STUDIA INFORMATICA
Nr 1/2(7) Systemy i technologie informacyjne 2006

A Multi-Party Scheme for Privacy-Preserving Clustering

Justin Zhan[1], Stan Matwin[2], and LiWu Chang[3]
[1] School of Information Technology and Engineering, University of Ottawa, Canada, zhizhan@site.uottawa.ca
[2] School of Information Technology and Engineering, University of Ottawa, Canada. Institute for Computer Science, Polish Academy of Sciences, Warsaw, Poland, stan@site.uottawa.ca
[3] Center for High Assurance Computer Systems, Naval Research Laboratory, USA, lchang@itd.nrl.navy.mil

Abstract. Preserving data privacy while conducting data clustering among multiple parties is a demanding problem. We address this challenging problem in the following scenario: without disclosing their private data to each other, multiple parties, each having a private data set, want to collaboratively conduct k-medoids clustering. To tackle this problem, we develop secure protocols for multiple parties to achieve this dual goal. The solution is distributed, i.e., there is no central, trusted party having access to all the data. Instead, we define a protocol using homomorphic encryption and digital envelope techniques to exchange the data while keeping it private.

Keywords. Privacy, security, clustering.

1 Introduction

Huge amount of data often locates in geographically distributed sources. To extract useful knowledge from these distributed data, many approaches [4] have been designed. Although these techniques are sufficient if each source would like to provide its actual data, it is not desirable when their data privacy comes into place. How can multiple parties still extract the useful information without comprising their data privacy is a challenging problem. In the field of knowledge discovery and data mining, the above problem is known as *Privacy Preserving Collaborative Data Mining*. Vidya and Clifton [10] provided the following convincing example in the area of automotive safety: Ford Explorers with Firestone tires from a specific factory had tread separation problems in certain situations. Early identification of the real problem could have avoided at least some of the 800 injuries that occurred in accidents attributed to the faulty tires. Since the tires did not have problems on other vehicles, and other tires on Ford Explorers did not pose a problem, neither side felt responsible. Both manufacturers had their own data, but only early extraction of useful knowledge based on both parties' data may have enabled Ford and Firestone to collaborate in resolving this safety problem.

Data mining contains many algorithms such as association rule mining, classification, and clustering. This paper focuses on data clustering. In recent years, there has been a surge of interest in clustering [12]. It is an active research area in many fields such as pattern recognition, statistics, and machine learning. It divides the data into groups of similar objects such that both intra-group similarity and inter-group dissimilarity are maximized. Although losing certain fine details, it simplifies the representation of data by several clusters. In practice, clustering plays a high-profile role in data mining applications such as scientific data exploration, medical diagnostics, and information retrieval, etc. Based on the existing clustering technologies, we study the problem of clustering on private data. More precisely, we consider the scenario where data are vertically partitioned, and the problem is defined as follows: multiple parties want to conduct data clustering on a data set that consists of private data of all the parties, but none of the parties is willing to disclose its actual data to one another or any other parties. More specifically, the data consist of instances, all parties have data about all the instances involved, but each party has its own view of the instances – each party works with its own attribute set. We develop secure protocols, based on homomorphic encryption and digital envelope techniques, to tackle the problem. An important feature of our approach is its distributed character, i.e. there is no single, centralized authority that all parties need to trust. Instead, the computation is distributed among parties, and the use of homomorphic encryption and digital envelope techniques ensures privacy of the data.

The paper is organized as follows: We describe the clustering procedure in Section 2. We then present our proposed secure protocols and detailed analysis in Section 3. The related works are discussed in Section 4. We give our conclusion in Section 5.

2 Clustering On Private Data

There are many clustering algorithms [4] such as k-means method, k-medoids method, probabilistic clustering, etc. We focus on k-medoids method since it allows arbitrary objects that are not limited to numerical attributes [12]. In k-medoids clustering, a cluster is denoted by one of its points. It is an easy solution in that it covers any attribute type and that medoids are resistant against outliers. Once medoids are chosen, clusters are defined as subsets of points close to respective medoids, and the objective function is described as the distance between a point and its medoid. In this paper, our goal is to provide a privacy-preserving algorithm for multiple parties to collaboratively conduct data clustering using k-medoids method without compromising their data privacy.

2.1 Notations

We define the following notations for illustration purposes.

- n: the total number of parties. We assume $n \geq 3$.

- P_j : Party j.
- k: the total number of medoids.
- t: a non-medoid instance.
- m_{C_i} : the medoid of the cluster C_i .
- M: a general term for medoids. It contains all possible medoids.
- NM: a general term for non-medoids. It contains all possible medoids.
- $TD(C_i)$: the measure of the compactness for a cluster C_i .
- TD: the measure of the compactness of a clustering that contains all the clusters.

