Article

Not peer-reviewed version

Lattice-Based Multi-Key Homomorphic Encryption Scheme Without CRS

Hongyi Zhang , Mengxue Shang , Hanzhuo Liu * , Dandan Zhang

Posted Date: 3 April 2025

doi: 10.20944/preprints202504.0302.v1

Keywords: multi-key homomorphic encryption; lattice; CRS model;

 doi:10.20944/preprints202504.0302.v1

Article

Lattice-Based Multi-Key Homomorphic Encryption Scheme Without CRS

Hongyi Zhang [1], Mengxue Shang [2], Hanzhuo Liu [1,*] and Dandan Zhang [2]

[1] Mineral Resources Information Center, Metallurgical Geology Bureau, Beijing, 100025, China

[2] School of Computer Science, Qufu Normal University, Rizhao, China

* Correspondence: liuhanzhuo@cmgb.cn

Abstract: Multi-key homomorphic encryption is widely applied into outsourced computing and privacy-preserving applications in multi-user scenarios. However, the existence of CRS weakens the ability of users to independently generate public keys, and it is difficult to implement in decentralized systems or scenarios with low trust requirements. In order to reduce excessive reliance on public parameters, a multi-key homomorphic encryption scheme without pre-setting CRS is proposed based on a distributed key generation protocol. The proposed scheme does not require the pre-generation and distribution of CRS, which enhances the security and decentralization of the scheme. Furthermore, in order to further protect the plaintext privacy from each user, by embedding the specified target user into the ciphertext, this paper proposes an enhanced multi-key homomorphic encryption scheme that only allows only the target user to decrypt. Finally, this paper applies the proposed lattice-based multi-key homomorphic encryption scheme into the data submission stage of the perceived users, and thereby proposes a crowd-sensing scheme with privacy preservation.

Keywords: multi-key homomorphic encryption; lattice; CRS model; crowd intelligence perception

1. Introduction

With the continuous advancement of technologies such as the Internet, the Internet of Things, big data, and artificial intelligence, the demand for computing power and storage resources by enterprises and individuals has increased exponentially. Traditional local computing and storage methods can no longer meet the needs of modern society for data processing speed and capacity [1]. Outsourcing computing allows users to entrust complex computing tasks or data processing work to a third party (such as a cloud service provider) to perform computing tasks through a cloud platform or distributed computing resources, saving users a lot of time and computing costs[2].

Although outsourced computing provides flexibility and efficiency, it is also accompanied by some potential risks. In outsourced computing, users usually need to upload data to cloud service providers for processing. These outsourced data may contain some sensitive user information, such as personal privacy, commercial secrets or key business data[3–5]. Homomorphic encryption technology allows specific operations (such as addition or multiplication) to be performed directly on encrypted data without decrypting the data. The decrypted result is consistent with the result of performing the same operation on the plaintext[6]. User data remains encrypted during the calculation process, ensuring the privacy of the data throughout the calculation process, and users do not need to trust the cloud service provider.

Multi-key homomorphic encryption solves the above problem by allowing each user to encrypt data with his or her own key, while still supporting joint computing of encrypted data. The computing results can be obtained by collaborative decryption of the private keys of multiple users[7–9].

In a multi-key homomorphic encryption scheme, in order to enable multiple users to jointly perform homomorphic operations on ciphertext (for example, perform homomorphic operations

 doi:10.20944/preprints202504.0302.v1

such as addition and multiplication) and collaboratively decrypt the ciphertext after homomorphic operations, a mechanism is needed to coordinate and combine the public keys of different users[10]. The Common Random String (CRS) provides a shared public parameter based on which all users can generate their own public keys. Through the public parameters provided by the CRS, multiple public keys with the same parameters are integrated into an aggregate public key. Multiple users can perform homomorphic operations on ciphertext under the aggregate public key and collaboratively decrypt through a distributed decryption protocol.

CRS can simplify the scheme design and key generation process, but it also brings some problems: On the one hand, the existence of CRS means that the system relies on a public, predefined random string. This assumption may affect the independence and flexibility of the encryption scheme, which is difficult to meet in decentralized systems or scenarios with low trust requirements. On the other hand, the security and correctness of the scheme directly depend on the integrity and reliability of CRS. Dependence on CRS seriously affects the credibility of the scheme and even causes security vulnerabilities[12].

In order to avoid excessive reliance on public parameters, a multi-key homomorphic encryption scheme without pre-setting CRS is proposed based on a distributed key generation protocol. Furthermore, based on ciphertext expansion technology, a distributed ciphertext decryption method is proposed. In order to further protect the plaintext messages of each user, this paper proposes an enhanced multi-key homomorphic encryption scheme that only allows the target user to decrypt, by embedding the specified target user into the ciphertext.

Finally, by applying the proposed the lattice-based multi-key homomorphic encryption scheme into crowd sensing scenario, a crowd sensing scheme is proposed to protect the privacy of crowd sensing data.

The contributions of this paper are as follows:

1. In order to avoid excessive reliance on public parameters, this paper proposes a multi-key homomorphic encryption scheme based on a distributed key generation protocol. Each user independently generates his or her own public and private key pair, and enhances the security and decentralization of the scheme. Based on ciphertext expansion technology, this paper proposes a distributed ciphertext decryption method suitable for multi-key scenarios. By expanding the ciphertext structure, multiple users can collaboratively participate in the decryption process.
2. In order to further protect the plaintext privacy from each user, by embedding the specified target user into the ciphertext, this paper proposes an enhanced multi-key homomorphic encryption scheme that only allows only the target user to decrypt.
3. By applying the proposed lattice-based multi-key homomorphic encryption scheme into the data submission stage, a crowd-sensing scheme is proposed, protecting the privacy of the users. This ensures that the data is not leaked during transmission and processing, and all entities except the data requester cannot obtain the perception results.

