Mutual Privacy Preserving k-Means Clustering in Social Participatory Sensing

Kai Xing, *Member, IEEE,* Chunqiang Hu, *Member, IEEE,* Jiguo Yu, *Member, IEEE,* Xiuzhen Cheng, *Fellow, IEEE,* Fengjuan Zhang

Abstract—In this paper, we consider the problem of mutual privacy-protection in social participatory sensing in which individuals contribute their private information to build a (virtual) community. Particularly, we propose a mutual privacy preserving k-means clustering scheme that neither discloses individual's private information nor leaks the community's characteristic data (clusters). Our scheme contains two privacy-preserving algorithms called at each iteration of the k-means clustering. The first one is employed by each participant to find the nearest cluster while the cluster centers are kept secret to the participants; and the second one computes the cluster centers without leaking any cluster center information to the participants while preventing each participant from figuring out other members in the same cluster. An extensive performance analysis is carried out to show that our approach is effective for k-means clustering, can resist collusion attacks, and can provide mutual privacy protection even when the data analyst colludes with all except one participant.

Index Terms—Privacy Preservation; k-Means Clustering; Social Participatory Sensing; Homomorphic Encryption; Social Networking Big Data.

1 Introduction

Online/mobile social networking and the sensors (pedometer, cardio watch, etc.) embedded in various smart devices have become deeply involved in our daily lives, which consequently triggers a large variety of social participatory sensing applications. Participatory sensing is a process of acquisition, integration, and analysis of big and heterogeneous data, which is generated by a diversity of sources, such as smart devices and sensors [1], [2], and it can reduce the resource cost as data collection no longer depends on the large-scale deployment of sensors. With such applications, people show more or less interests in performing social comparisons with others, which could help them get an accurate view about themselves and motivate them to achieve more, according to the social psychology theory [3]. For example, people constantly engage in comparison with their friends on emotional moods, travelled locations, walking distances, fitness status, etc. One question that is frequently asked is: ***where am I in my community?*** A study on the community data with data mining techniques could answer such questions. However, although the underlying data about the participants are greatly helpful, such information may trigger serious concerns on privacy leakage (location, emotion, health, etc.). Consequently there is a conflict between data privacy and the wide adoption of social participatory sensing applications.

For a typical social participatory sensing application, it is important to motivate participation while at the same time the participatory sensing process should not disclose the private information of any participating party (the private data) or the community (patterns, distribution, etc.). Therefore it is essential to develop a mutual privacy preserving data mining technique to protect both the users and the community. In other words, we need a technique that can mutually protect the privacy of both the participants and the community, i.e., a technique that allows the data analyst (the social application server) to extract information about the community without accessing any user's private data while no participatory participant can obtain any information about other participants and the community.

However, existing privacy-preserving techniques either focus on protecting the privacy of one single side only, namely the participating users, or leak intermediate results during community learning to potential privacy attackers, or cannot resist collusion attacks. For example, the collaborating participants could identify the candidate clusters within each iteration of the k-means clustering in [4]. Exposing such intermediate information or the final community characteristics data can possibly put individual's privacy at risk, or cause panic and extreme actions among the participants in the community.

In this paper, we propose a mutual privacy preserving clustering scheme based on a well-known data mining method, the k-means algorithm [5], [6], which groups similar entities into clusters with the goal of minimizing intra-group distance and maximizing inter-group distance. Specifically, k-means clustering is an iterative algorithm with each iteration consisting of two steps: assigning each participant to the nearest cluster, and updating the center of each cluster. The iteration terminates in a fixed number of rounds or until the change of the cluster centers meets a given threshold [6].

Our main contributions can be summarized as follows:

- We propose two privacy-preserving algorithms called at

Manuscript received Oct. 31, 2016; revised Dec., 17, 2016; revised Feb., 28, 2017; accepted April 1, 2017.

- *Kai Xing and Fengjuan Zhang are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei Anhui, P.R.China. E-mail: kxing@ustc.edu.cn, fjzhang@mail.ustc.edu.cn*
- *Chunqiang Hu (Corresponding Author) and Xiuzhen Cheng are with the Department of Computer Science, The George Washington University, Washington DC, USA.E-mail: {chu,cheng}@gwu.edu*
- *Jiguo Yu (Co-corresponding Author) is with the School of Information Science & Engineering, Qufu Normal University, China. E-mail: jiguoyu@sina.com.*

each iteration of the k-means clustering. The first one is employed by each participant to find the nearest cluster and the second one updates cluster centers. Security and privacy analysis demonstrates that our k-means clustering scheme can resist collusion attacks.

- The proposed algorithms enable mutual privacy protection in clustering not only via keeping individuals' information private, but also by restraining the leakage of any cluster center information to the participants and preventing each participant from figuring out other members of the cluster, which is different from traditional approaches [7].
- To our best knowledge, this is the first mutual privacy preserving and collusion-resistant k-means clustering scheme, and we believe that its basic idea can be generalized to other iterative data mining algorithms for privacy protection.
- We carry out an extensive experimental study to validate our design. The results indicate that our privacy-preserving k-means clustering scheme is effective in clustering, and can provide mutual privacy protection.

The rest of the paper is organized as follows. Section 2 introduces the most related existing work on privacy-preserving k-means clustering. Section 3 presents the preliminary knowledge about k-means clustering and introduces the assumptions adopted by this paper. Our privacy-preserving k-means clustering algorithm is detailed in Section 4, and its corresponding privacy and cost analyses are carried out in Section 5. Experimental studies are reported in Section 6. We conclude this paper with a future research discussion in Section 7.

2 Related Work

Many existing privacy-preserving k-means clustering algorithms have been developed for different data distributions such as (*horizontally partitioning data*, *vertically partitioning data*, and *arbitrarily partitioned data* [4]) to protect the privacy of the users and the communities in a social participatory sensing application. In this subsection, we mainly summarize the state-of-the-art of existing privacy-preserving k-means clustering algorithms.

In a vertically partitioned data distribution, the data of an entity is distributed to different participants in such a way that each party obtains a portion of the attributes owned by the entity. Hence there is a need for the participating participants to disclose their private data during the computation of k-means clustering. The intermediate assignments of entities to their nearest clusters also pose a threat to privacy. The first privacy-preserving k-means algorithm for vertically partitioned data was proposed by Vaidya in [7], in which the distance between participants is securely computed based on the secure permutation scheme proposed by Du [8] and homomorphic encryption. However, this protocol requires the existence of three non-colluding participants, an assumption that cannot be easily guaranteed in many real-life applications. In [9], additive secret sharing [10] was adopted as a cryptographic primitive to implement a secure multi-party computation protocol, so as to realize privacy preserving clustering. To avoid the requirement of non-colluding participants, Samet [11] proposed an algorithm to compute the sum of distances by adopting the secure sum scheme. There also exists other research [12], [13] on vertically partitioned data distribution; but none of them can protect the number of entities in a cluster from being leaked. This problem is solved by our scheme proposed in this paper.

In a horizontally partitioned data distribution, each entity is owned by a single party; thus the distance to cluster centers can be computed without violating privacy. However, privacy disclosure can happen during the computation of intermediate cluster centers. Inan *et al.* [14] introduced a protocol for securing multi-party computation of a dissimilarity matrix over horizontally partitioned data, which constructs the dissimilarity matrix of objects from different sites in a privacy preserving manner. Jha *et al.* [15] presented a privacy-preserving k-means clustering algorithm based on oblivious polynomial evaluation and homomorphic encryption. This approach can be applied in a multi-party environment, but the intermediate centers are often exposed to potential privacy attacks. Our privacy preserving k-means clustering proposed in this paper can be applied to horizontally partitioned data and to prevent the disclosure of any additional information about intermediate centers and the cluster label of each entity.

Arbitrarily partitioned data was first considered in [4], in which a privacy preserving k-means clustering was proposed for arbitrarily partitioned data distributed between two participants. The idea is to split all the intermediate results into random partitions. Yu *et al.* [16] applied the concept of parallel computing to tackle the privacy-preserving multi-party k-means clustering problem, which can speed up the clustering process. This approach was designed for both vertically partitioned and horizontally partitioned data. Bunn *et al*. [17] proposed a two-party k-means clustering protocol based on homomorphic encryption to guarantee privacy in arbitrarily partitioned data, in which the protocol does not disclose the intermediate results and cluster assignments. Moreover, they designed a secure protocol for randomly selecting k initial centers. However, if the protocol is extended to multi-party k-means clustering, it may bring new security and privacy risks. For example, the protocol cannot resist collusion attacks when more than $\frac{n}{2}$ participants have collusion activities, where there are n participants.

