Privacy-Preserving and Outsourced Multi-Party K-Means Clustering Based on Multi-Key Fully Homomorphic Encryption

Peng Zhang, Teng Huang, Xiaoqiang Sun, Wei Zhao, Hongwei Liu, Shangqi Lai, and Joseph K. Liu

Abstract—The clustering algorithm is a useful tool for analyzing medical data. For instance, the k-means clustering can be used to study precipitating factors of a disease. In order to implement the clustering algorithm efficiently, data computation is outsourced to cloud servers, which may leak the private data. Encryption is a common method for solving this problem. But cloud servers are difficult to calculate ciphertexts from multiple parties. Hence, we choose multi-key fully homomorphic encryption (FHE), which supports computations on the ciphertexts that have different secret keys, to protect the private data. In this paper, based on Chen's multi-key FHE scheme, we first propose secure squared euclidean, comparison, minimum, and average protocols. Then, we design the basic and advanced schemes for implementing the secure multi-party k-means clustering algorithm. In the basic scheme, the implementation of homomorphic multiplication includes the process of transforming ciphertexts under different keys. In order to implement homomorphic multiplication efficiently, the advanced scheme uses an improved method to transform ciphertexts. Meanwhile, almost all computations are completely outsourced to cloud servers. We prove that the proposed protocols and schemes are secure and feasible. Simulation results also show that our improved method is helpful for improving the homomorphic multiplication of Chen's multi-key FHE scheme.

Index Terms—K-means, multi-key fully homomorphic encryption, multi-party, outsourced computing, privacy preserving

1 Introduction

Medical data usually include the patients' full names, date of births, addresses, medical history, medication records, etc. The analysis of medical data is helpful for improving medical diagnostics, drug discovery, clinical trials, and patient outcomes. The clustering algorithm is an efficient analysis method, which can identify underlying relationships and rules from these data. As a typical clustering algorithm, k-means is used to study precipitating factors of a disease. Because the patients lack of powerful computation capability, medical data are usually outsourced to cloud servers for the efficient implementation of the k-means clustering algorithm. However, cloud servers may illegally leak, falsify, or delete medical data.

For the security of medical data, differential privacy [1] and homomorphic encryption [2] are introduced. Differential privacy is a popular privacy model, which can guarantee that the removal or addition of any individual item in a data set does not significantly affect the analysis result on the data set. In addition, the security of differential privacy is independent on the attacker's background knowledge and computing ability. Homomorphic encryption is an encryption algorithm that allows computation on the encrypted data without decryption. This algorithm makes it possible that the k-means clustering algorithm can be implemented by the untrusted cloud server without leaking medical data. In addition, the communication cost between the patient and cloud server is low.

However, differential privacy does not support computations on the ciphertexts. Homomorphic encryption can only deal with the ciphertexts, which are decrypted by the same secret key. In the realistic scenario of medical diagnosis, there usually exist multiple source-restrained patients. In addition, each patient has his own health data, which include different biomedical indicators that are represented in rational numbers. Each patient's private data cannot be leaked to other untrusted third parties. Hence, differential privacy or homomorphic encryption is not suitable for this scenario.

Multi-key fully homomorphic encryption (FHE) [3] supports arbitrary computations on the ciphertexts, whose secret keys are different from each other. Based on the usage of

- *Peng Zhang and Wei Zhao are with the Guangdong Key Laboratory of Intelligent Information Processing, College of Electronics, Information Engineering, Shenzhen University, Shenzhen 518060, China. E-mail: zhangp@szu.edu.cnins, zhaowei@email.szu.edu.cn.*
- *Teng Huang is with the Institute of Artificial Intelligence, Blockchain, Guangzhou University, Guangzhou 510006, China, and also with Beihang University, Beijing 100191, China. E-mail: huangteng1220@buaa.edu.cn.*
- *Xiaoqiang Sun is with the School of Computer Science, Shenzhen Institute of Information Technology, Shenzhen 518172, China. E-mail: xqsun@sziit.edu.cn.*
- *Hongwei Liu is with the College of Big Data, Internet, Shenzhen Technology University, Shenzhen 518118, China. E-mail: liuhongwei@sztu.edu.cn.*
- *Shangqi Lai and Joseph K. Liu are with the Faculty of Information Technology, Monash University, Clayton, VIC 3800, Australia. E-mail: shangqi.lai@monash.eduins, joseph.liu@monash.edu.*

Manuscript received 7 Jan. 2021; revised 1 Mar. 2022; accepted 14 May 2022. Date of publication 10 June 2022; date of current version 13 May 2023. This work was supported in part by the National Natural Science Foundation of China under Grants 61702342 and 62002074, and in part by the Science and Technology Program Project of Shenzhen under Grant SZWD2021012. (Corresponding author: Teng Huang.) Digital Object Identifier no. 10.1109/TDSC.2022.3181667

multi-key FHE, the patients' medical data is encrypted separately. Then, generated ciphertexts are transmitted to the cloud server. If there exists a need of medical diagnosis, the cloud server implements the k-means clustering algorithm on the ciphertexts, and outputs a medical diagnosis result.

In this paper, we endeavor to study the security of breast cancer data in the above realistic scenario and focus on the secure implementation of the k-means clustering algorithm. Taking into account the privacy and computation of breast cancer data on the untrusted cloud servers, we adopt multi-key FHE as the main encryption primitive to carry out our research. Eventually, we make following three contributions:

1) In order to implement the privacy-preserving multi-party k-means clustering, based on Chen's multi-key FHE scheme [4], we design four secure unit protocols, namely, secure squared euclidean distance protocol, secure comparison protocol, secure minimum protocol, and secure average protocol.
2) Based on the proposed protocols, we design a basic multi-party k-means clustering scheme, so that k-means clustering can be implemented over the medical data from multi-party patients securely. In this scheme, the patients are required to keep online in the whole process of implementing this scheme.
3) In order to outsource the whole computation process of k-means clustering to cloud servers completely, we propose an advanced multi-party k-means clustering scheme by using an improved method to transform ciphertexts that have different secret keys. This scheme has the advantage that the patients can be off-line.

The rest of this paper is organized as follows. Section 2 introduces related work about the privacy-preserving k-means clustering scheme. Section 3 introduces preliminaries. Our protocols are shown in Section 4. Then, Section 5 provides the details of our secure k-means clustering schemes. Section 6 discusses the security and efficiency of our schemes. Finally, we conclude our work in Section 7.

2 Related Work

2.1 Privacy-Preserving K-Means Clustering

In this subsection, we describe existing privacy-preserving k-means clustering algorithms, which are mostly constructed based on the differential privacy or homomorphic encryption technique.

Based on the differential privacy technique, there exist some privacy-preserving k-means clustering schemes. In 2013, Li *et al.* [5] proposed a privacy-preserving k-means clustering scheme with an improved method for the initial center point. In 2016, Li *et al.* [6] proposed a more efficient scheme under the MapReduce framework. However, these schemes did not consider data features. In 2017, based on the differential privacy technique, Ren *et al.* [7] proposed a novel k-means clustering mechanism, which improves the selection of initial center points. But it will result in large deviation by adding the same noise to different clusters in the same iteration. In order to address this problem, Zhang *et al.* [8] proposed a privacy-preserving k-means clustering scheme by using contour coefficients.

Based on the homomorphic encryption technique, there exist some privacy-preserving k-means clustering schemes. In 2007, Bunn and Ostrovsky [9] proposed a privacy-preserving two-party k-means clustering scheme. However, if the number of involved party is more than two, this scheme is no longer secure. In 2014, Liu *et al.* [10] proposed an outsourced k-means clustering scheme, which is constructed based on the proposed fully homomorphic encryption (FHE) algorithm. In this scheme, the honest-but-curious third party can arrange ciphertexts in order by using dynamic trapdoors. But the data owner needs to recalculate the trapdoors in each iteration of the k-means clustering algorithm. In 2015, Rao *et al.* [11] proposed a novel scheme based on the Pailliar cryptosystem. In 2017, based on Brakerski's scheme [12], Theodouli *et al.* [13] proposed a framework for implementing the k-means clustering algorithm securely. In 2018, Kim and Chang [14] constructed an efficient secure comparison protocol. Then, based on this protocol and Paillier cryptosystem [15], the authors proposed a privacy-preserving k-means clustering scheme. Based on Paillier cryptosystem, Xing *et al.* [16] proposed the first mutual privacy-preserving and collusion-resistant k-means clustering scheme. In addition, this scheme does not incur much computation and communication costs on the participants. In 2019, based on the Brakerski's cryptosystem [17], Sakellariou and Gounaris [18] proposed a secure k-means clustering scheme with low client-side load. In this scheme, the trusted server implements secure distance comparison operations. Based on fully homomorphic encryption, Wu *et al.* [19] proposed a secure and efficient outsourced k-means clustering scheme. Based on homomorphic encryption, Fan *et al.* [20] provided a privacy-preserving multi-party k-means clustering scheme. But, this scheme needs extra interactions.

2.2 Multi-Key FHE

In this subsection, we describe existing multi-key FHE schemes, which are shown as follows.

In 2012, based on NTRU cryptosystem [21], Lopez-Alt *et al.* [3] constructed a multi-key homomorphic encryption scheme without fully proved security. Then, Clear and McGoldrick [22] proposed a new multi-key FHE scheme by using Gentry's cryptosystem [23]. Moreover, the authors constructed a 3-round secure multiparty computation protocol under the common random string (CRS) model. In 2016, based on Gentry's cryptosystem [23] and the threshold decryption protocol [24], Mukherjee and Wichs [25] presented a 2-round secure multiparty computation protocol under the CRS model.