2.2 Overview of k-medoids Clustering Algorithm

The k-medoids method divides a distance-space into k clusters. A medoid [12], that is selected from the dataset, represents a cluster. The algorithm chooses k medoids to denote the k clusters. Clusters are then created by assigning each of the remaining instances to the nearest medoid. As in the k-means method *k* needs to be predefined. Unlike k-means method where each mean is the average of certain instances, k-medoids are exactly k instances selected from the dataset. We describe the k-medoids clustering algorithm in the following.

1. Arbitrarily select k instances from the dataset as medoids.
2. Assign each remaining (non-medoid) instance to the cluster with the nearest medoid.
3. Compute the compactness of a clustering, denoted by $TD_{current}$.

$$TD = \sum_{i=1}^{k} TD(C_i) \qquad (1)$$

$$TD(C_i) = \sum_{t \in C_i} dist(t, m_{C_i}) \qquad (2)$$

4. For each pair (medoid M and non-medoid NM)
   - Compute the value of TD for the partition that results from swapping M with NM, denoted by $TD_{NM \leftrightarrow M}$.
5. Select the non-medoid NM for which $TD_{NM \leftrightarrow M}$ is minimal.
6. If $TD_{NM \leftrightarrow M} < TD_{currrent}$
   - Swap NM with M
   - Set $TD_{current}$ to be $TD_{NM \leftrightarrow M}$.
   - Go to Step 4.

The algorithm requires a distance function. For instance, the distances can be defined in terms of standard Euclidean distance. As we will discuss, each party computes her own portion of the distance and utilization of certain distance measure does not cause privacy violation. Therefore, other distance functions can be applied as well.

2.3 The Scenarios Where the Private Data May Be Exposed

The key step of the k-medoids clustering algorithm is the computation of the distance between each non-medoid t and its medoid m_{C_i} without disclosing their private data. There are two cases where we need secure computations: (1) Assign each non-medoid instance to the cluster with the nearest medoid. Since each party holds only a portion of attributes for each instance, each party computes its portion of the distance measure (called the *distance portion*) according to its attribute set. To decide the nearest medoid of t, all the parties need to sum their distance portions together, then compare the summation. For example, assume that the distance portions between t and the medoid instance m_{C_i} are $s_{11}, s_{12}, \cdots, s_{1n}$; and the distance portions between t and the medoid instance $m_{C_i} (i \neq j)$ are $s_{21}, s_{22}, \cdots, s_{2n}$ where s_{1j} and s_{2j} belong to P_j for $j \in [1, n]$. To compute whether the distance between the medoid instance m_{C_i} and t is larger than the distance between the medoid instance m_{C_i} and t, we need to evaluate the expression $\sum_{i=1}^{n} s_{1i} \geq \sum_{i=1}^{n} s_{2i}$. (2) Compute TD. That is, for a particular cluster, computing the distances between each non-medoid instance and its medoid; then adding all the distances together to obtain $TD(C_i)$. TD can then be computed by summation of $TD(C_i)$ for all k clusters. Given a non-medoid instance t, multiple parties want to compute the distance between t and its medoid instances m_{C_i}. Secure protocols are developed in the next section to enforce such computations without sacrificing data privacy.

3 Secure Computing Protocol

3.1 Introducing Homomorphic Encryption

In our secure protocols, we use homomorphic encryption [17] keys to encrypt the parties' private data. In particular, we utilize the following character of the homomorphic encryption functions: $e(a_1) \times e(a_2) = e(a_1 + a_2)$ where e is an encryption function; a_1 and a_2 are the data to be encrypted. Because of the property of associativity, $e(a_1 + a_2 + \cdots \_ a_n)$ can be computed as $e(a_1) \times e(a_2) \times \cdots \times e(a_n)$ where $e(a_i) \neq 0$. That is

$$e(a_1 + a_2 + \cdots + a_n) = e(a_1) \times e(a_2) \times \cdots \times e(a_n) \qquad (3)$$

We observe that some homomorphic encryption schemes [7] are not robust against chosen cleartext attacks. However, we base our algorithm on [17], which is semantically secure [10].

A Secure Protocol for Computing the Nearest Medoid

Assuming P_j has a private distance portion of the ith instance, s_{ij}, for $i \in [1,k], j \in [1,n]$, the problem is to decide whether $\sum_{j=1}^{n} s_{ij} \le \sum_{j=1}^{n} s_{lj}$ for $i,l \in [1,k](i \ne l)$ and select the smallest value $TD(C_i)$, without disclosing each distance portion. We will provide a solution which uses homomorphic encryption and digital envelope techniques.