There exist many other research on privacy-preserving k-means clustering. Rao *et al.* [18] described a two-party k-means clustering protocol, which can be implemented by utilizing any semantically secure homomorphic encryption scheme. Liu *et al.* [19] presented an outsourced k-means clustering, which proposed an encryption algorithm to generate trapdoor information; but this scheme only protects one party's privacy. Lin [20] applied the linear transformation and the random perturbation of the kernel matrix for privacy preservation in k-means clustering. Patel *et al.* [21] adopted secret sharing and the code-based zero-knowledge identification scheme to construct a distributed privacy-preserving k-means clustering scheme under the malicious adversarial model, in contrast with the mainstream research that assumes the semi-honest adversarial model. In [22], an overview on existing privacy preserving k-means clustering algorithm based on secure multi-party computation was provided. To ensure I/O efficiency, [23] and [24] proposed a privacy-preserving version of the simple deterministic algorithm Recluster; but the cluster centers are exposed to both participants in the protocol. Erkin *et al.* [25] distributed trust among a number of helper users instead of relying on a single party to obtain k-means clustering without leaking the clusters' privacy; but this approach requires a strong assumption of non-colluding activities among the participants. Samanthula *et al.* [18] proposed a privacy-preserving distributed clustering mechanism via outsourcing multi-user k-means clustering to cloud servers; unfortunately this scheme still cannot resist collusion attacks.

In summary, existing privacy preserving clustering solutions cannot provide mutual privacy protection as either the private data of the participants or the (intermediate) cluster information may be disclosed; moreover, most of them cannot resist collusion attacks. In this paper, we propose a robust mutual privacy preserving k-means clustering algorithm that can preserve the privacy of the participating participants as well as the intermediate results of the cluster centers while resisting collusion attacks. Our research was motivated by social participatory sensing applications where the community information should be kept confidential since otherwise it may result in panic or extreme actions among the community members, and the personal data of the social participants should be kept private as regular community members usually do not want to disclose their private information; meanwhile, such applications are vulnerable to collusion attacks.

3 Preliminaries and Assumptions

For better elaboration, the notations used in this paper and their semantic meanings are presented in Table 1.

TABLE 1
The Notations and Their Semantic Meanings

Notations	means
U_i	The ith cluster, $1 \leq i \leq k$
$\mathbf{u_j}$	The jth cluster center
p_1, q_1	Two prime integers
$E(\cdot)$	Encryption operation
$D(\cdot)$	Decryption operation
PU	Public key
PR	Private key

3.1 The k-Means Clustering Algorithm

The process of k-means clustering is based on (1) and (2). Assume that there are n participants with each participant a_i holding a q-dimensional sample data $\mathbf{a_i}$. Suppose that the participants need to be grouped into k clusters $U_1, ..., U_k$, with the center of the j-th cluster denoted by $\mathbf{u_j}$. Initially, each cluster center is arbitrarily and randomly assigned. Note that cluster centers can also be initialized with the method proposed in [26]. A sample data $\mathbf{a_i}$ belongs to a cluster U_j if the center $\mathbf{u_j}$ is the closest among all centers to $\mathbf{a_i}$ according to (1), where $\mathbf{u_j}$ is the mean of the samples in U_j computed from (2). There are many criterions to measure the distance between a sample and the related cluster center. In this paper, we adopt the Euclidean distance as our criterion. At each iteration, the k-means algorithm re-assigns the sample data to their nearest centers following (1) and re-computes the cluster centers $\mathbf{u_1}, ..., \mathbf{u_k}$ according to (2). The iteration terminates when there is no or little change in the cluster centers.

$$c_i := \arg\min_j ||\mathbf{a_i} - \mathbf{u_j}||^2, \tag{1}$$

$$\mathbf{u_j} = \frac{\sum_{i=1}^{n} I\{c_i = j\}\mathbf{a_i}}{\sum_{i=1}^{n} I\{c_i = j\}} \tag{2}$$

where $1 \leq j \leq k$, $1 \leq i \leq n$, and $I\{c_i = j\}$ is the index function that equals 1 if $c_i = j$ and 0 otherwise.

3.2 Homomorphic Encryption

Homomorphic encryption [27] allows certain computation over encrypted data. Paillier cryptosystem [28] is a popular Homomorphic encryption scheme that provides fast encryption and decryption, which is a probabilistic asymmetric algorithm based on the decisional composite residuosity problem. It is adopted by the secure scalar product, which has been widely used in privacy preserving data mining. The Paillier cryptosystem is briefly introduced as follows:

- **Key generation:** An entity selects two large primes p_1 and q_1 and computes $N = p_1 q_1$ and $\lambda = lcm(p_1 - 1, q_1 - 1)$, where lcm stands for the least common multiple. It then chooses an integer g such that $gcd(L(g^\lambda \mod N^2), N) = 1$, where gcd stands for the greatest common divisor, $g \in \mathbb{Z}_N^*$, and $L(x) = \frac{x-1}{N}$. The public key and private key are respectively $\{N, g\}$ and $\{\lambda\}$.
- **Encryption:** Let $m \in \mathbb{Z}_N^*$ be a plaintext and $r \in \mathbb{Z}_N^*$ be a random number. The ciphertext of m is computed by

$$E(m) = g^m \cdot r^N \mod N^2, \tag{3}$$

  where $E()$ denotes the encryption operation using public key $\{N, g\}$.
- **Decryption:** For the ciphertext $E(m)$, the corresponding plaintext can be computed by

$$D(E(m)) = \frac{L(E(m)^\lambda \mod N^2)}{L(g^\lambda \mod N^2)} \mod N, \tag{4}$$

  where $D()$ denotes the decryption operation using private key $\{\lambda\}$.

- **Homomorphic :** The Paillier cryptosystem is additively homomorphic as it satisfies the following conditions: Given $\{m_1, m_2\} \in \mathbb{Z}_N^*$, we have:

$$E(m_1) \cdot E(m_2) = E(m_1 + m_2), \tag{5}$$

  Furthermore, given $E(m)$ and a constant c, $E(c \cdot m)$ can be computed by:

$$E(c \cdot m) = E(m)^c. \tag{6}$$

3.3 Clustering Model

We consider a clustering problem consisting of n participants $a_1, ..., a_n$. Each participant a_i holds its own private information $\mathbf{a_i}$, a q-dimensional vector describing its features or activities. As shown in Fig. 1, there also exists a data analyst A who is responsible for grouping those participants with similar properties/activities into one cluster. However, due to privacy concerns, the data analyst cannot access the private information of any participant.

Our goal is to build mutual privacy protection between the data analyst A and the participants $a_1, .., a_n$ when computing the k centers of the private data $\mathbf{a_1}, \mathbf{a_2}, \cdots, \mathbf{a_n}$. Neither of A and the participants should deduce any private information about the other side except those being published as a result of the k-means clustering scheme. In other words, the data analyst A cannot learn any private information owned by the participants since a compromised A may potentially sell the data to others. Similarly, malicious participants should learn nothing about the data analyst A's data (the centers) and should not be able to disrupt A's computation.

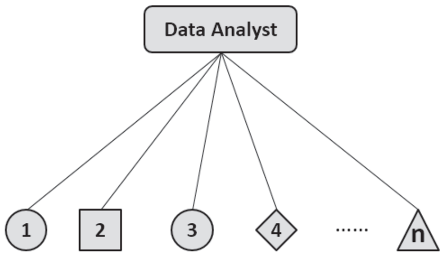


Fig. 1. Clustering Model

We assume that:

- any pair of the participants a_i and a_j share a unique pairwise key;
- the data analyst A shares a unique pairwise key with each participant a_i;
- the data analyst A generates a public and private key pair (PU, PR) that can be used for additive homomorphic encryption, and distributes its public key PU to all the participants.

Messages from each participant a_i to A need to be encrypted with PU. Let m denote the plaintext, and m' the corresponding ciphertext. We have $m' = E(PU, m)$ and $m = D(PR, m')$. Due to homomorphic property, we have

$$E(PU, m_1) \cdot E(PU, m_2) = E(PU, m_1 + m_2) \tag{7}$$

4 Privacy Preserving K-Means Clustering

In this section, we present our privacy preserving k-means clustering algorithm. As mentioned earlier, there exist two steps within each iteration: assigning each participant to its closest center, and computing the new center of each cluster in a secure way.

4.1 Stage 1: Assign Participants to Their Nearest Centers

This step assigns participants to their nearest centers within an iteration. The cluster centers $(\mathbf{u_1}, ..., \mathbf{u_k})$ are initialized and updated by the data analyst A.