2. Materials and Methods

2.1. Symbols and Definitions

In this paper, λ is used to denote the security parameter, and the dot product of two vectors u and v is denoted by $< u, v >$. Let Ω denote a finite field and X be a probability distribution defined on Ω, then $\omega \leftarrow X$ denotes that an element ω is randomly selected from the distribution X[13]. $R_q = Z_q[X]/\Phi_M(X)$ denotes a cyclotomic polynomial ring, where $Z_q[X]$ is a ring of polynomials whose coefficients are taken from Z_q, and $\Phi_M(X) = X^{M/2} + 1$ denotes a cyclotomic polynomial of order M[14].

Definition 1. *B-bounded distribution. Let D be a random distribution. If any x sampled from D satisfies $Pr_{x \leftarrow D}[\|x\| > B] = negl(\lambda)$, then D is called a B-bounded distribution*[15].

 doi:10.20944/preprints202504.0302.v1

Definition 2. *RLWE Problem. RLWE is a generalization of the LWE problem, which extends the vector operations in LWE to polynomial ring*[16]. *Given a polynomial ring* $R = Z[X]/f(x)$, *where* $f(x)$ *is an irreducible polynomial, define a ring* $R_q = R/qR = Z_q[X]/f(x)$ *modulo* q. *Select a secret vector* $s \in R_q$, *and give a RLWE sample pair* $(a, b) \in R_q \times R_q$, *where* $b = a \cdot s + e \ (mod\ q)$, *and* e *is a random noise sampled from the noise distribution* χ[17]. *Depending on the goal, the RLWE problem is divided into two types: search RLWE problem and decision RLWE problem*[18].

Definition 3. *Decision RLWE Problem. The goal of the decision RLWE problem is to distinguish between two distributions: distribution 1 is an RLWE distribution, where the sample pair* $(a_i, b_i) \in R_q \times R_q$ *satisfies* $b_i = a_i \cdot s + e_i \ (mod\ q)$, *where* $s \in R_q$ *is the secret vector and* $e \in \chi$ *is the random error term; distribution 2 is a uniform distribution, where* a_i *and* b_i *in the sample pair* $(a_i, b_i) \in R_q \times R_q$ *are independently and uniformly randomly sampled from* R_q[8]. *The RLWE assumption means that there is no effective polynomial algorithm that can distinguish between these two distributions, that is, for a probabilistic polynomial time algorithm* $\mathcal{B}$ *and security parameter* λ, *we have*

$$Adv(\mathcal{B}) := \left|Pr\left[\mathcal{B}^{A_{s,\chi}}(1^\lambda) = 1\right] - Pr\left[\mathcal{B}^{R_q \times R_q}(1^\lambda) = 1\right]\right| = negl(\lambda) \quad (1)$$

2.2. Multi-Key Homomorphic Encryption

Multi-key homomorphic encryption allows users to encrypt their data using their own public keys and perform homomorphic operations on the ciphertext. At the same time, the calculation results are decrypted by all users collaboratively, which is more suitable for multi-user collaborative scenarios[19,20]. A multi-key homomorphic encryption scheme usually consists of six polynomial time algorithms, namely $MFHE.Setup$, $MFHE.Keygen$, $MFHE.Enc$, $MFHE.Expand$, $MFHE.Eval$, and $MFHE.Dec$. The specific descriptions are as follows:

- $MFHE.Setup(1^\lambda)$: Input security parameter λ and output public parameter $params$.
- $MFHE.KeyGen(params)$: Input public parameters $params$ and output the user's public key and private key (pk, sk).
- $MFHE.Enc(pk, m)$: For the plaintext m that needs to be encrypted, input the public key pk and output a ciphertext ct.
- $MFHE.Expand((pk_1, \cdots, pk_N), i, ct_i)$: Input the public keys of N users $pk_1, \cdots pk_N$ and the ciphertext ct_i encrypted by the i-th public key pk_i, and output the expanded ciphertext $\widehat{ct_i}$.
- $MFHE.Eval(params, f, (\widehat{ct_1}, \cdots, \widehat{ct_l}))$: Given a function f, input l extended ciphertexts $\widehat{ct_1}, \cdots, \widehat{ct_l}$, and output the ciphertext $\hat{ct}$ after homomorphic operation.
- $MFHE.Dec(params, (sk_1, \cdots, sk_N), \hat{ct})$: Input the private keys of N users $sk_1, \cdots, sk_N$ and the homomorphic operation ciphertext $\hat{ct}$, and output the plaintext m. The decryption process is divided into two steps, as follows:
  - $MFHE.PartDec(i, sk_i, \hat{ct})$: Input the private key sk_i of the i-th user and the homomorphic operation ciphertext $\hat{ct}$, and output the partial decryption result p_i.
  - $MFHE.FinDec(p_1, \cdots, p_N)$: Input the partial decryption results $p_1, \cdots, p_N$ of N users and output the plaintext m.

3. Lattice-Based Multi-Key Homomorphic Encryption Scheme Without CRS

In order to reduce the dependence on public parameters and enhance the ability of users to independently generate public keys, this section proposes a lattice-based multi-key homomorphic encryption scheme without CRS. Through a distributed key generation protocol, all users independently generate their keys. Based on the ciphertext expansion technology, a distributed ciphertext decryption method in a multi-key scenario is proposed, thereby realizing cross-user homomorphic addition operations without public parameters. In order to further protect the plaintext messages of each user, this section embeds the target user's information in the ciphertext,

 doi:10.20944/preprints202504.0302.v1

so that the encryption process supports the designated target user as the only decryptor, providing more flexible privacy preservation.

