

Covert Federated Learning under Regulation based on Threshold Anamorphic Encryption

Wenxuan Xu, Huaqun Wang, Debiao He

Abstract—When communication systems are subject to strict external control, a powerful authority may monitor all transmitted messages and compel users to surrender their secret keys, thereby undermining user autonomy and the confidentiality of keys in encrypted communication. Anamorphic encryption (AE) enables covert communication under such surveillance by embedding hidden messages into innocent-looking ciphertexts. However, existing lattice-based AE constructions remain limited and typically rely on lattice trapdoor techniques, which impose restrictive parameter requirements and hinder their deployment in practical post-quantum cryptosystems such as Kyber. Moreover, existing constructions do not address the challenge of enabling covert communication among multiple parties under dictator-controlled environments. In this work, we propose Threshold Anamorphic Encryption (TAE), a new cryptographic primitive that extends receiver-AE to the N -out-of- N threshold setting, where the covert message can be recovered only through the collaboration of all N participants. Then we propose TAKyber, a concrete instantiation of TAE constructed from the KyberPKE framework. TAKyber embeds covert information into the randomness of ciphertexts rather than the public matrix structure, avoiding the large aspect-ratio requirement imposed by lattice trapdoor techniques and enabling deployment on Kyber and other lattice-based encryption schemes whose public matrices do not satisfy such requirements. Furthermore, TAKyber distributes covert information into multiple components during encryption and enables all N participants to jointly reconstruct the covert ciphertext, while preventing any subset of fewer than N participants without the double key from obtaining any information about the covert message. Finally, we apply TAKyber to privacy-preserving federated learning, where participants can securely exchange encrypted model gradients while simultaneously transmitting covert information through the anamorphic channel under dictator-controlled environments.

Index Terms—Anamorphic Encryption; Threshold Anamorphic Encryption; Kyber; Post-Quantum Cryptography; Module Lattice; Federated Learning

I. INTRODUCTION

UNDER normal circumstances, an encryption scheme is typically built upon the following fundamental assumptions: users are free to choose the messages they wish to encrypt, and their secret keys remain confidential. However, under the rule of a dictator, these assumptions may no longer hold. A dictator may compel users to surrender all of their keys and subsequently use them to monitor communications, thereby preventing users from transmitting messages that are

not approved by the dictator. Moreover, the dictator may force a user to send a message specified by the dictator. To address these challenges, the cryptographic primitive known as Anamorphic Encryption (AE), introduced in [1], aims to enable users to circumvent the dictator's surveillance and achieve covert communication under the regulation of a dictator.

A. Related Works

Persiano *et al.* [1] introduced the concept of Anamorphic Encryption (AE), a cryptographic primitive enabling covert communication under dictator surveillance. Their framework consists of two variants: Sender-Anamorphic Encryption (Sender-AE) and Receiver-Anamorphic Encryption (Receiver-AE). Since this work focuses on the receiver-anamorphic setting, we briefly review Receiver-AE.

A Receiver-AE scheme consists of a normal encryption scheme (KG, Enc, Dec) and an anamorphic encryption scheme $(aKG, aEnc, aDec)$. The receiver generates an anamorphic key pair (apk, ask) and an additional double key dk . The distribution of (apk, ask) is computationally indistinguishable from a normal key pair (pk, sk) , while the covert message can only be recovered using dk . Therefore, even if the dictator obtains ask , only the innocent-looking message is revealed.

Several follow-up works have extended AE from different perspectives. [2] revisited Sender-AE and introduced stronger security notions with connections to public-key steganography and algorithm-substitution attacks. [3] strengthened Receiver-AE by supporting post-deployment double-key generation and robustness, with instantiations based on ElGamal, Cramer-Shoup, and RSA-OAEP. [4] extended AE to hybrid and homomorphic encryption settings and supported multiple independent covert channels. [5], [6] introduced Anamorphic-Resistant encryption to study the limitations of AE and construct schemes resisting anamorphic extensions. [7] proposed Anamorphic Key Encapsulation Mechanisms (AKEMs) with strong IND-CCA security for covert channels based on KEMs. [8] showed that several LWE-based cryptosystems, including Regev, Dual Regev, and Dual GSW, support fully asymmetric AE and investigated robustness in the homomorphic setting. [9] refined Sender-AE security definitions by explicitly considering encryption randomness leakage.

Although several AE schemes have been proposed, lattice-based constructions remain limited, with only [1], [4], [8] exploring lattice-based AE. Specifically, [1] constructs sender-AE schemes from LWE-PKE and Dual-LWE-PKE using rejection sampling, while [4] and [8] extend AE to hybrid,

Wenxuan Xu is with the College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China (e-mail: 2025040340@njupt.edu.cn).

Huaqun Wang is with the College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China (e-mail: wanghuaqun@aliyun.com).

Debiao He is with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China (e-mail: hedebiao@163.com).

homomorphic, and fully asymmetric settings. However, these constructions rely on additional lattice structures such as trapdoors, which are not naturally supported by many practical lattice-based encryption schemes.

Moreover, existing AE schemes only consider the non-threshold setting, where the covert message can be independently recovered by any party possessing the double key. To the best of our knowledge, no existing work has investigated threshold extensions of AE.

B. Motivation

Despite the recent progress in lattice-based AE, existing constructions still rely heavily on lattice trapdoor techniques. In particular, the constructions in [8] require embedding a lattice trapdoor into the public matrix $\mathbf{A}$. Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. To guarantee the correctness and security of the trapdoor, the parameters are typically required to satisfy $m \gg n$. However, such a parameter requirement is incompatible with many practical lattice-based cryptosystems and may itself raise suspicion under the rule of dictator. Since the dictator is assumed to regulate both the cryptographic schemes and the public parameters adopted by communicating parties, restrictions may be imposed on the structure of public matrices or the choice of underlying encryption schemes.

This limitation is particularly evident for Kyber, one of the NIST-standardized post-quantum cryptographic schemes. Kyber employs a public matrix satisfying $\mathbf{A} \in R_q^{l \times l}$, where the matrix is inherently square and does not satisfy the large aspect-ratio requirement commonly required by lattice trapdoor constructions. Consequently, existing trapdoor-based AE constructions cannot be directly applied to Kyber or other lattice-based encryption schemes with similar structures.

Moreover, existing AE schemes are designed for the non-threshold setting, where the covert message can be independently recovered by any party possessing the double key. However, many practical applications, such as secure multi-party computation and federated learning, require the decryption capability to be distributed among multiple parties, such that no individual participant can independently recover the covert message. When these applications operate under the regulation of a dictator, existing AE schemes cannot provide an appropriate covert communication mechanism. Therefore, it is necessary to extend AE to the threshold setting, where multiple participants collaboratively recover covert messages while preserving indistinguishability from normal communications.

To address the above challenges, we introduce a new cryptographic primitive called Threshold Anamorphic Encryption (TAE). Specifically, TAE distributes the covert message recovery capability among multiple participants by embedding different components of a covert ciphertext into multiple independent innocent-looking ciphertexts. In the threshold setting, decryption participants consist of N partial decryption participants and one final combiner. Each partial decryption participant performs partial decryption on its received innocent-looking ciphertext using its secret key and obtains a corresponding innocent-looking fragment, while the final combiner reconstructs the covert ciphertext after collecting all

required fragments. In this work, we focus on the N -out-of- N threshold setting, where all N partial decryption participants are required to provide their corresponding fragments for successful recovery of the covert message.

We instantiate TAE from the KyberPKE framework, i.e., the public-key encryption (PKE) version of CRYSTALS-Kyber [10], and obtain a concrete construction called TAKyber. Since TAKyber does not rely on lattice trapdoor techniques, it does not impose any additional requirements on the public matrix $\mathbf{A}$, such as the large aspect-ratio condition required by trapdoor-based constructions. Therefore, TAKyber can be instantiated with the original Kyber parameter settings or other valid parameter settings without modifying the fundamental structure of Kyber, supporting practical deployment over NIST-standardized post-quantum cryptographic schemes.

Finally, we apply TAKyber to federated learning (FL) [11]–[13] under the dictator model, enabling covert gradient exchange while preserving indistinguishability of innocent-looking communications and protecting gradients privacy.

C. Contribution

Our contributions are summarized as follows:

- We introduce **Threshold Anamorphic Encryption (TAE)**, a new cryptographic primitive that extends receiver-AE to the N -out-of- N threshold setting.
- We instantiate TAE from KyberPKE as **TAKyber**, which avoids imposing additional restrictions on the structure.
- We construct a privacy-preserving federated learning scheme based on TAKyber, enabling covert gradient exchange among multiple parties under the regulation of dictator.

D. Organization

The remainder of this paper is organized as follows. Section II reviews the preliminaries. Section III presents TAKyber and its security analysis. Section IV introduces the covert federated learning framework based on TAE. Section V evaluates the performance of the proposed scheme. Finally, Section VI concludes the paper and discusses future research directions.

II. PRELIMINARIES

A. Basic Definitions

Let q be a positive integer modulus, define $\mathbb{Z}_q = [-\frac{q}{2}, \frac{q}{2}) \cap \mathbb{Z}$. For $x \in \mathbb{Z}$, $[x]_q$ denotes its centered reduction modulo q .

Bold lowercase and uppercase letters denote vectors and matrices, respectively, and $(\cdot)^T$ denotes transpose. For a real number r , $\lceil r \rceil$, $\lfloor r \rfloor$, and $\lceil r \rceil$ denote ceiling, floor, and rounding operations. For a polynomial $\mathbf{r} \in R_q$, $\|\mathbf{r}\|_\infty$ denotes its infinity norm, which naturally extends to polynomial vectors.

B. Rings and Distributions

Let $R = \mathbb{Z}[x]/(x^n + 1)$ and $R_q = \mathbb{Z}_q[x]/(x^n + 1)$, where n is a power of two. Arithmetic in R_q is performed coefficient-wise modulo q . We write R_q^l for the set of polynomial vectors of length l over R_q , and $R_q^{k \times l}$ for the set of $k \times l$ polynomial matrices over R_q .

The notation $\mathcal{U}(R_q)$ denotes the uniform distribution over

R_q . For a positive integer δ , the restricted uniform distribution $\mathcal{U}_\delta(R_q)$ is defined such that each coefficient is independently and uniformly sampled from $9] [-\delta, \delta]$. In addition, $\mathcal{B}_\eta(R_q)$ denotes the polynomial distribution over R_q whose coefficients are independently sampled from the centered binomial distribution $\mathcal{B}_\eta$, where $\mathcal{B}_\eta = \sum_{i=1}^\eta (a_i - b_i)$, and $a_i, b_i \in \{0, 1\}$ are independent Bernoulli random variables with parameter $1/2$.

C. Operations on Module Lattice

Let $\mathbf{x} \in R_q^l$ be a module-lattice vector. In the following, we introduce four basic operations used to manipulate module-lattice vectors in our construction.

Definition 1. For positive integers B and q satisfying $B \ll q$, the decompose algorithm **Decompose**($\mathbf{x}, q, B$) takes a vector $\mathbf{x} \in R_q^l$ as input and outputs a set of $\lceil \log_B q \rceil$ vectors $\{\mathbf{x}_j\}_{j=1}^{\lceil \log_B q \rceil}$. Algorithm 1 describes the decomposition procedure for a single coefficient $x \in \mathbb{Z}_q$. The decomposition of a vector $\mathbf{x} \in R_q^l$ is performed coefficient-wise by applying Algorithm 1 independently to every coefficient of every polynomial. The output vectors satisfy $\mathbf{x}_j \in R_B^l$ and

$$\mathbf{x} = \sum_{j=1}^{\lceil \log_B q \rceil} B^{j-1} \mathbf{x}_j \quad (1)$$

Definition 2. Given the output $\{\mathbf{x}_j\}_{j=1}^{\lceil \log_B q \rceil}$ of the decompose algorithm in Definition 1, the composition algorithm **Compose**($\{\mathbf{x}_j\}_{j=1}^{\lceil \log_B q \rceil}, q, B$) reconstructs the original vector by computing $\sum_{j=1}^{\lceil \log_B q \rceil} B^{j-1} \mathbf{x}_j$, and the following relation holds:

$$\text{Compose}(\text{Decompose}(\mathbf{x}, q, B), q, B) = \mathbf{x} \quad (2)$$

Definition 3. For a vector $\mathbf{x} \in R_q^L$, where $L \geq l$, the slice algorithm **Slice**($\mathbf{x}, L, l$) first padding $\mathbf{x}$ into $\hat{\mathbf{x}} = (\mathbf{x}, \mathbf{r})$ where $\mathbf{r}$ is randomly sampled from $\mathcal{U}(R_q^{\lceil L/l \rceil - L})$. Then $\hat{\mathbf{x}}$ is partitioned into $\lceil L/l \rceil$ vectors $\{\mathbf{x}_k\}_{k=1}^{\lceil L/l \rceil}$, where each vector satisfies $\mathbf{x}_k \in R_q^l$. The partition satisfies

$$\hat{\mathbf{x}} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{\lceil L/l \rceil}) \quad (3)$$

Definition 4. The merge algorithm **Merge**($\{\mathbf{x}_k\}_{k=1}^{\lceil L/l \rceil}, L, l$) takes the output vectors $\{\mathbf{x}_k\}_{k=1}^{\lceil L/l \rceil}$ generated by the slice algorithm in Definition 3 as input and reconstructs the original vector $\mathbf{x} \in R_q^L$. Specifically, it first obtains $\hat{\mathbf{x}}$ according to (3), and then removes the last $\lceil L/l \rceil - L$ padding components from $\hat{\mathbf{x}}$ to recover $\mathbf{x}$. The following relation holds:

$$\text{Merge}(\text{Slice}(\mathbf{x}, L, l), L, l) = \mathbf{x} \quad (4)$$

D. Anamorphic Encryption

We now briefly review the formal framework and security definitions of anamorphic encryption (AE). An AE scheme

Algorithm 1 Decompose(x, q, B)

```

1: for  $i$  from 0 to  $\lceil \log_B q \rceil - 1$ :
2:    $d_i \leftarrow x \bmod B$ ;
3:   if  $d_i > \delta$ :
4:      $d_i \leftarrow d_i - B$ ;
5:    $x \leftarrow (x - d_i)/B$ ;
6: return  $\{d_i\}_{i=0}^{\lceil \log_B q \rceil - 1}$ ;
```

RealG$_{\Pi}(\lambda, \mathcal{D})$ 1. $(pk, sk) \leftarrow \mathbf{KG}(1^\lambda)$; 2. Return $\mathcal{D}^{\mathcal{O}_e(pk, \cdot, \cdot)}(pk, sk)$, where $\mathcal{O}_e(pk, m, M) = \mathbf{Enc}(pk, m)$.
AnaG$_{\Sigma}(\lambda, \mathcal{D})$ 1. $(apk, ask, dk) \leftarrow \mathbf{aKG}(1^\lambda)$; 2. Return $\mathcal{D}^{\mathcal{O}_a(apk, \cdot, \cdot)}(apk, ask)$, where $\mathcal{O}_a(apk, m, M) = \mathbf{aEnc}(apk, dk, m, M)$.

