Efficient Distributed Privacy Preserving Clustering

Mahir Can Doğanay

Submitted to the Graduate School of Sabancı University
in partial fulfillment of the requirements for the degree of
Master of Science

Sabanci University

August, 2008

Efficient Distributed Privacy Preserving Clustering

Approved by:

Assist.Prof. Yücel Saygın (Thesis Advisor)	.....
Assoc.Prof. Erkay Savaş	.....
Assoc.Prof. Albert Levi	.....
Assoc.Prof. Mehmet Keskinöz	.....
Assist.Prof. Hüsnü Yenigün	.....

Date of Approval: .....

Efficient Distributed Privacy Preserving Clustering

Mahir Can Doğanay
Computer Science & Engineering, Master's Thesis, 2008
Thesis Supervisor: Yücel Saygın

Keywords: Privacy Preserving Data Mining, Clustering, Secure Multiparty Computation, Secret Sharing

Abstract

With recent growing concerns about data privacy, researchers have focused their attention to developing new algorithms to perform privacy preserving data mining. However, methods proposed until now are either very inefficient to deal with large datasets, or compromise privacy with accuracy of data mining results.

Secure multiparty computation helps researchers develop privacy preserving data mining algorithms without having to compromise quality of data mining results with data privacy. Also it provides formal guarantees about privacy. On the other hand, algorithms based on secure multiparty computation often rely on computationally expensive cryptographic operations, thus making them infeasible to use in real world scenarios.

In this thesis, we study the problem of privacy preserving distributed clustering and propose an efficient and secure algorithm for this problem based on secret sharing and compare it to the state of the art. Experiments show that our algorithm has a lower communication overhead and a much lower computation overhead than the state of the art.

Verimli Mahremiyeti Koruyan Dağıtık Kümeleme

Mahir Can Doğanay
Bilgisayar Bilimi ve Mühendisliği, Yüksek Lisans Tezi 2008
Tez Danışmanı: Yücel Saygın

Anahtar Kelimeler : Mahremiyeti Koruyan Veri Madenciliği, Kümeleme,
Güvenli Çok Partili Hesaplama, Paylaşımlı Şifreleme

Özet

İnsanların son yıllarda artan mahremiyet kaygıları veri madenciliği alanında çalışan araştırmacıları mahremiyeti koruyan veri madenciliği algoritmaları geliştirmeye zorluyor. Ancak şu ana kadar ortaya atılan algoritmalar ya çok büyük veri yığınları üzerinde etkisiz kalmakta, ya da insanları mahremiyet ve sonuç kalitesi arasında bir seçim yapmaya zorlamaktadır.

Güvenli çok partili hesaplama yöntemleri veri madenciliği algoritmalarının sonuç kalitelerini bozmadan veri mahremiyetini korumaya imkan tanımaktadır. Ancak güvenli çok partili hesaplama yöntemi ile geliştirilmiş mahremiyeti koruyan veri madenciliği algoritmaları, açık anahtar şifreleme teknikleri kullanılarak geliştirilmektedir. Bu da bu algoritmaların gerçek hayatta büyük veri yığınları üstünde kullanılmasını imkansız kılmaktadır.

Bu tezde açık anahtar şifreleme yerine paylaşımlı şifreleme tekniğine dayanan bir dağıtık kümeleme algoritması önerilmiştir. Yapılan testlere göre algoritma şu ana kadar ortaya atılan eni iyi algoritmalardan çok daha hızlı çalışmakta ve partiler arası daha az veri transferi gerektirmektedir.

Acknowledgments

Firstly I would like to express my gratitude to Dr. Yücel Saygın for his support during my entire graduate study. He has always been supportive and understanding and always provided me with good advice on any matter. I would like to especially thank him for encouraging me to register for conferences and summer schools and pay all the expenses. This really helped me to broden my vision in the area of computer science.

I would like thank TÜBİTAK, for providing me the financial support with their excellent scholarship program for graduate studies. I am proud to be a recipient of this scholarship.

I would like thank Dr. Erkay Savaş and Dr. Albert Levi, who has shown great interest in my work and guided me with their helpful comments all the way.

I would like to thank Thomas B. Pedersen, most of the ideas in this thesis were inspired by our on-board discussions with him.

Finally, I would like to thank my fellows, Ulvi, Ayşegül, Övünç, Ersin, İsmail Fatih, Sezin, Selim, Cengiz, Ömer, Hakan, the people whom I spent great time with in Cryptography & Information Security Lab.

Contents

List of Figures

1 Introduction

Data mining can best be described as the process of turning data into useful knowledge. Massive amounts of data are collected for various reasons by many organizations with the hope that data mining technology will extract useful knowledge from the collected data and turn it into something beneficial. Data mining technology has been proven successful in numerous applications such as business intelligence, health sciences, traffic management and security related topics like credit card fraud detection.

However, increasing privacy concerns have begun to put large scale data mining projects in jeopardy, largely due to the projects carried out by the Department of Homeland Security in U.S. Privacy concerns even lead to cancellation of large scale projects because they failed to meet privacy concerns. According to a recent article in Computer World by Jaikumar Vijayan "The chairman of the House Committee on Homeland Security, has asked Department of Homeland Security Secretary Michael Chertoff to provide a detailed listing of all IT programs that have been canceled, discontinued or modified because of privacy concerns"[25]. Due to these privacy concerns, data mining researchers have focused their attention to developing techniques that would enable data mining while preserving the privacy of individuals and started a popular branch of research named "privacy preserving data mining"[1].

