\ &- \Pr[\mathcal{A}((a, b, mk), \mathsf{ct}_1) = 1]| > 1/t(\lambda).\end{aligned} \tag{1}$$

- **Hybrid** H_1: The hybrid H_1 is identical to H_0 except for that b of the public key and the modified multiplication key mk are chosen to be uniformly random from R_q.

In H_1, the public and modified multiplication keys are uniformly random. In addition, mk is independent of (c_0, c_1). Also, (a, c_0) and (b, c_1) are computationally indistinguishable from the uniform distribution over $\left(\prod_{i=0}^{L} R_{q_i}\right)^2$ since they can be viewed as two RLWE samples of secret v. Thus, we obtain that

$$\mathsf{Adv}_{H_1}[\mathcal{A}] = \mathsf{negl}(\lambda). \tag{2}$$

Now, we claim that

$$|\mathsf{Adv}_{H_0}[\mathcal{A}] - \mathsf{Adv}_{H_1}[\mathcal{A}]| \le \mathsf{negl}(\lambda). \tag{3}$$

A ciphertext is generated by adding an encoded plaintext to a random encryption of zero. Hence we consider the random variables (a, b, d, c_0, c_1) over $\left(\prod_{i=0}^{L} R_{q_i}\right)^5$ defined by

$$\begin{aligned} a &\leftarrow \prod_{i=0}^{L} R_{q_i} \\ b &\leftarrow -a \cdot s + x \in \prod_{i=0}^{L} R_{q_i} \\ mk &\leftarrow a \cdot x + e + s \in \prod_{i=0}^{L} R_{q_i}, \end{aligned}$$

where $s \leftarrow \chi_{\text{sec}}$, $x, e \leftarrow \chi_{\text{err}}$, and $(c_0, c_1) = v \cdot (a, b) + (e_0, e_1)$ for $v \leftarrow \chi_{\text{sec}}$ and $e_0, e_1 \leftarrow \chi_{\text{err}}$. Now, we change the definition of (b, mk) as

$$b \leftarrow \prod_{i=0}^{L} R_{q_i} \text{ and } mk \leftarrow \prod_{i=0}^{L} R_{q_i}.$$

Then it is computationally indistinguishable by the ReRLWE assumption with parameter pp from Theorem 4. This means that

$$|\mathsf{Adv}_{H_0}[\mathcal{A}] - \mathsf{Adv}_{H_1}[\mathcal{A}]| \le \mathsf{negl}(\lambda).$$

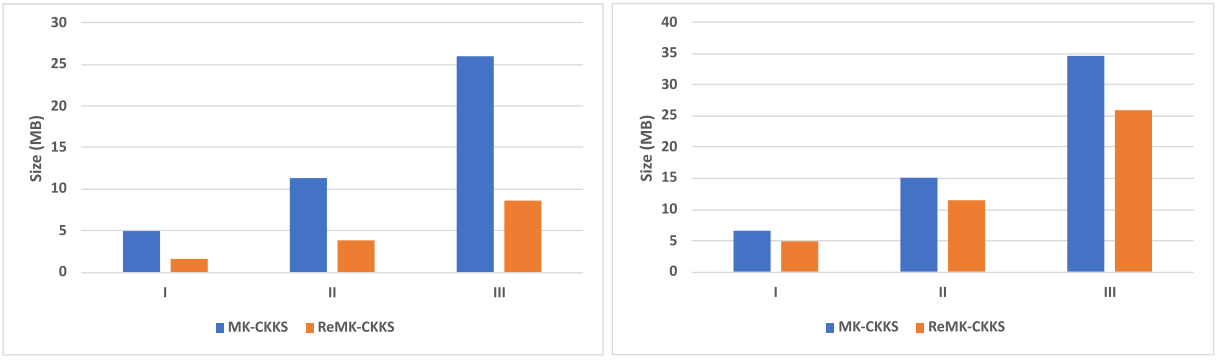
By combining (2) and (3), we obtain

$$\begin{aligned}\mathsf{Adv}_{H_0}[\mathcal{A}] \le \mathsf{Adv}_{H_1}[\mathcal{A}] &+ |\mathsf{Adv}_{H_0}[\mathcal{A}] \\ &- \mathsf{Adv}_{H_1}[\mathcal{A}]| = \mathsf{negl}(\lambda),\end{aligned}$$