Fig. 1. Real and anamorphic games of AE.

RobustG$_{\mathcal{A}, \Pi, \Sigma}^b(\lambda)$ 1. $(apk, ask, dk) \leftarrow \mathbf{aKG}(1^\lambda)$; 2. Return $\mathcal{A}^{\mathcal{O}_b(apk, ask, dk, \cdot)}(apk, ask)$, where: $\mathcal{O}_0(apk, ask, dk, m) = \mathbf{aDec}(dk, ask, \mathbf{Enc}(apk, m))$ and $\mathcal{O}_1(apk, ask, dk, m) = \perp$.

Fig. 2. Robustness game of AE.

is defined based on a public-key encryption (PKE) scheme $\Pi = (\mathbf{KG}, \mathbf{Enc}, \mathbf{Dec})$, where $\mathbf{KG}$, $\mathbf{Enc}$, and $\mathbf{Dec}$ are the standard key generation, encryption, and decryption algorithm respectively. In addition, it is equipped with an anamorphic triplet $\Sigma = (\mathbf{aKG}, \mathbf{aEnc}, \mathbf{aDec})$ described as follows.

Definition 5. A triplet consisting of three probabilistic polynomial time (PPT) algorithms $\Sigma = (\mathbf{aKG}, \mathbf{aEnc}, \mathbf{aDec})$ is anamorphic if:

- $\mathbf{aKG}$ takes the security parameter 1^λ as input, then outputs an anamorphic public key apk , an anamorphic secret key ask , and a double key dk .
- $\mathbf{aEnc}$ takes apk , dk , an innocent-looking message m , and a covert message M as input, then outputs an anamorphic ciphertext act .
- $\mathbf{aDec}$ takes dk , ask and act as input, then outputs the covert message M or the failure symbol $\perp$.

Definition 6. A PKE scheme $\Pi = (\mathbf{KG}, \mathbf{Enc}, \mathbf{Dec})$ is anamorphic if it satisfies IND-CPA security and there exists an anamorphic triplet Σ such that, for any PPT dictator $\mathcal{D}$:

$$\text{Adv}_{\mathcal{D}, \Pi, \Sigma}^{\text{ana}}(\lambda) = |\Pr[\text{RealG}_{\Pi}(\lambda, \mathcal{D}) = 1] - \Pr[\text{AnaG}_{\Sigma}(\lambda, \mathcal{D}) = 1]| \leq \text{negl}(\lambda)$$

where $\text{negl}(\lambda)$ is a negligible function of λ , and the real and ideal games are defined in Figure 1. Here, $\mathcal{D}$ denotes a PPT distinguisher, while $\mathcal{O}_e$ and $\mathcal{O}_a$ denote the normal and anamorphic encryption oracles, respectively.

Definition 7. An anamorphic PKE scheme $\Pi = (\mathbf{KG}, \mathbf{Enc}, \mathbf{Dec})$ endowed with an anamorphic triplet $\Sigma = (\mathbf{aKG}, \mathbf{aEnc}, \mathbf{aDec})$ is robust if for any PPT adversary $\mathcal{A}$:

$$\text{Adv}_{\mathcal{A}, \Pi, \Sigma}^{\text{robust}}(\lambda) = |\Pr[\text{RobustG}_{\mathcal{A}, \Pi, \Sigma}^0(\lambda) = 1] - \Pr[\text{RobustG}_{\mathcal{A}, \Pi, \Sigma}^1(\lambda) = 1]| \leq \text{negl}(\lambda)$$

where the game $\text{RobustG}_{\mathcal{A}, \Pi, \Sigma}^b(\lambda)$ for $b \in \{0, 1\}$ is defined in Figure 2.

E. Kyber

Kyber is a module-lattice-based encryption scheme [10] whose security relies on the hardness of the MLWE problem.

```

KyberPKE.KG( $1^\lambda, \text{PP}_{\text{Kyber}}$ )  $\rightarrow pk, sk$ :
1)  $\mathbf{A} \leftarrow \mathcal{U}(R_q^{l \times l})$ ;
2)  $(\mathbf{s}, \mathbf{e}) \leftarrow \mathcal{B}_{\eta_1}(R_q^l) \times \mathcal{B}_{\eta_1}(R_q^l)$ ;
3)  $\mathbf{b} = [\mathbf{A}\mathbf{s} + \mathbf{e}]_q$ ;
4) return  $pk = (\mathbf{A}, \mathbf{b}), sk = \mathbf{s}$ ;

KyberPKE.Enc( $m, pk, \text{PP}_{\text{Kyber}}$ )  $\rightarrow \mathbf{c}_m$ :
1)  $\mathbf{r} \leftarrow \mathcal{U}_\delta(R_q^l)$ ;
2)  $(\mathbf{e}_1, \mathbf{e}_2) \leftarrow \mathcal{B}_{\eta_2}(R_q^l) \times \mathcal{B}_{\eta_2}(R_q^l)$ ;
3)  $\mathbf{u} = [\mathbf{A}^T \mathbf{r} + \mathbf{e}_1]_q$ ;
4)  $v = [\mathbf{b}^T \mathbf{r} + \mathbf{e}_2 + \Delta m]_q$ ;
5) return  $\mathbf{c}_m = (\mathbf{u}, v)$ ;

KyberPKE.Dec( $\mathbf{c}_m = (\mathbf{u}, v), sk, \text{PP}_{\text{Kyber}}$ )  $\rightarrow m$ :
1)  $m = \left\lfloor \frac{v - \mathbf{s}^T \mathbf{u}}{\Delta} \right\rfloor$ ;
2) return  $m$ ;

```

Fig. 3. Implementation of KyberPKE.

Compared with cryptosystems constructed over integer lattices or ideal lattices, Kyber achieves security comparable to that of integer-lattice-based constructions while significantly reducing the sizes of public keys and ciphertexts. Moreover, by fixing the underlying polynomial ring and adjusting the module rank, module lattices provide a flexible trade-off between security and efficiency. Owing to these advantages, Kyber has been standardized by the National Institute of Standards and Technology (NIST) as a post-quantum cryptographic primitive.

KyberPKE is the PKE variant of Kyber. Figure 3 illustrates the key generation, encryption, and decryption procedures of KyberPKE, where

$$\text{PP}_{\text{Kyber}} = \{n, q, l, t, \eta_1, \eta_2, \delta, \Delta = \left\lfloor \frac{q}{t} \right\rfloor\}$$

denotes the public parameters of Kyber. In our construction, the sampling distribution of the random vector $\mathbf{r}$ is instantiated as a discrete uniform distribution, while the remaining procedures follow the original KyberPKE.

III. THRESHOLD ANAMORPHIC ENCRYPTION

A. System Model

We consider an N -out-of- N threshold anamorphic encryption (TAE) system consisting of an encryptor, N partial decryptors, a final combiner, and a dictator. The N partial decryptors and the final combiner are collectively referred to as decryption participants.

During system initialization, the key generation algorithm generates N key pairs $\{(apk_i, ask_i)\}_{i=1}^N$, a key pair (cpk, csk) , and a double key dk . The key pairs $\{(apk_i, ask_i)\}_{i=1}^N$ are assigned to the partial decryptors, while (cpk, csk) is assigned to the final combiner. The N partial decryptors hold their respective secret keys $\{ask_i\}_{i=1}^N$, and the final combiner holds the secret key csk . No decryption participant obtains the secret information of another decryption participant. In contrast, the double key dk and all public keys $\{apk_i\}_{i=1}^N$ and cpk are shared among the encryptor and decryption participants. The dictator can compel all decryption participants to surrender their secret keys. Consequently, the dictator obtains $\{(apk_i, ask_i)\}_{i=1}^N$ and (cpk, csk) . However, the dictator is unaware of the existence of the double key dk .

Given a covert message M , for every $i \in [1, N]$, the encryptor generates an innocent-looking ciphertext $\mathbf{c}_{M,i}$ for the i -th partial decryptor using apk_i and dk . Each partial decryptor performs partial decryption using ask_i , dk , and cpk , and obtains an innocent-looking fragment $\mathbf{f}_{M,i}$, which is then sent to the final combiner.

After receiving all valid innocent-looking fragments $\{\mathbf{f}_{M,i}\}_{i=1}^N$, the final combiner executes the combination algorithm using dk and csk to recover the covert message M . If any partial decryptor fails to provide a valid innocent-looking fragment, the final combiner outputs the failure symbol $\perp$.

It is worth noting that our TAE model differs from conventional threshold encryption schemes (e.g., [14], [15]) in two aspects. First, unlike conventional threshold encryption, where multiple participants jointly decrypt a common ciphertext using their secret key shares, TAE distributes different anamorphic ciphertexts to different partial decryptors and achieves threshold recovery by aggregating the corresponding innocent-looking fragments. Although generating a single ciphertext and distributing decryption shares is more efficient in conventional threshold encryption, such a design is unsuitable for covert communication under the dictator model. Specifically, if the same ciphertext is transmitted to multiple partial decryptors, the repeated communication pattern may raise suspicion of the dictator. Since the communication between the encryptor and each partial decryptor should appear independent and indistinguishable from normal encryption processes, TAE generates independent innocent-looking ciphertexts for different partial decryptors to preserve the covert nature of communication.

Second, in conventional threshold encryption, partial decryption fragments can be directly transmitted to the final combiner for aggregation. However, such fragments are typically distinguishable from normal ciphertexts and may reveal the existence of covert communication under dictator surveillance. Therefore, TAE introduces an additional key pair (cpk, csk) for the final combiner, where cpk enables partial decryptors to generate innocent-looking fragments and csk allows the final combiner to recover the hidden message from the aggregated fragments. Consequently, both the anamorphic ciphertexts and the transmitted fragments remain indistinguishable from ordinary ciphertexts, preserving the covert nature of communication against the dictator.

B. Formal and Security Definition

Definition 8. A quadruple consisting of four PPT algorithms $\Sigma_N = (\mathbf{taKG}, \mathbf{taEnc}, \mathbf{taPDec}, \mathbf{taCDec})$ is N -out-of- N threshold anamorphic if:

- $\mathbf{taKG}(1^\lambda)$ takes the security parameter 1^λ as input, then outputs N anamorphic key pairs $\{(apk_i, ask_i)\}_{i=1}^N$ for partial decryptors, a key pair (cpk, csk) for the final combiner, and a double key dk .

- $\mathbf{taEnc}$ takes an integer $i \in [1, N]$, the double key dk , the public key apk_i , an arbitrarily chosen message m_i , and a covert message M as input, then outputs an innocent-looking ciphertext $\mathbf{c}_{M,i}$.

- $\mathbf{taPDec}$ takes an integer $i \in [1, N]$ and $dk, ask_i, cpk, \mathbf{c}_{M,i}$ as input, then outputs the corresponding innocent-looking

fragment $\mathbf{f}_{M,i}$ or the failure symbol $\perp$.

• **taCDec** takes the double key dk , the final combiner's secret key csk , and a set of innocent-looking fragments F as input. If $\{\mathbf{f}_{M,i}\}_{i=1}^N \subseteq F$, it outputs the covert message M ; otherwise, it outputs the failure symbol $\perp$.

Definition 9. A PKE scheme $\Pi = (\mathbf{KG}, \mathbf{Enc}, \mathbf{Dec})$ is N -out-of- N threshold anamorphic if it satisfies IND-CPA security and there exists an N -out-of- N threshold anamorphic quadruple Σ_N such that, for every $1 \leq i \leq N$ and any PPT dictator $\mathcal{D}$, the following two advantages are negligible:

$$\text{Adv}_{\mathcal{D}, \Pi, \Sigma_N}^{\text{c-thana}}(\lambda, i) = \left| \Pr[\text{RealG}_{\Pi}(\lambda, \mathcal{D}) = 1] - \Pr[\text{TAnaG}_{1, \Sigma_N}(\lambda, \mathcal{D}, i) = 1] \right| \leq \text{negl}(\lambda) \quad (5)$$

$$\text{Adv}_{\mathcal{D}, \Pi, \Sigma_N}^{\text{f-thana}}(\lambda, i) = \left| \Pr[\text{RealG}_{\Pi}(\lambda, \mathcal{D}) = 1] - \Pr[\text{TAnaG}_{2, \Sigma_N}(\lambda, \mathcal{D}, i) = 1] \right| \leq \text{negl}(\lambda) \quad (6)$$

The real game is defined in Figure 1, while the two threshold anamorphic games are defined in Figure 4. Here, $\mathcal{O}_{1,ta}$ and $\mathcal{O}_{2,ta}$ denote the threshold anamorphic encryption oracle and the threshold anamorphic partial decryption oracle for the i -th partial decryptor, respectively.

Definition 10. An anamorphic PKE scheme $\Pi = (\mathbf{KG}, \mathbf{Enc}, \mathbf{Dec})$ endowed with an N -out-of- N threshold anamorphic quadruple $\Sigma_N = (\mathbf{taKG}, \mathbf{taEnc}, \mathbf{taPDec}, \mathbf{taCDec})$ is robust if, for every $1 \leq i \leq N$ and any PPT adversary $\mathcal{A}$, the following two advantages are negligible:

$$\text{Adv}_{\mathcal{A}, \Pi, \Sigma_N}^{\text{c-robust}}(\lambda, i) = \left| \Pr[\text{TRobustG}_{1, \Sigma_N}^0(\lambda, \mathcal{A}, i) = 1] - \Pr[\text{TRobustG}_{1, \Sigma_N}^1(\lambda, \mathcal{A}, i) = 1] \right| \leq \text{negl}(\lambda) \quad (7)$$

$$\text{Adv}_{\mathcal{A}, \Pi, \Sigma_N}^{\text{f-robust}}(\lambda, i) = \left| \Pr[\text{TRobustG}_{2, \Sigma_N}^0(\lambda, \mathcal{A}, i) = 1] - \Pr[\text{TRobustG}_{2, \Sigma_N}^1(\lambda, \mathcal{A}, i) = 1] \right| \leq \text{negl}(\lambda) \quad (8)$$

The robustness games are defined in Figure 5. The first game captures the robustness of partial decryption, requiring that a normal ciphertext cannot be recognized as an anamorphic ciphertext by **taPDec**. The second game captures the robustness of innocent-looking fragment reconstruction, requiring that for any set F satisfying $\{\mathbf{f}_{M,i}\}_{i=1}^N \not\subseteq F$, the probability that **taCDec**(dk, csk, F) outputs correct M is negligible.