Digital envelope A digital envelope is a random number (or a set of random numbers) only known by the owner of private data. To hide the private data in a digital envelope, we conduct a set of mathematical operations between a random number (or a set of random numbers) and the private data. The mathematical operations could be addition, subtraction, multiplication, etc. For example, assume the private data value is a. There is a random number R which is only known the owner of a. The owner can hide a by adding this random number, e.g., a+R.

Highlight of the protocol Our protocol has four steps. (1) Key and digital envelope generation: multiple parties select one of them, e.g., P_n, as the key generator, which creates a cryptographic key pair (e, d) of a semantically-secure homomorphic encryption scheme. Each party generates k digital envelopes. (2) Computing $e(\sum_{j=1}^{n}(s_{ij} + r_{ij}))$ for $i \in [1,k]$: each party puts its private distance portion into a digital envelope and sends it to P_{n-1}. (3) Computing $e(\sum_{j=1}^{n} r_{ij})$, for all $i \in [1,k]$: each party encrypts its digital envelopes and sends them to P_1. (4) P_1, P_{n-1} and P_n jointly compute the nearest medoid: there are 4 sub-steps. The details on how P_{n-1} and P_n compute the smallest element in the last step are described following the protocol.

Protocol 1.

Step I: Key and digital envelope generation.
1. P_js for $j \in [1,n]$ randomly select a key generator, e.g., P_n.

2. P_n generates a cryptographic key pair (e, d) of a semantically-secure homomorphic encryption scheme and publishes its public key e. Let e(.) denote encryption and d(.) denote decryption.
3. Each party independently generates k digital envelopes, i.e., P_j generates k digital envelopes r_{ij}, for all $i \in [1,k], j \in [1,n]$.

Step II: Computing $e(\sum_{j=1}^{n}(s_{ij}+r_{ij}))$ for $i \in [1,k]$.

1. P_1 computes $e(s_{i1}+r_{i1})$, for $i \in [1,k]$, and sends them to P_2.
2. P_2 computes $e(s_{i1}+r_{i1}) \times e(s_{i2}+r_{i2}) = e(s_{i1}+s_{i2}+r_{i1}+r_{i2})$, where $i \in [1,k]$, and sends them to P_3.
3. Repeat steps 1, 2 until P_{n-1} obtains $e(s_{i1}+s_{i2}+\cdots+s_{i(n-1)}+r_{i1}+r_{i2}+\cdots+r_{i(n-1)})$, for all $i \in [1,k]$.
4. P_n computes $e(s_{in}+r_{in})$ for $i \in [1,k]$, and sends them to P_{n-1}.
5. P_{n-1} computes $e(s_{i1}+s_{i2}+\cdots+s_{i(n-1)}+r_{i1}+r_{i2}+\cdots+r_{i(n-1)}) \times e(s_{in}+r_{in})$

$e(s_{i1}+s_{i2}+\cdots+s_{i(n-1)}+s_{in}+r_{i1}+r_{i2}+\cdots+r_{i(n-1)}+r_{in}) = e(\sum_{j=1}^{n}(s_{ij}+r_{ij}))$, $i \in [1,k]$. Let e(S + R) denote the k encrypted elements as follows:
[$e(S_1+R_1), e(S_2+R_2), \cdots, e(S_k+R_k)$], where $S_i = \sum_{j=1}^{n} s_{ij}$ and $R_i = \sum_{j=1}^{n} r_{ij}$.

Step III: Computing $e(\sum_{j=1}^{n} r_{ij})$ for all $i \in [1,k]$.

1. P_n computes $e(r_{in})$ for $i \in [1,k]$ and sends them to P_{n-1}.
2. P_{n-1} computes $e(r_{in}) \times e(r_{i(n-1)}) = e(r_{in}+r_{i(n-1)})$ for $i \in [1,k]$, and sends them to P_{n-2}.
3. Repeat steps 1, 2 until P_1 obtains $e(r_{i1}+r_{i2}+\cdots+r_{i(n-1)}) \times e(r_{in}) = e(\sum_{j=1}^{n} r_{ij})$, for all $i \in [1,k]$. The k encrypted elements are denoted by e(R) that contains the following: $[e(R_1), e(R_2), \cdots, e(R_k)]$ where $R_i = \sum_{j=1}^{n} r_{ij}$.

Step IV: Computing the nearest medoid.