Today, the need for efficient privacy preserving data mining methods is more than ever. With many governmental and private institutions collecting data about their clients and users, nowadays personal data about an individual resides on several servers. In some cases, it is useful to combine this data and perform data mining on the union of this personal data when

analysis of this data is of mutual interest to all data holders. In this case, protecting privacy becomes not only the individuals' concern, but data holders also must pay attention to protecting the indivuals' data they posses due to contractual obligations. Thus, there is a need for privacy preserving data mining algorithms which can work in a distributed scenario.

Over the years, algorithms based on statistics and cryptography were proposed for privacy preserving data mining problems such as classification, clustering, and pattern mining that work in centralized and distributed environments. However, privacy preserving data management in general, is still an ongoing research topic, and efficient, as well as provably secure, methods without strong assumptions are yet to be proposed.

In this thesis, we address privacy preserving clustering of data sets vertically partitioned among multiple sites. We propose a new secure multiparty computation based algorithm for distributed privacy preserving clustering that is both more efficient and faster than the state of the art for this problem. As a case study we chose k-means clustering of moving object trajectories to show the effectiveness of the proposed scheme. In this scheme, we eliminate the computationally heavy public key encryption operations and base our algorithm on secret sharing as the cryptographic primitive. The main contributions of this work can be summarized as follows:

- We propose a new and generic scheme which has lower communication cost than state of the art up to more than 60 parties, which is reasonable for most applications.

- We implemented and tested our scheme as well as the algorithm in [24]. Our experiments, confirm that our algorithm has significantly lower

computational overhead due to the fact that we replace computationally expensive public key encryption operations with secret sharing.

- We demonstrate that our scheme is applicable in any clustering algorithm which uses a dissimilarity matrix, such as hierarchical clustering and k-medoids clustering.

- We apply our algorithm to clustering of moving object trajectories and show that our algorithm is able to deal with trajectories with local time and distance shifts.

The rest of this thesis is organized as follows. In section 2, we focus on the necessary background, explaining clustering and secure multiparty computation. In section 3 we explain privacy preserving data mining and give related work on the subject. In section 4, we explain our privacy preserving clustering methods, explaining how it differs from state of the art. Section 5, we show how our methods preserve privacy and security, explaining its strengths by comparing it to state of the art. In section 6, we show an in depth computation and communication cost analysis and finally in section 7, we give our conclusions.

2 Background

2.1 Clustering

Clustering is one of the most fundamental data mining problems. Clustering can be defined as grouping a set of data objects into classes of similar objects. A cluster is a set of data objects where an object is *similar* to the objects in the same cluster and is *dissimilar* to the objects in other clusters. Clustering has a very diverse application area, ranging from market research(e.g. customer segmentation) to biology(e.g. finding genes with similar functionality). A number of clustering algorithms have been proposed so far with new ones still continuing to be proposed. Clustering algorithms can be roughly divided into 3 categories.

2.1.1 Partitioning based methods

Algorithms in this group partition a set of n data objects and create k partitions of it where $k \leq n$. The number of clusters,k, is given as a parameter to the algorithms in this group. Given k, algorithms in this group create an initial partitioning. Then they iteratively try to improve the quality of the partitioning until a termination criteria is reached. K-means is a prime example of a partitioning based clustering algorithm. Another notable example k-medoids, very similar in terms of basic idea to k-means clustering. The main difference between k-means and k-medoids is that in k-means an imaginary mean object is selected as a cluster representative whereas in k-medoids one of the actual objects in a cluster is selected as the representative object for that cluster. This makes k-medoids more robust to outliers, yet it

also decreases its computational efficiency.

2.1.2 Hierarchical Clustering

Algorithms in this group create a hierarchical decomposition of the given set data objects. Depending on how the hierarchical order is computed, a hierarchical clustering algorithm is further classified as *agglomerative* or *divisive*. Agglomerative approach($bottom - up$) starts with putting each data object in a group of its own and merges objects or groups of objects that are close to one another until a termination criteria is met. Divisive approach($top-down$) is the opposite, it starts with all the data objects in one group successively divides each group into smaller groups until it terminates. The main problem with hierarchical clustering algorithms is that, when an object is assigned to a false cluster, it is impossible to go back and correct it, unlike partitioning methods. Some examples of algorithms in this category are AGNES(AGglomerative NESting) and BIRCH.

2.1.3 Density based Clustering

While partitioning based and hierarchical clustering algorithms are based on distance, algorithms in this group are based on the notion of density. The general idea behind algorithms in this class is to keep growing the given cluster as long as the density in the neighborhood of the cluster is over some threshold. In other words, for each data object in a given cluster, there has to be more than a minimum number of points in the cluster. Such a method is more robust to noise and can also find clusters with arbitrary shape. OPTICS[2] and DBSCAN[8] are two prime examples of density based

clustering methods.

It is very difficult to find the most effective clustering algorithm. Usually each algorithm is tailor-made to work for a specific type of data, the choice of the clustering algorithm that gives best results is highly problem specific.

2.2 Spatio-Temporal Clustering

In theory, all the clustering algorithms that work on numeric data can be applied to spatio-temporal data(e.g. moving object trajectory data). However, it has been understood that when dealing with moving object trajectories, treating trajectories as numerical data does not yield much success, time attribute of moving object trajectories should be considered as well. A regression model based algorithm by Gaffney et al.[11] considers trajectories as a whole rather than as a set of points. A more recent work by Lee,Han,Whang[18] proposed an algorithm that can detect similar portions of trajectories that otherwise are not close to each other.