which contradicts the result of (1). ∎

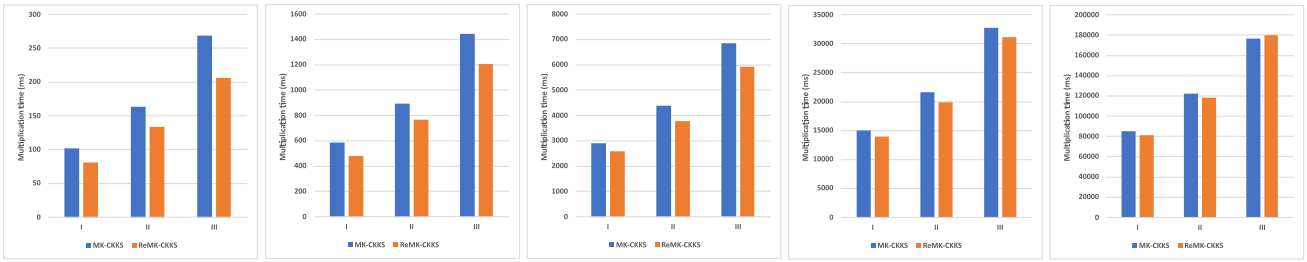
C. COMPARISON

In this subsection, the numerical results of the proposed ReMK-CKKS and ReTMK-CKKS schemes are compared with MK-CKKS adapted from [9] and TMK-CKKS adapted from [14], respectively. In our implementation, each number is stored as an unsigned 64-bit integer. Also, our implementation is based on the RNS-HEAAN library and is performed on a computer with AMD Ryzen Threadripper PRO 3995WX CPU @ 2.70GHz processor on a multi-threaded mode. Table 3 lists the parameters used in all schemes.



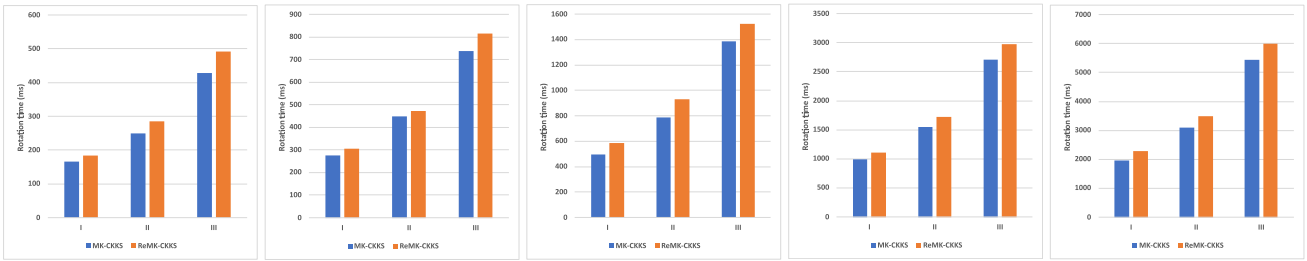
(a) The size of multipication key for one client (b) The size of rotation key of two rotation indices for one client

FIGURE 1. Comparison of the size of evaluation keys to be generated by one user for various parameter sets between MK-CKKS and ReMK-CKKS.



(a) 2 clients (b) 4 clients (c) 8 clients (d) 16 clients (e) 32 clients

FIGURE 2. Comparison of the time of homomoprhic multiplication for each parameter of MK-CKKS and ReMK-CKKS.



(a) 2 clients (b) 4 clients (c) 8 clients (d) 16 clients (e) 32 clients

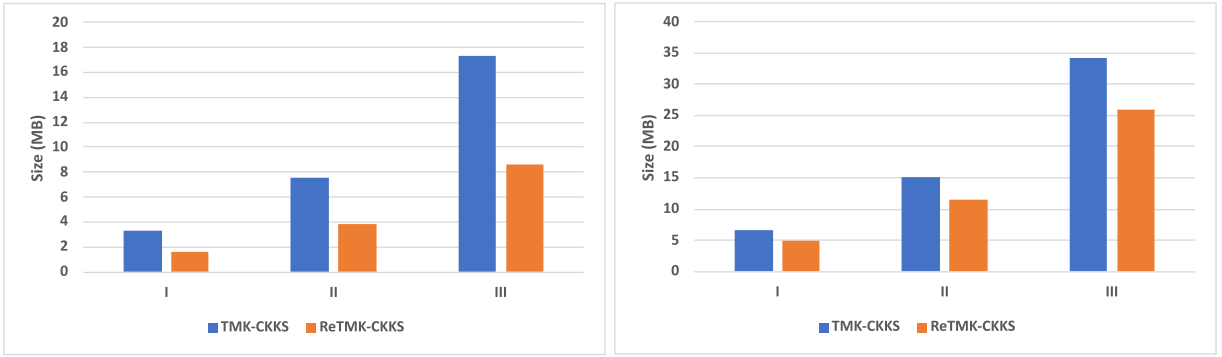
FIGURE 3. Comparison of the time of homomoprhic rotation for each parameter of MK-CKKS and ReMK-CKKS. The time of homomorphic rotation is calculated as the sum of homomorphic rotation times for the two rotation indices.