Both Clear's scheme [22] and Mukherjee' scheme [25] are single-hop. They need to identify all parties from the beginning. In addition, any new party is not allowed to participate in the computation process. Hence, Peikert and Shiehian [26] proposed the first multi-hop multi-key FHE scheme that any new party can join the computation process dynamically. Brakerski and Perlman [27] proposed a similar notion, which is called fully dynamic.

Above multi-key FHE schemes [22], [25], [26], [27] are constructed based on the same Gentry's cryptosystem [23]. Although homomorphic multiplication of these schemes does not result in an expansion of ciphertext size, these

TABLE 1
Symbols and Explanations

Symbols	Explanations
$n \in \mathbb{Z}$	Number of users
$d \in \mathbb{Z}$	Dimension of the plaintext
$k \in \mathbb{Z}$	Number of clusters
$a \in \mathbb{Z}$	Polynomial dimension of the ring R
χ	Noise distribution
$p \in \mathbb{Z}$	Plaintext space modulus
$q \in \mathbb{Z}$	Ciphertext space modulus
$\beta_q = \lfloor \log_2 q \rfloor + 1$	Bit length of q
$\mu \in R_q$	Ring element
pk	Public key
sk	Secret key
$\mathbf{m}$	Plaintext
$\boldsymbol{\mu}$	Encoded plaintext
$\mathbf{C}(x)$	Ciphertext whose plaintext is the ring element x
$\underline{\mathbf{C}}$	Ciphertext
$\overline{\mathbf{C}}$	Expanded ciphertext

schemes only support a single bit plaintext. In 2017, based on Brakerski's scheme [28], Chen *et al.* [4] proposed a multi-key FHE scheme, which, for the first time, supports the single instruction multiple data technique. Besides, this scheme's extension of ciphertext size is simpler than that of above schemes.

3 Preliminaries

3.1 Notations

In this paper, we write vectors in bold lowercase letters, for example, vectors $\mathbf{x}$ and $\mathbf{y}$. $\mathbf{x}[i]$ denotes the ith entry of $\mathbf{x}$. We write matrix in bold uppercase letters, for example, matrix $\mathbf{M}$. $\mathbf{M}[i,:]$ denotes the ith row of $\mathbf{M}$. $\mathbf{M}[i,j]$ denotes the entry in the ith row and jth column. $|$ denotes the cascade of vectors, for example, $\mathbf{x}|\mathbf{y}$ denotes the cascade of $\mathbf{x}$ and $\mathbf{y}$. $\otimes$ denotes the tensor product, for example, $\mathbf{x} \otimes \mathbf{y}$ denotes the tensor product of $\mathbf{x}$ and $\mathbf{y}$. All the other symbols, which will be used in the rest of this paper, are summarized in Table 1.

3.2 K-Means Clustering Algorithm

In this subsection, we describe the implementation details of the k-means clustering algorithm (Algorithm 1), which are described as follows. In this algorithm, input parameters include the users' data $\mathbf{y}_0, \ldots, \mathbf{y}_{n-1}$, k, and the maximum number of iterations ω. The number of iterations s is initialized as 0. The cluster center $\mathbf{o}_j$ is selected from $\mathbf{y}_0, \ldots, \mathbf{y}_{n-1}$ randomly, where $j = 0, \ldots, k-1$. Next, the user P_i computes the euclidean distance $ed(\mathbf{y}_i, \mathbf{o}_j)$ between $\mathbf{y}_i$ and $\mathbf{o}_j$ at each iteration, where $ed(\mathbf{y}_i, \mathbf{o}_j) = \sqrt{\sum_{z=0}^{d-1} (\mathbf{y}_i[z] - \mathbf{o}_j[z])^2}$, $i = 0, \ldots, n-1$, $j = 0, \ldots, k-1$. P_i compares above euclidean distances. It finds $\mathbf{o}_j$ that has the minimum euclidean distance with $\mathbf{y}_i$. Then, $\mathbf{y}_i$ is assigned to the jth cluster. At the end of each iteration, cluster centers are updated by computing the average of all the data in each cluster. For example, there is a cluster that includes m different data $\mathbf{y}_0, \ldots, \mathbf{y}_{m-1}$. The average of all the data in this cluster is defined as $aver(K_1) = \frac{1}{m}\sum_{i=1}^{m} \mathbf{y}_i$. If s exceeds ω, the execution of this algorithm is ended. Finally, this algorithm outputs k cluster centers.

3.3 Chen's Multi-Key FHE Scheme

In this subsection, we briefly introduce Chen's multi-key FHE scheme $CFHE = (Setup,\ KeyGen,\ Enc, Dec, CExt,\ EvkGen)$, which is described as follows.

- $CFHE.Setup(\lambda, n, L)$: Input the security parameter λ, the number of users n, and the circuit depth L, this algorithm outputs the public parameters pp.
- $CFHE.KeyGen(pp, i \in [0, n))$: Input pp and the index i, this algorithm outputs the ith party's public keys $\{pk_{l,i}\}_{l=0,\ldots,L-1}$, secret keys $\{sk_{l,i}\}_{l=0,\ldots,L-1}$, and evaluation key generation materials $em_i = \{em_{l,i}\}_{l=0,\ldots,L-1}$.
- $CFHE.Enc(pp, pk_{l,i}, \mu)$: Input pp, $pk_{l,i}$, and a message $\mu \in R_p$, this algorithm outputs a ciphertext $\mathbf{C}$, where p denotes a small constant integer.
- $CFHE.Dec(pp, \mathbf{C}, sk_{l,i})$: Input pp, a ciphertext $\mathbf{C}$, and $sk_{l,i}$, this algorithm computes $\mu = (\langle \mathbf{C}, sk_{l,i} \rangle \bmod q_l) \bmod p$, where q_l denotes the modulus.
- $CFHE.CExt(pp, \mathbf{C})$: Input a ciphertext $\mathbf{C}$, this algorithm outputs a new ciphertext $\overline{\mathbf{C}}$, which satisfies that $\langle \overline{\mathbf{C}}, sk_{l,\{0,\ldots,n-1\}} \rangle = \langle \mathbf{C}, sk_{l,i} \rangle$, where $sk_{l,\{0,\ldots,n-1\}} = sk_{l,0}| \cdots |sk_{l,n-1}$..
- $CFHE.EvkGen(pp, em_i, pk_{l,i})$: Input pp, em_i, and $pk_{l,i}$, this algorithm outputs the evaluation key $\tau_{sk_{l,i} \to sk_{l-1,i}}$, which can convert $\mathbf{C}$ to a new ciphertext $\mathbf{C}'$ that has the secret key $sk_{l-1,i}$, where $l \geq 1$.

Algorithm 1. K-Means Clustering Algorithm

Input: $\mathbf{y}_0, \ldots, \mathbf{y}_{n-1}$, k, and ω.
Output: k cluster centers.
Choose k initial cluster centers $\mathbf{o}_0, \ldots, \mathbf{o}_{k-1}$ randomly;
Set $s = 0$;
Repeat
  For $i = 0$ to $n-1$ **do**
    For $j = 0$ to $k-1$ **do**
      Calculate the euclidean distance between $\mathbf{y}_i$ and $\mathbf{o}_j$;
    End for
    Compare euclidean distances;
    Find $\mathbf{o}_j$ that has the minimum euclidean distance with $\mathbf{y}_i$;
    Assign $\mathbf{y}_i$ to the jth cluster;
  End for
  For $j = 0$ to $k-1$ **do**
    Calculate the average of all data which belong to the jth cluster;
    Let the average be the jth cluster center for the next iterations;
  End for
  Set $s = s + 1$.
Until $s \geqslant \omega$

In addition, this scheme's homomorphic addition and homomorphic multiplication are described as follows.

We suppose that there are two ciphertexts $\mathbf{C}_0$ and $\mathbf{C}_1$, where the plaintexts of $\mathbf{C}_0$ and $\mathbf{C}_1$ are μ_0 and μ_1, and the secret keys of $\mathbf{C}_0$ and $\mathbf{C}_1$ are $sk_{l,0}$ and $sk_{l,1}$, respectively.

Then, they are extended to the ciphertexts $\overline{\mathbf{C}}_0$ and $\overline{\mathbf{C}}_1$, which share the same secret key $sk_{l,\{0,1\}}$, where $sk_{l,\{0,1\}}$ is $(sk_{l,0}|sk_{l,1})$. Based on the usage of the evaluation key $\tau_{sk'_{l,\{0,1\}}\rightarrow sk_{l,\{0,1\}}}$, the ciphertext $\overline{\mathbf{C}}_{mul} = \overline{\mathbf{C}}_0 \times \overline{\mathbf{C}}_1$, whose secret key is $sk'_{l,\{0,1\}} = sk_{l,0} \otimes sk_{l,1}$, is transformed to a ciphertext $\widetilde{\mathbf{C}}_{mul}$ that has the secret key $sk_{l,\{0,1\}}$. The ciphertext of homomorphic addition is $\overline{\mathbf{C}}_{add} = \overline{\mathbf{C}}_0 + \overline{\mathbf{C}}_1$, which decryption is

$$\begin{aligned} CFHE.Dec(\overline{\mathbf{C}}_{add}, sk_{l,\{0,1\}}) &= CFHE.Dec(\mathbf{C}_0, sk_{l,0}) + \\ &= CFHE.Dec(\mathbf{C}_1, sk_{l,1}) \end{aligned}$$

The ciphertext of homomorphic multiplication is $\widetilde{\mathbf{C}}_{mul}$, which decryption is

$$\begin{aligned} CFHE.Dec(\widetilde{\mathbf{C}}_{mul}, sk_{l,\{0,1\}}) &= CFHE.Dec(\mathbf{C}_0, sk_{l,0}) \\ &\times CFHE.Dec(\mathbf{C}_1, sk_{l,1}) \end{aligned}$$

We can use the modulus switching technique [28] to reduce the noise of $\widetilde{\mathbf{C}}_{mul}$. In addition, the modulus of $\widetilde{\mathbf{C}}_{mul}$ is reduced from q_l to q_{l-1}.