1. Computation between P_1 and P_n.
(a) P_1 randomly permutes $e(R_1), e(R_2), \cdots, e(R_k)$, then sends the permuted elements to P_n.

(b) P_n decrypts each element and sends them to P_1 in the same order as P_1 did.

(c) P_1 computes R that contains the following: $[R_1, R_2, \cdots R_k]$. Note that P_1 can do it since it has the permutation function.

2. Computation between P_{n-1} and P_n.

(a) P_{n-1} randomly permutes $e(S_1), e(S_2), \cdots, e(S_k)$, then sends the permuted elements to P_n.

(b) P_n decrypts each element and sends them to P_{n-1} in the same order as P_{n-1} did.

(c) P_{n-1} computes $[S_1 + R_1, S_2 + R_2, \cdots, S_k + R_k]$ denoted by S+R. Note that: (1) P_{n-1} can do it since it has the permutation function. (2) The permutation function that P_1 used is independent of the permutation function that P_{n-1} used.

3. P_{n-1} and P_1 compute $e(S_i - S_l) = e(\sum_{j=1}^{n} s_{ij} - \sum_{j=1}^{n} s_{lj})$, for $i, l \in [1, k](i \neq l)$, and collects the results into a sequence ϕ which contains k(k-1) elements. This computation can be achieved via the following process:

(a) P_1 computes $e(R_l)$ and $e(-R_i)$ for $i, l \in [1, k](i \neq l)$, then sends them to P_{n-1}.

(b) P_{n-1} computes $e(S_i - S_l)$ for $i, l \in [1, k](i \neq l)$ as follows:

- $e(S_i + R_i) \times e(-R_i) = e(S_i)$.
- $e(-S_l - R_l) \times e(R_l) = e(-S_l)$.
- $e(S_i) \times e(-S_l) = e(S_i - S_l)$.

4. Computation between P_{n-1} and P_n.

(a) P_{n-1} randomly permutes this sequence ϕ and obtains the permuted sequence denoted by ϕ', then sends ϕ' to P_n. Note that the permutation is independent of the ones it used.

(b) P_n decrypts each element in sequence ϕ'. It assigns the element +1 if the result of decryption is not less than 0, and -1, otherwise. Finally, it obtains a +1/-1 sequence denoted by ϕ''.

(c) P_n sends ϕ'' to P_{n-1} who computes the smallest element. (Details are given right after this protocol.) It is the nearest medoid for a given non-medoid instance t. It then decides the cluster to which t belongs.

How To Compute the Smallest Element P_{n-1} is able to remove permutation effects from ϕ'' (the resultant sequence is denoted by ϕ''') since it has the permutation function that it used to permute ϕ, so that the elements in ϕ and ϕ''' have the same order. It means that if the qth position in sequence ϕ denotes $e(\sum_{j=1}^{n} s_{ij} - \sum_{j=1}^{n} s_{lj})$, then the qth position in sequence ϕ''' denotes the evaluation results of $\sum_{j=1}^{n} s_{ij} - \sum_{j=1}^{n} s_{lj}$. We encode it as +1 if $\sum_{j=1}^{n} s_{ij} \geq \sum_{j=1}^{n} s_{lj}$, and as -1 otherwise. P_{n-1} has two sequences: one is the ϕ, the sequence of $e(\sum_{j=1}^{n} s_{ij} - \sum_{j=1}^{n} s_{lj})$, for $i, l \in [1, k] (i \neq l)$, and the other is ϕ''', the sequence of +1/-1. The two sequences have the same number of elements. P_{n-1} knows whether or not $\sum_{j=1}^{n} s_{ij}$ is larger than $\sum_{j=1}^{n} s_{lj}$ by checking the corresponding value in the ϕ''' sequence. For example, if the first element ϕ''' is -1, P_{n-1} concludes $\sum_{j=1}^{n} s_{ij} < \sum_{j=1}^{n} s_{lj}$. P_{n-1} examines the two sequences and constructs the index table (Table 1) to compute the nearest medoid.