TABLE 3. Proposed parameter sets. L and k denote the number of RNS primes and the number of special primes, respectively. $\log q$ and $\log q_i$ denote the bit length of the largest RLWE modulus and individual RNS primes, respectively.

ID	$\log N$	L	k	$\log q$	$\log q_i$
I	13	12	13	660	55
II	14	14	15	756	55
III	15	16	17	881	55

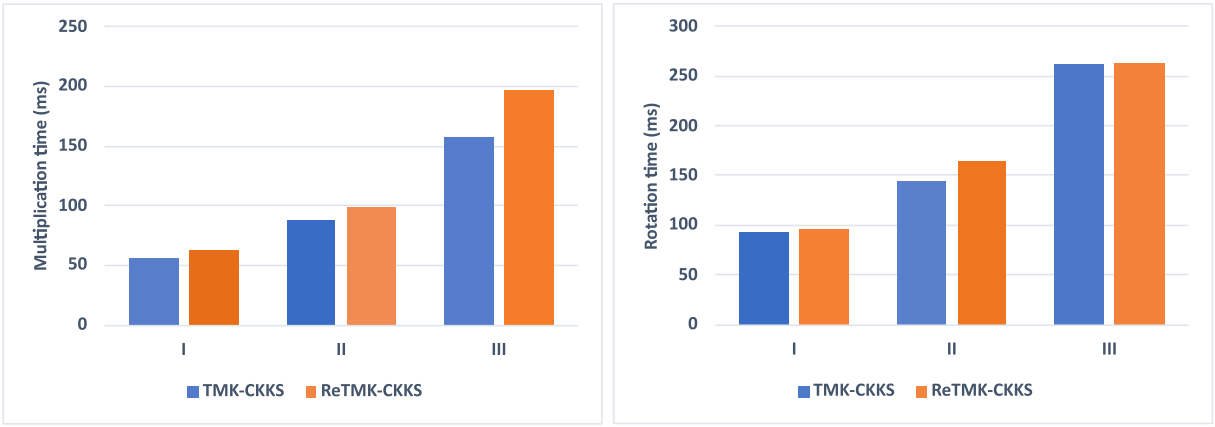
1) COMPARISON OF MK-CKKS AND ReMK-CKKS

In Fig. 1, the size of the keys for the MK-CKKS and the ReMK-CKKS schemes are compared. The size of the multiplication key of ReMK-CKKS can be reduced by 66% compared to that of MK-CKKS for each client. The size of rotation key for two rotation indices of ReMK-CKKS can be reduced by 25% compared to that of MK-CKKS for each client. Figs. 2 and 3 compare the homomorphic evaluation times of MK-CKKS and ReMK-CKKS. The multiplication operation time of ReMK-CKKS is slightly reduced compared to that of MK-CKKS from Figs. 2(a) to 2(d). This is because of the operations ModUp and ModDown of ReMK-CKKS.Relin are used fewer than those of MK-CKKS.Relin. However, the rotation operation time of ReMK-CKKS is slightly increased compared to that of MK-CKKS.



(a) The size of multiplication key for one client (b) The size of rotation key of two rotation indices for one client

FIGURE 4. Comparison of the size of multiplication and rotation key to be generated by one client for various parameter sets between TMK-CKKS and ReTMK-CKKS.



(a) The homomorphic multiplication time (b) The homomorphic rotation time

FIGURE 5. Comparison of the time of the homomorphic evaluation for each parameter of TMK-CKKS and ReTMK-CKKS.

2) COMPARISON OF TMK-CKKS AND ReTMK-CKKS

Since TMK-CKKS and ReTMK-CKKS schemes operate like a single-key HE, we simulate the case of two clients. In Fig. 4, the size of the evaluation keys for the TMK-CKKS and ReTMK-CKKS schemes are compared. The size of the multiplication key of ReTMK-CKKS can be reduced by 50% compared to that of TMK-CKKS. Also, as in the comparison of MK-CKKS and ReMK-CKKS, the size of rotation key for two rotation indices of ReTMK-CKKS can be reduced by 25% compared to that of TMK-CKKS. Fig. 5 compares the time of the homomorphic evaluation operations. The homomorphic evaluation operation time of ReTMK-CKKS is slightly increased than that of TMK-CKKS.