More details about Chen's multi-key FHE scheme can be found in [4].

3.4 Vector Decomposition

Vector decomposition is used to convert vectors into their unique bit representations. It consists of two algorithms which are defined as follows.

- $BitDecomp(\mathbf{x}, q)$: Input a vector $\mathbf{x}$ and a modulus q, this algorithm outputs the vector $(\mathbf{w}_0, \ldots, \mathbf{w}_{\beta_q-1}) \in \{0,1\}^{n\cdot\beta_q}$, where $\mathbf{w}_i \in \{0,1\}^n$, $\mathbf{x} = \sum_{i=0}^{\beta_q-1} 2^i \mathbf{w}_i$.
- $Powerof2(\mathbf{y})$: Input a vector $\mathbf{y}$, this algorithm outputs the vector $(\mathbf{y}, 2\mathbf{y}, \ldots, 2^{\beta_q-1}\mathbf{y}) \in \mathbb{Z}_q^{n\beta_q}$.

According to the Claim 3.4 in [17], for all $q \in \mathbb{Z}$ and $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$, it holds that $\langle \mathbf{x}, \mathbf{y} \rangle = \langle BitDecomp(\mathbf{x}), Powerof2(\mathbf{y}) \rangle \bmod q$.

3.5 Key Switching

As a noise management technique, key switching is introduced in [28] first. It consists of two subroutines, which are defined as follows.

- $RBGV.SwitchKeyGen(pp, sk_1, pk_2)$: Input a secret key sk_1 and a public key pk_2 that is associated with another secret key sk_2, this algorithm outputs the evaluation key $\tau_{sk_1\rightarrow sk_2} = \mathbf{A} + Powerof2(sk_1)$, where $\mathbf{A}$ is a matrix that satisfies $\mathbf{A} \cdot sk_2 = 2\mathbf{e}$, $\mathbf{e} \in \chi$.
- $RBGV.SwitchKey(\tau_{sk_1\rightarrow sk_2}, \mathbf{C}_1)$: Input $\tau_{sk_1\rightarrow sk_2}$ and a ciphertext $\mathbf{C}_1$, this algorithm outputs a refreshed ciphertext $\mathbf{C}_2 = \langle BitDecomp(\mathbf{C}_1), \tau_{sk_1\rightarrow sk_2} \rangle$, where $\mathbf{C}_1$'s secret key is sk_1, and $\mathbf{C}_2$'s secret key is sk_2.

It can be proved that $\mathbf{C}_1$ and $\mathbf{C}_2$ have the same plaintext. More details about key switching can be found in [28].

3.6 Data Encoding

Data encoding technique is used to convert the rational data into its ring element representation. In this paper, we adopt Dowlin's data encoding technique [29], which is described as follows. Input parameters include a rational number b and a polynomial ring $R = \mathbb{Z}[X]/(x^a + 1)$, where a denotes the dimension of ring R. b is first decomposed as its unique decimal presentation $b_{a_1} \cdots b_0 . b_{-1} \cdots b_{-a_2}$, where $b_{a_1} \cdots b_0$ represents the integer part of b, $b_{-1} \cdots b_{-a_2}$ denotes the fractional part of b, and $a = a_1 + a_2$. Then, b is encoded as a ring element μ by computing $\mu = \sum_{i=0}^{a_1} b_i x^i - \sum_{i=0}^{a_2} b_{-i} x^{a-i} \in R$. More details about this data encoding technique can be found in [29].

4 Unit Protocols

As described before, FHE supports homomorphic addition and homomorphic multiplication over the ciphertexts, but the privacy-preserving k-means clustering scheme needs the euclidean distance computation, comparison computation, minimum computation, and average computation over the ciphertexts. In order to fill the research gap, we propose four secure computation protocols, namely secure squared euclidean distance protocol, secure comparison protocol, secure minimum protocol, and secure average protocol, which are described as follows.

4.1 Secure Squared Euclidean Distance (SSED) Protocol

The euclidean distance is defined as $ed(\mathbf{y}_i, \mathbf{o}_j) = \sqrt{\sum_{z=0}^{d-1} (\mathbf{y}_i[z] - \mathbf{o}_j[z])^2}$, where $\mathbf{y}_i$ is a data point, $\mathbf{o}_j$ is a cluster center, $i = 0, \ldots, n-1$, and $j = 0, \ldots, k-1$. In order to avoid the square root operation, we design a secure squared euclidean distance protocol (Protocol 1). There are two cases.

Protocol 1. Secure Squared Euclidean Distance (SSED) Protocol

Input: $\mathbf{C}_i, \mathbf{C}_j$.
Output: $ed^2(\mathbf{C}_i, \mathbf{C}_j)$.
For the cloud server
If $pk_{l,i} \neq pk_{l,j}$
  Extend $\mathbf{C}_i, \mathbf{C}_j$ to $\overline{\mathbf{C}}_i, \overline{\mathbf{C}}_j$;
  Compute $ed^2(\overline{\mathbf{C}}_i, \overline{\mathbf{C}}_j) = \sum_{z=0}^{d-1} (\overline{\mathbf{C}}_{i,z} - \overline{\mathbf{C}}_{j,z}) \times (\overline{\mathbf{C}}_{i,z} - \overline{\mathbf{C}}_{j,z})$;
  Transform $ed^2(\overline{\mathbf{C}}_i, \overline{\mathbf{C}}_j)$ to $\widetilde{ed}^2(\overline{\mathbf{C}}_i, \overline{\mathbf{C}}_j)$;
Else
  Compute $ed^2(\mathbf{C}_i, \mathbf{C}_j) = \sum_{z=0}^{d-1} (\mathbf{C}_{i,z} - \mathbf{C}_{j,z}) \times (\mathbf{C}_{i,z} - \mathbf{C}_{j,z})$.

Case 1: $\mathbf{C}_i$ and $\mathbf{C}_j$ have the secret keys $sk_{l,i}$ and $sk_{l,j}$ respectively, where $i \neq j$, $\mathbf{C}_i = (\mathbf{C}_{i,0}, \ldots, \mathbf{C}_{i,d-1})$ denotes the encrypted $\mathbf{y}_i$, $\mathbf{C}_j = (\mathbf{C}_{j,0}, \ldots, \mathbf{C}_{j,d-1})$ denotes the encrypted cluster center $\mathbf{o}_j$, and d is the dimension. We suppose $\mathbf{C}_i$ and $\mathbf{C}_j$ are stored in the cloud server. $\mathbf{C}_i$ and $\mathbf{C}_j$ are extended to $\overline{\mathbf{C}}_i = (\overline{\mathbf{C}}_{i,0}, \ldots, \overline{\mathbf{C}}_{i,d-1})$ and $\overline{\mathbf{C}}_j = (\overline{\mathbf{C}}_{j,0}, \ldots, \overline{\mathbf{C}}_{j,d-1})$ respectively, namely $\overline{\mathbf{C}}_i \leftarrow CFHE.CExt(pp, \mathbf{C}_i)$, $\overline{\mathbf{C}}_j \leftarrow CFHE.CExt(pp, \mathbf{C}_j)$. $\overline{\mathbf{C}}_i$ and $\overline{\mathbf{C}}_j$ have the same secret key $sk_{l,\{i,j\}} = sk_{l,i}|sk_{l,j}$. Then, the cloud server calculates the encrypted squared euclidean distance

$$ed^2(\overline{\mathbf{C}}_i, \overline{\mathbf{C}}_j) = \sum_{z=0}^{d-1} (\overline{\mathbf{C}}_{i,z} - \overline{\mathbf{C}}_{j,z}) \times (\overline{\mathbf{C}}_{i,z} - \overline{\mathbf{C}}_{j,z}),$$

where the secret key of $ed^2(\overline{\mathbf{C}}_i, \overline{\mathbf{C}}_j)$ is $sk'_{l,\{i,j\}} = sk_{l,i} \otimes sk_{l,j}$. Based on the evaluation key $\tau_{sk'_{l,\{i,j\}}\rightarrow sk_{l,\{i,j\}}}$, $ed^2(\overline{\mathbf{C}}_i, \overline{\mathbf{C}}_j)$ is transformed to the ciphertext $\widetilde{ed}^2(\mathbf{C}_i, \mathbf{C}_j)$ that has the secret key $sk_{l,\{i,j\}}$.

Case 2: $\mathbf{C}_i$ and $\mathbf{C}_j$ are generated by the same public key, where $pk_{l,i} = pk_{l,j}$. We do not need to extend the ciphertexts to the same key. Then, the cloud server computes

$$ed^2(\mathbf{C}_i, \mathbf{C}_j) = \sum_{z=0}^{d-1} (\mathbf{C}_{i,z} - \mathbf{C}_{j,z}) \times (\mathbf{C}_{i,z} - \mathbf{C}_{j,z}),$$

where the secret key of $ed^2(\mathbf{C}_i, \mathbf{C}_j)$ is $sk_{l,i}$.