2.2.1 Trajectory Distance Measures

The difficulty of trajectory clustering is a result of the fact that the notion of "similar trajectories" can change per application. Therefore, depending on the notion of "similar trajectories", appropriate distance measures should be used. Also trajectories in real world may be of different length, they may have different observation intervals, and trajectories may be subject to local time and distance shifts. A distance measure should take these issues into account. Various distance measures for trajectories have been proposed and this topic is still an active research area.

The naive method to compare trajectories is to use Euclidean or Manhattan distances, based on comparing the corresponding location observations of trajectories of equal length. They are also sensitive to local time shifts and noise. However, the value they output is metric(i.e. triangle inequality holds). Also they have a $\theta(n)$ complexity, making it very efficient for very long trajectories.

EDR(Edit Distance on Real Sequence)[5] and LCSS(Longest Common Subsequence)[26] are distances based on quantized penalty. They try to match the corresponding location observations of two trajectories. Although they are resistant to time shifts, the value they output is not metric. Also both of them have a $\theta(n^2)$ complexity.

Edit Distance with Real Penalty(ERP)[6] is a combination of Euclidean Distance and EDR. It calculates the minimum cost of transforming one trajectory to the other after finite number of operations where the cost of each operation is the spatial shift between the corresponding location observations. It is resistant to noise, and local time shifts and it is a metric distance measure. The complexity of ERP is $\theta(n^2)$.

2.3 Secure Multiparty Computation

The Secure Multipary Computation was introduced by Yao in[28], with "millionaires problem". In this problem, two millionaires want to compare their wealth, but at the same time do not want to reveal the precise amount of wealth they got. Formal definition of the problem is as follows. A number of players $(P_1, P_2, ..., P_n)$ have their private data $(x_1, x_2, ..., x_n)$ respectively. They want to compute the value $y = f(x_1, x_2, ..., x_n)$,such that f is a public

function, so that no player can learn anything than the result of the computation and the public function f reveal. Two important building blocks of Secure multiparty computation is homomorphic encryption and secret sharing.

2.3.1 Homomorphic Encryption

Homomorphic encryption is a type of encryption that allows us to do computation on encrypted inputs without having to decrypt the input, thus it is a fundamental block of Secure multiparty computation. A public key encryption scheme is homomorphic with respect to a binary operator $\bullet$ if there is a binary operator $*$ such that $E(a) * E(b) = E(a \bullet b)$. One example of such an encryption scheme is Paillier public key encryption, which is homomorphic with respect to addition. With modulus n and and base g, the encryption of the message a will be

$$E(a) = g^a r_1^n \ mod \ n^2$$

The encryption of the message will be

$$E(b) = g^b r_2^n \ mod \ n^2$$

From two equations above, it follows that

$$E(a)E(b) = g^{a+b}(r_1 r_2)^n \ mod \ n^2 = E(a + b \ mod n)$$

2.3.2 Secret Sharing

A (t, n) secret sharing scheme is a set of two functions S and R. The function S is a *sharing function* and takes a secret s as input and creates n *secret*

shares: $S(s) = (s_1, \ldots, s_n)$. The two functions satisfy that for any set $I \subseteq \{1, \ldots, n\}$ of t indices $R(I, s_{I_1}, \ldots, s_{I_t}) = s$. Furthermore we require that it is impossible to recover s from a set of $t-1$ secret shares. A secret sharing scheme is *additively homomorphic* if $R(I, s_{I_1} + s'_{I_1}, \ldots, s_{I_t} + s'_{I_t}) = s + s'$.

A very simple (n, n) secret sharing scheme which is additively homomorphic is $S(s) = (r_1, \ldots, r_{n-1}, r)$, where r_i is random for $i \in \{1, \ldots, n-1\}$, and $r = s - \sum_{i=1}^{n-1} r_i$. To recover s all secret shares are added: $s = r + \sum_{i=1}^{n-1} r_i$. If even one secret share is missing nothing is known about s. We use this simple additive secret sharing scheme in our algorithms in this thesis.

3 Privacy Preserving Data Mining

Privacy preserving data mining is a relatively young research area with around 8 years of history. Aggrawal and Srikant initiated the research on privacy preserving data mining with their seminal paper about constructing classifications models while preserving privacy [1]. The aim of privacy preserving data mining is ensuring the privacy of the individuals is preserved while maintaining the efficiency and accuracy of data mining algorithms. Preserving privacy and performing data mining are two conflicting goals, if one wants to perform privacy preserving data mining, he has to either sacrifice accuracy of data mining results or computational efficiency of the algorithm. Sometimes data mining results themselves can be in breach of privacy[14]. All the work in privacy preserving data mining field try to achieve maximum privacy protection with minimum loss of accuracy/efficiency.

Privacy preserving data mining algorithms can be divided into two categories, (1) Random perturbation-based and (2) Secure Multiparty computation based algorithms.

3.1 Data Perturbation Based Algorithms

Perturbation techniques mix additive or multiplicative noise with the data so that actual values in the data set are not learned, yet the data mining results gathered from the perturbed data will not deviate significantly from the results gathered from the original data. The work of Aggrawal and Srikant[1] fall in this category. Another notable work in this category is privacy preserving mining of association rules by Effimievski et. al. [9]. One

recent significant work by Kargupta et al.[20] shows that random projection based multiplicative data perturbation is a very efficient way to perform privacy preserving clustering. The results obtained from perturbed data have below 5% error rate as compared to the results obtained from the original data.

Algorithms in this category present a very practical and efficient way of performing privacy preserving data mining, bringing little or no computational overhead to the data mining algorithms. However, these algorithms are based on trading accuracy of the data mining results with privacy of individuals. In addition, while the noise added to the data may preserve one statistical feature of the dataset, other statistical properties are likely to be affected. Also most of these algorithms consider a centralized scenario, i.e. cannot deal with data distributed over multiple data holders. Furthermore, algorithms in this category do not preserve privacy in any formal sense, i.e. one cannot easily calculate how much effort and resources are needed to filter out the noise and breach privacy.