3.1. Securrity Model

IND-CPA security requirement: For any probabilistic polynomial time adversary $\mathcal{A}$, its advantage under the "chosen plaintext attack" is negligible. IND-CPA security is defined by the interactive game $Game_{\mathcal{A}}$ between challenger $\mathcal{C}$ and adversary $\mathcal{A}$. The specific steps are as follows.

1. Initialization phase: Input security parameter λ, $\mathcal{C}$ runs $params \leftarrow Setup(1^{\lambda}, 1^{L})$ algorithm to generate system public parameter $params$. $\mathcal{C}$ runs $((pk_i, sk_i), (pk_T, sk_T)) \leftarrow KeyGen(params)$ algorithm to generate key pairs $\{(pk_i, sk_i)\}_{i=1}^{N}$ for N users and key pair (pk_T, sk_T) for target user T, and sends $\{pk_i\}_{i=1}^{N}$, pk_T to $\mathcal{A}$.
2. Query phase: $\mathcal{C}$ maintains a query record table Q, which is empty at initialization and records all ciphertext query indexes initiated by $\mathcal{A}$ during the entire query process. $\mathcal{A}$ can adaptively select any plaintext m_i and initiate a query request. $\mathcal{C}$ runs the $\{c_i \leftarrow Enc(pk_i, pk_T, m_i)\}_{i \in S}$ algorithm to generate the ciphertext c_i and returns it to $\mathcal{A}$. This phase allows $\mathcal{A}$ to perform a polynomial number of queries.
3. Challenge phase: After $\mathcal{A}$ finishes the query, it requests the challenge ciphertext. $\mathcal{A}$ selects two plaintexts m_0, m_1 of equal length and the target public key set $S^* \in \{1,2,\cdots,k\}$, and sends them to $\mathcal{C}$. $\mathcal{C}$ randomly selects a bit $b \leftarrow \{0,1\}$, calculates the challenge ciphertext $c^* = Enc(\{pk_i\}_{i \in S^*}, pk_T, m_b)$, and returns c^* to $\mathcal{A}$.
4. Guessing stage: $\mathcal{A}$ outputs a guess bit $b^* \in \{0,1\}$ based on c^*. If $b^* = b$, $\mathcal{A}$ wins and the game output is 1; otherwise, the output is 0.

If and only if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $negl(\lambda)$ such that:

$$\left| Pr[Game_{\mathcal{A}} = 1] - \frac{1}{2} \right| \leq negl(\lambda) \tag{2}$$

Where λ is a security parameter, the multi-key homomorphic encryption scheme without CRS is IND-CPA secure, that is, it satisfies semantic security.

3.2. Scheme Construction

The lattice-based multi-key homomorphic encryption scheme without CRS includes nine algorithms, namely: $Setup$ algorithm, $KeyGen$ algorithm, $Encode$ algorithm, Enc algorithm, $Expand$ algorithm, $AddEval$ algorithm, $PartDec$ algorithm, $FinDec$ algorithm, and $Decode$ algorithm. The specific description of each algorithm is as follows.

1. System Initialization $Setup(1^{\lambda}, 1^{L})$

Step 1. Let the security parameter be λ, the circuit depth be L, and the number of users be N. Let the dimension of the polynomial ring $R_q = \mathbb{Z}_q[X]/(X^K + 1)$ be K, and the ciphertext modulus be q. Let $\chi = \chi(\lambda)$ be the key distribution on R_q, and $\psi = \psi(\lambda)$ be the error distribution on R.

Step 2. Returns the system common parameters $params = (K, q, \chi, \psi)$.

2. Key generation algorithm $KeyGen(params)$

Step 1. U_i selects $s_i \leftarrow \chi$ and sets its private key to $sk_i = (1, s_i)$. U_i randomly samples $e_i \leftarrow \psi$, $a_i \leftarrow R_q$, calculates $b_i = -a_i \cdot s_i + e_i \ mod \ q \in R_q$, and sets its public key to $pk_i = (b_i, a_i)$.

Step 2. U_T selects $s_T \leftarrow \chi$ and sets its private key to $sk_T = (1, s_T)$. U_T randomly samples $e_T \leftarrow \psi$, $a_T \leftarrow R_q$, calculates $b_T = -a_T \cdot s_T + e_T \ mod \ q \in R_q$, and sets its public key to $pk_T = (b_T, a_T)$.

3. Coding $Encode(z_i, \Delta)$

Step 1. The message of user U_i is a complex vector $z_i = (z_{i,1}, z_{i,2}, \cdots, z_{i,K/2})$, where K is the dimension of the polynomial ring. The complex vector z_i is scaled to retain decimal precision, and $z_i = \Delta \cdot z_i$ is calculated, where Δ is the scaling factor.

Step 2. The complex vector z_i is mapped to the polynomial ring $R = \mathbb{Z}[X]/(X^N + 1)$ through τ mapping, that is, $m = \lfloor \tau^{-1}(\lfloor z_i \rceil) \rceil$.

 doi:10.20944/preprints202504.0302.v1

Step 3. Output integer coefficient plaintext polynomial m_i.

4. Encryption algorithm $Enc(pk_i, pk_T, m_i)$

Step 1. U_i randomly samples $v_i \leftarrow \chi$, $e_0^i, e_1^i, e_2^i \leftarrow \psi$, and sets $a_i = a_i[1]$, $a_T = a_T[1]$, $b_i = b_i[1]$, $b_T = b_T[1]$.