4.2 Secure Comparison (SC) Protocol

In order to compare euclidean distances, we propose a secure comparison protocol (Protocol 2), which is described as follows. We suppose the users have two distances μ_1, μ_2, and p. Then, they are encrypted as $\mathbf{C}_1$, $\mathbf{C}_2$, and $\mathbf{C}(p)$ respectively, where $\mathbf{C}_1$'s secret key is $sk_{l,1}$, $\mathbf{C}_2$'s secret key is $sk_{l,2}$, and $\mathbf{C}(p)$'s secret key is $sk_{l,1}$. We suppose $\mathbf{C}_1$, $\mathbf{C}_2$, and $\mathbf{C}(p)$ are stored in the cloud server. First, $\mathbf{C}_1$, $\mathbf{C}_2$, and $\mathbf{C}(p)$ are extended to the ciphertexts $\overline{\mathbf{C}}_1$, $\overline{\mathbf{C}}_2$, and $\overline{\mathbf{C}}(p)$ respectively, namely $\overline{\mathbf{C}}_1 \leftarrow CFHE.CExt(pp, \mathbf{C}_1)$, $\overline{\mathbf{C}}_2 \leftarrow CFHE.CExt(pp, \mathbf{C}_2)$, and $\overline{\mathbf{C}}(p) \leftarrow CFHE.CExt(pp, \mathbf{C}(p))$, where $\overline{\mathbf{C}}_1$, $\overline{\mathbf{C}}_2$, and $\overline{\mathbf{C}}(p)$ have the same secret key $sk_{l,\{1,2\}} = sk_{l,1}|sk_{l,2}$. The order of ciphertexts is regarded as an intermediate result. In order to prevent the users from learning about the order of ciphertexts, the cloud server uses a random bit $b \in \{0, 1\}$. Namely, if $b = 0$, the cloud server computes $\overline{\mathbf{C}}(h) = \overline{\mathbf{C}}(p) + \overline{\mathbf{C}}_1 - \overline{\mathbf{C}}_2$; otherwise, the cloud server computes $\overline{\mathbf{C}}(h) = \overline{\mathbf{C}}(p) + \overline{\mathbf{C}}_2 - \overline{\mathbf{C}}_1$. Therefore, the users cannot get the order. $\overline{\mathbf{C}}(h)$ is transmitted to the users. The users decrypts $\overline{\mathbf{C}}(h)$ to obtain h by using $sk_{l,\{1,2\}}$. If the $\lfloor \log_2 h \rfloor$th bit is 0, we set the integer $v = 0$; otherwise, $v = 1$. v is sent to the cloud server. If $b = 0$, $v = 0$ or $b = 1$, $v = 1$, $\mu_1 \leq \mu_2$; otherwise, $\mu_1 \geq \mu_2$.

Protocol 2. Secure Comparison (SC) Protocol

Input: $\mathbf{C}_1$, $\mathbf{C}_2$, and $\mathbf{C}(p)$.
Output: The comparison result.
For the cloud server
Extend $\mathbf{C}_1$, $\mathbf{C}_2$, and $\mathbf{C}(p)$ to $\overline{\mathbf{C}}_1$, $\overline{\mathbf{C}}_2$, and $\overline{\mathbf{C}}(p)$ respectively;
Choose a random bit $b \in \{0, 1\}$;
If $b = 0$
Compute $\overline{\mathbf{C}}(h) = \overline{\mathbf{C}}(p) + \overline{\mathbf{C}}_1 - \overline{\mathbf{C}}_2$;
Else
Compute $\overline{\mathbf{C}}(h) = \overline{\mathbf{C}}(p) + \overline{\mathbf{C}}_2 - \overline{\mathbf{C}}_1$;
Send $\overline{\mathbf{C}}(h)$ to the users;
For the users
Decrypt $\overline{\mathbf{C}}(h)$ to obtain h;
If the $\lfloor \log_2 h \rfloor$th bit of h is 0
Set $v = 0$;
Else
Set $v = 1$;
Send v to the cloud server;
For the cloud server
If $b = 0$, $v = 0$ or $b = 1$, $v = 1$
$\mu_1 \leq \mu_2$;
Else
$\mu_1 \geq \mu_2$.

4.3 Secure Minimum (SM) Protocol

In order to find the minimum euclidean distance, we propose a secure minimum protocol (Protocol 3), which is described as follows. In this protocol, we suppose the cloud server stores k ciphertexts $\mathbf{C}_0, \ldots, \mathbf{C}_{k-1}$, whose secret keys are $sk_{l,0}, \ldots, sk_{l,k-1}$ respectively. The cloud server first extends $\mathbf{C}_i$ to $\overline{\mathbf{C}}_i$, namely $\overline{\mathbf{C}}_i \leftarrow CFHE.CExt(pp, \mathbf{C}_i)$, where $i = 0, \ldots, k-1$. All the extended ciphertexts share the same secret key $sk_{l,\{0,\ldots,k-1\}} = sk_{l,0}| \cdots |sk_{l,k-1}$. The cloud server finds out the minimum ciphertext through the following steps.

1) The cloud server creates a set M, which consists of k ciphertexts.
2) If k is even, the cloud server groups $\{\overline{\mathbf{C}}_i\}_{i=0,\ldots,k-1}$ to $s = \frac{k}{2}$ pairs as $(\overline{\mathbf{C}}_0, \overline{\mathbf{C}}_1), \cdots, (\overline{\mathbf{C}}_{k-2}, \overline{\mathbf{C}}_{k-1})$; otherwise, the cloud server groups $\{\overline{\mathbf{C}}_i\}_{i=0,\ldots,k-1}$ to $s = \frac{(k+1)}{2}$ pairs as $(\overline{\mathbf{C}}_0, \overline{\mathbf{C}}_1), \cdots, (\overline{\mathbf{C}}_{k-1})$.
3) For each pair of ciphertexts, based on our secure comparison protocol, the cloud server and users output the small one. We can obtain s small ciphertexts.
4) The cloud server replaces the elements in M by s small ciphertexts.
5) Steps 2-4 are repeated until M has only one ciphertext, which is the minimum ciphertext $\mathbf{C}_{min}$.

In [30], the secure minimum protocol requires at least $3(k-1)$ communication rounds. The secure minimum protocol in [7] requires at least $(k-1)$ communication rounds. Our secure minimum protocol only requires $(\lfloor \log_2 k \rfloor + 1)$ communication rounds.

Protocol 3. Secure Minimum (SM) Protocol

Input: $\{\mathbf{C}_i\}_{i=0,\ldots,k-1}$.
Output: $\mathbf{C}_{\min}$.
For the cloud server
For $i = 0$ to $k-1$
Extend $\mathbf{C}_i$ to $\overline{\mathbf{C}}_i$;
End For
Set $M = \{\overline{\mathbf{C}}_i\}_{i=0,\ldots,k-1}$;
Repeat
If k is even
The cloud server groups $\{\overline{\mathbf{C}}_i\}_{i=0,\ldots,k-1}$ to $s\big(s = \frac{k}{2}\big)$ pairs as
$(\overline{\mathbf{C}}_0, \overline{\mathbf{C}}_1), \cdots, (\overline{\mathbf{C}}_{k-2}, \overline{\mathbf{C}}_{k-1})$;
Else
The cloud server groups $\{\overline{\mathbf{C}}_i\}_{i=0,\ldots,k-1}$ to $s\Big(s = \frac{(k+1)}{2}\Big)$ pairs as $(\overline{\mathbf{C}}_0, \overline{\mathbf{C}}_1), \cdots, (\overline{\mathbf{C}}_{k-1})$;
The cloud server and users run the SC protocol to find the small
ciphertext in each pair;
The cloud server sets $k = s$;
All the small ciphertexts are assigned to the set $M = \{\overline{\mathbf{C}}_j\}_{j=0,\ldots,k-1}$.
Until $k = 1$

4.4 Secure Average (SA) Protocol

In order to compute the average of all data in each cluster, we propose a secure average protocol (Protocol 4), which is described as follows. We suppose that there are m ciphertexts $\mathbf{C}_0, \ldots, \mathbf{C}_{m-1}$ that belong to the same cluster K_j, whose secret keys are $sk_{l,0}, \ldots, sk_{l,m-1}$ respectively. First, the cloud server extends $\mathbf{C}_i$ to the ciphertext $\overline{\mathbf{C}}_i$, namely $\overline{\mathbf{C}}_i \leftarrow CFHE.CExt(pp, \mathbf{C}_i)$, where $\overline{\mathbf{C}}_i$'s secret key is $sk_{l,\{0,\ldots,m-1\}} = sk_{l,0}| \cdots |sk_{l,m-1}$, $i = 0, \ldots, m-1$. The average is regarded as

a private intermediate result. Hence, the cloud server encrypts a random vector $\mathbf{r}$, and computes $\overline{\mathbf{C}}(\mathbf{sum}) = \sum_{i=0}^{m-1}\overline{\mathbf{C}}_i + m\overline{\mathbf{C}}(\mathbf{r})$, where $\overline{\mathbf{C}}(\mathbf{sum})$ is the ciphertext of the computation result $\mathbf{sum}$, $\overline{\mathbf{C}}(\mathbf{r})$ is the ciphertext of $\mathbf{r}$, $\overline{\mathbf{C}}(\mathbf{r})$'s secret key is $sk_{l,\{0,\ldots,m-1\}}$. Then, the users decrypt $\overline{\mathbf{C}}(\mathbf{sum})$ and compute the average $\mathbf{ave} = \frac{1}{m}(\mathbf{sum})$. The users encrypt the average with each user's public key $pk_{l,i}$, respectively. Encrypted average $\mathbf{C}_{ave}$ is sent to the cloud server. Finally, the cloud server computes $\mathbf{C}_{ave} = \mathbf{C}_{ave} - \mathbf{C}(\mathbf{r})$ to get the encrypted averages.