Table 1

	$\sum_{l=1}^{n} s_{1l}$	$\sum_{l=1}^{n} s_{2l}$	$\sum_{l=1}^{n} s_{3l}$	……	$\sum_{l=1}^{n} s_{kl}$
$\sum_{l=1}^{n} s_{1l}$	+1	+1	-1	….	-1
$\sum_{l=1}^{n} s_{2l}$	-1	+1	-1	….	-1
$\sum_{l=1}^{n} s_{3l}$	+1	+1	+1	….	+1
……	…..	…...	…..	….	….
$\sum_{l=1}^{n} s_{kl}$	+1	+1	-1	….	+1

Table 2

	S_1	S_2	S_3	S_4	Weight
S_1	+1	-1	-1	-1	-2
S_2	+1	+1	-1	+1	+2
S_3	+1	+1	+1	+1	+4
S_4	+1	-1	-1	+1	0

In Table 1, +1 in entry ij indicates that the distance measure of the row (e.g., $\sum_{l=1}^{n} s_{il}$ of the ith row) is not less than the distance measure of a column (e.g., $\sum_{l=1}^{n} s_{jl}$ of the jth column); -1, otherwise. P_{n-1} sums the index values of each row and uses this number as the weight of the distance measure in that row. It then selects the one that corresponds to the smallest weight, as the nearest medoid.

To make it clearer, let us illustrate it by an example. Assume that: (1) there are 4 elements with $S_1 < S_4 < S_2 < S_3$; (2) the sequence ϕ is $[e(S_1 - S_2), e(S_1 - S_3), e(S_1 - S_4), e(S_2 - S_3), e(S_2 - S_4), e(S_3 - S_4)]$. The sequence ϕ''' will be [-1, -1, -1, -1, +1, +1]. According to ϕ and ϕ''', P_{n-1} builds the Table 2. From the table, P_{n-1} knows S_1 is the smallest element since its weight, which is -2, is the smallest.

Next, we will discuss how to securely compute TD.

A Secure Protocol for Computing TD

Once each non-medoid instance is assigned to the nearest medoid, we need to compute the compactness of a clustering.

$$TD = \sum_{i=1}^{k} TD(C_i) = \sum_{i=1}^{k} \sum_{t \in C_i} dist(t, m_{C_i}), \qquad (4)$$

To compute TD, each party computes its local distance portions between each non-medoid instance and the corresponding medoid instance, and adds them together. For the purpose of illustration, let us assume that P_j gets a distance portion v_j for $j \in [1, n]$. In order to compute TD, we need to compute $\sum_{j=1}^{n} v_j$. We develop the following protocol to tackle the problem. Note that Protocol 1 and Protocol 2 are independent protocols. Therefore, even if we use the similar symbols, e.g., e, d and r, to represent the keys and digital envelopes, they are independent.

Protocol 2.

Step I: Key and digital envelope generation.
1. P_js for $j \in [1, n]$ randomly select a key generator, e.g., P_n.

2. P_n generates a cryptographic key pair (e, d) of a semantically-secure homomorphic encryption scheme and publishes its public key e. Let e(.) denote encryption and d(.) denote decryption.
3. Each party independently generates a digital envelope, i.e., P_j generates a digital envelope r_j, for $j \in [1, n]$.

Step II: Computing $e(\sum_{j=1}^{n}(v_j + r_j))$.

1. P_1 computes $e(v_1 + r_1)$, and sends it to P_2.
2. P_2 computes $e(v_1 + r_1) \times e(v_2 + r_2) = e(v_1 + v_2 + r_1 + r_2)$, and sends it to P_3.
3. Repeat steps 1, 2 until P_{n-1} obtains $e(v_1 + v_2 + \cdots + v_{n-2} + r_1 + \cdots + r_{n-2}) \times e(v_{n-1} + r_{n-1}) = e(v_1 + \cdots + v_{n-1} + r_1 + \cdots + r_{n-1})$.
4. P_n computes $e(v_n + r_n)$, and sends it to P_{n-1}.
5. P_{n-1} computes $e(v_1 + \cdots + v_{n-1} + r_1 + \cdots + r_{n-1}) \times e(v_n + r_n)$ $e(v_1 + \cdots + v_n + r_1 + \cdots + r_n) = e(\sum_{j=1}^{n}(v_j + r_j))$. Let us denote it by e(V + R), where $V = \sum_{j=1}^{n} v_j$ and $R = \sum_{j=1}^{n} r_j$.

Step III: Computing $\sum_{j=1}^{n} v_j$.

1. P_n computes $e(-r_n)$, and sends it to P_{n-1}.
2. P_{n-1} computes $e(-r_n) \times e(-r_{n-1}) = e(-r_n - r_{n-1})$, and sends it to P_{n-2}.
3. Repeat steps 1, 2 until P_1 obtains $e(-R) = e(-r_1 - \cdots - r_n) = e(-\sum_{j=1}^{n} r_j)$.

Step IV: Computing $\sum_{j=1}^{n} v_j$.