VI. CONCLUSION

In this paper, we first proposed a variant of RLWE by reusing the error defining the RLWE problem, called the ReRLWE problem. To define this problem, we first defined the ReRLWE distribution. Second, we proved that there exists a reduction from RLWE to ReRLWE, which means that ReRLWE is also NP-hard. Under the ReRLWE assumption, we proposed a modified MK-CKKS and TMK-CKKS with reduced evaluation key sizes, called ReMK-CKKS and ReTMK-CKKS. The sizes of the multiplication key of ReMK-CKKS and ReTMK-CKKS were reduced by approximately 66% compared to that of MK-CKKS and 50% compared to that of TMK-CKKS, respectively. In addition, the sizes of the rotation keys of ReMK-CKKS and ReTMK-CKKS were reduced by 25% of those of MK-CKKS and TMK-CKKS, respectively. Due to the reduced size of the evaluation key, it is possible to perform the multi-key homomorphic operation even though client's computer resources are limited. In the future, the key size can be reduced by generating a more compact ReRLWE sample in a cryptographic scheme that requires many RLWE samples. Additionally, we will demonstrate whether the problem is difficult in the case of continued error reuse.

REFERENCES

[1] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 21st Annu. ACM Symp. Theory Comput.*, May 2009, pp. 169–178.

[2] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology—ASIACRYPT* (Lecture Notes in Computer Science), vol. 10624. Cham, Switzerland: Springer, 2017, pp. 409–437.

[3] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "A full RNS variant of approximate homomorphic encryption," in *Proc. Int. Conf. Sel. Areas Cryptogr. (SAC)*, Calgary, AB, Canada, 2018, pp. 347–368.
[4] J.-W. Lee, E. Lee, Y. Lee, Y.-S. Kim, and J.-S. No, "High-precision bootstrapping of RNS-CKKS homomorphic encryption using optimal minimax polynomial approximation and inverse sine function," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science). Manhattan, NY, USA: Springer, 2021, pp. 618–647.
[5] W. Jung, S. Kim, J. H. Ahn, J. H. Cheon, and Y. Lee, "Over 100x faster bootstrapping in fully homomorphic encryption through memory-centric optimization with GPUs," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2021, pp. 114–148, Aug. 2021.
[6] W. Xu, B. Wang, J. Liu, Y. Chen, P. Duan, and Z. Hong, "Toward practical privacy-preserving linear regression," *Inf. Sci.*, vol. 596, pp. 119–136, Jun. 2022.
[7] Y. Liu, Y. Luo, Y. Zhu, Y. Liu, and X. Li, "Secure multi-label data classification in cloud by additionally homomorphic encryption," *Inf. Sci.*, vol. 468, pp. 89–102, Nov. 2018.
[8] A. Aloufi, P. Hu, H. W. H. Wong, and S. S. M. Chow, "Blindfolded evaluation of random forests with multi-key homomorphic encryption," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 4, pp. 1821–1835, Jul. 2021.
[9] H. Chen, W. Dai, M. Kim, and Y. Song, "Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 395–412.
[10] H. Chen, I. Chillotti, and Y. Song, "Multi-key homomorphic encryption from TFHE," in *Advances in Cryptology—ASIACRYPT* (Lecture Notes in Computer Science), vol. 11922. Cham, Switzerland: Springer, 2019, pp. 446–472.
[11] T. Kim, H. Kwak, D. Lee, J. Seo, and Y. Song, "Asymptotically faster multi-key homomorphic encryption from homomorphic gadget decomposition," *Cryptol. ePrint Archive*, 2022, Paper 2022/347.
[12] A. Aloufi and P. Hu, "Collaborative homomorphic computation on data encrypted under multiple keys," 2019, *arXiv:1911.04101*.
[13] C. Mouchet, J. Troncoso-Pastoriza, J.-P. Bossuat, and J.-P. Hubaux, "Multiparty homomorphic encryption from ring-learning-with-errors," in *Proc. Privacy Enhancing Technol. (CONF)*, 2021, pp. 291–311.
[14] J. Park, "Homomorphic encryption for multiple users with less communications," *IEEE Access*, vol. 9, pp. 135915–135926, 2021.
[15] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, pp. 1–40, Sep. 2009.
[16] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 6110. Berlin, Germany: Springer-Verlag, 2010, pp. 1–23.
[17] B. Applebaum, D. Cash, C. Peikert, and A. Sahai, "Fast cryptographic primitives and circular-secure encryption based on hard learning problems," in *Advances in Cryptology—CRYPTO*, vol. 5677. Berlin, Germany: Springer, 2009, pp. 595–618.
[18] J. Blömer and J.-P. Seifert, "On the complexity of computing short linearly independent vectors and short bases in a lattice," in *Proc. 31st Annu. ACM Symp. Theory Comput.*, May 1999, pp. 711–720.
[19] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "Bootstrapping for approximate homomorphic encryption," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 10820. Cham, Switzerland: Springer, 2018, pp. 360–384.
[20] H. Chen, I. Chillotti, and J. Kilian, "Improved bootstrapping for approximate homomorphic encryption," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 11477. Cham, Switzerland: Springer, 2019, pp. 34–54.
[21] K. Han and D. Ki, "Better bootstrapping for approximate homomorphic encryption," in *Proc. Cryptogr. Track RSA Conf.*, 2020, pp. 364–390.
[22] A. Kim, A. Papadimitriou, and Y. Polyakov, "Approximate homomorphic encryption with reduced approximation error," in *Proc. Cryptogr. Track RSA Conf.*, 2022, pp. 120–144.
[23] V. Lyubashevsky, C. Peikert, and O. Regev, "A toolkit for ring-LWE cryptography," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 7881. Cham, Switzerland: Springer, 2013, pp. 35–54.
[24] P. Longa and M. Naehrig, "Speeding up the number theoretic transform for faster ideal lattice-based cryptography," in *Proc. Int. Conf. Cryptol. Netw. Secur.* Cham, Switzerland: Springer, 2016, pp. 124–139.
[25] C. E. Shannon, "Communication theory of secrecy systems," *Bell Syst. Tech. J.*, vol. 28, no. 4, pp. 656–715, Oct. 1949.