Protocol 4. Secure Average (SA) Protocol

Input: $\{\mathbf{C}_i\}_{i=0,\ldots,m-1}$.
Output: $\mathbf{C}_{ave}$.
For the cloud server
  For $i = 0$ to $m-1$
    Extend $\mathbf{C}_i$ to $\overline{\mathbf{C}}_i$;
  End For
    Choose a random d-dimensional vector $\mathbf{r}$. $\mathbf{r}$ is encrypted as $\overline{\mathbf{C}}(\mathbf{r})$;
    Compute $\overline{\mathbf{C}}(\mathbf{sum}) = \sum_{i=0}^{m-1}\overline{\mathbf{C}}_i + m\overline{\mathbf{C}}(\mathbf{r})$, and send the $\overline{\mathbf{C}}(\mathbf{sum})$ and m to the users;
    For the users
    Decrypt the $\overline{\mathbf{C}}(\mathbf{sum})$, and computes $\mathbf{ave} = \frac{1}{m}(\mathbf{sum})$;
    For $i = 0$ to $m-1$
      Encrypt $\mathbf{ave}$, and get $\mathbf{C}_{ave}$, and send $\mathbf{C}_{ave}$ to the cloud server;
    End For
  For the cloud server
    For $i = 0$ to $m-1$
      $\mathbf{r}$ is encrypted as $\mathbf{C}(\mathbf{r})$;
      Compute $\mathbf{C}_{ave} = \mathbf{C}_{ave} - \mathbf{C}(\mathbf{r})$.
    End For

In this protocol, the cloud server learns nothing about original data and average. The users also can not determine the average. In addition, according to the Algorithm 1, this protocol's outputs will be the new cluster centers of the SSED protocol. Because our SSED protocol involves homomorphic multiplications, the users encrypt the average with their public keys respectively. It is helpful for reducing the ciphertext size, computation cost, and communication cost for the later iterations.

5 Privacy-Preserving and Outsourced Multi-Party K-Means Clustering Schemes

In this section, based on our designed secure protocols in Section 4, we first propose a privacy-preserving multi-party k-means clustering scheme, which is named as the basic scheme. In this scheme, there are n source-restrained patients $P_0, \ldots, P_{n-1}$. Each patient P_i has his own breast cancer data, which are stored in a cloud server $CS1$. In order to make medical diagnosis over the data from multiple patients, the k-means clustering algorithm is implemented on their joint data sets. Then, in order to support outsourced computation, we extend the basic scheme to the advanced scheme by using an improved method of transforming ciphertexts with different secret keys.

5.1 The Basic Scheme

As shown in Fig. 1, we design the basic scheme, which consists of four phases, namely initialization, data-uploading, clustering, and result-acquisition.

Initialization. In this phase, the patients generate the public parameters pp, namely $pp \leftarrow CFHE.Setup(\lambda, n, L)$. Based on the usage of pp, the patients generate their key pairs $\{pk_{l,i}, sk_{l,i}\}_{l=0,\ldots,L-1}$ and evaluation key materials $em_i = \{em_{l,i}\}_{l=0,\ldots,L-1}$ independently, namely $(\{pk_{l,i}, sk_{l,i}\}_{l=0,\ldots,L-1}, em_i) \leftarrow CFHE.KeyGen(pp, i \in [0,n))$. Next, $\mathbf{y}_i$ is encoded as $\boldsymbol{\mu}_i$. Then, $\boldsymbol{\mu}_i$ is encrypted by $pk_{L-1,i}$. The generated ciphertext is $\mathbf{C}_i \leftarrow CFHE.Enc(pp, pk_{L-1,i}, \boldsymbol{\mu}_i)$.

Data-Uploading. In this phase, P_i uploads $\mathbf{C}_i$ and em_i to $CS1$. Then, $CS1$ stores the patients' data and generates the evaluation key $\tau_{sk'_{l,\{0,\ldots,n-1\}} \rightarrow sk_{l,\{0,\ldots,n-1\}}} \leftarrow CFHE.EvkGen(pp, \{em_i\}_{i\in[0,n)}, \{pk_{l,i}\}_{i\in[0,n)})$, where $l = 0, \ldots, L-1$, $sk'_{l,\{0,\ldots,n-1\}} = sk_{l,0} \otimes \cdots \otimes sk_{l,n-1}$, and $sk_{l,\{0,\ldots,n-1\}} = sk_{l,0} \,|\cdots|\, sk_{l,n-1}$.

Clustering. In this phase, based on our protocols, the patients and $CS1$ implement the k-means clustering algorithm. This phase consists of four steps, which are described as follows.

1) $CS1$ randomly chooses k encrypted data as the initial cluster centers $\{\mathbf{C}_j\}_{j=0,\ldots,k-1}$.
2) For each data record, based on the SSED protocol, $CS1$ computes the squared euclidean distance between the data and each cluster center.
3) Based on our SC and SM protocols, $CS1$ and the patients compare distances. The minimum distance is assigned to the closest group.
4) Based on the SA protocol, $CS1$ and the patients calculate the average. $CS1$ updates the k averages as the new cluster centers for the next iteration. Steps 2-4 are repeated until the number of iterations reaches the threshold.

At the end of this phase, $CS1$ gets the final k encrypted cluster centers $\{\mathbf{C}_j^{(i)}\}_{j=0,\ldots,k-1}$ for each patient.

Result-Acquisition. In this phase, the patients decrypt the final encrypted cluster centers independently, namely $\mathbf{o}_j \leftarrow CFHE.Dec(pp, \mathbf{C}_j^{(i)}, sk_{l,i})$, where $j = 0, \ldots, k-1$.

In our basic scheme, although the patients only need to spend a few computation costs, they have to keep online in the whole process of implementing the k-means clustering algorithm. In practice, due to the unpredictable network condition, some patients may off-line, which may affect the implementation of our basic scheme. In the next subsection, we extend the basic scheme to support outsourced computation. That is the following advanced scheme, where two non-collusion cloud servers are introduced, so that all patients do not have to keep online all the time.

5.2 The Advanced Scheme

In Chen's multi-key FHE scheme, the implementation of homomorphic multiplication includes ciphertext transformation. However, the ciphertext size increases exponentially after each ciphertext transformation. In order to implement homomorphic multiplication efficiently, we propose an efficient key switching method to transform ciphertexts that have different secret keys. Namely, based on the usage of key switching, the patients' original ciphertexts are

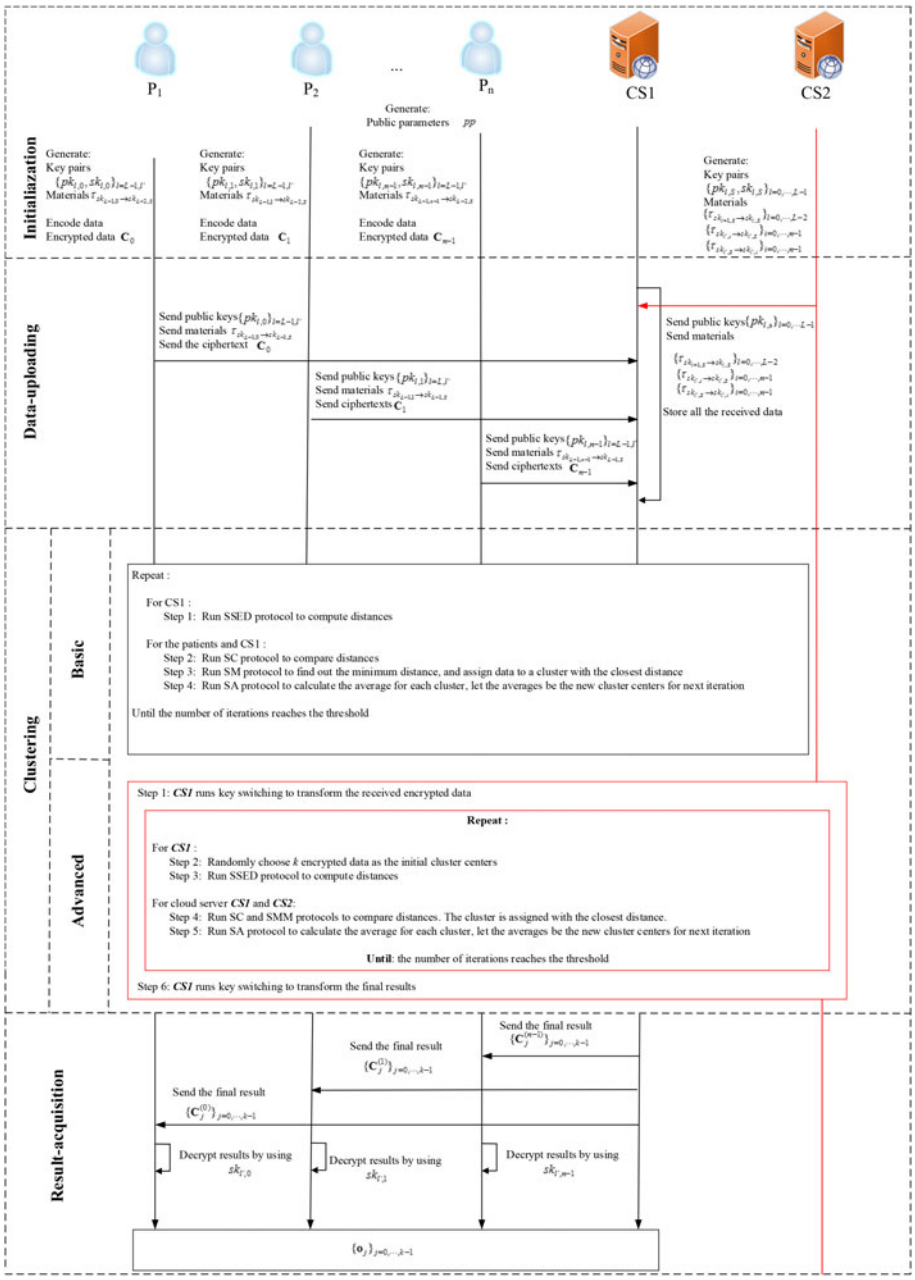


Fig. 1. The framework of our basic and advanced schemes (The red in the advanced scheme indicates changes from the basic scheme).

transformed into the refreshed ciphertexts without expanding ciphertext sizes and evaluation key sizes.