1. P_1 sends $e(-\sum_{j=1}^{n} r_j)$ to P_{n-1}.
2. P_{n-1} computes $e(\sum_{j=1}^{n}(v_j + r_j)) \times e(-\sum_{j=1}^{n} r_j) = e(\sum_{j=1}^{n} v_j)$, then sends it to P_n.
3. P_n computes $d(e(\sum_{j=1}^{n} v_j)) = \sum_{j=1}^{n} v_j$.

In the next section, we show that the outputs of the protocols are correct, we argue that the data privacy is preserved, and we analyze the complexity for each protocol.

The Analysis of Correctness, Privacy and Complexity

Analysis for Protocol 1:

Correctness Analysis

Assuming all of the parties follow the protocol, the protocol correctly finds the nearest medoid for a given non-medoid instance t.

In step II, P_{n-1} obtains $e(s_{i1}+r_{i1})\times e(s_{i2}+r_{i2})\times\cdots\times e(s_{in}+r_{in})$ $=e(s_{i1}+r_{i1}+\cdots+s_{in}+r_{in})=e(\sum_{j=1}^{n}(s_{ij}+r_{ij}))$, for $i\in[1,k]$ according to Eq.[3]. In step III, P_1 finds $e(r_{i1})\times\cdots\times e(r_{in})=e(\sum_{j=1}^{n}r_{ij})$, for $i\in[1,k]$, consistent with Eq.[3]. In step IV, during sub-step 1-3, P_{n-1} obtains $e(\sum_{j=1}^{n}s_{ij}-\sum_{j=1}^{n}s_{lj})$ for $i,l\in[1,k](i\neq l)$. Following the detailed description on how to compute the smallest element, we know that P_{n-1} finds the nearest medoid for a given non-medoid, which is the desired result.

Privacy Analysis Assuming P_1, P_{n-1} and P_n do not collude, one party's distance portion (i.e., private data) cannot be disclosed to other parties.

In the protocol, there are two levels of privacy protection: before one party sends his private data to any other parties, she firstly seals it by a digital envelope solely known by herself, then uses a semantically secure encryption scheme to encrypt the data. Thus, other parties cannot identify her private data. To make the discussion concrete, we analyze the protocol step by step. Step I does not disclose private data since there is no communication involving private data. In Step II, private data are communicated. However, prior to sending her private data to the other party, one party hides her private data by a two-level protector: a digital envelope known only by the owner of the private data and P_n's public key e. Since P_n does not receive any data in this step and other parties have no decryption key d, especially, no one knows the digital envelope except for the owner, the private data are securely hidden. In Step III, digital envelopes are communicated. Since each digital envelope is encrypted by e, and P_n does not receive any encrypted digital envelopes, each digital envelope is securely hidden. In Step IV, even though P_n has the decryption key d, what it can obtain is a permuted sequence of $\sum_{j=1}^{n}s_{ij}-\sum_{j=1}^{n}s_{lj}$, for $i,l\in[1,k](i\neq l)$. By knowing this sequence, it cannot identify other parties' private distance portions. Neither can P_{n-1} obtain private distance portions. Although P_{n-1} knows the sequences

ϕ and ϕ''', it cannot obtain other parties' private distance portions since it only knows the relation between $\sum_{j=1}^{n} s_{ij}$ and $\sum_{j=1}^{n} s_{lj}$ and does not know the exact values.

Complexity Analysis The communication cost of this protocol is $\alpha(3n+k^2+5k-3)$ where α is the number of bits for each encrypted element. The computational costs are contributed by: (1) the generation of kn digital envelopes; (2) additions; (3) $k^2+k+(2n-2)k$ multiplications; (4) 2kn encryptions; (5) $\frac{1}{2}k(k-1)$ decryptions; (6) 4k + k(k-1) permutations; (7) $\frac{1}{2}k(k-1)$ assignments when P_n computes ϕ''. Therefore, the total computational cost is $5kn+3k^2-3k$ where k is the number of clusters and n is the number of parties.

Analysis for Protocol 2:

Correctness Analysis Assuming all of the parties follow the protocol, to show $\sum_{i=1}^{n} v_i$ is correctly computed, we need to discuss it step by step. In step II, what P_{n-1} obtains is $e(v_1+r_1)\times e(v_2+r_2)\times\cdots\times e(v_n+r_n)$ which equals to $e(\sum_{j=1}^{n}(v_j+r_j))$ according to Eq.[3]. In step III, P_1 obtains $e(-r_1-\cdots-r_n)=e(-\sum_{j=1}^{n} r_j)$ consistent with Eq.[3]. In step IV, P_n finally gets

$d(e(\sum_{j=1}^{n} v_j+\sum_{j=1}^{n} r_j)\times e(-\sum_{j=1}^{n} r_j))=d(e(\sum_{j=1}^{n} v_j+\sum_{j=1}^{n} r_j-\sum_{j=1}^{n} r_j))=d(e(\sum_{j=1}^{n} v_j))=\sum_{j=1}^{n} v_j=TD$. This is the desired result that multiple parties want to obtain.