Let D_{ij} be the distance between the i-th participant a_i and the center of the j-th cluster U_j, i.e.,

$$D_{ij} = (\mathbf{a}_i - \mathbf{u}_j)^T(\mathbf{a}_i - \mathbf{u}_j)$$

Now consider a_i and its distances to the clusters U_j and $U_{j'}$. We have

$$\begin{aligned} & D_{ij} - D_{ij'} \\ = \; & (\mathbf{a}_i - \mathbf{u}_j)^T(\mathbf{a}_i - \mathbf{u}_j) - (\mathbf{a}_i - \mathbf{u}_{j'})^T(\mathbf{a}_i - \mathbf{u}_{j'}) \\ = \; & \mathbf{a}_i^T\mathbf{a}_i - 2\mathbf{a}_i^T\mathbf{u}_j + \mathbf{u}_j^T\mathbf{u}_j - (\mathbf{a}_i^T\mathbf{a}_i - 2\mathbf{a}_i^T\mathbf{u}_{j'} + \mathbf{u}_{j'}^T\mathbf{u}_{j'}) \\ = \; & \mathbf{u}_j^T\mathbf{u}_j - \mathbf{u}_{j'}^T\mathbf{u}_{j'} - 2\mathbf{a}_i^T(\mathbf{u}_j - \mathbf{u}_{j'}) \end{aligned} \tag{8}$$

Remark 1: If A can send $\mathbf{u}_j^T\mathbf{u}_j - \mathbf{u}_{j'}^T\mathbf{u}_{j'}$ and $\mathbf{u}_j - \mathbf{u}_{j'}$ to a_i, then a_i can calculate $D_{ij} - D_{ij'}$ according to (8). If $D_{ij} - D_{ij'} < 0$, a_i is closer to $\mathbf{u}_j$; otherwise, a_i is closer to $\mathbf{u}_{j'}$. This process can be repeated for $k-1$ times for a_i to identify the closest cluster center among all the k clusters.

However, if the data analyst A sends all the $\{\mathbf{u}_j - \mathbf{u}_{j'}\}$'s directly to a participant a_i, then a_i can collect $\mathbf{u_1} - \mathbf{u_2}$, $\mathbf{u_2} - \mathbf{u_3}$, $\cdots$, $\mathbf{u_{k-1}} - \mathbf{u_k}$, and thus the cluster centers can be leaked. Therefore, A should send randomly perturbed values of $\{\mathbf{u}_j - \mathbf{u}_{j'}\}$, as shown below, where the $\rho_{(i,j)} > 0$ values are random numbers:

$$\begin{cases} \rho_{(i,1)}[(\mathbf{u_1}^T\mathbf{u_1}) - (\mathbf{u_2}^T\mathbf{u_2})], \rho_{(i,1)}(\mathbf{u_1} - \mathbf{u_2}) \\ \rho_{(i,2)}[(\mathbf{u_1}^T\mathbf{u_1}) - (\mathbf{u_3}^T\mathbf{u_3})], \rho_{(i,2)}(\mathbf{u_1} - \mathbf{u_3}) \\ \cdots, \\ \rho_{(i,k-1)}[(\mathbf{u_1}^T\mathbf{u_1}) - (\mathbf{u_k}^T\mathbf{u_k})], \rho_{(i,k-1)}(\mathbf{u_1} - \mathbf{u_k}) \\ \rho_{(i,k)}[(\mathbf{u_2}^T\mathbf{u_2}) - (\mathbf{u_3}^T\mathbf{u_3})], \rho_{(i,k)}(\mathbf{u_2} - \mathbf{u_3}) \\ \rho_{(i,k+1)}[(\mathbf{u_2}^T\mathbf{u_2}) - (\mathbf{u_4}^T\mathbf{u_4})], \rho_{(i,k+1)}(\mathbf{u_2} - \mathbf{u_4}) \\ \cdots, \\ \rho_{(i,2k-3)}[(\mathbf{u_2}^T\mathbf{u_2}) - (\mathbf{u_k}^T\mathbf{u_k})], \rho_{(i,k+1)}(\mathbf{u_2} - \mathbf{u_k}) \\ \cdots, \\ \cdots, \\ \rho_{(i,\frac{k(k-1)}{2})}[(\mathbf{u_{k-1}}^T\mathbf{u_{k-1}}) - (\mathbf{u_k}^T\mathbf{u_k})], \\ \qquad \rho_{(i,\frac{k(k-1)}{2})}(\mathbf{u_{k-1}} - \boldsymbol{u_k}) \end{cases} \tag{9}$$

Remark 2: Since $\rho_{(i,j)} > 0$, the randomized results have no influence on determining the sign of $D_{ij} - D_{ij'}$, as well as the closeness between the participant a_i and the clusters.

After obtaining this information, each participant a_i can identify the nearest center based on its own private data $\mathbf{a_i}$. The procedure is stated as follows. The participant a_i first computes the value $S_{(i,1)} = \rho_{(i,1)}[(\mathbf{u_1}^T\mathbf{u_1}) - (\mathbf{u_2}^T\mathbf{u_2})] - 2\rho_{(i,1)}\mathbf{a_i}^T[(\mathbf{u_1} - \mathbf{u_2})]$, which is related to the received information $\rho_{(i,1)}[(\mathbf{u_1}^T\mathbf{u_1}) - (\mathbf{u_2}^T\mathbf{u_2})]$ and $\rho_{(i,1)}(\mathbf{u_1} - \mathbf{u_2})$. If $S_{(i,1)} < 0$, a_i is closer to $\mathbf{u}_1$, and it can choose $\rho_{(i,2)}[(\mathbf{u_1}^T\mathbf{u_1}) - (\mathbf{u_3}^T\mathbf{u_3})]$ and $\rho_{(i,2)}(\mathbf{u_1} - \mathbf{u_3})$ to calculate the value of $S_{(i,2)} = \rho_{(i,2)}[(\mathbf{u_1}^T\mathbf{u_1}) - (\mathbf{u_3}^T\mathbf{u_3})] - 2\rho_{(i,2)}\mathbf{a_i}^T[(\mathbf{u_1} - \mathbf{u_3})]$; otherwise, a_i is closer to $\mathbf{u}_2$, and it can compute the value $S_{(i,k)} = \rho_{(i,k)}[(\mathbf{u_2}^T\mathbf{u_2}) - (\mathbf{u_3}^T\mathbf{u_3})] - 2\rho_{(i,k)}\mathbf{a_i}^T[(\mathbf{u_2} - \mathbf{u_3})]$ based on the received information $\rho_{(i,k)}[(\mathbf{u_2}^T\mathbf{u_2}) - (\mathbf{u_3}^T\mathbf{u_3})]$ and $\rho_{(i,k)}(\mathbf{u_2} - \mathbf{u_3})$. This process repeats until the nearest center is figured out. Then a_i informs the data analyst A which cluster (the one with the nearest center) it belongs to. Note that when this procedure terminates, A only knows which cluster a_i belongs to; it has no knowledge about a_i's private information $\mathbf{a_i}$.

The above process is summarized in Algorithm 1.

Note that each a_i only needs k rows of the information included in (9) to identify its nearest cluster. But we choose to let the data analyst A deliver all information included in (9) to each participant a_i for privacy protection as an interactive procedure asking only the required cluster center information may disclose the privacy of both $\mathbf{a_i}$ and the cluster centers.

4.2 Stage 2: Update the Cluster Centers

After each participant identifies its closest cluster with Algorithm 1 and informs the data analyst A, A should re-evaluate the cluster center $\mathbf{u_j}$ for each cluster U_j. Assume that there are n_j participants assigned to cluster U_j; then $n_1 + n_2 + \cdots + n_k = n$. After Stage 1, A can get the cluster membership list shown in Table 2, where $s_{(j,l)} \in \{a_1, ..., a_n\}$ is a participant indicating that this participant is the l-th participant in the j-th cluster, where $1 \leq l \leq n_k$ and $1 \leq j \leq k$.