Step 2. U_i uses public keys pk_i and pk_T to encrypt its plaintext m_i and do the following calculation.

$$c_0^i = v_i \cdot (b_i + b_T) + m_i + e_0^i (mod\ q) \tag{3}$$

$$c_1^i = v_i \cdot a_i + e_1^i (mod\ q) \tag{4}$$

$$c_2^i = v_i \cdot a_T + e_2^i (mod\ q) \tag{5}$$

The output ciphertext is $c_i = (c_0^i, c_1^i, c_2^i) \in R_q^3$.

5. Ciphertext expansion algorithm $Expand(c_i, i)$

Step 1. User U_i expands its ciphertext $c_i \in R_q^3$ to a higher dimension and outputs the expanded ciphertext $\hat{c}_i = (c_0^i, \underbrace{0, \cdots}_{i-1}, c_1^i, 0, \cdots, c_2^i) \in R_q^{N+2}$.

Step 2. U_i sends its extended ciphertext $\hat{c}_i$ to CSP for homomorphic operation.

6. Homomorphic operation algorithm $AddEval(\hat{c}_1, \hat{c}_2, \cdots \hat{c}_N)$

Step 1. After CSP collects the extended ciphertexts $\hat{c}_1, \hat{c}_2, \cdots \hat{c}_N$ of all users $U_i\{i = 1,2, \cdots, N\}$, it performs homomorphic computation as follows. $C_{sum_0} = \sum_{i=1}^N c_0^i$, $C_{sum_1} = (c_1^1, \cdots, c_1^N)$, $C_{sum_2} = \sum_{i=1}^N c_2^i$, and outputs the aggregated ciphertext $C_{sum} = (C_{sum_0}, c_1^1, \cdots, c_1^N, C_{sum_2})$.

Step 2. CSP sends the aggregate ciphertext C_{sum} to the target user U_T for decryption.

7. Partial decryption algorithm $PartDec(i, sk_i, c_i)$

Step 1. User U_i uses his private key sk_i to partially decrypt his ciphertext c_i and calculates his decryption share $p_i = s_i \cdot c_1^i + e_i^* (mod\ q)$, where $e_i^* \leftarrow \psi$.

Step 2. U_i sends its decrypted share p_i to U_T for final decryption.

8. Final decryption algorithm $FinDec(C_{sum}, p_1, p_2, \cdots, p_N)$

Step 1. After receiving the aggregate ciphertext C_{sum} and the decryption shares $p_1, p_2, \cdots, p_N$, U_T uses its own private key sk_T to perform the final decryption, calculate and output the aggregate plaintext value as follows.

$$m^* = C_{sum_0} + \sum_{i=1}^N p_i + s_T \cdot C_{sum_2} (mod\ q) \tag{6}$$

9. Decoding $Decode(m^*)$

Step 1. Use mapping τ to map m^* and calculate $m = \lfloor \tau(m^*) \rceil$.

Step 2. Perform an inverse scaling operation on m to restore the accuracy of the original data, that is, $m = \lfloor \Delta^{-1}(m) \rceil$, where Δ is the scaling factor used during encoding.

Step 3. Output the aggregate plaintext value m in the form of a complex vector.

3.3. Correctness Analysis

Given security parameter λ and circuit depth L, set modulus $q = 2^{\lambda L \cdot \omega(\log \lambda + \log L)}$, $B = \omega(\lambda L)$, and ψ is a B-bounded distribution on R. Given the ciphertext $c_{sum} = \left(\sum_{i=1}^N c_0^i, c_1^1, c_1^2, \cdots, c_1^N\right)$ under N user public keys and the private keys $sk = (1, s_1, s_2, \cdots, s_N)$ of N user connections, we have

$$\begin{aligned} \langle sk, c_{sum} \rangle &= (1, s_1, s_2, \cdots, s_N) \cdot \left(\sum_{i=1}^N c_0^i, c_1^1, c_1^2, \cdots, c_1^N\right) \\ &= \sum_{i=1}^N c_0^i + \sum_{i=1}^N s_i \cdot c_1^i\ (mod\ q) \\ &= \sum_{i=1}^N m_i + e' (mod\ q) \end{aligned}$$

 doi:10.20944/preprints202504.0302.v1

Among them, $e' = \sum_{i=1}^{N}(v_i \cdot e_i + e_0^i + s_i \cdot e_1^i)$, and $\|e'\|_\infty \le 2^{L \cdot O(\log \lambda + \log L)}$. Therefore, given the plaintext aggregation value m_{sum} and the corresponding aggregation ciphertext C_{sum}, according to the definition of $PartDec$ algorithm and $FinDec$ algorithm, we can calculate

$$\begin{aligned}
&C_{sum_0} + \sum_{i=1}^{N} p_i + s_T \cdot C_{sum_2} (mod\ q) \\
&= \sum_{i=1}^{N} c_0^i + \sum_{i=1}^{N} \left(s_i c_1^i + e_i^*\right) + s_T \sum_{i=1}^{N} c_2^i \ (mod\ q) \\
&= \sum_{i=1}^{N} \left(v_i(b_i + b_T) + m_i + e_0^i\right) + \sum_{i=1}^{N} \left(s_i\left(v_i a_i + e_1^i\right) + e_i^*\right) \\
&\qquad + s_T \sum_{i=1}^{N} \left(v_i \cdot a_T + e_2^i\right) (mod\ q) \\
&= \sum_{i=1}^{N} \left(v_i b_i + m_i + e_0^i + s_i\left(v_i \cdot a_i + e_1^i\right) + e_i^*\right) \\
&\qquad + \sum_{i=1}^{N} \left(v_i b_T + s_T\left(v_i a_T + e_2^i\right)\right) (mod\ q) \\
&= \sum_{i=1}^{N} \left(v_i \cdot e_i + m_i + e_0^i + s_i \cdot e_1^i + e_i^*\right) \\
&\qquad + \sum_{i=1}^{N} \left(v_i \cdot e_T + s_T \cdot e_2^i\right) (mod\ q) \\
&= m_{sum} + e' + e''(mod\ q)
\end{aligned}$$