Privacy Analysis Like in protocol 1, there are two levels of privacy protection. One is that the actual local TD portion of each party is hidden by a digital envelope, e.g., r_i; the other is the protection by semantically secure encryptions. Before any party sends anything related to their actual TD portions, the TD portions are concealed by this two-level protector. For example, prior to P_1 sending values, related to v_1, to P_2, it computes $e(v_1+r_1)$. Instead of sending v_2 to P_3, P_2 sends $e(v_1+v_2+r_1+r_2)$, etc. Since P_1, P_{n-1} and P_n play more important role than others, e.g., step IV only involves these three parties, we provide more analysis for these three parties. (1) In step II, P_{n-1} gets $e(\sum_{j=1}^{n}(v_j+r_j))$. Because each v_j is protected by a digital envelope r_j, and the summation of each TD portion with a digital envelope is encrypted by a semantic secure encryption, P_{n-1} cannot learn anything about each v_j for $j \in [1,2,\cdots,n-2,n]$. (2) In step III, P_1 obtains $e(-\sum_{j=1}^{n} r_j)$. Since it is the summation of all the digital envelopes and is encrypted by e, it cannot know anything about each digital envelope r_j for $j \in [2,n]$. (3) P_n finally obtains $\sum_{j=1}^{n} v_j$ which is the desired output of the protocol. It will be shared by all the parties. From the above analysis, we can see that the protocol disclose nothing about each private TD portion.

Complexity Analysis The communication cost of this protocol is $\alpha(3n-1)$ where α is the number of bits for each encrypted element, and n is the total number of parties. The computational costs are contributed by: (1) the generation of n digital envelopes; (2) n additions; (3) 2n-1 multiplications; (4) 2n encryptions; (5) 1 decryption. Thus, the computational costs are 6n.

An Interesting Case Let us discuss an interesting scenario where P_1, P_{n-1}, and P_n collude with each other. What we want to know is whether the private data can be disclosed. In this case, these three parties can gain more information than what they should according to the protocols. In protocol 1, the extra useful information they can obtain is $\sum_{j=1}^{n} s_{ij}$ for $i \in [1,k]$. In protocol 2, the extra information they can obtain is $\sum_{j=1}^{n} v_j$. Based on this information, other parties' individual private distance portions cannot be derived unless, among the remaining of n-3 parties, there are n-4 parties colluding with P_1, P_{n-1} and P_n. In other words, to break our two-level protection and gain private data that should not be disclosed, n-1 parties in total need to collude. Thus, although we assume that P_1, P_{n-1} and P_n do not collude, the assumption can be released to certain extent.

4 Related Work

In early work on privacy-preserving data mining, Lindell and Pinkas [14] propose a solution to privacy-preserving classification problem using oblivious transfer protocol, a powerful tool developed by secure multi-party computation (SMC) research [9, 21]. The techniques based on SMC for efficiently dealing with large data sets have been addressed in [10]. Randomization approaches were firstly proposed by Agrawal and Srikant in [2] to solve privacy-preserving data mining problem. Researchers proposed more random perturbation-based techniques to tackle the problems (e.g., [3, 6, 19]). In addition to perturbation, aggregation of data values [20] provides another alternative to mask the actual data values. In [1], authors studied the problem of computing the kth-ranked element. Dwork and Nissim [7] showed how to learn certain types of boolean functions from statistical databases in terms of a measure of probability difference with respect to probabilistic implication, where data are perturbed with noise for the release of statistics.

Recently, there are several endeavours on privacy preserving clustering [13, 15, 16, 17]. A framework for clustering distributed over horizontally partitioned data in unsupervised and semi-supervised scenarios using sampling techniques is provided in [15]. In [13], Klusch et. al. presented an approach to distributed data clustering based on sampling density estimates. Oliveira and Zaiane introduced a family of geometric data transformation methods that ensure the mining process does not violate privacy up to a certain degree of security in [16], and showed that a solution can be achieved by transforming a database using object similarity-based representation and dimensionality reduction-based transformation in [17]. Vaidya and Clifton's work [11] is an important contribution to the problem of privacy-preserving clustering over vertically partitioned data. Their approach was using the k-means method. In our paper, we focus on clustering using k-medoids method. Since the two algorithms are different, the design for the secure protocols are dissimilar. In our protocol, the digital envelope is distributed in that each party has its own digital envelope and one party does not know the other party's digital envelope. As we discussed in the previous section, there are two-level protections in our protocols. Even though P_1, P_{n-1} and P_n collude with one another, other parties' private data still remain securely hidden unless all of the parties collude except only one party.