$\text{TAnaG}_{1, \Sigma_N}(\lambda, \mathcal{D}, i)$ 1. $\{(apk_j, ask_j)\}_{j=1}^N, (cpk, csk), dk \leftarrow \mathbf{taKG}(1^\lambda);$ 2. Return $\mathcal{D}_{1,ta}^{\mathcal{O}_{1,ta}(i, apk_i, \cdot, \cdot)}(\{(apk_j, ask_j)\}_{j=1}^N, (cpk, csk))$, where $\mathcal{O}_{1,ta}(i, apk_i, m, M) = \mathbf{taEnc}(i, dk, apk_i, m, M)$.
$\text{TAnaG}_{2, \Sigma_N}(\lambda, \mathcal{D}, i)$ 1. $\{(apk_j, ask_j)\}_{j=1}^N, (cpk, csk), dk \leftarrow \mathbf{taKG}(1^\lambda);$ 2. Return $\mathcal{D}_{2,ta}^{\mathcal{O}_{2,ta}(i, apk_i, ask_i, \cdot, \cdot)}(\{(apk_j, ask_j)\}_{j=1}^N, (cpk, csk))$, where $\mathcal{O}_{2,ta}(i, apk_i, ask_i, m, M) = \mathbf{taPDec}(i, dk, ask_i, cpk, \mathbf{taEnc}(i, dk, apk_i, m, M))$.

Fig. 4. Threshold anamorphic games of TAE.

$\text{TRobustG}_{1, \Sigma_N}^b(\lambda, \mathcal{A}, i)$ 1. $\{(apk_j, ask_j)\}_{j=1}^N, (cpk, csk), dk \leftarrow \mathbf{taKG}(1^\lambda);$ 2. Return $\mathcal{A}_{1,b}^{\mathcal{O}_{1,b}(i, apk_i, ask_i, \cdot, \cdot)}(\{(apk_j, ask_j)\}_{j=1}^N, (cpk, csk))$, where $\mathcal{O}_{1,b}(i, apk_i, ask_i, m) = \mathbf{taPDec}(i, dk, ask_i, cpk, \mathbf{Enc}(apk_i, m))$ and $\mathcal{O}_{1,b}(i, apk_i, ask_i, m) = \perp$.
$\text{TRobustG}_{2, \Sigma_N}^b(\lambda, \mathcal{A}, i)$ 1. $\{(apk_j, ask_j)\}_{j=1}^N, (cpk, csk), dk \leftarrow \mathbf{taKG}(1^\lambda);$ 2. Randomly select $M, \{m_j\}_{j=1}^N;$ 3. for j from 1 to N : 4. $\mathbf{c}_{M,j} \leftarrow \mathbf{taEnc}(j, dk, apk_j, m_j, M);$ 5. $\mathbf{f}_{M,j} \leftarrow \mathbf{taPDec}(j, dk, ask_j, cpk, \mathbf{c}_{M,j});$ 6. Return $\mathcal{A}_{2,b}^{\mathcal{O}_{2,b}(i, \cdot, \cdot)}(\{(apk_j, ask_j)\}_{j=1}^N, cpk, csk)$, where $\mathcal{O}_{2,b}(i, \mathbf{f}) = \mathbf{taCDec}(dk, csk, \{\mathbf{f}_{M,j}\}_{1 \leq j \leq N, j \neq i} \cup \{\mathbf{f}\})$ and $\mathcal{O}_{2,b}(i, \mathbf{f}) = \perp$.

Fig. 5. Robustness games of TAE.

C. Construction of TAE over Module Lattice

We present the construction of TAKyber, a TAE scheme over module lattices which contains four PPT algorithms as shown in Figure 6. The set of parameters of TAKyber is

$$\text{PP}_{\text{TAKyber}} = \{\text{PP}_{\text{Kyber}}, \text{PP}_{\text{Kyber}}^{\text{ana}}, N, \text{tag}_{\text{TAE}}, \text{tag}_F\}$$

where

$$\begin{cases} \text{PP}_{\text{Kyber}} = \{n, q, l, t, \eta_1, \eta_2, \delta, \Delta = \lfloor \frac{q}{t} \rfloor\} \\ \text{PP}_{\text{Kyber}}^{\text{ana}} = \{n, Q, L, t', \eta'_1, \eta'_2, \delta', \Delta' = \lfloor \frac{Q}{t'} \rfloor\} \end{cases} \quad (9)$$

and

$$N = \left\lceil \frac{L}{l} \right\rceil \lceil \log_{2\delta+1} Q \rceil$$

denotes the number of partial decryptors required in the TAE construction.

Two hash functions, Hash_{e_1} and Hash_{e_2} , are employed to deterministically generate the error terms e_1 and e_2 , respectively. The identification tags $\text{tag}_{\text{TAE}} \in R_q$ and $\text{tag}_F \in R_q$ are incorporated into the input of Hash_{e_2} , which correspond to ciphertexts generated in the initial transmission phase and the fragment forwarding phase, respectively. By binding the ciphertext generation process with different transmission phases, these tags enable the distinction of innocent-looking ciphertexts carrying different hidden contents, thereby achieving the robustness property defined in Definition 10. The detailed constructions of these hash functions are presented in Appendix A.

The main idea of TAKyber is to construct a threshold anamorphic encryption scheme based on two independent Kyber instances with different functionalities. Specifically, the encryptor first encrypts the covert message M using the anamorphic Kyber instance to generate an anamorphic ciphertext $\mathbf{c}_M$, which is then decomposed into multiple components and distributed among the partial decryptors. Each partial decryptor only obtains one component of $\mathbf{c}_M$ through **taEnc**.

During partial decryption, each partial decryptor recovers its assigned component of $\mathbf{c}_M$ and re-encrypts it under the final combiner's public key cpk to generate an innocent-looking covert fragment $\mathbf{f}_{M,i}$. This re-encryption step prevents the

TAKyber.taKG(PP_{TAKyber}) $\rightarrow \{(apk_i, ask_i)\}_{i=1}^N$,
 cpk, csk, dk :

- 1) $dpk, dsk \leftarrow \text{KyberPKE.KG}(PP_{\text{Kyber}}^{\text{ana}})$;
- 2) for i from 0 to N :
- 3) while True:
- 4) $(\mathbf{A}_i, \mathbf{b}_i), \mathbf{s}_i \leftarrow \text{KyberPKE.KG}(PP_{\text{Kyber}})$;
- 5) if $\mathbf{A}_i$ is invertible:
- 6) $apk_i = (\mathbf{A}_i, \mathbf{b}_i), ask_i = (\mathbf{A}_i, \mathbf{b}_i, \mathbf{s}_i)$; break;
- 7) $cpk = apk_0, csk = ask_0, dk = (dpk, dsk)$;
- 8) return $\{(apk_i, ask_i)\}_{i=1}^N, cpk, csk, dk$;

TAKyber.taEnc($i, dk, apk_i, m_i, M, PP_{\text{TAKyber}}$) $\rightarrow \mathbf{c}_{M,i}$:

- 1) Parse $dk = (dpk, dsk), apk_i = (\mathbf{A}_i, \mathbf{b}_i)$;
- 2) $\mathbf{c}_M \leftarrow \text{KyberPKE.Enc}(M, dpk, PP_{\text{Kyber}}^{\text{ana}})$;
- 3) $\{\mathbf{d}_{\mathbf{c}_M, j}\}_{j=1}^{\lceil \log_{2\delta+1} Q \rceil} \leftarrow \text{Decompose}(\mathbf{c}_M, Q, 2\delta + 1)$;
- 4) $j \leftarrow \left\lfloor \frac{i-1}{\lceil L/l \rceil} \right\rfloor + 1$;
- 5) $\{\mathbf{d}_{\mathbf{c}_M, j, k}\}_{k=1}^{\lceil L/l \rceil} \leftarrow \text{Slice}(\mathbf{d}_{\mathbf{c}_M, j}, L, l)$;
- 6) $k \leftarrow i - (j-1)\lceil L/l \rceil$;
- 7) $\mathbf{r}_i \leftarrow \mathbf{d}_{\mathbf{c}_M, j, k}$;
- 8) $\mathbf{e}_{1,i} \leftarrow \text{Hash}_{\mathbf{e}_1}(apk_i, dk, m_i, i, \eta_2)$;
- 9) $\mathbf{e}_{2,i} \leftarrow \text{Hash}_{\mathbf{e}_2}(apk_i, dk, m_i, i, \eta_2, \text{tag}_{\text{TAE}})$;
- 10) $\mathbf{u}_i = [\mathbf{A}_i^T \mathbf{r}_i + \mathbf{e}_{1,i}]_q$;
- 11) $\mathbf{v}_i = [\mathbf{b}_i^T \mathbf{r}_i + \mathbf{e}_{2,i} + \Delta m_i]_q$;
- 12) return $\mathbf{c}_{M,i} = (\mathbf{u}_i, \mathbf{v}_i)$;

TAKyber.taPDec($i, dk, ask_i, cpk, \mathbf{c}_{M,i}, PP_{\text{TAKyber}}$)
 $\rightarrow \mathbf{f}_{M,i} / \perp$:

- 1) Parse $ask_i = (\mathbf{A}_i, \mathbf{b}_i, \mathbf{s}_i), \mathbf{c}_{M,i} = (\mathbf{u}_i, \mathbf{v}_i)$;
- 2) $m_i \leftarrow \text{KyberPKE.Dec}(\mathbf{c}_{M,i}, ask_i, PP_{\text{Kyber}})$;
- 3) $\mathbf{e}_{1,i} \leftarrow \text{Hash}_{\mathbf{e}_1}(apk_i, dk, m_i, i, \eta_2)$;
- 4) $\mathbf{r}_i \leftarrow [(\mathbf{A}_i^T)^{-1}(\mathbf{u}_i - \mathbf{e}_{1,i})]_q$;

5) if $\text{Hash}_{\mathbf{e}_2}(apk_i, dk, m_i, i, \eta_2, \text{tag}_{\text{TAE}}) \neq [\mathbf{v}_i - \mathbf{b}_i^T \mathbf{r}_i - \Delta m_i]_q$: return $\perp$;

6) Arbitrarily select $m'_i \in R_t$; Parse $cpk = (\mathbf{A}_0, \mathbf{b}_0)$;

7) $\mathbf{e}'_{1,i} \leftarrow \text{Hash}_{\mathbf{e}_1}(cpk, dk, m'_i, i, \eta_2)$;

8) $\mathbf{e}'_{2,i} \leftarrow \text{Hash}_{\mathbf{e}_2}(cpk, dk, m'_i, i, \eta_2, \text{tag}_F)$;

9) $\mathbf{u}'_i = [\mathbf{A}_0^T \mathbf{r}_i + \mathbf{e}'_{1,i}]_q$;

10) $\mathbf{v}'_i = [\mathbf{b}_0^T \mathbf{r}_i + \mathbf{e}'_{2,i} + \Delta m'_i]_q$;

11) return $\mathbf{f}_{M,i} = (\mathbf{u}'_i, \mathbf{v}'_i)$;

TAKyber.taCDec($dk, csk, F, PP_{\text{TAKyber}}$) $\rightarrow M / \perp$:

- 1) Parse $dk = (dpk, dsk), csk = (\mathbf{A}_0, \mathbf{b}_0, \mathbf{s}_0)$;
- 2) for i from 1 to N :
- 3) Initialize $\mathbf{r}_i \leftarrow \perp$;
- 4) for each $\mathbf{f} \in F$:
- 5) Parse $\mathbf{f} = (\mathbf{u}', \mathbf{v}')$;
- 6) $m' \leftarrow \text{KyberPKE.Dec}(\mathbf{f}, csk, PP_{\text{Kyber}})$;
- 7) $\mathbf{e}'_1 \leftarrow \text{Hash}_{\mathbf{e}_1}(cpk, dk, m', i, \eta_2)$;
- 8) $\mathbf{r} \leftarrow [(\mathbf{A}_0^T)^{-1}(\mathbf{u}' - \mathbf{e}'_1)]_q$;
- 9) if $\text{Hash}_{\mathbf{e}_2}(cpk, dk, m', i, \eta_2, \text{tag}_F) = [\mathbf{v}' - \mathbf{b}_0^T \mathbf{r} - \Delta m']_q$:
- 10) $\mathbf{r}_i \leftarrow \mathbf{r}$; break;
- 11) if $\mathbf{r}_i = \perp$: return $\perp$;
- 12) $j \leftarrow \left\lfloor \frac{i-1}{\lceil L/l \rceil} \right\rfloor + 1$;
- 13) $k \leftarrow i - (j-1)\lceil L/l \rceil$;
- 14) $\mathbf{d}_{\mathbf{c}_M, j, k} \leftarrow \mathbf{r}_i$;
- 15) for j from 1 to $\lceil \log_{2\delta+1} Q \rceil$:
- 16) $\mathbf{d}_{\mathbf{c}_M, j} \leftarrow \text{Merge}(\{\mathbf{d}_{\mathbf{c}_M, j, k}\}_{k=1}^{\lceil L/l \rceil}, L, l)$;
- 17) $\mathbf{c}_M \leftarrow \text{Compose}(\{\mathbf{d}_{\mathbf{c}_M, j}\}_{j=1}^{\lceil \log_{2\delta+1} Q \rceil}, Q, 2\delta + 1)$;
- 18) $M \leftarrow \text{KyberPKE.Dec}(\mathbf{c}_M, dsk, PP_{\text{Kyber}}^{\text{ana}})$;
- 19) return M ;

Fig. 6. Implementation of TAKyber.

transmission of partial decryption results from revealing the existence of covert communication to the dictator. After collecting all N valid fragments, the final combiner reconstructs the original ciphertext $\mathbf{c}_M$ through the merge and compose procedures, and finally recovers the covert message M using dsk , which is the secret key of the anamorphic Kyber instance.

D. Analysis of TAKyber

1) *Correctness Analysis*: We first analyze the correctness of TAKyber.