Then, based on this method, the above basic scheme is extended to the advanced scheme. In this advanced scheme, an additional cloud server $CS2$ is introduced for supporting outsourced computation of the secure k-means clustering. Hence, the patients can be off-line without affecting the implementation of the secure k-means clustering.

Based on our efficient method and secure protocols, we present an advanced privacy-preserving and outsourced multi-party k-means clustering scheme, which uses two non-collusion cloud servers $CS1$ and $CS2$. In this scheme, $CS2$ is regarded as a "special" patient P_S, which owns independent key pairs. As shown in Fig. 1, this scheme consists of four phases, which are described as follows.

Initialization. In this phase, $\{P_i\}_{i=0,\ldots,n-1}$ and $CS2$ generate the public parameters pp, namely $pp \leftarrow CFHE.Setup(\lambda, n, L)$. Based on the usage of pp, patients and $CS2$ generate their own key pairs and evaluation key materials $\{em_l\}_{l=0,\ldots,L-2}, l', L-1$. Unlike the basic scheme, each patient only needs to generate the key pairs $(pk_{l,'i}, sk_{l,'i})$ and $(pk_{L-1,i}, sk_{L-1,i})$, where $i = 0, \ldots, n-1$, l' denotes an index, $(pk_{L-1,i}, sk_{L-1,i})$ is used for original data encryption, and $(pk_{l,'i}, sk_{l,'i})$ is used for results decryption. $CS2$ generates L key pairs $\{pk_{l,S}, sk_{l,S}\}_{l=0,\ldots,L-1}$ for further computation, where S denotes an index. Then, each patient encodes his sensitive data and encrypts the encoded data.

Data-Uploading. In this phase, the patients upload their encrypted data and materials to $CS1$. $CS2$ uploads its materials to $CS1$. Then, $CS1$ generates the evaluation keys $\tau_{sk_{l+1,S} \to sk_{l,S}} \leftarrow CFHE.EvkGen(pp, em_l, pk_{l,S})$, $\tau_{sk_{L-1,i} \to sk_{L-2,S}}$

TABLE 2
Computation and Communication Costs of Our Protocols

	SSED	SC	SM	SA
Number of encryptions	×	×	×	m
Number of decryptions	×	1	$\frac{1}{2}(1+k/2)\log_2 k$	m
Number of homomorphic additions	1	2	$(1+k/2)\log_2 k$	$m+1$
Number of homomorphic multiplications	1	×	×	×
Number of interactions	×	2	$(1+k/2)\log_2 k$	2

$\leftarrow CFHE.EvkGen\ (pp, em_{L-1}, pk_{L-1,i})$, and $\tau_{sk_{l,'S}\rightarrow sk_{l,'i}} \leftarrow CFHE.EvkGen(pp, em_{l'}, pk_{l,'i})$, where $l=0,\ldots,\ L-2$, $\tau_{sk_{l+1,S}\rightarrow sk_{l,S}}$ is used for noise management, $\tau_{sk_{l,'S}\rightarrow sk_{l,'i}}$ is used for transforming the final results back to the ciphertexts that are encrypted with the patients' public keys respectively, $\tau_{sk_{l,'i}\rightarrow sk_{l,'S}}$ is used for transforming ciphertexts that have different private keys.

Clustering. In this phase, based on our protocols, $CS1$ and $CS2$ implement the secure k-means clustering algorithm. This phase consists of six steps, which are described as follows.

1) $CS1$ uses $\tau_{sk_{L-1,i}\rightarrow sk_{L-1,S}}$ to transform encrypted data $\mathbf{C}_i$. Namely, $\mathbf{C}_i' \leftarrow RBGV.\ SwitchKey(\tau_{sk_{L-1,i}\rightarrow sk_{L-1,S}}, \mathbf{C}_i)$.
2) $CS1$ randomly picks k encrypted data as the initial cluster centers $\{\overline{\mathbf{C}}_j\}_{j=0,\ldots,k-1}$.
3) For each data, based on the SSED protocol, $CS1$ computes the squared euclidean distance between the data and each cluster center.
4) Based on our SC and SM protocols, $CS1$ and $CS2$ compare distances. The minimum distance is assigned to the closest group.
5) Based on the SA protocol, $CS1$ and $CS2$ calculate the averages. $CS1$ updates the k averages as the new cluster centers for the next iteration.

   Steps 3-5 are repeated until the number of iterations reaches the predefined threshold. At the last iteration, $CS1$ gets the final k encrypted cluster centers $\{\overline{\mathbf{C}}_j\}_{j=0,\ldots,k-1}$, which are encrypted with $CS2$'s public key.
6) $CS1$ uses $\tau_{sk_{l,'S}\rightarrow sk_{l,'i}}$ to transform the final results back to the ciphertexts which are encrypted with the patients' public keys respectively. Namely, $\mathbf{C}_j^{(i)} \leftarrow RBGV.SwitchKey(\ \tau_{sk_{l,'S}\rightarrow sk_{l,'i}}, \overline{\mathbf{C}}_j), j=0,\ldots,k-1$.

At the end of this phase, $CS1$ gets the final k encrypted cluster centers $\{\mathbf{C}_j^{(i)}\}_{j\in[0,k)}$ for each patient P_i.

Result-Acquisition. In this phase, each patient extracts final results from $CS1$ and decrypts them independently. Therefore, P_i gets the final cluster center $\mathbf{o}_j \leftarrow CFHE.Dec\ (pp, \mathbf{C}_j^{(i)}, sk_{l,'i})$, where $i=0,\ldots,n-1$, $j=0,\ldots,k-1$.

In our advanced scheme, required computations are completely outsourced to cloud servers. Hence, all patients do not have to keep online all the time.

6 Security and Efficiency Analysis

6.1 Security Analysis

In this subsection, we prove the security of our protocols and schemes.

SSED protocol This protocol is non-interactive and its inputs are ciphertexts, which are received from P_i, where $i=0,\ldots,n-1$. Because $CS1$ does not have the secret key sk_i, it can not learn anything from these ciphertexts.

SC protocol This protocol's inputs are the ciphetexts of squared euclidean distances. Due to the semantic security of Chen's multi-key FHE scheme, $CS1$ learns nothing about distances. Furthermore, due to the random binary bit b, P_i learns nothing about the order and distances.

SM protocol This protocol is constructed based on the SC protocol. In this protocol, $CS1$ cannot learn anything about squared euclidean distances. In addition, P_i does not know the order about the distances.

SA protocol This protocol's inputs are the ciphertexts that come from P_i. Because $CS1$ does not have the secret keys sk_i, it does not learn anything from these ciphertexts. Besides, due to the blinding vector $\mathbf{r}$, P_i can not learn anything about intermediate averages.

Basic scheme In our basic scheme, P_i generates its own key pairs $\{(pk_{l,i}, sk_{l,i})\}_{l=0,\ldots,L-1}$ and encrypts data $\mathbf{y}_i$ with its own public key. Then, it uploads ciphetexts to $CS1$ through secure communication channels. The security follows the semantic security of Chen's multi-key FHE scheme. During the clustering phase, based on our protocols, P_i and $CS1$ implement the k-means clustering algorithm. In the results-acquisition phase, each patient downloads the results form $CS1$ and decrypts them by its own secret key individually. Hence, our basic scheme is secure in the semi-honest model.

Advanced scheme In our advanced scheme, based on our protocols, $CS1$ and $CS2$ implement the secure k-means clustering algorithm. Because $CS1$ does not have the patients' secret keys and $CS2$'s secret key, it can not know the patients' inputs and final outputs. Although $CS2$ can decrypt ciphertexts by using its own secret keys, it only has the blinded decryption. $CS2$ learns nothing about patients' inputs and final outputs. In addition, the required computations are totally outsourced to cloud severs. The patients learn nothing about the intermediate results of the k-means clustering algorithm. Consequently, our advanced scheme is secure in the semi-honest model.

6.2 Efficiency Analysis

In this subsection, we analyze computation and communication costs of our protocols and schemes, which is described as follows.