where $e'' = \sum_{i=1}^{N}\left(e_i^* + v_i \cdot e_T + s_T \cdot e_2^i\right)$ and $\|e''\|_\infty \le 2^{\lambda L \cdot O(\log \lambda + \log L)}$. Therefore, if $\|e' + e''\|_\infty < q/4$, the lattice-based multi-key homomorphic encryption scheme without CRS can be correctly decrypted.

3.4. Security Analysis

Theorem 3.1. *Assuming that the RLWE problem is difficult, if there is no adversary $\mathcal{A}$ that can win the following security game $Game_{\mathcal{A}}$ with non-negligible probability, then the lattice-based multi-key homomorphic encryption scheme without CRS is IND-CPA secure, that is, it satisfies semantic security.*

Proof of Theorem 3.1. Given an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, the theorem is proved by defining the following game sequence.

Game 0. Given public parameters $params = (K, q, \chi, \psi)$ and vector $a_i \leftarrow R_q$, challenger $\mathcal{C}$ runs $KeyGen(params)$ algorithm to generate public key $pk_i = (b_i, a_i)$, and sends pk_i to adversary $\mathcal{A}$, where $b_i = -a_i \cdot s_i + e_i \ mod\ q$. The distribution of pk at this stage is the same as that of MFHE scheme.

Game 1. Except for the key generation phase, the steps of other phases are the same as Game 0. The distribution of public keys is redefined in Game 1. Given public parameters $params = (n, q, \chi, \psi)$ and vector $a_i \leftarrow R_q$, generate public key $pk_i' = (b_i', a_i)$, where $b_i' \leftarrow R_q$. According to the difficulty and cyclic security assumed by RLWE, the computational difference between $pk_{Game\ 0}$ and $pk_{Game\ 1}$ cannot be distinguished, so b_i and b_i' are also computationally indistinguishable, so the advantage of the attacker distinguishing Game 0 from Game 1 can be ignored.

$$Adv(\mathcal{A}) = \left|Pr_{Game\ 0}\left[\mathcal{A}\left(1^\lambda, pk_i\right) = 1\right] - Pr_{Game\ 1}\left[\mathcal{A}\left(1^\lambda, pk_i'\right) = 1\right]\right| = negl(\lambda) \quad (7)$$

Within a certain period of time, $\mathcal{A}$ challenges $\mathcal{C}$ and sends the challenge plaintext $\mu_1, \mu_2 \in \{0,1\}$. $\mathcal{C}$ randomly selects $k \in \{0,1\}$, runs the $Enc(pk_i, pk_T, m_i)$ algorithm to output the challenge ciphertext c_i, and then sends the ciphertext c_i to $\mathcal{A}$. $\mathcal{A}$ outputs the guess result of the scheme and outputs $k' \in \{0,1\}$. If $k' = k$, output 1, otherwise output 0. Protection Since the probability of $\mathcal{A}$ distinguishing b_i and b_i' can be ignored, the multi-key homomorphic encryption scheme without CRS proposed in this paper is IND-CPA secure, that is, it satisfies semantic security. □

4. Crowd-sensing Scheme with privacy preservation

 doi:10.20944/preprints202504.0302.v1

Crowd sensing refers to a mode in which a large number of sensing devices (usually personal smartphones, wearable devices, sensors, etc.) distributed in different geographical locations work together to collect, process and share information[22]. This mode usually involves multiple participants collaborating to complete a task without central control, especially in the fields of environmental monitoring, urban management, intelligent transportation, etc.[23].

In a crowd sensing system, the task issued by the data requester requires multiple sensing users to upload sensing data to the sensing platform, and the platform aggregates and calculates these data to obtain the sensing results. However, the data uploaded by users may contain personal sensitive information, such as location information, health data, etc. On the other hand, the sensing platform cannot be fully trusted, that is, users are worried that the platform may abuse or leak the sensing result data. Multi-key homomorphic encryption allows the data of multiple users to be calculated in an encrypted state, which can achieve secure data calculation under the premise of protecting user privacy data. In order to solve the data privacy problem of sensing users, this section applies the lattice-based multi-key homomorphic encryption scheme without CRS to the data submission stage of sensing users, thereby designing a crowd sensing scheme with privacy protection. Specifically, users encrypt data before uploading it, and the perception platform only aggregates multiple data ciphertexts. The perception results are obtained by decryption by the data requester, ensuring that the data is not leaked during transmission and processing. At the same time, no other entity except the data requester can obtain the perception results.

4.1. System Model

This section proposes a crowd sensing scheme based on multi-key homomorphic encryption. The entities involved in this scheme are sensing users, sensing platforms, and data receivers.

1. Sensing users

Sensing users are data providers in the crowd sensing system, responsible for collecting data using their own devices (such as smartphones, wearable devices, environmental sensors, etc.). For example, smartphone users can provide data such as location, acceleration, and temperature; health monitoring device users can provide physiological data such as steps, heart rate, and sleep quality. Their data usually contains personal privacy information, so the data needs to be encrypted before uploading to the sensing platform.