Theorem 1. (Correctness of TAKyber) For any covert message $M \in R_t$, firstly let $\{(apk_i, ask_i)\}_{i=1}^N, cpk, csk, dk \leftarrow \text{TAKyber.taKG}(PP_{\text{TAKyber}})$ and let the i -th innocent-looking ciphertext be generated as

$$\mathbf{c}_{M,i} \leftarrow \text{TAKyber.taEnc}(i, dk, apk_i, m_i, M, PP_{\text{TAKyber}})$$

where $m_i \in R_t$ denotes an arbitrarily selected innocent-looking message. For every innocent-looking ciphertext $\mathbf{c}_{M,i}$, its corresponding innocent-looking fragment $\mathbf{f}_{M,i}$ is generated by partial decryption:

$$\mathbf{f}_{M,i} \leftarrow \text{TAKyber.taPDec}(i, dk, ask_i, cpk, \mathbf{c}_{M,i}, PP_{\text{TAKyber}})$$

Then, with overwhelming probability,

$$\text{TAKyber.taCDec}(dk, csk, \{\mathbf{f}_{M,i}\}_{i=1}^N, PP_{\text{TAKyber}}) = M$$

That is, when all partial decryptors correctly execute the partial

decryption algorithm, the final combiner can correctly recover the covert message M .

Proof. According to the construction of **TAKyber.taEnc**, the covert message M is first encrypted by KyberPKE using the public parameters $PP_{\text{Kyber}}^{\text{ana}}$, producing $\mathbf{c}_M \in R_Q^L$. Then, the ciphertext $\mathbf{c}_M$ is decomposed as

$$\{\mathbf{d}_{\mathbf{c}_M, j}\}_{j=1}^{\lceil \log_{2\delta+1} Q \rceil} = \text{Decompose}(\mathbf{c}_M, Q, 2\delta + 1) \quad (10)$$

According to the definitions of **Decompose** and **Compose** in Definition 1 and Definition 2, the original ciphertext satisfies

$$\mathbf{c}_M = \left[\sum_{j=1}^{\lceil \log_{2\delta+1} Q \rceil} (2\delta + 1)^{j-1} \mathbf{d}_{\mathbf{c}_M, j} \right]_Q \quad (11)$$

Since each partial decryptor only processes an l -dimensional component, each decomposition component $\mathbf{d}_{\mathbf{c}_M, j}$ is further divided into

$$\{\mathbf{d}_{\mathbf{c}_M, j, k}\}_{k=1}^{\lceil L/l \rceil} = \text{Slice}(\mathbf{d}_{\mathbf{c}_M, j}, L, l) \quad (12)$$

Therefore, for all the j, k , the total number of $\mathbf{d}_{\mathbf{c}_M, j, k}$ is $\lceil L/l \rceil \lceil \log_{2\delta+1} Q \rceil$, which is equal to the number of partial decryptors N . For the i -th partial decryptor, the decomposition level and slice index of its corresponding component $\mathbf{d}_{\mathbf{c}_M, j, k}$ are determined by

$$j = \left\lfloor \frac{i-1}{\lceil L/l \rceil} \right\rfloor + 1, \quad k = i - (j-1)\lceil L/l \rceil \quad (13)$$

Thus, the component corresponding to the i -th partial decryptor is denoted as $\mathbf{r}_i = \mathbf{d}_{\mathbf{c}_M, j, k}$. Before sending to the i -th partial decryptor, $\mathbf{r}_i$ is embedded into an innocent-looking ciphertext $\mathbf{c}_{M, i} = (\mathbf{u}_i, v_i)$ by

$$\mathbf{u}_i = [\mathbf{A}_i^T \mathbf{r}_i + \mathbf{e}_{1, i}]_q, \quad v_i = [\mathbf{b}_i^T \mathbf{r}_i + e_{2, i} + \Delta m_i]_q$$

Upon receiving $\mathbf{c}_{M, i}$, the i -th partial decryptor can obtain $\mathbf{e}_{1, i}$ and $e_{2, i}$ through $\text{Hash}_{\mathbf{e}_1}$ and $\text{Hash}_{\mathbf{e}_2}$, and since $\mathbf{A}_i$ is guaranteed to be invertible in the key generation phase, the i -th partial decryptor can successfully recover

$$\mathbf{r}_i = [(\mathbf{A}_i^T)^{-1}(\mathbf{u}_i - \mathbf{e}_{1, i})]_q = \mathbf{d}_{\mathbf{c}_M, j, k}$$

and verify whether $e_2 = [v_i - \mathbf{b}_i^T \mathbf{r}_i - \Delta m_i]_q$ or not.

After the verification succeeds, the i -th partial decryptor selects an arbitrary message $m'_i \in R_t$ and regenerates the corresponding error terms $\mathbf{e}'_{1, i}$ and $e'_{2, i}$ through the hash functions. Then, following the construction of taPDec , the recovered component $\mathbf{r}_i$ is re-embedded into an innocent-looking ciphertext fragment $\mathbf{f}_{M, i} = (\mathbf{u}'_i, v'_i)$ under the combiner's public key cpk :

$$\mathbf{u}'_i = [\mathbf{A}'_i^T \mathbf{r}_i + \mathbf{e}'_{1, i}]_q, \quad v'_i = [\mathbf{b}'_i^T \mathbf{r}_i + e'_{2, i} + \Delta m'_i]_q$$

Since the fragment generation procedure is similar to the KyberPKE encryption procedure except for the adopted key parameters, the final combiner can recover the embedded component from each valid fragment using the extraction and verification procedure in taCDec . Therefore, after receiving all N valid fragments, according to the mapping between i and (j, k) in Equation (13), the final combiner has obtained the set of all decomposition components:

$$\{\mathbf{d}_{\mathbf{c}_M, j, k}\}_{1 \leq j \leq \lceil \log_{2\delta+1} Q \rceil, 1 \leq k \leq \lceil L/l \rceil} = \{\mathbf{r}_i\}_{i=1}^N \quad (14)$$

Then the final combiner can recover the covert message M as follows. According to Equation (4) and Equation (10):

$$\mathbf{d}_{\mathbf{c}_M, j} = \text{Merge} \left(\{\mathbf{d}_{\mathbf{c}_M, j, k}\}_{k=1}^{\lceil L/l \rceil}, L, l \right)$$

According to Equation (2) and Equation (12):

$$\mathbf{c}_M = \text{Compose} \left(\{\mathbf{d}_{\mathbf{c}_M, j}\}_{j=1}^{\lceil \log_{2\delta+1} Q \rceil}, Q, 2\delta + 1 \right)$$

According to the correctness of **KyberPKE**:

$$M = \text{KyberPKE.Dec}(\mathbf{c}_M, \text{dsk}, \text{PP}_{\text{Kyber}}^{\text{ana}})$$

This completes the proof.

2) *Security Analysis*: Given $\Pi = \text{KyberPKE}$ and $\Sigma_N = \text{TAKyber}$, we analyze the security properties of TAKyber through the following lemmas and theorem.

Lemma 1. TAKyber satisfies the security requirement specified by Equation (5) in Definition 9.

Proof. In the game $\text{TAnaG}_{1, \Sigma_N}^b$, the dictator obtains the key pairs of all partial decryptors $\{(apk_j, ask_j)\}_{j=1}^N$, as well as the key pair of the final combiner (cpk, csk) , and is granted access to the oracle

$$\mathcal{O}_{1, \text{ta}}(i, apk_i, m, M) = \text{taEnc}(i, dk, apk_i, m, M)$$

The oracle returns the ciphertext $\mathbf{c}_{M, i} = (\mathbf{u}_i, v_i)$, where

$$\mathbf{u}_i = [\mathbf{A}_i^T \mathbf{r}_i + \mathbf{e}_{1, i}]_q, \quad v_i = [\mathbf{b}_i^T \mathbf{r}_i + e_{2, i} + \Delta m_i]_q$$

Since (apk_i, ask_i) is generated by **KyberPKE.KG**, its distribution is identical to that of a KyberPKE key pair, namely $\mathbf{A} \sim \mathcal{U}(R_q^{l \times l})$, $\mathbf{b} \sim \mathcal{U}(R_q^l)$. Furthermore, according

to the constructions of hash functions $\text{Hash}_{\mathbf{e}_1}$ and $\text{Hash}_{\mathbf{e}_2}$, we have $\mathbf{e}_{1, i} \sim \mathcal{B}_{\eta_2}(R_q^l)$, $e_{2, i} \sim \mathcal{B}_{\eta_2}(R_q)$. Moreover, although $\mathbf{r}_i$ is obtained from the ciphertext decomposition, its range is identical to that of the randomness $\mathbf{r}$ in the KyberPKE encryption algorithm, namely, $\mathbf{r}_i \in R_{2\delta+1}^l$, and therefore $\mathbf{A}_i^T \mathbf{r}_i \sim \mathcal{U}(R_q^l)$. Hence, $\mathbf{u}_i = [\mathbf{A}_i^T \mathbf{r}_i + \mathbf{e}_{1, i}]_q \sim \mathcal{U}(R_q^l)$. Similarly, since $\mathbf{b}_i \sim \mathcal{U}(R_q^l)$, it follows that $\mathbf{b}_i^T \mathbf{r}_i \sim \mathcal{U}(R_q)$, and consequently $v_i = [\mathbf{b}_i^T \mathbf{r}_i + e_{2, i} + \Delta m_i]_q \sim \mathcal{U}(R_q)$.

Therefore, the ciphertext $(\mathbf{u}_i, v_i)$ has the same joint probability distribution as a ciphertext generated by the KyberPKE encryption algorithm. Consequently, the oracle output is computationally indistinguishable from a legitimate KyberPKE ciphertext. Thus, for any PPT dictator $\mathcal{D}$, it holds that

$$\left| \Pr[\text{TAnaG}_{1, \Sigma_N}^1(\lambda, \mathcal{D}, i) = 1] - \Pr[\text{RealG}_{\Pi}(\lambda, \mathcal{D}) = 1] \right| \leq \text{negl}(\lambda)$$

This completes the proof.

Lemma 2. TAKyber satisfies the security requirement specified by Equation (6) in Definition 9.

Proof. In $\text{TAnaG}_{2, \Sigma_N}^b$, the oracle first invokes **taEnc** to generate an innocent-looking anamorphic ciphertext, and then executes **taPDec** to finally return a fragment $\mathbf{f}_{M, i} = (\mathbf{u}'_i, v'_i)$, where $\mathbf{u}'_i = [\mathbf{A}'^T \mathbf{r}_i + \mathbf{e}'_{1, i}]_q$, $v'_i = [\mathbf{b}'^T \mathbf{r}_i + e'_{2, i} + \Delta m'_i]_q$. Since (cpk, csk) is generated by **KyberPKE.KG**, we have $\mathbf{A}' \sim \mathcal{U}(R_q^{l \times l})$, $\mathbf{b}' \sim \mathcal{U}(R_q^l)$. And according to the construction of the hash functions, we have $\mathbf{e}'_{1, i} \sim \mathcal{B}_{\eta_2}(R_q^l)$, $e'_{2, i} \sim \mathcal{B}_{\eta_2}(R_q)$. Therefore $\mathbf{u}'_i \sim \mathcal{U}(R_q^l)$, $v'_i \sim \mathcal{U}(R_q)$.

Hence, the fragment $\mathbf{f}_{M, i}$ has the same joint probability distribution as a standard KyberPKE ciphertext. Therefore, for any PPT dictator $\mathcal{D}$, there is

$$\left| \Pr[\text{TAnaG}_{2, \Sigma_N}^1(\lambda, \mathcal{D}, i) = 1] - \Pr[\text{RealG}_{\Pi}(\lambda, \mathcal{D}) = 1] \right| \leq \text{negl}(\lambda)$$

This completes the proof.

Theorem 2. TAKyber is threshold anamorphic.

Proof. According to Definition 9, TAKyber is required to satisfy the security requirements specified by Equations (5) and (6). Lemma 1 and Lemma 2 prove that these two requirements are satisfied, respectively. Therefore, TAKyber is threshold anamorphic.

Lemma 3. TAKyber satisfies the security requirement specified by Equation (7) in Definition 10.

Proof. In the game $\text{TRobustG}_{1, \Sigma_N}^b$, the adversary obtains all partial decryptors' key pairs $\{(apk_j, ask_j)\}_{j=1}^N$ and the final combiner's key pair (cpk, csk) , and has access to the oracle $\mathcal{O}_{1,0}(i, apk_i, ask_i, m) = \text{taPDec}(i, dk, ask_i, cpk, \text{Enc}(apk_i, m))$.

According to the implementation of **taPDec** in Figure 6, the algorithm first decrypts the ciphertext using ask_i to recover the message m_i . It then computes

$$\mathbf{e}_{1, i} = \text{Hash}_{\mathbf{e}_1}(apk_i, dk, m_i, i, \eta_2)$$

$$\mathbf{r}_i = [(\mathbf{A}_i^T)^{-1}(\mathbf{u}_i - \mathbf{e}_{1, i})]_q$$

Finally, it verifies whether Equation (15) is true:

$$\begin{aligned} & \text{Hash}_{\mathbf{e}_2}(apk_i, dk, m_i, i, \eta_2, \text{tag}_{\text{TAE}}) \\ &= [v_i - \mathbf{b}_i^T \mathbf{r}_i - \Delta m_i]_q \end{aligned} \quad (15)$$

For a standard KyberPKE ciphertext, the error terms are independently generated by the KyberPKE encryption algorithm rather than derived from the above two hash functions. Consequently, the ciphertext does not satisfy the consistency relation in Equation (15), except with negligible probability.

Furthermore, according to the construction, both Hash_{e_1} and Hash_{e_2} are modeled as random oracles. Since their inputs contain the double key dk , which is unknown to the adversary, the hash outputs are independent random values from the adversary's perspective. Therefore, the adversary cannot predict or manipulate the required verification values in Equation (15) and can only succeed by randomly guessing the corresponding hash output. Hence, the probability of generating a ciphertext that passes the verification is negligible. Therefore,

$$\Pr[\text{TRobustG}_{1,\Sigma_N}^0(\lambda, \mathcal{A}, i) = 1] \leq \text{negl}(\lambda)$$

In $\text{TRobustG}_{1,\Sigma_N}^1$, the oracle always returns $\perp$, so

$$\Pr[\text{TRobustG}_{1,\Sigma_N}^1(\lambda, \mathcal{A}, i) = 1] = 0$$

Consequently,

$$\left| \Pr[\text{TRobustG}_{1,\Sigma_N}^0(\lambda, \mathcal{A}, i) = 1] - \Pr[\text{TRobustG}_{1,\Sigma_N}^1(\lambda, \mathcal{A}, i) = 1] \right| \leq \text{negl}(\lambda)$$

This completes the proof.