Next, we present an additive homomorphic encryption scheme to re-compute the cluster centers, in order to ensure that only the data analyst knows the intermediate cluster centers, and the participants are kept blind to such private information. While in

Algorithm 1 Identifying the nearest cluster for a participant a_i.

```
1: Initialize: There are $n$ participants and one data analyst $A$. Each participant $a_i$ owns a $q$-dimensional private data $\mathbf{a_i} = (a_{i1}, \cdots, a_{iq})$. There are $k$ cluster centers $\{\mathbf{u_1}, \cdots, \mathbf{u_k}\}$, with their initial values randomly determined by $A$.
2: $\tau = 1$
3: for $j = 1$ to $k-1$
4:     $\gamma = j+1$;
5:     for $\iota = \gamma$ to $k$
6:         $A$ sends to $a_i$: $\rho_{(i,\tau)}[\mathbf{u_j}^T\mathbf{u_j} - \mathbf{u}_\iota{}^T\mathbf{u}_\iota]$ and $\rho_{(i,\tau)}(\mathbf{u_j} - \mathbf{u}_\iota)$;
7:         $\tau = \tau + 1$
8:     endfor
9: endfor
10: $a_i$ identifies the nearest center based on the received data and its own private data as follows:
11: $a_i$ first compares its distance to $\mathbf{u_1}$ and $\mathbf{u_2}$ by computing $S_{(i,1)} = \rho_{(i,1)}[(\mathbf{u_1}^T\mathbf{u_1}) - (\mathbf{u_2}^T\mathbf{u_2})] - 2\rho_{(i,1)}\mathbf{a_i}^T[(\mathbf{u_1} - \mathbf{u_2})]$;
12: If $S_{(i,1)} < 0$, $a_i$ is closer to $\mathbf{u_1}$, and then $a_i$ compares the distances to $\mathbf{u_1}$ and $\mathbf{u_3}$ using the related received data and its own private data as in Step 11;
13: else if $a_i$ is closer to $\mathbf{u_2}$, then $a_i$ compares its distances to $\mathbf{u_2}$ and $\mathbf{u_3}$ as in Step 11;
14: EndIf
15: $a_i$ repeats Steps 11 to 14 to identify the nearest cluster.
16: $a_i$ notifies $A$ which cluster it belongs to.
```

TABLE 2
participant Label

participants / Cluster	participant Label
U_1 (n_1 participants)	$s_{(1,1)}, s_{(1,2)}, \cdots, s_{(1,n_1)}$
U_2 (n_2 participants)	$s_{(2,1)}, s_{(2,2)}, \cdots, s_{(2,n_2)}$
$\vdots$	$\vdots$
U_k (n_k participants)	$s_{(k,1)}, s_{(k,2)}, \cdots, s_{(k,n_k)}$

many existing privacy-preserving k-means clustering algorithms such as [7], [15] and [24] this information cannot be protected.

According to (2), a cluster center is the mean of the private data of the participants that belong to that cluster. From Table 2, one can see that the data analyst knows n_j, the number of participants in cluster U_j. Therefore we need to compute the sum of the private data of the participants belonging to U_j. Consider the cluster U_j and its member participants $s_{(j,1)}, s_{(j,2)}, \cdots, s_{(j,n_j)}$. First, the data analyst A randomly generates n_j q-dimensional vectors $\mathbf{V}_{(j,1)}, \mathbf{V}_{(j,2)}, \cdots, \mathbf{V}_{(j,n_j)}$ satisfying

$$\mathbf{V}_{(j,1)} + \mathbf{V}_{(j,2)} + \cdots + \mathbf{V}_{(j,n_j)} = \mathbf{0} \tag{10}$$

Then A securely sends $\{\mathbf{V}_{(j,l)}, +\}$ to each participant $s_{(j,l)} \in U_j$ encrypted with their pairwise key, where "$+$" denotes that $s_{(j,l)}$ belongs to the cluster currently under consideration. Meanwhile, A generates another $(n - n_j)$ q-dimensional vectors $\mathbf{R}_{(j',l)}$ for each $s_{(j',l)} \notin U_j$ satisfying:

$$\begin{aligned}\mathbf{R}_{(1,1)} + \mathbf{R}_{(1,2)} + \cdots + \mathbf{R}_{(1,n_1)} + \cdots \\ +\mathbf{R}_{(j-1,1)} + \mathbf{R}_{(j-1,2)} + \cdots + \mathbf{R}_{(j-1,n_{j-1})} \\ +\mathbf{R}_{(j+1,1)} + \mathbf{R}_{(j+1,2)} + \cdots + \mathbf{R}_{(j+1,n_{j+1})} \\ + \cdots + \mathbf{R}_{(k,1)} + \cdots + \mathbf{R}_{(k,n_k)} = \mathbf{0}\end{aligned} \tag{11}$$

Then A securely sends $\{\mathbf{R}_{(j',l)}, -\}$ to each $s_{(j',l)} \notin U_j$.

Next, all of the n participants should compute their encrypted data. Recall that each participant owns a q-dimensional private vector. For each $s_{(j,l)} \in U_j$, $s_{(j,l)}$ should encrypt the received vector $\mathbf{V}_{(j,l)}$ and its private vector $\mathbf{a}_{(j,l)}$ after receiving the message $\{\mathbf{V}_{(j,l)}, +\}$ with a label "+". The encryption is computed with the public key PU of the homomorphic encryption system. Then $s_{(j,l)}$ obtains:

$$\boldsymbol{Y}_{j,l} = E(PU, \mathbf{a}_{(j,l)} + \mathbf{V}_{(j,l)}). \tag{12}$$

While for each $s_{(j',l)} \notin U_j$, $s_{(j',l)}$ only encrypts the received vector upon receiving the message $\{\mathbf{R}_{(j',l)}, -\}$ with a label "$-$". That is, $s_{(j',l)}$ gets:

$$\boldsymbol{Y}_{j',l} = E(PU, \mathbf{R}_{(j',l)}). \tag{13}$$

After completing the encryption operation, each participant should share part of its ciphertext with others. This can be done as follows. Each participant $s_{(p,q)}$ first randomly divides its ciphertext $\boldsymbol{Y}_{p,q}$ (computed either from (12) or from (13)) into m components $\boldsymbol{Y}^1_{p,q}$, $\boldsymbol{Y}^2_{p,q}$, $\cdots$, $\boldsymbol{Y}^m_{p,q}$, where $1 \le m \le n$, satisfying

$$\boldsymbol{Y}^1_{p,q} \cdot \boldsymbol{Y}^2_{p,q} \cdot \cdots \cdot \boldsymbol{Y}^m_{p,q} = \boldsymbol{Y}_{p,q}.$$

Note that $s_{(p,q)}$ must keep one of the m components to itself. Then $s_{(p,q)}$ randomly selects $m-1$ participants in the network and sends one component to each chosen participant through the secure channel protected by their pairwise keys.

Note that each participant should complete the above slicing and distributing process on its ciphertext computed from (12) or (13). Meanwhile, each participant may receive components (slices of ciphertexts) from other participants. Then each participant applies the homomorphic operation on all the received cipher components as well as the slice kept to itself by multiplying them together to get $\mathbf{r}_{(p,q)}$, which should be sent to the data analyst A after the computation is over.

Lastly, A multiplies all the received data, and obtains the following result according to (7), (10), and (11):

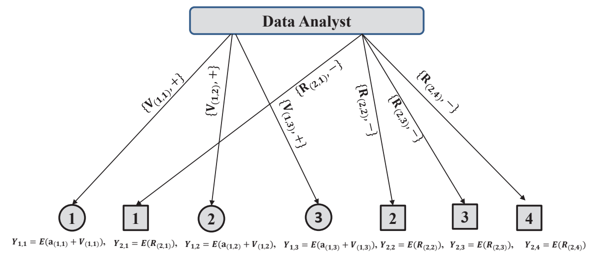
$$\begin{aligned}\boldsymbol{Y}_{j,1} \cdot \boldsymbol{Y}_{j,2} \cdots \boldsymbol{Y}_{j,n_j} \cdots \boldsymbol{Y}_{j',l} \cdots \\ = E(PU, \mathbf{a}_{(j,1)} + \mathbf{a}_{(j,2)} + \cdots + \mathbf{a}_{(j,n_j)} + \\ \mathbf{V}_{(j,1)} + \mathbf{V}_{(j,2)} + \cdots + \mathbf{V}_{(j,n_j)} + \cdots + \mathbf{R}_{(j',l)} + \cdots) \\ = E(PU, \mathbf{a}_{(j,1)} + \mathbf{a}_{(j,2)} + \cdots + \mathbf{a}_{(j,n_j)})\end{aligned} \tag{14}$$

Then A gets $\mathbf{a}_{(j,1)} + \mathbf{a}_{(j,2)} + \cdots + \mathbf{a}_{(j,n_j)}$ by decrypting with its private key PR, and finally computes the cluster center $\mathbf{u_j}$ by dividing it by n_j. The whole procedure is summarized by Algorithm 2.

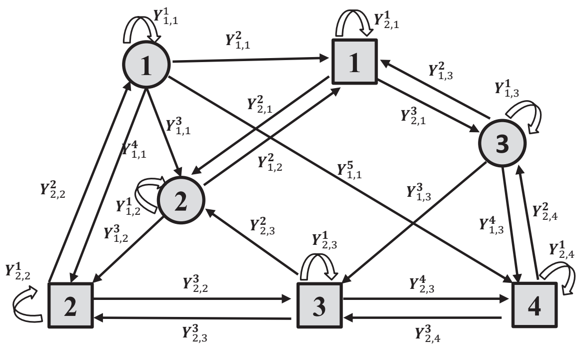
An example detailing the procedure is given in Fig. 2, which illustrates how to compute the center of the first cluster whose members include the circled participants $1, 2, 3$.