2. Sensing platform

Sensing platform is an intermediary platform between sensing users and data requesters in the crowd sensing system, responsible for receiving encrypted data from multiple sensing users and performing homomorphic operations, and feeding the results back to the data requester.

3. Data requester

Data requester is the subject that uses the crowd sensing results, usually a government department, enterprise, or individual. According to their own needs (such as traffic management, environmental monitoring, health management, etc.), they publish data request tasks, receive aggregated ciphertext from the perception platform and decrypt it, and then analyze the perception data to make decisions, provide services or optimize operations.

The crowd intelligence perception scheme based on multi-key homomorphic encryption proposed in this section is divided into four stages: initialization, perception data submission, ciphertext aggregation, and perception result decryption. Figure 1 shows the four stages of the scheme and the interaction process between perception users, perception platforms, and data requesters.

 doi:10.20944/preprints202504.0302.v1

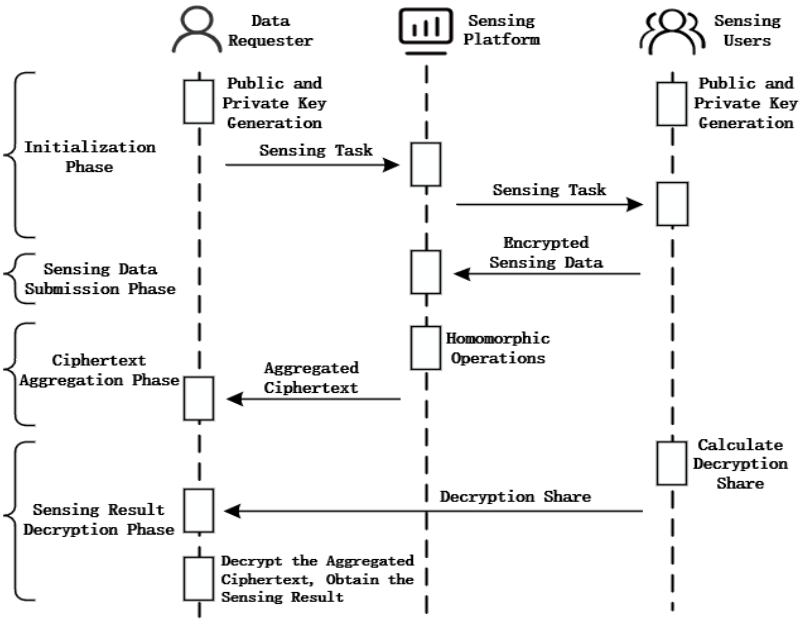

Figure 1. Flowchart of crowd-sensing scheme based on multi-key homomorphic encryption.

The crowd-sensing solution based on multi-key homomorphic encryption proposed in this section contains five core functional modules, namely task management module, data collection module, encryption module, ciphertext aggregation module, and access control module. The introduction of each functional module is as follows.

Task management module is responsible for allocating and coordinating user tasks to ensure the effectiveness of data collection. In crowd-sensing, different tasks need to be assigned to different perception users, and tasks need to be dynamically allocated, taking into full consideration factors such as user location and device capabilities.

Data collection module is responsible for the collection of environmental information or data by crowd-sensing terminals (such as smartphones and IoT devices). Data may need to be pre-processed such as denoising, format conversion, and data compression to reduce communication overhead and computing burden.

Encryption module is responsible for encrypting the collected data to ensure privacy preservation during data transmission and calculation. The perception user encrypts the data with his own public key and then submits the ciphertext.

Ciphertext aggregation module is the perception platform performs homomorphic calculations on the ciphertexts of multiple perception users without decrypting the data.

Access control module allows the perception user embeds the public key information of the data requester in the ciphertext to ensure that only the data requester has the right to decrypt the aggregated ciphertext and thus access the perception results to ensure privacy protection and security.

4.2. Construction of Crowd-sensing Scheme Based on Multi-key Homomorphic Encryption

4.2.1. Initialization Phase

Define data requester D and L perception users $U_1, \cdots, U_L$. Perception user U_i runs $(pk_i, sk_i) \leftarrow KeyGen(params)$ algorithm to generate its key pair (pk_i, sk_i), and data requester D runs $(pk_D, sk_D) \leftarrow KeyGen(params)$ algorithm to generate its key pair (pk_D, sk_D).

Data requesters publish perception tasks to the perception platform according to their needs. The perception platform is responsible for organizing appropriate users to collect and upload data

 doi:10.20944/preprints202504.0302.v1

according to the tasks. The perception platform selects perception users $U_1, \cdots, U_N$ that meet the task requirements and sends task invitations to the selected perception users U_i. Users can choose to accept or reject.

4.2.2. Perception Data Submission Phase

The perceptual user U_i who receives the task collects data through its perceptual device and runs the $m_i \leftarrow Encode(i, z_i, \Delta)$ algorithm to encode the collected data into m_i. U_i uses its own public key pk_i and the public key pk_D of the data requester D to encrypt m_i, and runs the $c_i \leftarrow Enc(pk_i, pk_D, m_i)$ algorithm to obtain the ciphertext $c_i = (c_0^i, c_1^i, c_2^i) = (v_i \cdot (pk_i + pk_D) + m_i + e_0^i, v_i \cdot a_i + e_1^i, v_i \cdot a_D + e_2^i)(mod\ q)$, where $v_i \leftarrow \chi$, $e_0^i, e_1^i, e_2^i \leftarrow \psi$, $a_i = a_i[0]$, $a_D = a_D[0]$, $pk_i = pk_i[0]$, $pk_D = pk_D[0]$.