Lemma 4. TAKyber satisfies the security requirement specified by Equation (8) in Definition 10.

Proof. In the game $\text{TRobustG}_{2,\Sigma_N}^b$, the oracle first generates all valid fragments $\{\mathbf{f}_{M,j}\}_{j=1}^N$. Then, the adversary can submit an arbitrary forged fragment $\mathbf{f}$, and the oracle invokes

$$\mathbf{taCDec}(dk, csk, \{\mathbf{f}_{M,j}\}_{1 \leq j \leq N, j \neq i} \cup \{\mathbf{f}\})$$

to perform combination and recover the covert message.

According to the implementation of $\mathbf{taCDec}$ in Figure 6, the algorithm first decrypts using the final combiner's secret key csk to obtain m' , and then computes

$$\begin{aligned} \mathbf{e}'_1 &= \text{Hash}_{e_1}(cpk, dk, m', i, \eta_2) \\ \mathbf{r} &= [(\mathbf{A}_0^T)^{-1}(\mathbf{u}' - \mathbf{e}'_1)]_q \end{aligned} \quad (16)$$

Finally, it verifies whether Equation (17) is true:

$$\text{Hash}_{e_2}(cpk, dk, m', i, \eta_2, \text{tag}_F) = [v' - \mathbf{b}_0^T \mathbf{r} - \Delta m']_q \quad (17)$$

Only if Equation (17) holds, the values $\mathbf{e}'_1$ and $\mathbf{r}$ obtained from Equation (16) are accepted as the valid values of $\mathbf{e}'_{1,i}$ and $\mathbf{r}_i$ for the corresponding fragment. Otherwise, the fragment is rejected.

Moreover, $\mathbf{taCDec}$ accepts the reconstructed message only when all N fragments satisfy the consistency verification in Equation (17). If any fragment fails the verification, the algorithm outputs $\perp$ instead of recovering the message M .

Since both Hash_{e_1} and Hash_{e_2} are modeled as random oracles and their inputs contain the double key dk , the adversary cannot predict the corresponding hash outputs. Therefore, the adversary cannot construct a forged fragment that satisfies the above consistency relation. The probability of successfully generating such a fragment is negligible. Hence,

$$\Pr[\text{TRobustG}_{2,\Sigma_N}^0(\lambda, \mathcal{A}, i) = 1] \leq \text{negl}(\lambda)$$

In $\text{TRobustG}_{2,\Sigma_N}^1$, the oracle always returns $\perp$, so

$$\Pr[\text{TRobustG}_{2,\Sigma_N}^1(\lambda, \mathcal{A}, i) = 1] = 0$$

Therefore,

$$\left| \Pr[\text{TRobustG}_{2,\Sigma_N}^0(\lambda, \mathcal{A}, i) = 1] - \Pr[\text{TRobustG}_{2,\Sigma_N}^1(\lambda, \mathcal{A}, i) = 1] \right| \leq \text{negl}(\lambda)$$

This completes the proof.

Theorem 3. TAKyber is robust.

Proof. According to Definition 10, in order to prove robustness, TAKyber is required to satisfy the security requirements specified by Equations (7) and (8). Lemma 3 and Lemma 4 prove that these two requirements are satisfied, respectively. This completes the proof.

3) Homomorphism Analysis:

Lemma 5. (Additive homomorphism of decomposition and composition) For a finite set of vectors $\mathcal{S}_x$ which satisfies $\forall \mathbf{x} \in \mathcal{S}_x, \mathbf{x} \in R_q^l$. Let

$$\{\mathbf{x}_j\}_{j=1}^{\lceil \log_B q \rceil} = \mathbf{Decompose}(\mathbf{x}, q, B), \quad \mathbf{x} \in \mathcal{S}_x$$

Then, the following relation holds:

$$\mathbf{Compose}\left(\left\{\sum_{\mathbf{x} \in \mathcal{S}_x} \mathbf{x}_j\right\}_{j=1}^{\lceil \log_B q \rceil}, q, B\right) = \sum_{\mathbf{x} \in \mathcal{S}_x} \mathbf{x}$$

Proof. We first consider the case of two vectors $\mathbf{x}, \mathbf{y} \in R_q^l$. According to Equation 1, we have

$$\mathbf{x} = \sum_{j=1}^{\lceil \log_B q \rceil} B^{j-1} \mathbf{x}_j, \quad \mathbf{y} = \sum_{j=1}^{\lceil \log_B q \rceil} B^{j-1} \mathbf{y}_j$$

According to Definition 2 of the **Compose** algorithm,

$$\begin{aligned} & \mathbf{Compose}\left(\{\mathbf{x}_j + \mathbf{y}_j\}_{j=1}^{\lceil \log_B q \rceil}, q, B\right) \\ &= \sum_{j=1}^{\lceil \log_B q \rceil} B^{j-1}(\mathbf{x}_j + \mathbf{y}_j) \\ &= \sum_{j=1}^{\lceil \log_B q \rceil} B^{j-1} \mathbf{x}_j + \sum_{j=1}^{\lceil \log_B q \rceil} B^{j-1} \mathbf{y}_j \\ &= \mathbf{x} + \mathbf{y} \end{aligned}$$

Therefore, the lemma holds for any two vectors. Since vector addition is associative, this result can be naturally extended to any finite set. This completes the proof.

Lemma 6. (Additive homomorphism of KyberPKE) For a finite set of plaintexts $\mathcal{S}_m$ which satisfies

$$\forall m \in \mathcal{S}_m, \mathbf{c}_m = \mathbf{KyberPKE.Enc}(m, pk, \text{PP}_{\text{Kyber}})$$

where all ciphertexts are generated under the same public key $pk = (\mathbf{A}, \mathbf{b})$. Then, the following relation holds:

$$\mathbf{KyberPKE.Dec}\left(\sum_{m \in \mathcal{S}_m} \mathbf{c}_m, sk, \text{PP}_{\text{Kyber}}\right) = \sum_{m \in \mathcal{S}_m} m$$

where $sk = \mathbf{s}$ is the corresponding secret key of pk .

Proof. We first consider the case of two plaintexts m_1, m_2 . According to the encryption algorithm of KyberPKE in Figure 3, the corresponding ciphertexts $\mathbf{c}_{m_1}$ and $\mathbf{c}_{m_2}$ can be expressed as $\mathbf{c}_{m_1} = (\mathbf{u}_1, v_1)$, $\mathbf{c}_{m_2} = (\mathbf{u}_2, v_2)$, where

$$\begin{aligned} \mathbf{u}_1 &= [\mathbf{A}^T \mathbf{r}_1 + \mathbf{e}_1^{(1)}]_q, & v_1 &= [\mathbf{b}^T \mathbf{r}_1 + e_2^{(1)} + \Delta m_1]_q \\ \mathbf{u}_2 &= [\mathbf{A}^T \mathbf{r}_2 + \mathbf{e}_1^{(2)}]_q, & v_2 &= [\mathbf{b}^T \mathbf{r}_2 + e_2^{(2)} + \Delta m_2]_q \end{aligned}$$

Therefore, the aggregated ciphertext $\mathbf{c}_{m_1} + \mathbf{c}_{m_2}$ satisfies

$$\mathbf{u}_1 + \mathbf{u}_2 = [\mathbf{A}^T(\mathbf{r}_1 + \mathbf{r}_2) + \mathbf{e}_1^{(1)} + \mathbf{e}_1^{(2)}]_q$$

$$v_1 + v_2 = [\mathbf{b}^T(\mathbf{r}_1 + \mathbf{r}_2) + e_2^{(1)} + e_2^{(2)} + \Delta(m_1 + m_2)]_q$$

Hence,

$$(v_1 + v_2) - \mathbf{s}^T(\mathbf{u}_1 + \mathbf{u}_2) = \Delta(m_1 + m_2) + e_2^{(1)} + e_2^{(2)} - \mathbf{s}^T \mathbf{e}_1^{(1)} - \mathbf{s}^T \mathbf{e}_1^{(2)}$$

When

$$\left\| e_2^{(1)} + e_2^{(2)} - \mathbf{s}^T \mathbf{e}_1^{(1)} - \mathbf{s}^T \mathbf{e}_1^{(2)} \right\|_\infty < \frac{\Delta}{2} \quad (18)$$

we have

$$\begin{aligned} & \text{KyberPKE.Dec}(\mathbf{c}_{m_1} + \mathbf{c}_{m_2}, sk, \text{PP}_{\text{Kyber}}) \\ &= \left\lfloor \frac{[v_1 + v_2 - \mathbf{s}^T(\mathbf{u}_1 + \mathbf{u}_2)]_q}{\Delta} \right\rfloor \\ &= m_1 + m_2 \end{aligned}$$

Therefore, the lemma holds for two plaintexts. This result can be naturally extended to any finite set of plaintexts. This completes the proof.

Theorem 4. (Homomorphism of TAKyber) For a finite set of messages $\mathcal{S}_M$ satisfying $\forall M \in \mathcal{S}_M, M \in R_t$, consider the generation of the innocent-looking ciphertexts for each $M \in \mathcal{S}_M$ and $1 \leq i \leq N$ as

$$\mathbf{c}_{M,i} \leftarrow \text{TAKyber.taEnc}(i, dk, apk_i, m_i, M, \text{PP}_{\text{TAKyber}})$$

Then, the i -th partial decryptor computes the aggregated innocent-looking fragment as $\mathbf{f}_{\widetilde{M},i} \leftarrow \text{TAKyber.taHPDec}(i, dk, ask_i, cpk, \{\mathbf{c}_{M,i}\}_{M \in \mathcal{S}_M}, \text{PP}_{\text{TAKyber}})$, where we define $\widetilde{M} = \sum_{M \in \mathcal{S}_M} M$. Then, with overwhelming probability:

$$\widetilde{M} = \text{TAKyber.taCDec}(dk, csk, \{\mathbf{f}_{\widetilde{M},i}\}_{i=1}^N, \text{PP}_{\text{TAKyber}})$$

Proof. According to **TAKyber.taEnc** in Figure 6, each message $M \in \mathcal{S}_M$ is first encrypted and decomposed as

$$\mathbf{c}_M = \text{KyberPKE.Enc}(M, dpk, \text{PP}_{\text{Kyber}}^{\text{ana}})$$

$$\{\mathbf{d}_{\mathbf{c}_M,j}\}_{j=1}^{\lceil \log_{2\delta+1} Q \rceil} = \text{Decompose}(\mathbf{c}_M, Q, 2\delta + 1)$$

For the i -th partial decryptor, the corresponding embedded component is denoted as $\mathbf{r}_{M,i} = \mathbf{d}_{\mathbf{c}_M,j,k}$. According to the **TAKyber.taHPDec** algorithm in Figure 7, after verifying

TAKyber.taHPDec($i, dk, ask_i, cpk, \{\mathbf{c}_{M,i}\}_{M \in \mathcal{S}_M},$

$\text{PP}_{\text{TAKyber}} \rightarrow \mathbf{f}_{\widetilde{M},i} / \perp$ ($\widetilde{M} = \sum_{M \in \mathcal{S}_M} M$):

- 1) Parse $ask_i = (\mathbf{A}_i, \mathbf{b}_i, \mathbf{s}_i)$, $cpk = (\mathbf{A}_0, \mathbf{b}_0)$;
- 2) For each $M \in \mathcal{S}_M$:
- 3) $m_i \leftarrow \text{KyberPKE.Dec}(\mathbf{c}_{M,i}, ask_i, \text{PP}_{\text{Kyber}})$;
- 4) $\mathbf{e}_{1,i} \leftarrow \text{Hash}_{e_1}(cpk, dk, m_i, i, \eta_2)$;
- 5) $\mathbf{r}_{M,i} \leftarrow [(\mathbf{A}_i^T)^{-1}(\mathbf{u}_i - \mathbf{e}_{1,i})]_q$;
- 6) if $\text{Hash}_{e_2}(apk_i, dk, m_i, i, \eta_2, \text{tag}_{\text{TAE}}) \neq [v_i - \mathbf{b}_i^T \mathbf{r}_{M,i} - \Delta m_i]_q$: **return** $\perp$;
- 7) Let $\widetilde{M} = \sum_{M \in \mathcal{S}_M} M$; $\mathbf{r}_{\widetilde{M},i} = \sum_{M \in \mathcal{S}_M} \mathbf{r}_{M,i}$;
- 8) Arbitrarily select $m'_i \in R_t$;
- 9) $\mathbf{e}'_{1,i} \leftarrow \text{Hash}_{e_1}(cpk, dk, m'_i, i, \eta_2)$;
- 10) $e'_{2,i} \leftarrow \text{Hash}_{e_2}(cpk, dk, m'_i, i, \eta_2, \text{tag}_F)$;
- 11) $\mathbf{u}'_i = [\mathbf{A}_0^T \mathbf{r}_{\widetilde{M},i} + \mathbf{e}'_{1,i}]_q$;
- 12) $v'_i = [\mathbf{b}_0^T \mathbf{r}_{\widetilde{M},i} + e'_{2,i} + \Delta m'_i]_q$;
- 13) **return** $\mathbf{f}_{\widetilde{M},i} = (\mathbf{u}'_i, v'_i)$;

all received ciphertexts, the i -th partial decryptor aggregates the recovered components as $\mathbf{r}_{\widetilde{M},i} = \sum_{M \in \mathcal{S}_M} \mathbf{r}_{M,i}$.