3.2 Secure Multiparty Computation Based Algorithms

Algorithms based on secure multiparty computation (SMC) do not have to trade accuracy with privacy and they preserve the security and privacy of the data of individuals in a formal way. It is well known from the SMC literature that any Turing computable function can be computed among distributed parties without revealing the private inputs[28, 3]. Therefore, theoretically it is possible to perform distributed data mining without revealing data at individual sites. However, generalized solutions to SMC problem like Yao's

secure circuit evaluation[28] have very high communication and computation overhead, thus making infeasible to use in data mining problems where data size can be hundreds of GBs. Clifton et.al.[7] proposed that data mining algorithms make use of a relatively small set of primitive functions and therefore generalized SMC algorithms for these primitive functions can be used to solve data mining problems. However, even this solution suffers from large communication and computation overheads. Therefore, specialized algorithms should be developed for specific data mining problems.

The first privacy preserving data mining algorithm based on SMC is the work of Lindell and Pinkas[19] on decision tree classification. This algorithm however requires each bit of information sent between parties is encrypted, thus has a very high communication and computation cost. Clifton and Kantarcıoğlu[13] proposed an algorithm for association rule mining, based on the commutative encryption. Clifton and Vaidya[24] proposed an algorithm for k-means clustering over vertically partitioned data with using homomorphic encryption. Common to all these algorithms are high computation and communication overhead. Inan et.al. [12] give an efficient algorithm for clustering over horizontally partitioned data, however it is not entirely privacy preserving. It requires a trusted 3rd party and if any party in this algorithm colludes with the 3rd party, he can learn every other players input.

All the algorithms above rely on public key encryption schemes, which bring high computation overhead. This high computation overhead can be avoided with the use of secret sharing. The use of secret sharing to perform secure multiparty data mining gained some momentum in recent years. One of the most notable examples is the work of Laur, Lipmaa and Mielikainen[17]

in which they use secret sharing for private support vector classification. One other notable use of secret sharing in a privacy preserving data mining algorithm is the work of Wright and Yang[27] to compute Bayesian networks over vertically partitioned data. Similar to these works, our algorithm also relies on secret sharing.

4 Privacy Preserving Clustering

In this chapter, we present our privacy preserving clustering algorithms over vertically partitioned data. First, we present our k-means clustering algorithm in section 4.1 Privacy preserving k-means clustering over vertically partitioned data has been studied in the literature by Clifton and Vaidya[24]. Here, we improve on their algorithm. Clifton and Vaidya make use of additive homomorphic encryption which requires public key encryptions. The public key encryptions are the bottleneck of the method described in [24]. In contrast, we use additive secret sharing to achieve privacy, which gives us lower computation and communication overhead.

In section 4.2, we also show how our method can be used to build a dissimilarity matrix and can be used in a clustering algorithm that takes dissimilarity matrix as an input. In section 4.3 we explain how our method can work with trajectories with spatial and temporal shifts.

4.1 K-means Clustering Algorithm

K-means clustering is the most prominent and widely used clustering algorithm, also referred in the literature as *Lloyds algorithm*. The algorithm gets a set of points in the d-dimensional space and assigns each point into one of k clusters such that the sum of distances to the nearest center is minimized. While the algorithm is simple and therefore efficient, it has a few drawbacks. The drawbacks are that it is sensitive to outliers, tends to form only spherical shaped clusters and the number of clusters, k must be provided in advance.

Our privacy-preserving clustering algorithm is an improvement of the one

proposed by Clifton and Vaidya in [24]. The central difference is the how we do the search for the closest cluster a given object. Another difference is that we use secret sharing for secure compuatation of closest cluster whereas hmomorphic encryption is used in [24]. The power of secret sharing in this setting is that communication and computation overhead is considerably lower. To make our presentation clear, we simplify some parts of the algorithm, but apply the same simplifications to [24] when we compare the efficiency of the two algorithms in Section 6.3.

4.1.1 Problem Definition

Suppose there are R data holders, such that $R \geq 4$ with each one of them holding some attributes of the each entity in the dataset. The number of entities in the dataset is n. The goal of the r parties is to perform k-means clustering on their *aggregated* data without revealing the values of the attributes they own to the other parties. At the end of the algorithm, each entity in the dataset is assigned to one of k clusters and each party learns the cluster means corresponding to their own attributes, and the index of the cluster into which each entity is assigned. Ideally, no party should learn anything other than this.

Let μ_c, $c \in \{1, ..., k\}$, represent the cluster means of the result. Let μ_{ci} be the projection of cluster mean c onto the attributes of party i ($\mu_c = (\mu_{c1}, \ldots, \mu_{cr})$). As output of privacy preserving k-means clustering party i gets:

- The final mean μ_{ci} for each cluster $c \in \{1, \ldots, k\}$.
- The cluster index for each entity $j \in \{1, ..., n\}$.

Each data holder involved in the algorithm is assumed to be semi-honest, meaning each data holder will follow the algorithm as expected, but may store any information he receives during the execution of the algorithm.

The algorithm starts with choosing initial cluster means, then all entities in the dataset are assigned to the closest initial clusters. After the initial assignment of clusters, the cluster means are recalculated and each entity is reassigned to the cluster with the closest cluster mean. The process continues until a termination criterion is met. The initial assignment of cluster means greatly affects the final clustering results, a detailed study is given in[4]. In our algorithm, we assume the initial assignment is done randomly, as in[24].