ZAHYUN KOO received the B.S. degree in mathematics from Dongguk University, Seoul, South Korea, in 2015, and the M.S. degree in mathematics from Seoul National University, Seoul, in 2018, where he is currently pursuing the Ph.D. degree in electrical and computer engineering. His research interests include lattice-based cryptography and error-correcting codes.

JOON-WOO LEE (Member, IEEE) received the B.S. and Ph.D. degrees in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2016 and 2022, respectively. He is currently an Assistant Professor with the School of Computer Science and Engineering, Chung-Ang University, Seoul. His research interests include privacy-preserving machine learning, homomorphic encryption, and post-quantum cryptography.

JONG-SEON NO (Fellow, IEEE) received the B.S. and M.S.E.E. degrees in electronics engineering from Seoul National University, Seoul, South Korea, in 1981 and 1984, respectively, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 1988. He was a Senior MTS with Hughes Network Systems, from 1988 to 1990. He was an Associate Professor with the Department of Electronic Engineering, Konkuk University, Seoul, from 1990 to 1999. He joined the Faculty of the Department of Electrical and Computer Engineering, Seoul National University, in 1999, where he is currently a Professor. His research interests include error-correcting codes, cryptography, sequences, LDPC codes, interference alignment, and wireless communication systems. He became an IEEE Fellow through the IEEE Information Theory Society, in 2012. He became a member of the National Academy of Engineering of Korea (NAEK), in 2015, where he served as the Division Chair of Electrical, Electronic, and Information Engineering, from 2019 to 2020. He was a recipient of the IEEE Information Theory Society Chapter of the Year Award, in 2007. From 1996 to 2008, he served as the Founding Chair for the Seoul Chapter of the IEEE Information Theory Society. He was the General Chair of Sequence and their Applications 2004 (SETA2004), Seoul. He served as the General Co-Chair for the International Symposium on Information Theory and its Applications 2006 (ISITA2006) and the International Symposium on Information Theory 2009 (ISIT2009), Seoul. He served as the Co-Editor-in-Chief for the IEEE JOURNAL OF COMMUNICATIONS AND NETWORKS, from 2012 to 2013.

YOUNG-SIK KIM (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from Seoul National University, in 2001, 2003, and 2007, respectively. He joined the Semiconductor Division, Samsung Electronics, where he performed research and development of security hardware IPs for various embedded systems, including modular exponentiation hardware accelerator called Tornado 2MX2 for RSA and elliptic curve cryptography in smart card products and mobile application processors of Samsung Electronics until 2010. He is currently a Professor with Chosun University, Gwangju, South Korea. He is a Submitter for two candidate algorithms (McNie and pqsigRM) in the first round for the NIST Post Quantum Cryptography Standardization. His research interests include post-quantum cryptography, the IoT security, physical layer security, data hiding, channel coding, and signal design. He is selected as one of 2025's 100 Best Technology Leaders (for Crypto-Systems) by the National Academy of Engineering of Korea.