Then, the aggregated component is re-embedded into an innocent-looking ciphertext fragment $\mathbf{f}_{\widetilde{M},i}$. Since $\mathbf{f}_{\widetilde{M},i}$ is generated following the same KyberPKE encryption procedure as the fragment generated in **TAKyber.taPDec**, the final combiner can extract the embedded component by executing the similar extraction procedure as in **TAKyber.taCDec**:

$$\mathbf{r}_{\widetilde{M},i} = [(\mathbf{A}_0^T)^{-1}(\mathbf{u}'_i - \mathbf{e}'_{1,i})]_q$$

Therefore, the final combiner obtains $\{\mathbf{r}_{\widetilde{M},i}\}_{i=1}^N$. Since for each $M \in \mathcal{S}_M$, according to Equation (14), there is

$$\{\mathbf{d}_{\mathbf{c}_M,j,k}\}_{1 \leq j \leq \lceil \log_{2\delta+1} Q \rceil, 1 \leq k \leq \lceil L/l \rceil} = \{\mathbf{r}_{M,i}\}_{i=1}^N$$

where the relation between j, k and i are in Equation (13). So

$$\mathbf{r}_{\widetilde{M},i} = \sum_{M \in \mathcal{S}_M} \mathbf{d}_{\mathbf{c}_M,j,k} \quad (19)$$

According to Equation (3) and Equation (12), for each $1 \leq j \leq \lceil \log_{2\delta+1} Q \rceil$ and $M \in \mathcal{S}_M$ we have

$$\mathbf{d}_{\mathbf{c}_M,j} = (\mathbf{d}_{\mathbf{c}_M,j,1}, \mathbf{d}_{\mathbf{c}_M,j,2}, \dots, \mathbf{d}_{\mathbf{c}_M,j,\lceil L/l \rceil})$$

Consequently for each $1 \leq j \leq \lceil \log_{2\delta+1} Q \rceil$, there is

$$\sum_{M \in \mathcal{S}_M} \mathbf{d}_{\mathbf{c}_M,j} = (\sum_{M \in \mathcal{S}_M} \mathbf{d}_{\mathbf{c}_M,j,1}, \dots, \sum_{M \in \mathcal{S}_M} \mathbf{d}_{\mathbf{c}_M,j,\lceil L/l \rceil})$$

Since Equation (19) has already given all $\sum_{M \in \mathcal{S}_M} \mathbf{d}_{\mathbf{c}_M,j,k}$, the final decryptor can apply the above equation repeatedly to obtain $\{\sum_{M \in \mathcal{S}_M} \mathbf{d}_{\mathbf{c}_M,j}\}_{j=1}^{\lceil \log_{2\delta+1} Q \rceil}$. Then, according to Equation (10):

$$\{\mathbf{d}_{\mathbf{c}_M,j}\}_{j=1}^{\lceil \log_{2\delta+1} Q \rceil} = \text{Decompose}(\mathbf{c}_M, Q, 2\delta + 1)$$

According to Lemma 5:

$$\text{Compose}(\{\sum_{M \in \mathcal{S}_M} \mathbf{d}_{\mathbf{c}_M,j}\}_{j=1}^{\lceil \log_{2\delta+1} Q \rceil}, Q, 2\delta + 1) = \sum_{M \in \mathcal{S}_M} \mathbf{c}_M$$

According to the homomorphic property of KyberPKE in Lemma 6, since $\mathbf{c}_M \leftarrow \text{KyberPKE.Enc}(M, dpk, \text{PP}_{\text{Kyber}}^{\text{ana}})$, the final combiner can recover $\widetilde{M}$ through

$$\begin{aligned} & \text{KyberPKE.Dec} \left(\sum_{M \in \mathcal{S}_M} \mathbf{c}_M, dsk, \text{PP}_{\text{Kyber}}^{\text{ana}} \right) \\ &= \sum_{M \in \mathcal{S}_M} M = \widetilde{M} \end{aligned}$$

This completes the proof.

The above additive homomorphism demonstrates that TAKyber enables partial decryptors to directly aggregate innocent-looking ciphertexts containing different covert messages during the partial decryption process, without processing each ciphertext individually. The resulting partial decryption fragments contain covert information equivalent to the sum of the covert messages embedded in all input ciphertexts. Therefore, this property not only reduces redundant computational overhead in large-scale data scenarios and improves system efficiency, but more importantly, further enhances privacy protection. Specifically, during the aggregated decryption process, the final combiner cannot obtain any individual covert message and can only recover the aggregated result of all covert messages by executing **taCDec**. Meanwhile, each partial decryptor only receives a single component obtained

Fig. 7. The homomorphic partial decryption algorithm.

after the covert ciphertext is encrypted and decomposed, and therefore cannot independently recover the corresponding covert message. Consequently, before the aggregated covert message is finally recovered, no entity other than the sender who generates the covert ciphertext can obtain the specific covert message content.

This property is of great significance for applications such as secure multi-party computation and privacy-preserving federated learning under dictator surveillance. In such scenarios, the system must not only conceal the existence of the covert communication channel to the dictator, but also prevent sensitive information from being leaked among participants. TAKyber simultaneously satisfies these requirements, making it suitable for privacy-preserving computation tasks in regulated environments. In the next section, we further introduce the application of TAKyber in privacy-preserving federated learning under dictator surveillance.

Furthermore, since the **TAKyber.taKG** phase in Figure 6 requires the generated matrix $\mathbf{A}$ to be invertible, we prove in Appendix C that $\mathbf{A}$ is invertible with high probability. Therefore, this requirement has a negligible impact on the execution efficiency of the scheme.

IV. COVERT FEDERATED LEARNING BASED ON TAE

A. System Model

We consider a federated learning system under the rule of a dictator, which is called covert federated learning. To establish a covert communication channel, the system employs the TAKyber scheme.

The system is composed of a server and a set of clients denoted by $\mathcal{C}$, where $|\mathcal{C}| = N_C \geq N$. The server acts as the global model aggregator and the final combiner of TAKyber, while all clients are responsible for local training, and N clients with sufficient computational capabilities are randomly selected as the partial decryptors of TAKyber.

In each training round, the server first anamorphically broadcasts the current global model to all clients. Then, each client $C \in \mathcal{C}$ updates its model and performs local training on its private dataset and obtains the corresponding local gradient, denoted by g_C .

To protect the privacy of local gradients and establish a covert communication channel under dictator surveillance, TAKyber is employed for transmitting gradient information among federated participants.

In practical federated learning systems, the gradient usually consists of floating-point values and its dimension may not exactly match the input requirement of TAKyber. Therefore, additional preprocessing procedures, such as quantization, encoding, and partitioning, are required before encryption. For simplicity, these preprocessing procedures are omitted in this framework, and g_C is directly regarded as the plaintext input of TAKyber.

Description of the detailed training procedures is as follows.

B. Framework Construction

Before the training process starts, the system first initializes the TAKyber scheme. Specifically, N clients with sufficient

computational capabilities are randomly selected as the partial decryptors and remain unchanged throughout the entire training process. The server and the selected partial decryptors execute the TAKyber key generation algorithm:

$$\{(apk_i, ask_i)\}_{i=1}^N, (cpk, csk), dk \\ \leftarrow \mathbf{TAKyber.KeyGen}(\lambda, N, \mathbf{PP}_{\mathbf{TAKyber}})$$

The public parameters $\mathbf{PP}_{\mathbf{TAKyber}}$, the double key dk , the public keys $\{apk_i\}_{i=1}^N$, and cpk are distributed to the server and all clients. Each selected partial decryptor i keeps its corresponding secret key ask_i , while the server owns csk . The dictator is assumed to be able to coerce all participants into revealing their keys. Therefore, the dictator obtains $\{(apk_i, ask_i)\}_{i=1}^N$, cpk , and csk . However, the double key dk is not disclosed to the dictator and remains known only to legitimate participants of covert federated learning.

At the beginning of each round, the server first broadcasts the current global gradient to all clients through the anamorphic sending algorithm **aSend** defined in Appendix B. Each client receives the gradient through the anamorphic receiving algorithm **aReceive** defined in Appendix B and updates its model accordingly.

After updating the model w , each client $C \in \mathcal{C}$ performs local training on its private dataset and obtains its local gradient through $g_C \leftarrow \mathbf{LocalTrain}(C, w)$.

Subsequently, each client C generates N innocent-looking ciphertexts for every $1 \leq i \leq N$:

$$c_{C,i} \leftarrow \mathbf{TAKyber.taEnc}(i, dk, apk_i, m_i, g_C, \mathbf{PP}_{\mathbf{TAKyber}})$$

where m_i is an arbitrary chosen innocent-looking message for preserving indistinguishability of the transmitted ciphertexts.

The generated innocent-looking ciphertexts are then transmitted to the corresponding partial decryptors. If one client itself serves as a partial decryptor, its corresponding ciphertext is directly retained locally without additional transmission.

Therefore, the i -th partial decryptor obtains a set of innocent-looking ciphertexts $\{c_{C,i}\}_{C \in \mathcal{C}}$ from all the clients.

According to the homomorphic property of TAKyber, partial decryptors do not need to process each ciphertext independently. Instead, the i -th partial decryptor executes

$$f_{\tilde{g},i} \leftarrow \mathbf{TAKyber.taHPDec}(i, dk, ask_i, cpk, \\ \{c_{C,i}\}_{C \in \mathcal{C}}, \mathbf{PP}_{\mathbf{TAKyber}})$$

to generate a innocent-looking fragment and send it to the server, where $\tilde{g} = \sum_{C \in \mathcal{C}} g_C$.

After receiving all valid fragments from the partial decryptors, the server executes the final combination algorithm:

$$\tilde{g} = \mathbf{TAKyber.taCDec}(dk, csk, \{f_{\tilde{g},i}\}_{i=1}^N, \mathbf{PP}_{\mathbf{TAKyber}})$$

The server then computes the averaged gradient $\bar{g} = \tilde{g}/N_C$ as the global gradient and updates the global model $w \leftarrow w - \rho \bar{g}$, where ρ denotes the global learning rate.

The latest global gradient is then prepared for the next round. The above procedure completes one round of the training process of our proposed covert FL framework. The training process is repeated until the global model converges.

V. PERFORMANCE EVALUATION

In this section, we evaluate the performance of TAKyber from both theoretical and practical perspectives. First, we analyze the parameter requirements of TAKyber and provide concrete parameter settings. Then, we evaluate the computational overhead of TAKyber under different parameter configurations. Finally, we demonstrate the effectiveness of TAKyber in the covert federated learning scenario through experimental evaluation.

A. Theoretical Analysis

In this section, we first analyze the theoretical constraints on the parameter selection of TAKyber. Then, based on the requirements of covert federated learning, we identify additional conditions for practical deployment and present several candidate parameter sets.

Since TAKyber is constructed based on the Kyber framework, its parameter selection can follow the settings adopted in [10]. For the TAKyber parameters defined in Equation (9), to enable efficient polynomial multiplication using NTT, both Q and q are required to be prime numbers and satisfy

$$Q \equiv 1 \pmod{2n}, \quad q \equiv 1 \pmod{2n} \quad (20)$$

Another important factor affecting parameter selection is noise growth. During homomorphic computations, the noise level may increase rapidly. In the covert federated learning framework introduced in Section IV, each partial decryptor performs homomorphic partial decryption on N_C received innocent-looking ciphertexts in each training round. Therefore, we analyze the parameter conditions required to keep the noise within a controllable range.

First, we analyze the noise distribution of KyberPKE in Figure 3. According to **KyberPKE.Dec**, we have

$$\begin{aligned} v - s^T u &= (As + e)^T r + e_2 + \Delta m - s^T (A^T r + e_1) \\ &= \Delta m + e^T r + e_2 - s^T e_1 \end{aligned}$$

Therefore, the decryption noise of KyberPKE is

$$E = e^T r + e_2 - s^T e_1 \quad (21)$$

According to the **TAKyber.taHPDec** algorithm in Figure 7, when the i -th partial decryptor conducting homomorphic partial decryption, there is

$$u'_i = [A_0^T r_{\widetilde{M},i} + e'_{1,i}]_q, \quad v'_i = [b_0^T r_{\widetilde{M},i} + e'_{2,i} + \Delta m'_i]_q$$

According to the TAKyber scheme in Figure 6, when the final combiner executing **TAKyber.taCDec**, there is

$$\begin{aligned} v'_i - s_0^T u'_i &= (A_0 s_0 + e_0)^T r_{\widetilde{M},i} + e'_{2,i} + \Delta m'_i \\ &\quad - s_0^T (A_0^T r_{\widetilde{M},i} + e'_{1,i}) \\ &= \Delta m'_i + e_0^T r_{\widetilde{M},i} + e'_{2,i} - s_0^T e'_{1,i} \end{aligned}$$

The total noise is $\widetilde{E} = e_0^T r_{\widetilde{M},i} + e'_{2,i} - s_0^T e'_{1,i}$.

Since Equation (19) gives $r_{\widetilde{M},i} = \sum_{M \in S_M} d_{c_M,j,k}$, where $d_{c_M,j,k}$ denotes the component obtained after the decomposition and slicing operations, we approximate its distribution as $d_{c_M,j,k} \sim \mathcal{U}_\delta(R_q^l)$. And because of the definition mentioned previously, we have

$$\begin{cases} e_0 \sim \mathcal{B}_{\eta_1}(R_q^l), s_0 \sim \mathcal{B}_{\eta_1}(R_q^l) \\ e'_{1,i} \sim \mathcal{B}_{\eta_2}(R_q^l), e'_{2,i} \sim \mathcal{B}_{\eta_2}(R_q^l) \end{cases}$$

Therefore, the distribution of any coefficient of the total noise $\widetilde{E}$ can be viewed as the combination of three independent parts. The first part is the accumulation of $|S_M|nl$ products between $\mathcal{B}_{\eta_1}$ -distributed terms and $\mathcal{U}_\delta$ -distributed terms. The second part follows $\mathcal{B}_{\eta_2}$. The third part is the accumulation of nl products between $\mathcal{B}_{\eta_1}$ -distributed terms and $\mathcal{B}_{\eta_2}$ -distributed terms. Because we have

$$\begin{cases} X \sim \mathcal{B}_{\eta_1} : & \mathbb{E}[X] = 0, \text{Var}(X) = \eta_1/2 \\ X \sim \mathcal{B}_{\eta_2} : & \mathbb{E}[X] = 0, \text{Var}(X) = \eta_2/2 \\ X \sim \mathcal{U}_\delta : & \mathbb{E}[X] = 0, \text{Var}(X) = \delta(\delta+1)/3 \end{cases}$$

According to the central limit theorem, any coefficient of $\widetilde{E}$ approximately follows a discrete Gaussian distribution with zero mean and a variance of

$$\sigma_{\widetilde{E}}^2 = |S_M|nl \frac{\eta_1 \delta(\delta+1)}{6} + \frac{\eta_2}{2} + nl \frac{\eta_1 \eta_2}{4} \quad (22)$$

In practical applications, the upper bound of the decryption failure probability, i.e., the probability that any coefficient is incorrect is typically set to $\epsilon = 2^{-128}$. Therefore, the equation is as follows:

$$1 - \left(1 - \Pr \left[\left\| \widetilde{E} \right\|_\infty > \frac{\Delta}{2} \right] \right)^n < \epsilon$$

Solution of the equation is:

$$\sigma_{\widetilde{E}} < \frac{\lfloor q/t \rfloor}{2\sqrt{2} \operatorname{erfc}^{-1}(1 - (1 - \epsilon)^{1/n})} \quad (23)$$

where $\operatorname{erfc}(\cdot)$ denotes the complementary error function.