Algorithm 1 shows the pseudo code of the privacy preserving k-means clustering algorithm.

The algorithm that we present in this paper terminates when there is no change in the assignment of clusters. The algorithm presented in [24] terminates when the change in cluster means between two iterations is less than a given threshold. Our termination criterion corresponds to setting the threshold to 0, which means the algorithm will terminate when there is no change in the cluster assignments.

Another factor to be considered is the distance measure used. For simplicity we use Euclidean distance, though our algorithm will work with every Minkowski distance. When we compare our work to the work of [24], we use Euclidean distance in their algorithm as well.

Since the data is vertically partitioned, each party can compute part of the distances between each of the n entities in the dataset and the cluster means. Since we use Euclidean distance the square of the total distance

Algorithm 1 Privacy Preserving k-means algorithm

do in parallel for each party $i \in \{1, \ldots, r\}$

  for each cluster $c \leftarrow 1, \ldots, k$ **do**

    initialize μ_{ci} randomly

  end for

end parallel

repeat

  for each entity $j \leftarrow 1, \ldots, n$ **do**

    Cluster[j] $\leftarrow$ SecurelyComputeClosestCluster(j)

  end for

  do in parallel for each party $i \in \{1, \ldots, r\}$

    for each cluster $c \leftarrow 1, \ldots, k$ **do**

      $\mu_{ci} \leftarrow$ mean of party i's attributes in cluster c

    end for

  end parallel

until termination criteria met

between an entity and a cluster mean is the sum of the squares of the sub-distances computed at the subspaces of each party:

$$\|x_i - \mu_c\|^2 = \sum_{p=1}^{r} \|x_{ip} - \mu_{cp}\|^2. \quad (1)$$

However, the parties cannot reveal their sub-distances in order to compute the sum of them, since the local sub-distances may contain private information. We therefore need to compute and compare the distances securely without revealing the individual sub-distances. This is done in the "SecurelyComputeClosestCluster" algorithm, which we will describe in the next subsection. It is in the secure computation of closest clusters that our algorithm varies from that of [24].

4.1.2 Secure Closest Cluster Computation

To compute the closest cluster mean for a given entity in the database we have to securely sum the sub-distances computed by each party and compare the results so that nothing other than the comparison result is learned. With n entities, the algorithm has to be invoked nt times, where t is the number of iterations in the standard k-means algorithm. In this section, we describe the algorithm for securely finding the cluster mean which is closest to a given entity.

The security of our algorithm relies on three ideas:

- Each party sends secret shares its sub-distance to all the other parties, and the sum of the sub-distances is computed on the secret shares.
- The comparison of distances is performed on secret shares so that only

the comparison result is learned. The actual values of the distances are not learned.

- The ordering of clusters is permuted so that for each entity in the database, only the index of the closest cluster is learned. Relative orderings of the entity's distances to each cluster mean μ_c cannot be learned. This is very simple when we work with secret shares, but in [24] this is the step that requires the highest amount of communication and computation since they rely heavily on public key encryptions.

The most important difference between our work and the work of Clifton and Vaidya is the secure computation of the closest cluster. In [24] the first party selects "disguising values" for each pair of cluster and party such that the sum of all disguising values corresponding to one cluster is zero. Then, the first party together with all other parties compute the encryption of the sub-distances plus the corresponding disguising value. Afterwards, the encrypted distances are permuted by the second party. Finally, the first party decrypts the distances, finds the minimal distance, and reveals the identity of the closest cluster with the help from party 2. The closest cluster corresponds to the new cluster assignment of the given entity. In contrast to Clifton and Vaidya, we use additive secret sharing, which allows us to compute all distances by locally adding up the correct shares. Our algorithm for securely computing the closest cluster mean for each entity has three phases. Pseudo code of these three phases are in Algorithm 2.

Phase 1: In the first phase of the secure closest cluster computation algorithm each party secret shares its sub-distance for each cluster mean with every other party. Let X_{ic} be the ith sub-distance between the entity

Algorithm 2 Secure Closest Cluster Computation

Require: entity e, cluster means $\mu_1, \ldots, \mu_k$

Ensure: Closest cluster to e

do in parallel for each party $i \in \{1, \ldots, r\}$

for each cluster $c \leftarrow 1, \ldots, k$ **do**

Phase 1 :

$X_{ic} \leftarrow$ local component of the distance from e to cluster mean μ_c

for every other party j **do**

$\alpha_{ij} \leftarrow$ random number

send α_{ij} to party j

end for

$\alpha_{ii} \leftarrow X_{ic} - \sum_{j \neq i} \alpha_{ij}$

Phase 2 :

$T_{ic} = \sum_j \alpha_{ji}$

send T_{ic} to party r (Party 1 does *not* send anything!)

end for

end parallel

Phase 3 : (Phase 3 only involves parties $1, 2, 3$ and r)

do in parallel for parties $i = 1, r$

for each cluster $c \leftarrow 1, \ldots, k$ **do**

Party 1: $D_{1c} \leftarrow T_{1c}$

Party r: $D_{rc} \leftarrow \sum_{i=2}^{r} T_{ic}$

end for

$(D'_{i1}, \ldots, D'_{ik}) \leftarrow$SecurePermute$(D_{i1}, \ldots, D_{ik})$

end parallel

return SecureFindMinimum$(D'_{i1}, \ldots, D'_{ik})$

that is being evaluated and the cluster mean c. Each party (i) creates a random number α_{ij}^c for every other party, and sends α_{ij}^c to party j for all $c \in \{1, \ldots, k\}$. The ith party keeps $\alpha_{ii}^c = X_{ic} - \sum_{j \neq i} \alpha_{ij}^c$ to himself. Note that X_{ic} cannot be computed unless all parties come together to recover it.