Following the above process, the data analyst can compute the new center for each cluster. The advantage of our algorithm is the ability to ensure mutual privacy preservation.

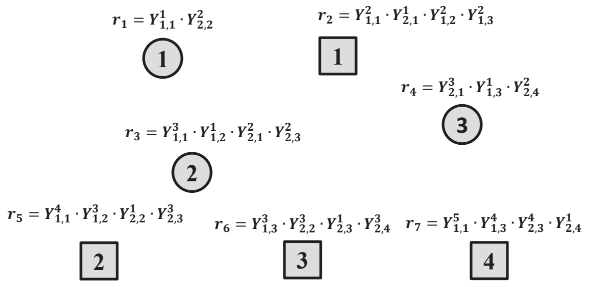
- The data analyst can obtain the sum of the data in a cluster without accessing the private information of each participant.


Data Analyst
$\{V_{(1,1)},+\}$
$\{V_{(1,2)},+\}$
$\{V_{(1,3)},+\}$
$\{R_{(2,1)},-\}$
$\{R_{(2,2)},-\}$
$\{R_{(2,3)},-\}$
$\{R_{(2,4)},-\}$
$Y_{1,1} = E(a_{(1,1)} + V_{(1,1)})$, $Y_{2,1} = E(R_{(2,1)})$, $Y_{1,2} = E(a_{(1,2)} + V_{(1,2)})$, $Y_{1,3} = E(a_{(1,3)} + V_{(1,3)})$, $Y_{2,2} = E(R_{(2,2)})$, $Y_{2,3} = E(R_{(2,3)})$, $Y_{2,4} = E(R_{(2,4)})$

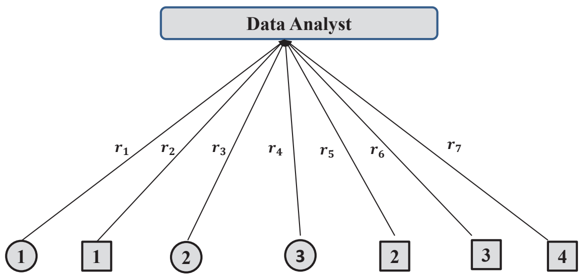
(a) The data analyst sends random data to each participant, satisfying $V_{(1,1)} + V_{(1,2)} + V_{(1,3)} = 0$, $R_{(2,1)} + R_{(2,2)} + R_{(2,3)} + R_{(2,4)} = 0$.



(b) Each participant slices its encrypted data, keeps one piece to itself, and sends the remaining to randomly chosen participants.


$r_1 = Y^1_{1,1} \cdot Y^2_{2,2}$
$r_2 = Y^2_{1,1} \cdot Y^1_{2,1} \cdot Y^2_{1,2} \cdot Y^2_{1,3}$
$r_4 = Y^3_{2,1} \cdot Y^1_{1,3} \cdot Y^2_{2,4}$
$r_3 = Y^3_{1,1} \cdot Y^1_{1,2} \cdot Y^2_{2,1} \cdot Y^2_{2,3}$
$r_5 = Y^4_{1,1} \cdot Y^3_{1,2} \cdot Y^1_{2,2} \cdot Y^3_{2,3}$
$r_6 = Y^3_{1,3} \cdot Y^3_{2,2} \cdot Y^1_{2,3} \cdot Y^3_{2,4}$
$r_7 = Y^5_{1,1} \cdot Y^4_{1,3} \cdot Y^4_{2,3} \cdot Y^1_{2,4}$

(c) Each the participant multiplies its received data and the component held by itself.



(d) All the participants send their results from Fig. 2(c) to the data analyst, who can compute the new cluster centers.

Fig. 2. The process of computing the new cluster centers within one iteration.

- The participants know nothing about each other. Particularly, they do not know who else are in the same cluster.
- The participants know nothing about the intermediate cluster centers. This information is protected by the random values (the ρ values) known only by the data analyst.

4.3 Stopping Criterion

Stage 1 and Stage 2 should be repeated iteratively until little or no change occurs in the clustering process. At the end of each iteration, the data analyst needs to compare the newly obtained cluster centers with those from the previous iteration. If they are "close enough" according to an application-specific parameter (e.g., the total distance change of the clusters and their corresponding participants is no more than a threshold between two iterations), the iteration process can terminate.

Algorithm 2 Computing the new center of cluster U_j.

```
1: Initial State: Cluster $U_j$ has the member participants $s_{(j,1)}$, $s_{(j,2)}, \cdots, s_{(j,n_j)}$, with $\mathbf{a}_{(j,l)}$ being the private data of member $s_{(j,l)}$.
2: The data analyst $A$ generates the random $\boldsymbol{V}$ values according to (10) and the random $\boldsymbol{R}$ values according to (11).
3: The data analyst $A$ sends $(\boldsymbol{V}_{(j,l)}, +)$ to each $s_{(j,l)\in U_j}$, and sends $\boldsymbol{R}_{(j',l)}$ to each participant $s_{(j',l)} \notin U_j$.
4: Each member computes its encrypted data with (12) if it belongs to cluster $U_j$, or with (13) otherwise.
5: Each member slices the ciphertext into $m$ components, and sends $m-1$ components to other participants randomly selected in the network.
6: Each participant multiplies the encrypted component it has kept for itself and all the received cipher components to compute $\mathbf{r}$.
7: Each participant sends $\mathbf{r}$ to the data analyst $A$.
8: The data analyst multiplies all the received data to get $\mathbf{a}_{(j,1)} + \mathbf{a}_{(j,2)} + \cdots + \mathbf{a}_{(j,n_j)}$ via decryption.
9: The new center of the cluster $U_j$ can be obtained by $\mathbf{a}_{(j,1)} + \mathbf{a}_{(j,2)} + \cdots + \mathbf{a}_{(j,n_j)}/n_j$.
```

5 Privacy and Efficiency Analysis

In this section, we discuss the ability of our scheme to ensure privacy preservation against potential passive and active attacks. Before delving into details, we first define our goals in privacy protection. In our consideration, each participant should not get the following information:

- cluster centers;
- other participants in the same cluster;
- the private data of the other participants.

The data analyst A knows where are the cluster centers, but it cannot access any private information of any participant. In other words, A knows that the participant $s_{(j,l)}$ belongs to the j-th cluster, but it does not know the private data $\mathbf{a}_{(\mathbf{j},1)}$ of $s_{(j,l)}$ nor the distance from $s_{(j,l)}$ to the associated cluster center $\mathbf{u_j}$.

5.1 Privacy Analysis on the Stage of Assigning Participants to Their Nearest Clusters

In Stage 1, each participant only notifies A which cluster is the closest; therefore the data analyst knows nothing about the private information of the participants. Now considering the worst case when $n-1$ participants collude with each other to detect the cluster centers. Without loss of generality, we assume that these $n-1$ participants are denoted by $a_1, a_2, \cdots, a_{n-1}$ and they combine their information to compute the cluster center $\mathbf{u}_1$. All these colluding $n-1$ participants can construct the following equations using the received data from the data analyst A and their own data:

$$\begin{cases} \rho_{(1,1)}(\mathbf{u_1}^T\mathbf{u_1} - \mathbf{u_2}^T\mathbf{u_2}) - 2\rho_{(1,1)}\mathbf{a_1}^T(\mathbf{u_1} - \mathbf{u_2}) = S_{(1,1)} \\ \rho_{(2,1)}(\mathbf{u_1}^T\mathbf{u_1} - \mathbf{u_2}^T\mathbf{u_2}) - 2\rho_{(2,1)}\mathbf{a_2}^T(\mathbf{u_1} - \mathbf{u_2}) = S_{(2,1)} \\ \rho_{(3,1)}(\mathbf{u_1}^T\mathbf{u_1} - \mathbf{u_2}^T\mathbf{u_2}) - 2\rho_{(3,1)}\mathbf{a_3}^T(\mathbf{u_1} - \mathbf{u_2}) = S_{(3,1)} \\ \cdots, \\ \rho_{(n-1,1)}(\mathbf{u_1}^T\mathbf{u_1} - \mathbf{u_2}^T\mathbf{u_2}) - 2\rho_{(n-1,1)}\mathbf{a_{n-1}}^T(\mathbf{u_1} - \mathbf{u_2}) \\ = S_{(n-1,1)} \\ \rho_{(1,2)}(\mathbf{u_1}^T\mathbf{u_1} - \mathbf{u_3}^T\mathbf{u_3}) - 2\rho_{(1,2)}\mathbf{a_1}^T(\mathbf{u_1} - \mathbf{u_3}) = S_{(1,2)} \\ \rho_{(2,2)}(\mathbf{u_1}^T\mathbf{u_1} - \mathbf{u_3}^T\mathbf{u_3}) - 2\rho_{(2,2)}\mathbf{a_2}^T(\mathbf{u_1} - \mathbf{u_3}) = S_{(2,2)} \\ \rho_{(3,2)}(\mathbf{u_1}^T\mathbf{u_1} - \mathbf{u_3}^T\mathbf{u_3}) - 2\rho_{(3,2)}\mathbf{a_3}^T(\mathbf{u_1} - \mathbf{u_3}) = S_{(3,2)} \\ \cdots, \\ \rho_{(n-1,2)}(\mathbf{u_1}^T\mathbf{u_1} - \mathbf{u_3}^T\mathbf{u_3}) - 2\rho_{(n-1,2)}\mathbf{a_{n-1}}^T(\mathbf{u_1} - \mathbf{u_3}) = S_{(n-1,2)} \\ \cdots, \\ \rho_{(1,k-1)}(\mathbf{u_1}^T\mathbf{u_1} - \mathbf{u_k}^T\mathbf{u_k}) - 2\rho_{(1,k-1)}\mathbf{a_1}^T(\mathbf{u_1} - \mathbf{u_k}) = S_{(1,k-1)} \\ \rho_{(2,k-1)}(\mathbf{u_1}^T\mathbf{u_1} - \mathbf{u_k}^T\mathbf{u_k}) - 2\rho_{(2,k-1)}\mathbf{a_2}^T(\mathbf{u_1} - \mathbf{u_k}) = S_{(2,k-1)} \\ \rho_{(3,k-1)}(\mathbf{u_1}^T\mathbf{u_1} - \mathbf{u_k}^T\mathbf{u_k}) - 2\rho_{(3,k-1)}\mathbf{a_3}^T(\mathbf{u_1} - \mathbf{u_k}) = S_{(3,k-1)} \\ \cdots, \\ \rho_{(n-1,k-1)}(\mathbf{u_1}^T\mathbf{u_1} - \mathbf{u_k}^T\mathbf{u_k}) - 2\rho_{(n-1,k-1)}\mathbf{a_{n-1}}^T(\mathbf{u_1} - \mathbf{u_k}) \\ = S_{(n-1,k-1)} \end{cases} \tag{15}$$

For the participants, the coefficients $\{\rho_{(1,1)}, \rho_{(2,1)}, ..., \rho_{(n-1,1)}, \rho_{(1,2)}, \rho_{(2,2)}, ..., \rho_{(n-1,2)}, ..., \rho_{(1,k-1)}, \rho_{(2,k-1)}, ..., \rho_{(n-1,k-1)}\}$ and $\{\mathbf{u_1}, \mathbf{u_2}, \mathbf{u_3}, ..., \mathbf{u_k}\}$ are unknowns in (15), which come from the data analyst A, and the cluster center is a q-dimensional vector. Thus there are $(k-1)(n-1)+kq$ unknown parameters in (15), but there exist only $(k-1)(n-1)$ equations; therefore the participants cannot figure out the cluster $\mathbf{u_1}$. Similarly, the colluding participants cannot find out other cluster centers $\{\mathbf{u_2}, \mathbf{u_3}..., \mathbf{u_k}\}$. Actually, when all the available information from the $n-1$ colluding participants are combined, $(n-1)\cdot\frac{k(k-1)}{2}$ equations can be obtained but there are $(n-1)\cdot\frac{k(k-1)}{2}+kq$ number of unknowns, which again proves that it is impossible for the participants to recover the cluster centers $\{\mathbf{u_1}, \mathbf{u_2}, \mathbf{u_3}..., \mathbf{u_k}\}$. Thus we conclude that our scheme can resist the collusion attacks launched by any number of participants without leaking any information about the cluster centers.

Cost Analysis on Stage 1:

We next analyze the communication and computation overheads of our algorithm in Stage 1. It should be noted that both communication and computation overheads of the k-means clustering algorithm depend on the size of the data set. The number of iterations being conducted to meet the stopping criterion depends on the dataset size and the initial cluster centers. A clustering algorithm targeting real world applications should not incur too much communication and computation overheads. A simple analysis on the complexity of Algorithm 1 reveals the following results: In Stage 1, each participant needs to do multiplication at the complexity of $O(q(k-1))$ to find the nearest cluster. For each participant, the communication complexity is $O(kq(k-1)+1)$. The total communication complexity is $O(kq(k-1)n+n))$.

5.2 Privacy Analysis on the Stage of Computing New Cluster Centers

In Algorithm 2, a public-key based additive homomorphic encryption scheme is adopted to compute the cluster centers. The encrypted data $Y_i = E(PU, \mathbf{a}_i + \mathbf{V}_i)$ or $Y_i = E(PU, \mathbf{R}_i)$ held by the participant a_i is semantically secure, since only the data analyst A knows the private key PR. As Y_i is sliced, shared, and finally sent to A, the process is secure for both a_i and A. The data Y_i is mixed with a random vector $\mathbf{V}_i$ only known by A and secured by the pubic-key cryptosystem; thus Y_i is secured against other participants. As the data that A obtains has already been mixed and operated by the participatory participants, A knows nothing about the private data of a_i.

At each round of Algorithm 2, the data analyst A sends a random vector to each participant. Then a participant a_i slices its encrypted data Y_i into m components and sends $m-1$ components to $m-1$ randomly selected participants. Participant a_i reserves the remaining component to itself. As a result, no one knows which participants are in the same cluster, and no one knows who are selected to receive the slices. If an attacker wants to acquire the private data held by a_i, it must break all the $m-1$ outgoing slices and other incoming slices. Since $1 \le m \le n$, the maximum number of incoming slices for a participant is $n-1$, and the maximum number of outgoing slices is $n-1$; and the private data of a participant may be disclosed only if an attacker has the ability to break all the $2n-2$ slices in this case. Let p denote the possibility that a single slice is leaked to an attacker. Then the possibility that the private slices held by the participant a_i may be disclosed is:

$$P_i = p^{m-1}.$$

Cost Analysis on Stage 2:

In Stage 2, some computation is done by the data analyst, while the participants need to perform very little calculation. Consider the computational complexity of Algorithm 2. Each participant does one public-key encryption in step 3, then takes $O(n*q)$ time for each of step 4 and 5. The data analyst conducts the public-key decryption and takes $O(n*q)$ time on the multiplication operation in step 7, and takes $O(q)$ time in step 8.

Now we consider the communication complexity of Algorithm 2. We observe that step 2 takes $O(n*q)$ time, step 4 takes $O(n^2*q)$ time, and step 6 takes $O(n*q)$ time. To update all the cluster centers, the data analyst needs to execute the corresponding steps in Algorithm 2 for k times. Therefore the communication complexity for updating all cluster centers is $O(n^2qk)$.

5.3 Security Against Collusion Attacks

Our algorithm makes no assumption on non-colluding participants. In the following, we will show that our algorithm can resist against collusion attacks. We assume that a semi-honest data analyst and participants follow the proposed protocol.

5.3.1 Collusion Between the Data Analyst and Participants

Our method ensures that a participant can hide its private data safely, even if the data analyst is an adversary. In Algorithm 2, the data analyst A receives $\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_n$ from the n participants. Although A can decrypt the data $\mathbf{r}_i$, no further information can be derived since $\mathbf{r}_i$ is the result of the sliced components mixed with other random data.

Consider the scenario when the data analyst A colludes with one or more participants. At maximum, there can be as many as $n-1$ participants colluding with A to deduce the private information of the remaining participant a_i. We assume a_i has sliced its encrypted data into m pieces, then $m-1$ of the m components will be taken by the $n-1$ colluded participants.

Although A knows the decryption key and the random vector held by a_i, A cannot get a_i's private information since one of the m slices is reserved by a_i. Thus the collusion alliance still cannot get $\mathbf{a}_i$, which illustrates that our algorithm is robust even in the worst case where there are as many as $n-1$ participants colluding with the data analyst. Therefore we conclude that our protocol can protect the privacy of each participant under the collusion of the data analyst and the participants.

5.3.2 Collusion Among the Participants

We have argued that the data analyst can hide the cluster centers with a random vector in our algorithm in Section 5.1. According to the analysis in Section 5.1, even if there are $n-1$ participants colluding together, they still cannot deduce any information about the cluster centers, because there exist $(n-1)\cdot\frac{k(k-1)}{2}+kq$ unknowns for the cluster centers while there are only $(n-1)\cdot\frac{k(k-1)}{2}$ equations. Thus we claim that our algorithm can securely protect the data analyst's privacy.