The noise constraint under PP_{Kyber} has been analyzed above. For $\text{PP}_{\text{Kyber}}^{\text{ana}}$, the noise analysis is much simpler. In this case, the terms in Equation (21) e, r, e_2 , and e_1 satisfy $e, r \sim \mathcal{B}_{\eta_1}(R_q^L)$, $e_2 \sim \mathcal{B}_{\eta_2}(R_q)$, and $e_1 \sim \mathcal{B}_{\eta_2}(R_q^L)$. Therefore any coefficient of E approximately follows a discrete Gaussian distribution with zero mean and a variance of

$$\sigma_E^2 = nL \frac{\eta_1' \delta'(\delta'+1)}{6} + \frac{\eta_2'}{2} + nL \frac{\eta_1' \eta_2'}{2} \quad (24)$$

And the following condition should be satisfied:

$$\sigma_E \leq \frac{\lfloor Q/t' \rfloor}{2\sqrt{2} \operatorname{erfc}^{-1}(1 - (1 - \epsilon)^{1/n})} \quad (25)$$

Furthermore, the parameters of PP_{Kyber} and $\text{PP}_{\text{Kyber}}^{\text{ana}}$ should achieve classical and quantum security levels higher than those of the light setting in [10], according to the results obtained by running the `MLWE_security.py` from the code provided in [10]. Meanwhile, the resulting number of partial decryptors N should satisfy $N \leq N_C$. The detailed security levels are given in the parameter selection table in the next section.

B. Parameter Selection

When TAKyber is applied in covert federated learning, the number of aggregated covert messages satisfies $|S_M| = N_C$. Accordingly, under the constraints in Equations (21)(22)(23)(24)(25), we present concrete parameter configurations for $N_C = 5, 10$, and 20 in Table I, respectively. These configurations enable TAKyber to support different practical federated learning application scenarios.

During federated learning, we transform the gradient vector into an integer vector with coefficients in $[-128, 128]$, and then decompose it into a longer integer vector with coefficients in

TABLE I
PARAMETER SELECTION OF TAKYBER

Parameter	$N_C = 5$	$N_C = 10$	$N_C = 20$
n	256	256	256
q	12289	7681	7681
l	3	3	3
t	2	2	2
η_1	2	1	2
η_2	2	1	2
δ	3	3	1
Q	12289	23041	40961
L	3	3	3
t'	11	21	41
η'_1	2	2	2
η'_2	2	2	2
δ'	1	1	1
N	5	6	10
fail-prob	$2^{-161}, 2^{-172}$	$2^{-167}, 2^{-166}$	$2^{-238}, 2^{-137}$
security-level	(153,139)(153,139)	(148,135)(142,129)	(162,147)(133,121)

$[-1, 1]$. Thus, after aggregating N_C clients, each coefficient of the resulting sum vector lies in $[-N_C, N_C]$. To avoid modular wrap-around, we set $t' = 2N_C + 1$. Accordingly, $t' = 11, 21$, and 41 are selected for $N_C = 5, 10$, and 20, respectively.

Moreover, we optimize the parameter selection to improve the embedding efficiency of TAKyber by reducing the scale of the padding part and maximizing the embedded covert message capacity through a larger decomposition range δ and anamorphic plaintext modulus t' , while keeping q, Q as small as possible under the correctness and security guarantees.

In Table I, the first part lists the parameters of PP_{Kyber} , while the second part presents the parameters of $PP_{\text{Kyber}}^{\text{ana}}$. Here, N denotes the number of partial decryptors in TAKyber. The fail-prob row reports the decryption failure probabilities of PP_{Kyber} , where the depth of the addition operation is N_C , and $PP_{\text{Kyber}}^{\text{ana}}$, respectively. The security-level row provides the classical and quantum security levels of both parameter sets. As shown in the table, all decryption failure probabilities and security levels meet the required criteria.

C. Performance Evaluation of TAKyber

In this section, we evaluate the computational overhead of TAKyber. The experiments are conducted on a computer equipped with a 13th Gen Intel(R) Core(TM) i5-13490F CPU and 32 GB RAM. The running time under different parameter settings is summarized in Table II.

Since TAKyber is constructed based on KyberPKE, the additional overhead mainly comes from matrix inversion of $\mathbf{A}$, ciphertext decomposition, hash computation, and partial decryption and combination procedures. Since $\mathbf{A}$ only appears in the public key, its inverse can be precomputed during key generation. Although this introduces an initialization cost, **taKG** is executed only once and thus has limited impact on practical applications. The running time of **taKG** in Table II includes matrix inversion. Consequently, we mainly evaluate the running time of **taEnc**, **taPDec**, **taCDec**, and **taHPDec**. The results show that the overhead is acceptable for practical applications.

D. Experiments of Application in Covert Federated Learning

In this section, we evaluate the computational overhead introduced by TAKyber in covert federated learning. Under the same hardware configuration, we implement FedAvg [12] on the MNIST dataset using a three-layer CNN model with $N_C = 5, 10$, and 20 clients. For each setting, the model is trained for 20 communication rounds. We record the running time of each computational stage and compare three schemes: the original FedAvg, FedAvg with KyberPKE under PP_{Kyber} , and FedAvg with TAKyber under PP_{TAKyber} . The results are presented in Table III.

As shown in Table III, incorporating TAE into federated learning introduces additional computational overhead compared with both the original FedAvg and FedAvg with KyberPKE. This overhead mainly comes from the partial decryption process required by the threshold anamorphic encryption mechanism. Nevertheless, the resulting running time remains acceptable for practical deployment. More importantly, in scenarios where federated learning operates under the surveillance and control of a dictator, such additional overhead is necessary to achieve privacy-preserving gradient exchange while maintaining the covert nature of the communication.

VI. CONCLUSION

In this paper, we proposed a module-lattice-based Threshold Anamorphic Encryption (TAE) framework for covert communication under dictator-controlled environments. Unlike existing anamorphic encryption schemes that mainly consider the single-receiver setting, TAE extends receiver-AE to the threshold setting, enabling multiple participants to collaboratively recover covert messages while preserving the indistinguishability of innocent-looking communications.

Based on the proposed TAE framework, we constructed a concrete scheme, named TAKyber, from the KyberPKE framework. By embedding covert information into ciphertext randomness rather than modifying the public matrix structure,

TABLE II
RUNNING TIME OF TAKYBER ALGORITHMS

Algorithm	$N_C=5, N=5$	$N_C=10, N=6$	$N_C=20, N=10$
taKG	14984.4 ms	18524.0 ms	30730.6 ms
taEnc	216.1 ms	268.8 ms	462.9 ms
taPDec	46.7 ms	49.8 ms	48.9 ms
taCDec	4.8 ms	4.9 ms	4.9 ms
taHPDec	239.5 ms	461.3 ms	924.8 ms

TABLE III
RUNNING TIME OF FEDAVG WITH DIFFERENT CRYPTOGRAPHIC SCHEMES

Scheme	Stage	$N_C=5$	$N_C=10$	$N_C=20$
FedAvg	train	0.117 s	0.076 s	0.067 s
	aggregate	0.001 s	0.001 s	0.001 s
	total	0.118 s	0.077 s	0.068 s
FedAvg + KyberPKE	train + encrypt	1.866 s	2.656 s	2.685 s
	aggregate+decrypt	0.508 s	0.947 s	1.138 s
	total	2.410 s	3.603 s	3.823 s
FedAvg + TAKyber	train + encrypt	40.2 s	47.6 s	70.7 s
	partial decrypt	41.6 s	83.9 s	148.9 s
	combine decrypt	0.62 s	0.64 s	0.66 s
	total	82.42 s	132.1 s	220.3 s

TAKyber avoids the additional lattice trapdoor requirements and can be directly instantiated over practical lattice-based encryption schemes such as Kyber. Furthermore, through ciphertext decomposition, hash-based binding, and threshold partial decryption, TAKyber enables secure reconstruction of covert messages while preventing intermediate parties without the double key from obtaining any information about them. We formally analyzed the correctness, security, and robustness of TAKyber, and evaluated its parameter selection and computational overhead. Moreover, we applied TAKyber to a privacy-preserving federated learning scenario, demonstrating its capability of supporting covert gradient exchange with acceptable computational overhead.

As future work, we will investigate more flexible threshold structures, such as t -out-of- N threshold anamorphic encryption based on [16], to improve the adaptability of the proposed framework. We will also explore optimization methods to further reduce the computational and communication overhead of TAKyber and extend its applications to larger-scale distributed learning and other privacy-preserving scenarios.

REFERENCES

- [1] G. Persiano, D. H. Phan, and M. Yung, "Anamorphic encryption: Private communication against a dictator," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn. (EUROCRYPT)*, Cham, Switzerland: Springer, 2022, pp. 34–63.
- [2] Wang Y, Chen R, Huang X, et al. Sender-anamorphic encryption reformulated: Achieving robust and generic constructions[C]//International Conference on the Theory and Application of Cryptology and Information Security. Singapore: Springer Nature Singapore, 2023: 135-167.
- [3] Banfi F, Gegier K, Hirt M, et al. Anamorphic encryption, revisited[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Cham: Springer Nature Switzerland, 2024: 3-32.
- [4] Catalano D, Giunta E, Migliaro F. Anamorphic encryption: New constructions and homomorphic realizations[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Cham: Springer Nature Switzerland, 2024: 33-62.
- [5] Carnemolla D, Catalano D, Giunta E, et al. Anamorphic resistant encryption: the good, the bad and the ugly[C]//Annual International Cryptology Conference. Cham: Springer Nature Switzerland, 2025: 472-503.
- [6] Dodis Y, Goldin E. Anamorphic-resistant encryption; or why the encryption debate is still alive[C]//Annual International Cryptology Conference. Cham: Springer Nature Switzerland, 2025: 440-471.
- [7] Wang S, Ning J, Li Q, et al. Anamorphic Encryption with CCA Security: A Standard Model Construction[J]. arXiv preprint arXiv:2604.07771, 2026.
- [8] Deo A, Libert B. Fully asymmetric anamorphic homomorphic encryption from LWE[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Cham: Springer Nature Switzerland, 2026: 609-639.]
- [9] Tanishita Y, Matsuda T, Matsuura K. Revisiting Security Definitions of Sender-Anamorphic Encryption[C]//IACR International Conference on Public-Key Cryptography. Cham: Springer Nature Switzerland, 2026: 130-161.
- [10] J. Bos, L. Ducas, E. Kiltz, et al., "CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, 2018, pp. 353–367.
- [11] Mothukuri V, Parizi R M, Pouriyeh S, et al. A survey on security and privacy of federated learning[J]. *Future Generation Computer Systems*, 2021, 115: 619-640.
- [12] McMahan B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data[C]//Artificial intelligence and statistics. Pmlr, 2017: 1273-1282.
- [13] Bonawitz K, Ivanov V, Kreuter B, et al. Practical secure aggregation for privacy-preserving machine learning[C]//proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017: 1175-1191.
- [14] Boudgoust K, Scholl P. Simple threshold (fully homomorphic) encryption from LWE with polynomial modulus[C]//international conference on the theory and application of cryptology and information security. Singapore: Springer Nature Singapore, 2023: 371-404.
- [15] Passelègue A, Stehlé D. Low communication threshold fully homomorphic encryption[C]//International conference on the theory and application of cryptology and information security. Singapore: Springer Nature Singapore, 2024: 297-329.
- [16] Shamir A. How to share a secret[J]. 1979.

APPENDIX A IMPLEMENTATION OF HASH FUNCTIONS

This appendix specifies the constructions of Hash_{e_1} and Hash_{e_2} used in TAKyber. Both hash functions deterministically generate the error terms e_1 and e_2 , respectively.

We give the constructions of $\text{Hash}_{e_1}(apk_i, dk, m, i, \eta)$ and $\text{Hash}_{e_2}(apk_i, dk, m_i, i, \eta, \text{tag})$.

Let

$$apk_i = (\mathbf{A}_i, \mathbf{b}_i), \quad dk = (\mathbf{A}, \mathbf{b})$$

where

$$\mathbf{A}_i \in R_q^{l \times l}, \mathbf{A} \in R_q^{L \times L}, \mathbf{b}_i \in R_q^l, \mathbf{b} \in R_q^L$$

For fixed apk_i, dk, m , and i , we first construct an aggregate polynomial by adding all polynomial components contained in $\mathbf{A}_i, \mathbf{b}_i, \mathbf{A}, \mathbf{b}$, and m . Specifically, define

$$Z_{e_1} = \sum_{a=1}^l \sum_{b=1}^l \mathbf{A}_i^{(a,b)} + \sum_{a=1}^l \mathbf{b}_i^{(a)} + \sum_{a=1}^L \sum_{b=1}^L \mathbf{A}^{(a,b)} + \sum_{a=1}^L \mathbf{b}^{(a)} + m$$

and

$$Z_{e_2} = \sum_{a=1}^l \sum_{b=1}^l \mathbf{A}_i^{(a,b)} + \sum_{a=1}^l \mathbf{b}_i^{(a)} + \sum_{a=1}^L \sum_{b=1}^L \mathbf{A}^{(a,b)} + \sum_{a=1}^L \mathbf{b}^{(a)} + m + \text{tag}$$

Since $Z_{e_1}, Z_{e_2} \in R$, they can be uniquely represented as

$$\begin{cases} Z_{e_1}(x) = \sum_{j=0}^{n-1} z_{e_1,j} x^j, & z_{e_1,j} \in \mathbb{Z} \\ Z_{e_2}(x) = \sum_{j=0}^{n-1} z_{e_2,j} x^j, & z_{e_2,j} \in \mathbb{Z} \end{cases}$$

For each Z_{e_1} and Z_{e_2} , the n coefficients are serialized in their canonical order. Specifically, define

$$S_{e_1} = \text{str}(z_{e_1,0}) \parallel \text{str}(z_{e_1,1}) \parallel \cdots \parallel \text{str}(z_{e_1,n-1}) \parallel i$$

and

$$S_{e_2} = \text{str}(z_{e_2,0}) \parallel \text{str}(z_{e_2,1}) \parallel \cdots \parallel \text{str}(z_{e_2,n-1}) \parallel i$$

where $\text{str}(\cdot)$ denotes the decimal representation of an integer coefficient and $\parallel$ denotes string concatenation. The coefficient ordering and representation are fixed and publicly specified.