As shown in Table 2, we can observe computation and communication costs of our protocols, where m denotes the the number of data in the cluster K_j, k denotes the number of clusters, and d denotes the dimension of the plaintext. Our

TABLE 3
Total Computation and Communication Costs of Our Basic Scheme

	Initialization	Data-uploading	Clustering	Result-acquisition
Number of key generations	nL	$\times$	$\times$	$\times$
Number of encryptions	n	$\times$	$mk\omega$	$\times$
Number of decryptions	$\times$	$\times$	$\frac{\omega}{2}(1+nk/2)\log_2 nk + k\omega m$	nk
Number of homomorphic additions	$\times$	$\times$	$\omega(nk+(1+nk/2)\log_2 nk + mk + k)$	$\times$
Number of homomorphic multiplications	$\times$	$\times$	$k\omega n$	$\times$
Number of key switchings	$\times$	$\times$	$k\omega n$	$\times$
Number of interactions	$\times$	n	$\omega(1+nk/2)\log_2 nk + 2k\omega$	n

TABLE 4
Computation and Communication Costs of One Patient in Our Basic Scheme

	Initialization	Data-uploading	Clustering	Result-acquistion
Number of key generations	L	$\times$	$\times$	$\times$
Number of encryptions	1	$\times$	$mk\omega/n$	$\times$
Number of decryptions	$\times$	$\times$	$(\frac{\omega}{2}(1+nk/2)\log_2 nk + k\omega m)/n$	k
Number of homomorphic additions	$\times$	$\times$	$\omega(nk+(1+nk/2)\log_2 nk + mk + k)/n$	$\times$
Number of homomorphic multiplications	$\times$	$\times$	$\times$	$\times$
Number of key switching	$\times$	$\times$	$\times$	$\times$
Number of interactions	$\times$	1	$(\omega(1+nk/2)\log_2 nk + 2k\omega)/n$	1

TABLE 5
Total Computation and Communication Costs of Our Advanced Scheme

	Initialization	Data-uploading	Clustering	Result-acquisition
Number of key generations	$L+n$	$\times$	$\times$	$\times$
Number of encryptions	n	$\times$	$mk\omega$	$\times$
Number of decryptions	$\times$	$\times$	$\frac{\omega}{2}(1+nk/2)\log_2 nk + k\omega m$	nk
Number of homomorphic additions	$\times$	$\times$	$\omega(nk+(1+nk/2)\log_2 nk + mk + k)$	$\times$
Number of homomorphic multiplications	$\times$	$\times$	$k\omega n$	$\times$
Number of key switchings	$\times$	n	$k\omega n$	$\times$
Number of interactions	$\times$	n	$\times$	n

SSED protocol only needs one homomorphic addition and one homomorphic multiplication. The proposed SC protocol requires one decryption, two homomorphic additions, and two interactions without any homomorphic multiplication. The proposed SM protocol needs $\frac{1}{2}(1+k/2)\log_2 k$ decryptions, $(1+k/2)\log_2 k$ homomorphic additions, and $(1+k/2)\log_2 k$ interactions with no homomorphic multiplications. Our SA protocol only needs m encryptions, m decryptions, $m+1$ homomorphic additions, and two interactions.

Table 3 shows total computation and communication costs of our basic scheme, where ω denotes the threshold number of iterations, n is the number of users, and L denotes the circuit depth. In our basic scheme, the initialization phase only needs nL key generations and n encryptions. The data-uploading phase only requires n interactions. The clustering phase requires high computation and communication costs, including $mk\omega$ encryptions, $k\omega n$ homomorphic multiplications, and $k\omega n$ key switchings. The result-acquisition phase needs nk decryptions and n interactions.

Table 4 describes computation and communication costs of one patient in our basic scheme. As shown in Table 4, the initialization phase only needs L key generations and one encryption. The data-uploading phase only requires one interaction. The clustering phase, there exist high computation and communication costs, including $mk\omega/n$ encryptions. The result-acquisition phase needs k encryptions and one interaction.

Table 5 shows total computation and communication costs of our advanced scheme. In our advanced scheme, the initialization phase only needs $L+n$ key generations and n encryptions. The data-uploading phase only requires n key switchings and n interactions. The clustering phase requires high computation and communication costs, including $mk\omega$ encryptions, $n\omega k$ homomorphic multiplications, and $k\omega n$ key switchings. The result-acquisition phase needs nk decryptions and n interactions.

Table 6 shows computational costs of one patient in our advanced scheme. As shown in Table 6, the initialization phase only needs L key encryptions, one encryption, and one interaction. The result acquisition phase only requires k decryptions and one interaction. Data uploading and clustering phases require no computation and communication costs.

From Table 2, we can observe that there are high computation costs in the SSED protocol, which involves several homomorphic multiplications. From Tables 3 and 4, it can

TABLE 6
Computation and Communication Costs of One Patient in Our Advanced Scheme

	Initialization	Data-uploading	Clustering	Result-acquistion
Number of key generations	L	×	×	×
Number of encryptions	1	×	×	×
Number of decryptions	×	×	×	k
Number of homomorphic additions	×	×	×	×
Number of homomorphic multiplications	×	×	×	×
Number of key switching	×	×	×	×
Number of interactions	1	×	×	1

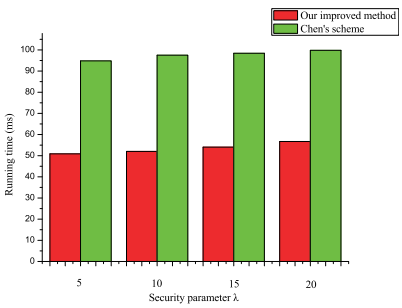


Fig. 2. The efficiency comparison of homomorphic multiplication between our improved method and Chen's multi-key FHE scheme.

be observed that the patients have to do complex computations in our basic scheme. From Tables 5 and 6, there is a conclusion that the patients only need to do key generation, encryption and decryption in our advanced scheme. This conclusion verifies that the implementation of the secure k-means clustering algorithm is completely outsourced to the cloud server. In addition, except for data uploading and result acquisition, the patients do not need additional communication costs in our advanced scheme.

6.3 Simulation Results

In this subsection, we compare the efficiency of homomorphic multiplication between our improved method and Chen's multi-key FHE scheme [4]. Then, we analyze the running time of our protocols and advanced scheme. Experiments are carried out on the same personal computer's virtual machine without the GPU hardware platform. The experimental environment is set as follows: the operating system is microsoft windows 7, featuring two Intel (R) Core (TM) i5-3470 CPU processors, running at 3.20 GHz, with 4.00 GB RAM, and the virtual machine's operation system is ubuntu 12.04, featuring single Intel (R) Core (TM) i5-3470 CPU processor, with 956 MB RAM. Our implementation uses NTL large number library for high

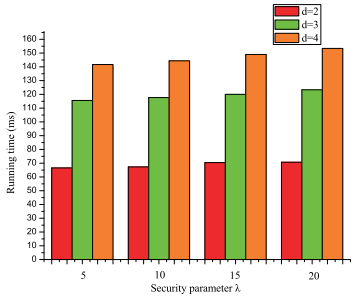


Fig. 3. The efficiency of our secure squared euclidean distance protocol.

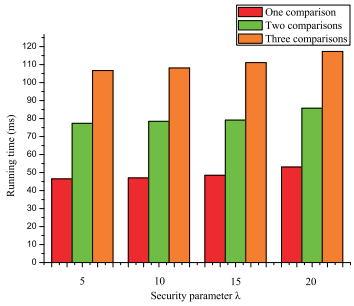


Fig. 4. The efficiency of our secure comparison protocol.

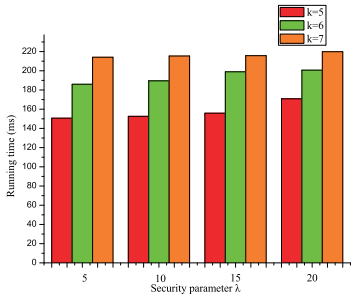


Fig. 5. The efficiency of our secure minimum protocol.

level numeric algorithms. The experimental code is compiled on the GCC platform by the C++ language. We adopt Wisconsin Breast Cancer data [1] for the experiments. For convenience, we set the security parameter λ ranging from 5 to 20.

Fig. 2 shows the efficiency comparison of homomorphic multiplication between our improved method and Chen's multi-key FHE scheme. As shown in this figure, because our improved method does not expand the ciphertext size in the process of transforming the ciphertext, the efficiency of our improved method's homomorphic multiplication is more efficient than that of Chen's multi-key FHE scheme [4] with the increasing of the security parameter λ. Fig. 3 shows the efficiency of our secure squared euclidean distance protocol, where d ranges from 2 to 4. It can be easily observed that our secure squared euclidean distance protocol's running time increases significantly with the increasing of d. Fig. 4 shows the efficiency of our secure comparison protocol, where the number of comparisons ranges from 1 to 3.

1. http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original)

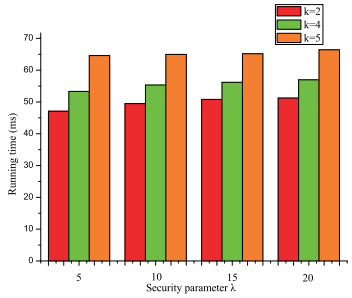


Fig. 6. The efficiency of our secure average protocol.

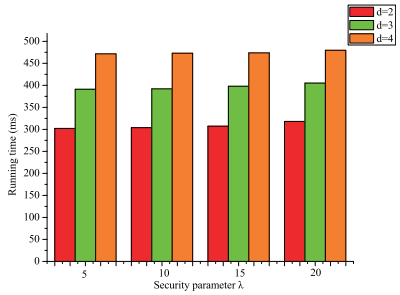


Fig. 7. The efficiency of our advanced multi-party k-means clustering scheme.

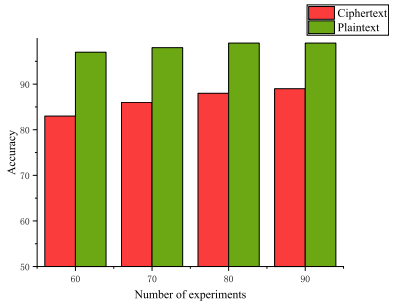


Fig. 8. The accuracy of our advanced multi-party k-means clustering scheme and k-means clustering that is directly applied on the plaintext.