Phase 2: After the completion of Phase 1, for every cluster c, we let T_{ic} denote party i's secret share of the distance between the given entity and cluster mean c ($T_{ic} = \sum_{j=1}^{r} \alpha_{ji}^c$).

Since all α_{ij}^c, $i \neq j$, are random numbers, T_{ic} is also a random number, so nothing can be learned from it. Every party, apart from the first party, now sends T_{ic} to the rth party.

Phase 3: This phase only involves parties $1, 2, 3$ and r. Party r adds the values T_{ic} to compute $D_{rc} = \sum_{i=2}^{r} T_{ic}$. Party one defines $D_{1c} = T_{1c}$. The distance between the given entity and cluster mean c is now $D_c = D_{1c} + D_{rc}$. Since party 1 did not send T_{1c}, party r cannot learn the distance. The task of party 1 and r is now to find the minimum element from the list $(D_1, \ldots, D_k)$, where party 1 knows D_{1c}, and party r knows D_{rc} of each element. This is done by comparing each of the elements with the current smallest element one by one. The comparison presents two problems:

1. How do parties 1 and r compare two numbers $D_c = D_{1c} + D_{rc}$ and $D_{c'} = D_{1c'} + D_{rc'}$ such that neither of them will learn the values of D_c and $D_{c'}$?

2. How do we prevent parties 1 and r from learning the ordering of all the distances? They should only learn the minimum distance?

The first problem is solved by observing that $D_c < D_{c'}$ if and only if

$D_{1c} - D_{1c'} < D_{rc'} - D_{rc}$. Party 1 knows the left hand side, and party r knows the right hand side. They can now compute the result of the comparison by applying the so called "Yao's millionaires problem". We will describe this in detail in Sec. 4.1.4.

To solve the second problem above, parties 1 and r permute the order of the elements in the vector $(D_1, \ldots, D_k)$ with the help of party 2 and party 3. We discuss this further in Sec. 4.1.3 below.

Once Phase 3 is completed parties 1 and r can reveal the index of the cluster which is closest to the entity being considered.

4.1.3 Secure Permutation

Since parties 1 and r obtain the result of all the comparisons of the distances from a given entity to all the cluster-means they will not only know which cluster is closer to the given entity, but also know which cluster is furthest away, second furthest away and so forth. This clearly gives parties 1 and r an advantage over the other parties. To prevent parties 1 and r from obtaining this information, we first make a "secure permutation" of all the cluster distances. After finding the minimum element in the permuted list of distances parties 1 and r are only told the true identity of the cluster at minimum distance.

The secure permutation is the bottleneck for the algorithm by Clifton and Vaidya since it amounts to more than 90% if the computation time. The reason being that in their algorithm party 1 has to create kr public-key encryptions for each entity in the database.

In our algorithm we take advantage of secret sharing to make a consider-

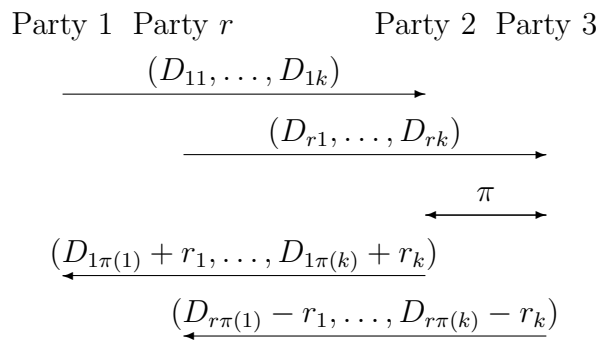

Figure 1: Secure Permutation Protocol

ably more efficient permutation algorithm. Parties 1 and r need the help of two other parties, parties 2 and 3, for the permutation. Party 1 simply sends his secret shares to party 2, and party r sends his secret shares to party 3. Then parties 2 and 3 agree on a permutation and each of them apply the permutation to the vector of shares they received. To make sure that parties 1 and r cannot recognize the elements in the vector they get back, party 2 adds a random number r_i to element i, while party 3 subtracts r_i (recall that each element of the vectors are additive secret shares of a distance, so adding and subtracting the same random number will not change the result of the algorithm). After applying the permutations, they send the vector back to the parties they got them from. The algorithm can be seen in Fig. 1.

4.1.4 Secure Minimum Element

In the last step of the "Secure Closest Cluster Computation" parties 1 and r compare the distances between the current entity and all the cluster means. Each of them has a secret share of the k entries in the permuted vector of these distances. To find the minimum, they perform $k - 1$ comparisons

with the current minimum element (they do not compare the first distance). After finding the minimal element, party 1 informs party 2 of the permuted identity of the closest cluster. Since party 2 knows the permutation, she can announce the real identity of the closest cluster to all parties.

Clifton and Vaidya[24] suggest using Yao's circuit evaluation[28] for the comparisons. They argue that even though Yao's algorithm is very inefficient, it may be plausible to use it for the comparison, since they only perform kn comparisons in each iteration of the k-means algorithm. Nonetheless, in our experiments, we use an algorithm proposed by Savaş, Pedersen, and Kaya in [16] in both our algorithm and our implementation of Clifton and Vaidya's algorithm.