5 Conclusion and Future Work

In this paper, we provide a novel solution for data clustering using k-medoids method over vertically partitioned data. Instead of using data transformation, we define a protocol using homomorphic encryption and digital envelope techniques to exchange the data while keeping it private. As we discussed in the previous sections, in our protocol, there is a two-level privacy protection. Even if the non-desired situation occurs where P_1, P_{n-1} and P_n collude with one another, other parties' private data are still securely hidden unless all of the parties collude except only one party. On the other hand, the bit-wise communication cost of our protocol 1 is $\alpha(3n+k^2+5k-3)$ and $\alpha(3n-1)$ for protocol 2.

References

1. Aggarwal G., Mishra N., and Pinkas B. *Secure computation of the kth-rankerd element*. In EUROCRYPT, 40-55, 2005.
2. Agrawal R. and Srikant R. *Privacy-preserving data mining*. In Proceedings of the ACM SIGMOD Conference on Management of Data, 439-450, ACM Press, May 2000.
3. Gehrke J. E., Evfimievshi A., and Srikant R. *Limiting privacy breaches in privacy preserving data mining*. In Proceedings of the 22nd ACM SIGMOD Symposium on Principles of Database Systems, San Diego, CA, June 2003.
2. Berkhin P. Survey of clustering data mining techniques. Technical Report, Accrue Software, San Jose, CA, 2002.
3. Domingo-Ferrer J. *A provably secure additive and multiplicative privacy homomorphism*. In Information Security Conference, 471-483, 2002.
4. Du W., and Zhan Z. *Using randomized response techniques for privacy-preserving data mining*. In Proceedings of The 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24-27, 2003.
5. Dwork C., and Nissim K. *Privacy-preserving data mining on vertically partitioned databases*. In CRYPTO 2004, 528-544.
6. Goethals B, Laur S., Lipmaa H., and Mielikainen T. *On secure scalar product computation for privacy-preserving data mining*. In Proceedings of The 7th Annual International Conference of Information Security and Cryptology, volume 3506 of Lecture Notes in Computer Science, 104-120, Seoul, Korea, December 2-3, 2004, Springer-Verlag.
7. Goldreich O. *Secure multi-party computation (working draft)*. http://www.wisdom,weizmann.ac.il/home/oded/public_html/foc.html, 1998.
8. Vaidya J., and Clifton C. *Privacy preserving association rule mining in vertically partitioned data*. In Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada.
9. Vaidya J., and Clifton C. *Privacy preserving k-means clustering over vertically partitioned data*. In Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003, Washington DC, USA.
10. Kaufman L., and Rousseeuw. Finding groups in data. Wiley, New York, NY, 1990.
11. Klusch M., Lodi S., and Moro G-L. Distributed clustering based on sampling local density estimates. In Proceedings of International Joint Conference on Artificial Intelligence, Mexico, 2003.
12. Lindell Y., and Pinkas B. *Privacy preserving data mining*. In Advances in Cryptology – Crypto2000, Lecture Notes in Computer Science, volume 1880, 2000.

13. Merugu S., and Ghosh J. Privacy-preserving distributed clustering using generative models. In Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 24-27, 2003, Washington, DC, USA.
14. Oliveira S., and Zaiane O. Privacy preserving clustering by data transformation. In Proceedings of the 18th Brazilian Symposium on Databases, 304-318, Manaus, Brazil, October 6-8, 2003.
15. Oliveira S., and Zaiane O. Privacy preserving clustering by data object similarity-based representation and dimensionality reduction transformation. In Workshop on Privacy and Security Aspects of Data Mining in conjuction with the 4th IEEE International Conference on Data Mining, 21-30, Brighton, UK, November 1, 2004.
16. Paillier P. *Public-key cryptosystems based on composite degree residuosity classes*. In Advances in Cryptography – EUROCRYPT99, 223-238, Prague, Czech Republic, May 1999.
17. Rizvi S., and Haritsa J. *Maintaining data privacy in association rule mining*. In Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002.
18. Sweeney L. *k-anonymity: a model for protecting privacy*. In International Journal on Uncertainty, fuzziness and Knowledge-based Systems, 10(5), 557-570, 2002.
19. Yao A. *Protocols for secure computations*. In Proceedings of the 23rd Annual IEEE Symposium on foundations of Computer Science, 1982.