6 Experimental Study

We use three datasets for the experiments. The first one is a health dataset that includes systolic pressure and heart rate. This data set is collected from 20 elderly people with high blood pressure and surveying 70 healthy students. The clustering results can help the subjects perceive their health conditions in their community. The second dataset is about location [29], and users can figure out the population distribution in the vicinity from the results of our clustering algorithm. In the last experiment, we consider mobile users and select the *Human Activity Recognition on Smart-phones* [30] as the test dataset. This dataset contains mobile location data which describes the historical location of some mobile users belonging to a telecommunications operator. It records about 900 users' latitude and longitude data in several days. This dataset is important to reflect the spatial distribution of travel demands.

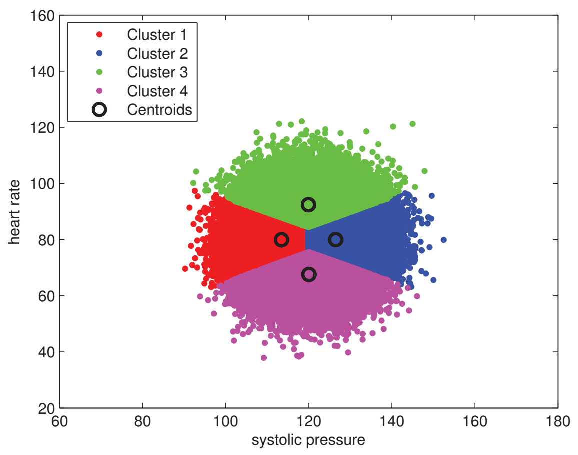


Fig. 3. Clustering results from our algorithm.

Fig. 3 shows our clustering results on the health data. If the clustering results indicate a very bad health condition in the neighborhood, then users may be alerted to start a healthy diet habit or look for health advisory, which can help promote a healthy life while preserving all users' privacy. Fig. 4 compares the cluster centers computed from our scheme and the classic k-means algorithm. The results show that our scheme can achieve almost the same accuracy while being able to preserve individual's privacy. Table 3 shows the accuracy and recall of our scheme compared with the clustering results which are computed from the original private data. One can conclude that our privacy preserving clustering scheme can compute cluster centers with the same accuracy.

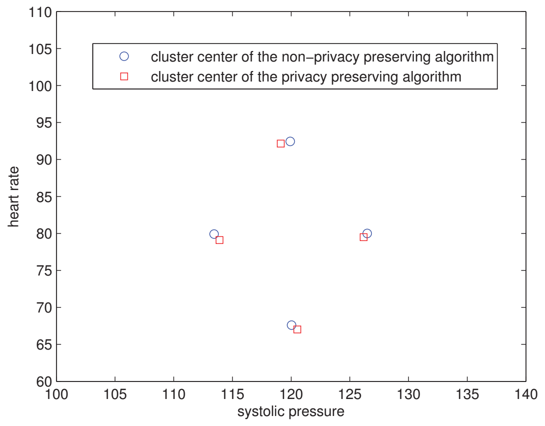


Fig. 4. Comparison on the cluster centers computed from our algorithm and the general k-means algorithm.

TABLE 3
Evaluation on the cluster assignment in our algorithm.

cluster	accuracy	recall
1	99.89%	99.81%
2	99.92%	99.87%
3	99.91%	99.98%
4	99.97%	99.90%

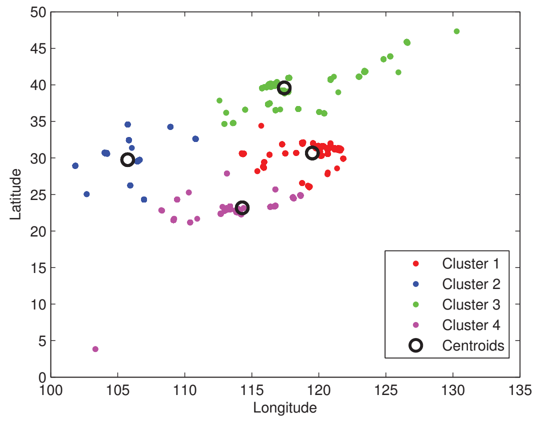


Fig. 5. Clustering distribution result without leaking each person's location.

Fig. 5 shows our clustering results on the location data. The results demonstrate that a user can be clustered with others in vicinity even though no location information is disclosed to each other. With our scheme, users can perceive the population distribution without leaking anyone's location information. Fig. 6 indicates that our scheme can compute cluster centers at the same accuracy as the non-privacy preserving algorithm. Table 4

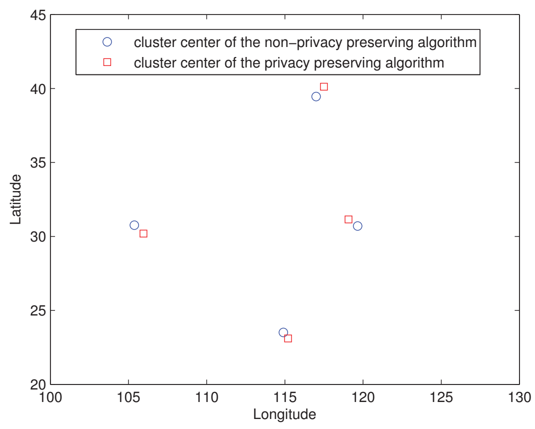


Fig. 6. Cluster center comparison of our scheme with the general k-means algorithm.

TABLE 4
Cluster assignment evaluation of our scheme.

cluster	accuracy	recall
1	99.51%	99.18%
2	97.1%	99.8%
3	99.67%	98.82%
4	99.39%	99.71%

indicates our scheme can get the same accurate clustering results without disclosing private location information.

In our last experiment, we select the *Human Activity Recognition on Smart-phones* [30] as our test dataset. This data was collected from a group of 30 volunteers within an age bracket of 19-48 years. Each person carries a smart-phone with an embedded accelerometer and a gyroscope. The data contains 3-axial linear acceleration and 3-axial angular velocity. Each of the 7352 samples contains 561 features. A certain part of the samples reflect one particular type of activity. Participants do not want to disclose their activities or location information, but want to play with others participating in similar activities. Next we conduct experiments to show that our solution allows the mobile users to recognize the others participating in the same activity without leaking the private activity information.

We perform privacy preserving clustering on the test data to group people's activities into several categories. Fig. 7 shows the clustering results from our scheme compared with the ground truth grouping.

One can conclude from Fig. 7 that our clustering solution can effectively group participants with similar activities in a secure way. The mobile users can easily find people who share the same interests without leaking anyone's individual private data.

Table 5 presents a simple comparison study over the three

!htb

TABLE 5
The comparisons among [7], [12], [19] and our scheme.

Properties	[7]	[12]	[19]	Our scheme
Collusion attack resistance	No	No	No	Yes
Individual's information protection	Yes	Yes	Yes	Yes
Cluster center protection	No	No	No	Yes
Intermediate result leak	Yes	Yes	Yes	No

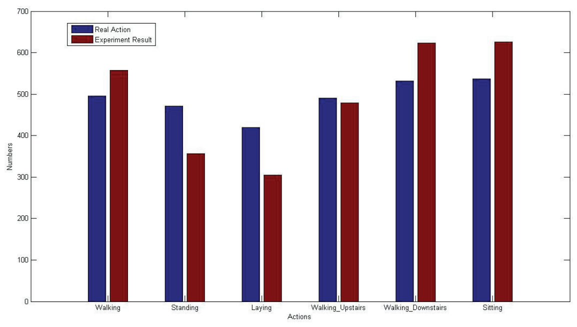


Fig. 7. Result comparison between our experiment result and users' ground truth action.

schemes proposed in [31], [12], and [19]. From Table 5, one can notice the significant advantages of our scheme compared to the other three schemes in terms of privacy preservation and collusion attack resistance.

7 Conclusion and Future Research

In this paper we propose an efficient privacy-preserving clustering scheme that ensures no leak of any intermediate result. Our scheme can securely compute the nearest cluster center for each participant without disclosing any cluster information to the participants. In addition, our scheme can update the cluster centers at each iteration without exposing any participants' information to the data analyst.

Our scheme can achieve more privacy preservation goals when compared to other existing works. No participant can obtain any private information of other participants or the cluster centers. Furthermore, the cluster centers can be computed without leaking the cluster label of each participant. And participants have no knowledge about the other participants in the same cluster. Through an extensive security analysis, we conclude that even in the existence of collusion participants, no private information of the remaining participants is released. Finally, we conclude that our scheme does not incur much communication and computation overheads on the participants.

In our future research, we shall consider the mutual privacy-protection of other clustering algorithms such as the Gaussian Mixture Modeling for social participatory sensing. On the other hand, we will investigate how to figure out the relative position of a participant in a community when the community model is available while protecting the privacy of both the participant and the community.

Acknowledgement

The authors would like to thank all the reviewers for their helpful comments. This research was partially supported by the National Science Foundation of the US under grants CNS-1318872, CCF-1442642 and IIS-1343976, and the National Natural Science Foundation of China under grants 61672119, 61672321, and 61373027.