In the comparison algorithm of [15, 16], two players who wish to compare two integers, each create two bitwise secret shares of their own inputs and send these shares to two semi-honest non-colluding third parties. Two $k-$bit integers x, y are compared according to the formula :

$$x > y \iff \bigvee_{i=1}^{k} (x_i \wedge \neg y_i \bigwedge_{j=i+1}^{k} (x_j \oplus y_j)) \tag{2}$$

The algorithm utilizes the fact that additive secret sharing is homomorphic with respect to addition, thus bitwise additive secret sharing is homomorphic with respect to bitwise addition, XOR operation of bits. The algorithm of [16], like Fischlin's algorithm[10], uses a trick by Sander, Yung, Young [23] to convert XOR homomorphic secret sharing into AND homomorphic secret sharing without disclosing the secret. For the proofs of correctness and security, the readers are referred to [16].

We have modified the algorithm from [16] slightly, by observing that the

two semi-honest third parties are actually not needed in the algorithm. In the original algorithm, the parties involved in secure comparison secret-share their inputs with 2 semi-honest third parties, sending one share to a third party and the remaining share to the other third party. This is equivalent to each party secret-sharing their inputs among themselves; each party sends a secret share of its input to the other party while keeping the remaining secret share to itself. Therefore, in our algorithm, the parties who are involved in the secure comparison (parties 1 and r) apply the algorithm in [16] by secret-sharing their inputs among themselves. In the original algorithm[16], two third parties end up with secret shares of a binary vector. The vector is as long as the inputs that are compared. If the first input to the comparison is greater than the other, then the vector has exactly one 1-bit at the first position where the first input is greater than the second input. Since the position of the 1-bit gives information about the relative difference between these two inputs, the two third parties agree on a permutation and permute the vector before it is sent to the party who will learn the result of the comparison. In our algorithm, parties 1 and r do not perform the permutation themselves since they are the ones that will learn the result of the comparison. Instead, we use the permutation algorithm described above in Sec. 4.1.3.

4.2 Other Clustering Methods

In this section, we show that our privacy preserving clustering scheme can be applied to any clustering algorithm that takes a dissimilarity matrix as an input, such as k-medoids and agglomerative hierarchical clustering(AGNES).

Pieces of our k-means algorithm, with little modification can be also used to compute a dissimilarity matrix of entities in the dataset.

Suppose there are n entities in the dataset. A dissimilarity matrix D is an $N \times N$ matrix, with one entry in the dissimilarity matrix, $D[a][b]$, corresponds to the distance between entities a and b in the dataset. Since the data is vertically partitioned, each party can compute part of the distances between the entity a and b in the dataset. Then, just as in our k-means algorithm each party will secret share its calculated subdistance with every other party. If this step is repeated for all the a, b entity pairs, then each party will have a $N \times N$ matrix D, with $D[a][b]$ containing a secret share of the distance between the points a and b. It can easily be seen that this step is exactly the same as step 1 of the algorithm "FindClosestCluster" given in section 4.1.2.

Once this dissimilarity matrix computation step is completed, it can be given as input to a dissimilarity matrix based clustering algorithm, such as AGNES. AGNES algorithm starts with every single entity in the dataset put into a cluster of its own. Then in bottom up fashion, closest pair of entities are merged into bigger clusters at every level. With a dissimilarity matrix given as an input to AGNES, closest pairs of entities and groups can be easily computed. However, revealing the dissimilarity matrices may cause some threats to privacy. Therefore we cannot simply sum the matrices each party holds and give to some party to continue with the AGNES algorithm. Instead, we can use Secure Comparison protocols we discussed in sections 4.1.4. Suppose we want to find out the whether entity a is closer to the entity b or entity c, we should compare if $D[a][b] < D[a][c]$, where D is the global dissimilarity matrix. Like step 2 in our "FindClosestCluster"

algorithm, every party apart from P_1 will send $D_i[a][b]$ and $D_i[a][c]$ values to P_r, where $D_i[a][b]$ and $D_i[a][c]$ are secret shares of $D[a][b]$ and $D[a][c]$ party i has. Afterwards, P_1 and P_r can privately compute $D[a][b] < D[a][c]$ and reveal the result to every other party.

4.3 Privacy Preserving Trajectory Clustering

Our k-means clustering algorithm can be applied to trajectories without any change, if trajectories are optimally aligned. Consider two trajectories T_A and T_B. Let points on T_A be (x_i, y_i) and (p_i, q_i). The distance between their positions ,d, on time i will be $d_i = \sqrt{(x_i - p_i)^2 + (y_i - q_i)^2}$ if we use Euclidean distance or $d_i = (x_i - p_i) + (y_i - q_i)$ if we use Manhattan distance. The total distance between T_A and T_B will be sum of d_i values. If Euclidean or Manhattan distance measures are used, our algorithm can be directly applied to trajectories as if it was ordinary numerical data.

4.3.1 Spatially Shifted Trajectories

Two trajectories can be very similar except for a small constant spatial shift. Then Euclidean distance or Manhattan distance may fail to realize the closeness of these two trajectories. The figure 2 shows two trajectories with a spatial shift between them.