The perceptual user U_i sends the ciphertext c_i to the perceptual platform. The platform can only store and homomorphically compute encrypted data and cannot view the user's plaintext data m_i.

4.2.3. Ciphertext Aggregation Phase

The perception platform receives the ciphertext $c_1, \cdots, c_N$ from the perception users $U_1, \cdots, U_N$ and performs homomorphic computation without decryption. The perception platform runs the $C_{sum} \leftarrow AddEval(c_1, c_2, \cdots, c_N)$ algorithm to calculate the aggregated ciphertext $C_{sum} = \sum_{i=1}^{N} c_i = (C_{sum_0}, C_{sum_1}, C_{sum_2}) = (\sum_{i=1}^{N} c_0^i, \sum_{i=1}^{N} c_1^i, \sum_{i=1}^{N} c_2^i)$. The aggregated ciphertext is still encrypted, and the perception platform cannot decrypt it to obtain the perception result. The perception platform sends C_{sum} to the data requester D for result decryption.

4.2.4. Perception Result Decryption Phase

The decryption phase of the perception result is divided into two steps: partial decryption and final decryption. In the partial decryption step, the perception user U_i runs the $p_i \leftarrow PartDec(i, c_i, sk_i)$ algorithm to calculate the decryption share $p_i = s_i \cdot c_1^i + e_i^*$, where $e_i^* \leftarrow \psi$, and then sends p_i to the data requester D. In the final decryption step, after receiving the decryption shares $p_1, \cdots, p_N$ of all perception users, the data requester D uses its own private key sk_D to decrypt the aggregated ciphertext, runs the $m^* \leftarrow FinDec(C_{sum}, p_1, p_2, \cdots, p_N)$ algorithm for final decryption, and obtains the perception resultm^*. The data m obtained after decoding m^* is the perception result required by the data requester D. The perception result is the aggregated perception data, not the data of a single perception user, to ensure user privacy.

5. Security Analysis of Crowd-sensing Scheme Based on Multi-key Homomorphic Encryption

In the crowd sensing scheme based on multi-key homomorphic encryption, the entire sensing process is completed through information transmission between three entities: the sensing user, the sensing platform, and the data requester. Therefore, the security of the scheme will be discussed from two aspects: the sensing user and the sensing platform.

Theorem 5.1. *In the crowd sensing scheme based on multi-key homomorphic encryption, no entity can obtain the plaintext data of a single sensing user, that is, the privacy data of the sensing user is safe.*

Proof of Theorem 5.1. In the crowd sensing scheme based on multi-key homomorphic encryption, the sensing user does not need to trust the sensing platform or other users, and generates its key independently according to the distributed key generation protocol, and the data is encrypted locally. The plaintext data m_i of the sensing user U_i is encrypted locally into the ciphertext $c_i = Enc(pk_i, pk_D, m_i)$, and c_i is uploaded to the sensing platform through the network. U_i's data remains encrypted during transmission. According to Theorem 3.1, the ciphertext $c_i \leftarrow Enc(pk_i, pk_D, m_i)$ is

 doi:10.20944/preprints202504.0302.v1

computationally indistinguishable from the uniform distribution on R_q. The security of IND-CPA based on the RLWE problem ensures that the ciphertext c_i cannot be cracked, that is, the plaintext data m_i cannot be recovered from c_i. Therefore, even if an attacker or data requester intercepts the ciphertext of the perceived user U_i, no information related to m_i can be inferred from it. □

Theorem 5.2. *In the crowd-sensing scheme based on multi-key homomorphic encryption, no entity other than the data requester can decrypt the aggregated ciphertext to obtain the perception result, that is, the perception result is secure.*

Proof of *Theorem 5.2.* In the crowd-sensing scheme based on multi-key homomorphic encryption, the perception platform only stores and homomorphically calculates the ciphertext $c_1, c_2, \cdots, c_N$, but does not hold any user's private key sk_i, so it is impossible to decrypt the ciphertext of a single user. The result C_{sum} after homomorphic calculation is still encrypted, and the perception platform cannot deduce the plaintext data through calculation. The decryption of the aggregated ciphertext C_{sum} requires the decryption shares p_i of all users and the private key sk_D of the data requester. Only the data requester can decrypt and obtain the perception result. Even if the perception platform obtains the aggregated ciphertext C_{sum} and the decryption shares $p_1, p_2, \cdots, p_N$, its calculation

$$
\begin{aligned}
C_{sum_0} + \sum_{i=1}^{N} p_i \,(mod\ q) &= \sum_{i=1}^{N} c_0^i + \sum_{i=1}^{N} \left(s_i \cdot c_1^i + e_i^*\right) (mod\ q) \\
&= \sum_{i=1}^{N} \left(v_i \cdot (pk_i + pk_D) + m_i + e_0^i\right) \\
&\qquad + \sum_{i=1}^{N} \left(s_i \cdot \left(v_i \cdot a_i + e_1^i\right) + e_i^*\right) (mod\ q) \\
&\approx \sum_{i=1}^{N} m_i + \sum_{i=1}^{N} v_i \cdot pk_D \,(mod\ q)
\end{aligned}
$$

In addition to the aggregated plaintext $\sum_{i=1}^{N} m_i$, the calculation result also contains the partial ciphertext $\sum_{i=1}^{N} v_i \cdot pk_D \,(mod\ q)$ encrypted by pk_D. Since the perception platform does not have the private key $\sum_{i=1}^{N} v_i \cdot pk_D \,(mod\ q)$ of the data requester, it cannot eliminate $\sum_{i=1}^{N} v_i \cdot pk_D \,(mod\ q)$ in the calculation result, so it is impossible to obtain the aggregated plaintext data through calculation. □

6. Conclusion

In multi-user scenarios, CRS, as a centralized public information, not only provides a basis for collaboration for participating users, but also simplifies the process of key generation and management, so that the encrypted data of multiple users can be effectively operated in the same computing environment. However, the existence of CRS weakens the ability of users to independently generate public keys, and it is difficult to achieve in decentralized systems or scenarios with low trust requirements. This section proposes a lattice-based multi-key homomorphic encryption scheme without CRS, aiming to eliminate the dependence on public parameters and improve the system's anti-attack capability. The proposed scheme not only solves the problems of privacy preservation and data security, but also maintains high efficiency and scalability in large-scale distributed systems. Multi-key full homomorphic encryption scheme will be our research direction in future for more wide application.