We can observe that our secure comparison protocol's running time increases rapidly with the increasing number of comparisons. Fig. 5 shows the efficiency of our secure minimum protocol, where k ranges from 5 to 7. Besides, this figure shows a marked increase of our secure minimum protocol's running time with the increasing of k. Fig. 6 shows the efficiency of our secure average protocol, where k ranges from 2 to 5. With the increasing of d, more running time is needed for the implementation of our secure average protocol. Fig. 7 shows the efficiency of our advanced multi-party k-means clustering scheme, where d ranges from 2 to 4. In addition, this figure shows the changing trend of our scheme's running time with the increasing of d. Fig. 8 shows the accuracy comparison of our advanced k-means clustering scheme and k-means clustering that is directly applied on the plaintext, where the number of experiments ranges from 60 to 90. This figure shows that the accuracy of our advanced k-means clustering scheme is close to the accuracy of the k-means clustering that is directly applied on the plaintext.

7 Conclusion

In this paper, we have designed secure squared euclidean distance, comparison, minimum, and average protocols. Based on the usage of our protocols, we have proposed the basic and advanced schemes for the implementation of the privacy-preserving multi-party k-means clustering algorithm. In our basic scheme, the patients have to do complex computations. Then, we propose the advanced scheme, which outsources the required computations completely to cloud servers. In addition, in order to improve the efficiency of homomorphic multiplication, we propose an efficient method for transforming ciphertexts that have different secret keys. We also have proved that our schemes are secure in the semi-honest model. Simulation results show that our method is helpful for improving the efficiency of homomorphic multiplication in Chen's multi-key FHE scheme. In addition, our protocols and secure k-means clustering scheme are feasible.

References

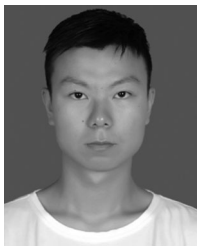
[1] C. Dwork, "Differential privacy," *Encyclopedia Cryptogr. Secur.*, pp. 338–340, 2011.
[2] R. L. Rivest *et al.*, "On data banks and privacy homomorphisms," *Found. Secure Comput.*, vol. 4, no. 11, pp. 169–180, 1978.
[3] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *Proc. 44th Annu. ACM Symp. Theory Comput.*, 2012, pp. 1219–1234.
[4] L. Chen, Z. Zhang, and X. Wang, "Batched multi-hop multi-key FHE from ring-LWE with compact ciphertext extension," in *Proc. Theory Cryptogr. Conf.*, 2017, pp. 597–627.
[5] Y. Li, Z. Hao, W. Wen, and G. Xie, "Research on differential privacy preserving k-means clustering," *Comput. Sci.*, vol. 40, no. 3, pp. 287–290, 2013.
[6] H. Li, X. Wu, and Y. Chen, "K-means clustering method preserving differential privacy in MapReduce framework," *J. Commun.*, vol. 37, no. 2, pp. 124–130, 2016.
[7] J. Ren, J. Xiong, Z. Yao, M. Rong, and M. Lin, "DPLK-means: A novel differential privacy k-means mechanism," in *Proc. IEEE 2nd Int. Conf. Data Sci. Cyberspace*, 2017, pp. 133–139.
[8] Y. Zhang, N. Liu, S. Wang, and M. Z. Asghar, "A differential privacy protecting k-means clustering algorithm based on contour coefficients," *PLoS One*, vol. 13, no. 11, 2018.
[9] P. Bunn and R. Ostrovsky, "Secure two-party k-means clustering," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 486–497.
[10] D. Liu, E. Bertino, and X. Yi, "Privacy of outsourced k-means clustering," in *Proc. 9th ACM Symp. Inf., Comput. Commun. Secur.*, 2014, pp. 123–134.
[11] F.-Y. Rao, B. K. Samanthula, E. Bertino, X. Yi, and D. Liu, "Privacy-preserving and outsourced multi-user k-means clustering," in *Proc. IEEE Conf. Collaboration Internet Comput.*, 2015, pp. 80–89.
[12] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," *SIAM J. Comput.*, vol. 43, no. 2, pp. 831–871, 2014.
[13] A. Theodouli, K. A. Draziotis, and A. Gounaris, "Implementing private k-means clustering using a LWE-based cryptosystem," in *Proc. IEEE Symp. Comput. Commun.*, 2017, pp. 88–93.
[14] H.-J. Kim and J.-W. Chang, "A privacy-preserving k-means clustering algorithm using secure comparison protocol and density-based center point selection," in *Proc. IEEE 11th Int. Conf. Cloud Comput.*, 2018, pp. 928–931.
[15] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, 1999, pp. 223–238.
[16] K. Xing, C. Hu, J. Yu, X. Cheng, and F. Zhang, "Mutual privacy preserving k-means clustering in social participatory sensing," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 2066–2076, Aug. 2017.

[17] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP," in *Proc. Adv. Cryptol.*, 2012, pp. 868–886.
[18] G. Sakellariou and A. Gounaris, "Homomorphically encrypted k-means on cloud-hosted servers with low client-side load," *Computing*, vol. 101, pp. 1813–1836, 2019.
[19] W. Wu, J. Liu, H. Wang, J. Hao, and M. Xian, "Secure and efficient outsourced k-means clustering using fully homomorphic encryption with ciphertext packing technique," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 10, pp. 3424–3437, Oct. 2021.
[20] Y. Fan *et al.*, "PPMCK: Privacy-preserving multi-party computing for K-means clustering," *J. Parallel Distrib. Comput.*, vol. 154, pp. 54–63, 2021.
[21] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," in *Proc. Int. Algorithmic Number Theory Symp.*, 1998, pp. 267–288.
[22] M. Clear and C. McGoldrick, "Multi-identity and multi-key leveled FHE from learning with errors," in *Proc. Adv. Cryptol.*, 2015, pp. 630–656.
[23] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Proc. Adv. Cryptol.*, 2013, pp. 75–92.
[24] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs, "Multiparty computation with low communication, computation and interaction via threshold FHE," in *Proc. Adv. Cryptol.*, 2012, pp. 483–501.
[25] P. Mukherjee and D. Wichs, Two round MPC from LWE via multi-key FHEIACR Cryptol. ePrint Arch., 2015. [Online]. Available: https://eprint.iacr.org/2015/345
[26] C. Peikert and S. Shiehian, "Multi-key FHE from LWE, revisited," in *Proc. Theory Cryptogr. Conf.*, 2016, pp. 217–238.
[27] Z. Brakerski and R. Perlman, "Lattice-based fully dynamic multi-key FHE with short ciphertexts," in *Proc. Adv. Cryptol.*, 2016, pp. 190–213.
[28] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *ACM Trans. Comput. Theory*, vol. 6, no. 3, pp. 1–14, 2014.
[29] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Manual for using homomorphic encryption for bioinformatics," *Proc. IEEE*, vol. 105, no. 3, pp. 552–567, Mar. 2017.
[30] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, vol. 2014, 2015, Art. no. 331.

Peng Zhang received the PhD degree in signal and information processing from Shenzhen University, China, in 2011. She is currently an associate professor with the College of Electronics and Information Engineering, Shenzhen University, China. Her current research interests include cryptography technology and privacy-preserving computing. She has published more than 30 academic journal and conference papers.

Teng Huang received the PhD degree in computer science from Beihang University, Beijing, China, in 2019. Currently, he is working with the Institute of Artificial Intelligence and Blockchain, Guangzhou University, Guangzhou, China. His research interests include deep learning, image processing and understanding.

Xiaoqiang Sun received the BS degree in mathematics and applied mathematics from Fuyang Normal University, in 2011 and the PhD degree in information and communication engineering from Shenzhen University, in 2018. He was a post-doctoral fellow with the F. Richard Yus Group, Carleton University, Ottawa, from 2019 to 2020. He was a post-doctoral fellow with the College of Electronics and Information Engineering, Shenzhen University, from 2018 to 2021. He is currently a lecturer with the School of Computer Science, Shenzhen Institute of Information Technology. He has published more than 10 academic journal articles and conference papers. His research interests include cryptography, information security, and fully homomorphic encryption.



Wei Zhao received the BS degree in electronic and information engineering from Xiangtan University, China, in 2017, and the MS degree in information and telecommunication engineering from Shenzhen University, China, in 2020. He is currently working toward the postgraduate degree with the College of Electronics and Information Engineering, Shenzhen University, China. His current research interests include fully homomorphic encryption and secure multi-party computation.

Hongwei Liu received the PhD degree in signal and information processing from Xidian University, Xian, China, in 2009. He is a professor with the College of Big Data and Internet, Shenzhen Technology University, China. His research interests include cryptography and information security.

Shangqi Lai received the BE degree from the Nanjing University of Aeronautics and Astronautics, in 2014, the MS degree from the University of Hong Kong, in 2015, and the PhD degree from Monash University, in 2020. He is a research fellow with the Faculty of Information Technology, Monash University. His research interests include data privacy and cloud security.

Joseph K. Liu received the PhD degree from the Chinese University of Hong Kong, in 2004. He is currently an associate professor with the Faculty of Information Technology, Monash University. His research areas include cyber security, blockchain and applied cryptography. He has received more than 8500 citations and with more than 200 publications in top venues such as CRYPTO, ACM CCS, NDSS, INFOCOM. He has been given the prestigious ICT researcher of the Year 2018 Award by the Australian Computer Society (ACS), the largest professional body in Australia representing the ICT sector, for his contribution to the blockchain and cyber security community.