References

[1] Y. Zheng, "Methodologies for cross-domain data fusion: An overview," *IEEE Transactions on Big Data*, vol. 1, no. 1, pp. 16–34, 2015.

[2] C. Hu, W. Li, X. Cheng, J. Yu, S. Wang, and R. Bie, "A secure and verifiable access control scheme for big data storage in clouds," *IEEE Transactions on Big Data*, 2017.

[3] L. Festinger, "A theory of social comparison processes," in *Human Relations*, vol. 7, 1954, pp. 117–140.

[4] G. Jagannathan and R. N. Wright, "Privacy-preserving distributed k-means clustering over arbitrarily partitioned data," in *Proceedings of the eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. ACM, 2005, pp. 593–599.

[5] Z. Erkin, A. Piva, S. Katzenbeisser, R. L. Lagendijk, J. Shokrollahi, G. Neven, and M. Barni, "Protection and retrieval of encrypted multimedia content: When cryptography meets signal processing," *EURASIP Journal on Information Security*, vol. 2007, pp. 1–20, 2007.

[6] W. Bi, M. Cai, M. Liu, and G. Li, "A big data clustering algorithm for mitigating the risk of customer churn," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 3, pp. 1270–1281, 2016.

[7] J. Vaidya and C. Clifton, "Privacy-preserving k-means clustering over vertically partitioned data," in *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2003, pp. 206–215.

[8] W. Du and M. J. Atallah, "Privacy-preserving cooperative statistical analysis," in *Proceedings 17th Annual Computer Security Applications Conference (ACSAC)*. IEEE, 2001, pp. 102–110.

[9] M. C. Doganay, T. B. Pedersen, Y. Saygin, E. Savaş, and A. Levi, "Distributed privacy preserving k-means clustering with additive secret sharing," in *Proceedings of the 2008 International Workshop on Privacy and Anonymity in Information Society*. ACM, 2008, pp. 3–11.

[10] P.-Y. Lin, "Distributed secret sharing approach with cheater prevention based on qr code," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 1, pp. 384–392, 2016.

[11] S. Samet, A. Miri, and L. Orozco-barbosa, "Privacy Preserving k-Means Clustering in Multi-Party Environment," in *Security and Cryptography*, 2007, pp. 381–385.

[12] X. Yi and Y. Zhang, "Equally contributory privacy-preserving k-means clustering over vertically partitioned data," *Information systems*, vol. 38, no. 1, pp. 97–107, 2013.

[13] Z. Lin and J. W. Jaromczyk, "Privacy preserving two-party k-means clustering over vertically partitioned dataset," in *2011 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2011, pp. 187–191.

[14] A. Inan, S. V. Kaya, Y. Saygın, E. Savaş, A. A. Hintoğlu, and A. Levi, "Privacy preserving clustering on horizontally partitioned data," *Data & Knowledge Engineering*, vol. 63, no. 3, pp. 646–666, 2007.

[15] S. Jha, L. Kruger, and P. McDaniel, "Privacy preserving clustering," in *Computer Security–ESORICS 2005*. Springer, 2005, pp. 397–417.

[16] T.-K. Yu, D. Lee, S.-M. Chang, and J. Zhan, "Multi-party k-means clustering with privacy consideration," in *2010 International Symposium on Parallel and Distributed Processing with Applications (ISPA)*. IEEE, 2010, pp. 200–207.

[17] P. Bunn and R. Ostrovsky, "Secure two-party k-means clustering," in *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 2007, pp. 486–497.

[18] F.-Y. Rao, B. K. Samanthula, E. Bertino, X. Yi, and D. Liu, "Privacy-preserving and outsourced multi-user k-means clustering," in *2015 IEEE Conference on Collaboration and Internet Computing (CIC)*. IEEE, 2015, pp. 80–89.

[19] D. Liu, E. Bertino, and X. Yi, "Privacy of outsourced k-means clustering," in *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*. ACM, 2014, pp. 123–134.

[20] K.-P. Lin, "Privacy-preserving kernel k-means clustering outsourcing with random transformation," *Knowledge and Information Systems*, pp. 1–24, 2016.

[21] S. Patel, V. Patel, and D. Jinwala, "Privacy preserving distributed k-means clustering in malicious model using zero knowledge proof," in *Distributed Computing and Internet Technology*. Springer, 2013, pp. 420–431.

[22] F. Meskine and S. N. Bahloul, "Privacy preserving k-means clustering: a survey research." *Int. Arab J. Inf. Technol.*, vol. 9, no. 2, pp. 194–200, 2012.

[23] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright, "A new privacy-preserving distributed k-clustering algorithm." in *SDM*. SIAM, 2006, pp. 494–498.

[24] G. Jagannathan, K. Pillaipakkamnatt, R. N. Wright, and D. Umano, "Communication-efficient privacy-preserving clustering," *Trans. Data Privacy*, vol. 3, no. 1, pp. 1–25, 2010.

[25] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, "Privacy-preserving distributed clustering," *EURASIP Journal on Information Security*, vol. 2013, no. 1, pp. 1–15, 2013.

[26] P. S. Bradley and U. M. Fayyad, "Refining initial points for k-means clustering." in *ICML*, vol. 98. Citeseer, 1998, pp. 91–99.

[27] C. Fontaine and F. Galand, "A survey of homomorphic encryption for nonspecialists," *EURASIP Journal on Information Security*, vol. 2007, no. 1, pp. 1–10, 2007.

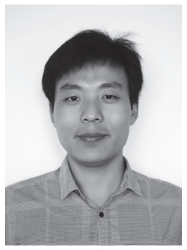
[28] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology-EUROCRYPT'99*. Springer, 1999, pp. 223–238.

[29] [Online]. Available: http://www.datatang.com/data/2991

[30] Http://archive.ics.uci.edu/ml/datasets/Human Activity Recognition Using Smartphones.

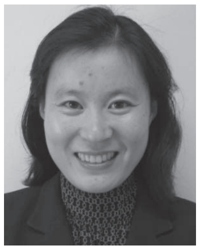
[31] J. Shi, R. Zhang, Y. Liu, and Y. Zhang, "Prisense: privacy-preserving data aggregation in people-centric urban sensing systems," in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–9.

Kai Xing is an Associate Professor at Department of Computer Science and Technology, University of Science and Technology of China. He received his M.S. degree and Ph.D. degree in Computer Science from the George Washington University in 2006 and 2009, respectively. His current research interests include cyber physical networking systems, wireless networks, mobile computing, mesh, ad hoc and sensor networking, in-network information processing, and network security. He is a member of the IEEE and the ACM.



Chunqiang Hu obtained his M.S degree and first PhD degree in computer Science and Technology from Chongqing University, Chongqing, China, in 2009 and 2013, respectively. He received his B.S. degree in Computer Science and Technology from Southwest University, Chongqing, China, in 2006. He was a visiting scholar at The George Washington University from Jan., 2011 to Dec., 2011, and then got his second PhD degree in Computer Science, The George Washington University, Washington DC in 2016. He won the Best Paper Award in ACM PAMCO 2016. His current research interests include applied cryptography, big data security and privacy, privacy-aware computing, wireless and mobile security, and algorithm design and analysis. He is a member of IEEE and ACM.

Jiguo Yu received his Ph.D. degree in School of mathematics from Shandong University in 2004. He became a full professor in the School of Computer Science, Qufu Normal University, Shandong, China in 2007. Currently he is a full professor in the School of Information Science and Engineering, Qufu Normal University. His main research interests include privacy-aware computing, wireless networking, distributed algorithms, peer-to-peer computing, and graph theory. Particularly, he is interested in designing and analyzing algorithms for many computationally hard problems in networks. He is a member of the IEEE and ACM, and a senior member of the CCF (China Computer Federation).



Xiuzhen Cheng received her M.S. and Ph.D. degrees in computer science from the University of Minnesota – Twin Cities in 2000 and 2002, respectively. She is a professor in the Department of Computer Science, The George Washington University, Washington, DC. Her current research interests include privacy-aware computing, wireless and mobile security, cyber physical systems, mobile computing, and algorithm design and analysis. She has served on the editorial boards of several technical journals and the technical program committees of various professional conferences/workshops. She also has chaired several international conferences. She worked as a program director for the US National Science Foundation (NSF) from April to October in 2006 (full time), and from April 2008 to May 2010 (part time). She received the NSF CAREER Award in 2004. She is a member of ACM, and a Fellow of IEEE.

Fengjuan Zhang graduated from University of Science and Technology of China in 2015 as a master. She is an engineer at Baidu now. Fengjuan Zhang focus mainly on convex optimization, collaborative filtering and knowledge graph.