The two serialized strings are processed by SHAKE-256 with output lengths of $2n\eta l$ bits and $2n\eta$ bits, respectively:

$$B_{e_1} = \text{SHAKE256}(S_{e_1}, 2n\eta l)$$

and

$$B_{e_2} = \text{SHAKE256}(S_{e_2}, 2n\eta)$$

Thus, $B_{e_1} \in \{0, 1\}^{2n\eta l}, B_{e_2} \in \{0, 1\}^{2n\eta}$.

The bit strings B_{e_1} and B_{e_2} are divided into multiple consecutive blocks of 2η bits as:

$$\begin{cases} B_{e_1} = B_{e_1,1} \| B_{e_1,2} \| \cdots \| B_{e_1,nl} \\ B_{e_2} = B_{e_2,1} \| B_{e_2,2} \| \cdots \| B_{e_2,nl} \end{cases}$$

where $B_{e_1,j} \in \{0,1\}^{2\eta}$, $B_{e_2,j} \in \{0,1\}^{2\eta}$.

For each $j \in \{1, \dots, nl\}$, let

$$B_{e_1,j} = (b_{e_1,j,1}, \dots, b_{e_1,j,\eta}, b_{e_1,j,\eta+1}, \dots, b_{e_1,j,2\eta})$$

The corresponding coefficient is generated according to the centered binomial distribution:

$$e_{e_1,j} = \sum_{h=1}^{\eta} b_{e_1,j,h} - \sum_{h=\eta+1}^{2\eta} b_{e_1,j,h}$$

The nl coefficients are sequentially arranged into l polynomials as:

$$e_{1,k}(x) = \sum_{j=0}^{n-1} e_{e_1,kn+j+1} x^j, \quad 1 \leq k \leq l$$

Thus,

$$\mathbf{e}_1 = (e_{1,1}, e_{1,2}, \dots, e_{1,l}) \in R_q^l$$

Similarly, for each $j \in \{1, \dots, n\}$, let

$$B_{e_2,j} = (c_{e_2,j,1}, \dots, c_{e_2,j,\eta}, c_{e_2,j,\eta+1}, \dots, c_{e_2,j,2\eta})$$

The corresponding coefficient is generated as

$$e_{2,j} = \sum_{h=1}^{\eta} c_{e_2,j,h} - \sum_{h=\eta+1}^{2\eta} c_{e_2,j,h}$$

The resulting coefficients form

$$e_2(x) = \sum_{j=0}^{n-1} e_{2,j+1} x^j \in R_q$$

Therefore, the two hash functions are instantiated as

$$\text{Hash}_{e_1}(apk_i, dk, m, i, \eta) = \mathbf{e}_1 \sim \mathcal{B}_\eta(R_q^l)$$

and

$$\text{Hash}_{e_2}(apk_i, dk, m, i, \eta, \text{tag}) = e_2 \sim \mathcal{B}_\eta(R_q)$$

All instances of Hash_{e_1} and Hash_{e_2} used in this paper follow the procedures specified in this appendix.

APPENDIX B

ANAMORPHIC TRANSMISSION ALGORITHMS

The anamorphic transmission algorithms **aSend** and **aReceive** in Figure 8 are application-specific transmission procedures designed for the covert federated learning framework. They are not part of the formal definitions of TAE or TAKyber, nor are they fundamental cryptographic algorithms of TAKyber. Instead, they are lightweight procedures built upon the covert communication capability of TAKyber to efficiently transmit global gradients.

Unlike sensitive messages, the global gradient does not need to remain confidential from legitimate participants in our setting. It only needs to remain covert from the dictator. Therefore, any legitimate participant holding the double key dk can recover the global gradient using **aReceive**, without performing the complete threshold recovery procedure of TAKyber. Specifically, **aSend** directly embeds the gradient

vector $\mathbf{m}$ into a TAKyber-style innocent-looking ciphertext. Given an innocent-looking message m_i , the sender deterministically generates $\mathbf{e}_{1,i}$ and $e_{2,i}$ using the hash functions specified in Appendix A, and computes

$$\mathbf{u} = [\mathbf{A}_i^T \mathbf{m} + \mathbf{e}_{1,i}]_q, \quad v = [\mathbf{b}_i^T \mathbf{m} + e_{2,i} + \Delta m_i]_q$$

The tag tag_{SEND} binds the ciphertext to the anamorphic transmission phase and prevents direct cross-phase substitution.

At the receiving side, **aReceive** first recovers the innocent-looking message m_i using the corresponding secret key and regenerates $\mathbf{e}_{1,i}$, $e_{2,i}$. It then recovers the covert message as

$$\mathbf{m} = [(\mathbf{A}_i^T)^{-1}(\mathbf{u} - \mathbf{e}_{1,i})]_q$$

and verifies the communication using $e_{2,i}$. If the verification fails, the algorithm outputs $\perp$; Otherwise, it outputs $\mathbf{m}$.

The use of **aSend** and **aReceive** substantially reduces the overhead of transmitting global gradients compared with directly applying TAKyber. Direct TAKyber transmission requires the gradient to be processed through the underlying KyberPKE encryption and then decomposed and sliced from the resulting high-dimensional R_Q -valued ciphertext. In contrast, **aSend** directly embeds the gradient vector and therefore avoids the encryption, decomposition, and threshold fragment-combination procedures required by full TAKyber. Moreover, a single transmission can carry a vector in $R_{2\delta+1}^l$, making it particularly suitable for the high-dimensional global gradients in federated learning. This optimization does not modify the core construction or formal security properties of TAKyber and is introduced solely for application-specific gradient transmission.

The correctness of **aSend** and **aReceive** follows directly from their consistent generation of the two ciphertext components. Under correct decryption of the underlying KyberPKE, the receiver obtains the same innocent-looking message m_i as the sender. Hence, both parties deterministically generate the same $\mathbf{e}_{1,i}$ and $e_{2,i}$. Consequently,

$$[\mathbf{u} - \mathbf{e}_{1,i}]_q = [\mathbf{A}_i^T \mathbf{m}]_q$$

Since $\mathbf{A}_i$ is invertible, we have

$$[(\mathbf{A}_i^T)^{-1}(\mathbf{u} - \mathbf{e}_{1,i})]_q = \mathbf{m}$$

aSend($i, dk, apk_i, m_i, \mathbf{m}, \text{PP}_{\text{TAKyber}}$) $\rightarrow \mathbf{c}_m$:

- 1) Parse $dk = (dpk, dsk)$, $apk_i = (\mathbf{A}_i, \mathbf{b}_i)$;
- 2) $\mathbf{e}_{1,i} \leftarrow \text{Hash}_{e_1}(apk_i, dk, m_i, i, \eta_2)$;
- 3) $e_{2,i} \leftarrow \text{Hash}_{e_2}(apk_i, dk, m_i, i, \eta_2, \text{tag}_{\text{SEND}})$;
- 4) $\mathbf{u} = [\mathbf{A}_i^T \mathbf{m} + \mathbf{e}_{1,i}]_q$;
- 5) $v = [\mathbf{b}_i^T \mathbf{m} + e_{2,i} + \Delta m_i]_q$;
- 6) **return** $\mathbf{c}_m = (\mathbf{u}, v)$;

aReceive($i, dk, ask_i, \mathbf{c}_m, \text{PP}_{\text{TAKyber}}$) $\rightarrow \mathbf{m} / \perp$:

- 1) Parse $ask_i = (\mathbf{A}_i, \mathbf{b}_i, \mathbf{s}_i)$, $\mathbf{c}_m = (\mathbf{u}, v)$;
- 2) $m_i \leftarrow \text{KyberPKE.Dec}(\mathbf{c}_m, ask_i, \text{PP}_{\text{Kyber}})$;
- 3) $\mathbf{e}_{1,i} \leftarrow \text{Hash}_{e_1}(apk_i, dk, m_i, i, \eta_2)$;
- 4) $e_{2,i} \leftarrow \text{Hash}_{e_2}(apk_i, dk, m_i, i, \eta_2, \text{tag}_{\text{SEND}})$;
- 5) $\mathbf{m} \leftarrow [(\mathbf{A}_i^T)^{-1}(\mathbf{u} - \mathbf{e}_{1,i})]_q$;
- 6) if $e_{2,i} \neq [v - \mathbf{b}_i^T \mathbf{m} - \Delta m_i]_q$: **return** $\perp$;
- 7) **return** $\mathbf{m}$;

Fig. 8. Implementations of **sSend** and **aReceive**.

Furthermore, there is

$$[v - \mathbf{b}_i^T \mathbf{m} - \Delta m_i]_q = e_{2,i}$$

So the verification succeeds for an honestly generated ciphertext. For a modified ciphertext, an inconsistent covert message, or an incorrect transmission tag, the regenerated $e_{2,i}$ fails to match the value implied by the ciphertext except with negligible probability. Thus, **aSend** and **aReceive** correctly recover the covert message and detect modified ciphertexts.

APPENDIX C

INVERTIBILITY OF RANDOM MATRICES OVER R_q

We next analyze the probability that a uniformly random matrix $\mathbf{A} \in R_q^{l \times l}$ is invertible. Assume that n is a power of 2 and $q \equiv 1 \pmod{2n}$. Let $\zeta \in \mathbb{F}_q$ be an element of order $2n$. Then

$$x^n + 1 = \prod_{i=0}^{n-1} (x - \zeta^{2i+1}) \pmod{q}$$

Since the roots are distinct in $\mathbb{F}_q$, the polynomial $x^n + 1$ splits completely into distinct linear factors over $\mathbb{F}_q$. Therefore, by the Chinese Remainder Theorem, we have

$$R_q \cong \prod_{i=0}^{n-1} \mathbb{F}_q$$

More explicitly, the isomorphism is induced by the evaluation map

$$f(x) \mapsto (f(\zeta), f(\zeta^3), \dots, f(\zeta^{2n-1}))$$

The above ring isomorphism naturally extends to matrices, yielding

$$R_q^{l \times l} \cong \prod_{i=0}^{n-1} \mathbb{F}_q^{l \times l}$$

Accordingly, any matrix $\mathbf{A} \in R_q^{l \times l}$ can be represented as

$$\mathbf{A} \longleftrightarrow (\mathbf{A}_0, \mathbf{A}_1, \dots, \mathbf{A}_{n-1}), \quad \mathbf{A}_i \in \mathbb{F}_q^{l \times l}$$

Since matrix multiplication is preserved componentwise under this isomorphism, $\mathbf{A}$ is invertible over R_q if and only if every component $\mathbf{A}_i$ is invertible over $\mathbb{F}_q$. Hence,

$$\mathbf{A} \in GL_l(R_q) \iff \mathbf{A}_i \in GL_l(\mathbb{F}_q), \quad \forall i \in \{0, \dots, n-1\}$$

Suppose that $\mathbf{A}$ is sampled uniformly from $R_q^{l \times l}$. Under the above isomorphism, the corresponding components $\mathbf{A}_0, \mathbf{A}_1, \dots, \mathbf{A}_{n-1}$ are independently and uniformly distributed over $\mathbb{F}_q^{l \times l}$. Therefore,

$$\Pr[\mathbf{A} \in GL_l(R_q)] = \prod_{i=0}^{n-1} \Pr[\mathbf{A}_i \in GL_l(\mathbb{F}_q)]$$

It remains to determine the probability that a uniformly random matrix in $\mathbb{F}_q^{l \times l}$ is invertible. The number of invertible $l \times l$ matrices over $\mathbb{F}_q$ is

$$|GL_l(\mathbb{F}_q)| = \prod_{k=0}^{l-1} (q^l - q^k)$$

Indeed, the first column can be any nonzero vector, giving $q^l - 1$ choices. After k linearly independent columns have been

selected, their span contains q^k vectors, so the next column has $q^l - q^k$ valid choices. Thus,

$$\Pr[\mathbf{A}_i \in GL_l(\mathbb{F}_q)] = \frac{|GL_l(\mathbb{F}_q)|}{q^{l^2}} = \prod_{k=0}^{l-1} \left(1 - \frac{1}{q^{l-k}}\right)$$

Equivalently,

$$\Pr[\mathbf{A}_i \in GL_l(\mathbb{F}_q)] = \prod_{k=1}^l \left(1 - \frac{1}{q^k}\right)$$

Combining the above results gives the exact probability

$$\Pr[\mathbf{A} \in GL_l(R_q)] = \left(\prod_{k=1}^l \left(1 - \frac{1}{q^k}\right) \right)^n$$

Consequently, the probability that $\mathbf{A}$ is non-invertible is

$$\Pr[\mathbf{A} \notin GL_l(R_q)] = 1 - \left(\prod_{k=1}^l \left(1 - \frac{1}{q^k}\right) \right)^n$$

For sufficiently large q , we have

$$\prod_{k=1}^l \left(1 - \frac{1}{q^k}\right) = 1 - \frac{1}{q} + O\left(\frac{1}{q^2}\right)$$

Therefore,

$$\Pr[\mathbf{A} \notin GL_l(R_q)] = \frac{n}{q} + O\left(\frac{n}{q^2} + \frac{n^2}{q^2}\right)$$

which tends to zero as q increases. Hence, a uniformly sampled matrix $\mathbf{A} \in R_q^{l \times l}$ is invertible with probability close to 1 for sufficiently large q .

In TAKyber, the key generation algorithm therefore employs rejection sampling to ensure that each public matrix $\mathbf{A}_i$ is invertible. Specifically, a matrix is repeatedly sampled from $R_q^{l \times l}$ until $\mathbf{A}_i \in GL_l(R_q)$. Let X denote the number of sampling trials required to obtain an invertible matrix. Since each trial is independent and succeeds with probability $\Pr[\mathbf{A} \in GL_l(R_q)]$, X follows a geometric distribution. Therefore, its mathematical expectation is

$$\mathbb{E}[X] = \frac{1}{\Pr[\mathbf{A} \in GL_l(R_q)]} = \left(\prod_{k=1}^l \left(1 - \frac{1}{q^k}\right) \right)^{-n}$$

Since the invertibility probability is close to one for sufficiently large q , the expected number of sampling trials is also close to one. Thus, the rejection sampling introduces only a small expected overhead in the key generation procedure.

Using the parameters in Table I, where $n = 256, l = 3$, the invertibility probability and the expected number of sampling trials for $q = 7681$ and $q = 12289$ are respectively:

$$\begin{cases} \Pr[\mathbf{A} \in GL_3(R_{7681})] \approx 0.967214, & \mathbb{E}[X] \approx 1.033897 \\ \Pr[\mathbf{A} \in GL_3(R_{12289})] \approx 0.979381, & \mathbb{E}[X] \approx 1.021053 \end{cases}$$

Hence, in both parameter settings, an invertible matrix is obtained after only slightly more than one sampling trial on average.