Acknowledgement: This study is sponsored by the Science and Technology Innovation Program of China Metallurgical Geology Bureau (Grant no. CMGBKY202407).

References

1. Liu, L.; Zhang J; Song, S.H. Client-edge-cloud hierarchical federated learning. ICC 2020-2020 IEEE International Conference on Communications (ICC).

 doi:10.20944/preprints202504.0302.v1

2. Kim, M.; Harmanci, A.O.; Bossuat, J.P. Ultrafast homomorphic encryption models enable secure outsourcing of genotype imputation. Cell systems, 2021, 12(11): 1108-1120. e4.
3. Kim, H.I.; Kim, H.J.; Chang, J.W.; A secure kNN query processing algorithm using homomorphic encryption on outsourced database. Data & knowledge engineering, 2019, 123: 101602.
4. Yang, Y.; Huang, X.; Liu, X. A comprehensive survey on secure outsourced computation and its applications. IEEE Access, 2019, 7: 159426-159465.
5. Sun, J.; Xu, G.; Zhang, T. Verifiable, fair and privacy-preserving broadcast authorization for flexible data sharing in clouds. IEEE Transactions on Information Forensics and Security, 2022, 18: 683-698.
6. Kadykov, V.; Levina, A.; Voznesensky, A. Homomorphic encryption within lattice-based encryption system. Procedia Computer Science, 2021, 186: 309-315.
7. Chen, H.; Dai, W.; Kim, M. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. 2019: 395-412.
8. Xu, K.; Tan, B.H.M.; Wang, L.P. Multi-key fully homomorphic encryption from NTRU and (R) LWE with faster bootstrapping. Theoretical Computer Science, 2023, 968: 114026.
9. Biswas, C.; Dutta, R. Secure and efficient multi-key FHE scheme supporting multi-bit messages from LWE preserving non-interactive decryption. Journal of Ambient Intelligence and Humanized Computing, 2023, 14(12): 16451-16464.
10. Zhou, T.; Chen, L.; Che, X. Multi-key Fully Homomorphic Encryption Scheme with Compact Ciphertexts. Cryptology ePrint Archive, 2021.
11. Li, H.; Li, X.; Gao, J. Multi-hop Multi-key Homomorphic Encryption with Less Noise Under CRS Model. International Symposium on Cyberspace Safety and Security. Cham: Springer International Publishing, 2022: 342-357.
12. Luo. F.; Wang, H.; Saif, A.K. Multi-key fully homomorphic encryption without CRS from RLWE. Computer Standards & Interfaces, 2023, 86: 103742.
13. Ma, J.; Naas, S.A.; Sigg, S. Privacy-preserving federated learning based on multi-key homomorphic encryption. International Journal of Intelligent Systems, 2022, 37(9): 5880-5901.
14. Ganesh, B.; Palmieri, P. Secure Search over Multi-key Homomorphically Encrypted Data. 2023 7th International Conference on Cryptography, Security and Privacy (CSP). IEEE, 2023: 145-151.
15. Li, X.; Li, H.; Gao, J. Privacy preserving via multi-key homomorphic encryption in cloud computing. Journal of Information Security and Applications, 2023, 74: 103463.
16. Chen, Y.; Dong, S.; Li, T. Dynamic multi-key FHE in asymmetric key setting from LWE. IEEE Transactions on Information Forensics and Security, 2021, 16: 5239-5249.
17. Antwi-Boasiako, E.; Zhou, S.; Liao, Y. An LWE-Based Multi-Key Privacy-Preserving Distributed Deep Learning. 2021 IEEE 23rd Int Conf on High Performance Computing & Communications, 2021: 533-542.
18. Che, X.; Zhou, H.; Yang, X. Efficient multi-key homomorphic encryption scheme on ring LWE. Journal of Xidian University, 2023, 48(1): 87-95.
19. Li, N.; Zhou, T.; Che, X. Research on multi-key homomorphic encryption. Journal of Cryptologic Reshearch, 2020, 7(6): 713-734.
20. Pathak, V. Lattices, homomorphic encryption, and ckks. arXiv preprint arXiv:2205.03511, 2022.
21. Qiu, F.; Yang, H.; Zhou, L. Privacy preserving federated learning using ckks homomorphic encryption. International Conference on Wireless Algorithms, Systems, and Applications. Cham: Springer Nature Switzerland, 2022: 427-440.
22. Li, J.; Zhu, Y.; Hua, Y. Crowdsourcing sensing to smartphones: A randomized auction approach. IEEE Transactions on Mobile Computing, 2017, 16(10): 2764-2777.
23. Zheng, X.; Cui, L.; Zhang, L. The perception results based on encryption technology can verify the privacy preservation group intelligence perception scheme. Journal of Beijing Institute of Technology(Nature Edition), 2024, 44(4): 413-420.

 doi:10.20944/preprints202504.0302.v1

disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.