Needham and Boyle[21] propose a metric for handling two such trajectories. They propose that the optimal shift,δ, between two trajectories T_A and T_B can be calculated by the formula :

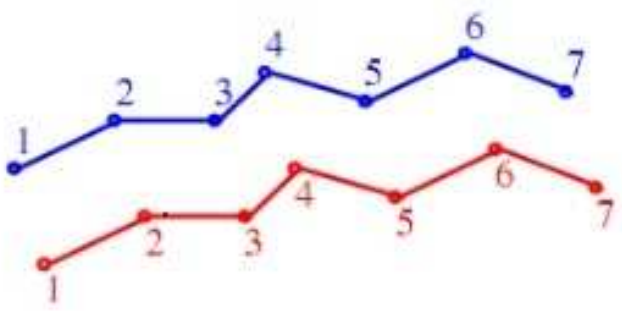

Figure 2: Spatially shifted trajectories

$$\delta = \mu(d_i) = \frac{1}{n}\sum_i d_i \tag{3}$$

The new distance between the trajectories can be calculated as Euclidean distance between T_B and T_A with all points of T_a shifted by δ. Since the data is vertically partitioned, every player P_i will be able to compute part of $\delta_i = \frac{1}{n}\sum_i d_i$ on the attributes he owns. δ is equal to sum of δ_i's of every player. Since δ_i values are private, every player secret share its own δ_i with every other player. This step is the same as step 1 of the algorithm 2, now every player can sum their δ_i's and reveal δ .

4.3.2 Temporally Shifted Trajectories

Two trajectories can be very similar except for a small constant time shift between them. Consider two moving objects, one following the same trajectory of the other, but only after some time. Euclidean distance may not be able to classify these two trajectories as similar trajectories. Figure 3 shows

two such trajectories.

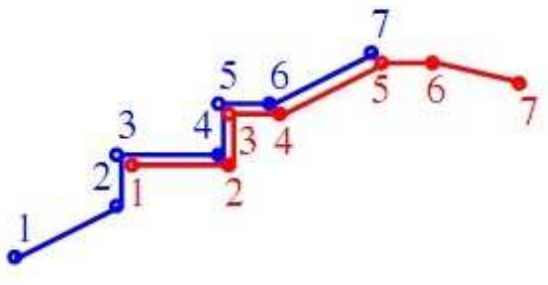

Figure 3: Two temporally shifted trajectories

Needham and Boyle[21] propose a metric for handling two such trajectories. The optimal time shift j between two trajectories T_A and T_B can be calculated by the formula

$$j = arg\ min_k(\frac{1}{n-k}\sum_i |(x_{i+k}, y_{i+k}) - (p_i, q_i)|). \tag{4}$$

Since the data is vertically partitioned (x_{i+k}, y_{i+k}) and (p_i, q_i) could be held by different players. In such case, the players secret share their points with each other. Suppose player a holds (p_i, q_i) points and player b holds (x_{i+k}, y_{i+k}) and they want to compute. To compute $x_{i+k} - p_i$, player b sends a random number r_b to player a, his share becomes $x_{i+k} - r_b$. Player a sends a random number r_a to player b, his share becomes $p_i - r_a$. Player b calculates

$$x_{i+k} - r_b - r_a.$$

Player a calculates

$$r_b - (p_i - r_a).$$

If player a and b add their shares, they will get $x_{i+k} - p_i$. At the end the each player will have a secret share of $\frac{1}{n-k} \sum_i |(x_{i+k}, y_{i+k}) - (p_i, q_i)|$. Using the Secure Minimum Element described in section 4.1.4, they can compute the minimum k without anyone learning other's input. Once the time shift j is known, the distance between T_B and the shifted trajectory T_A, j is calculated.

5 Privacy Discussion

In this section, we show why our algorithm preserves privacy(sec 5.1) and compare the security of our algorithm with that of Clifton and Vaidya[24](sec 5.2). Before going into the discussion about privacy, we have to define what we mean by private information in the algorithm. Above we set as our goal that the only information party i will learn after the algorithm is:

- The final mean μ_{ci} for each cluster $c \in \{1, \ldots, k\}$.
- The cluster index for each entity $j \in \{1, ..., n\}$.

No information other than these two should be learned out of algorithm. The actual values of data attributes belonging to the data holders are obviously private information. Since these values are not shared among data holders during the execution of our algorithm, we may say that these values are kept private. Portions of distances(X_{ic} values) calculated by each party according to their set of attributes are also deemed private information since one may recover the actual values of the entities by knowing the distances of the entities to the cluster means.

5.1 Privacy in our algorithm

In order to examine how our method protects privacy, we focus on the "Securely Computing the Closest Cluster" algorithm which is the only part of the algorithm in which there are interactions among the parties. In Phase 1 of the algorithm, all parties secret-share their local distance values with the other parties. Since each party keeps one share for himself and since *all*

shares are needed to recover the local distances, no information will leak even if all the remaining $r-1$ parties collude.

In Phase 2 of the algorithm all parties send their T_{ij} values to party r. Since T_{ij} are secret-shares of the total distance (it is the sum of the secret-shares of the local distances) and since party 1 does not send T_{1j} to party r, parties 1 and r cannot gain any information unless they collude. Notice that T_{1j} contains shares from all other parties' local distance components.

Phase 3 of the algorithm consists of the permutation and secure comparison sub-phases. In the permutation phase parties 1 and r send their secret shares of the distance to parties 2 and 3 respectively. Now parties 2 and 3 are in the same situation as parties 1 and r were after Phase 2. Collusion between parties 2 and 3 allow them to learn an entity's distance to each cluster mean, therefore the permutation phase is privacy-preserving under the assumption that parties 2 and 3 are non-colluding. The secure comparison is privacy-preserving under the assumption that parties 1 and r are non-colluding; detailed proof is explained in [16].

Finally the "Securely Computing the Closest Cluster" algorithm returns the closest cluster to the given entity. Clearly this is exactly what the parties are supposed to learn in the last round. However, all parties will see how entities change cluster in each iteration of the k-means clustering algorithm. An entity, which fluctuates between two clusters, can be assumed to be approximately half-way between the two cluster means. This is more information than is strictly allowed. However, since the data holders need to learn intermediate results to compute the new means, mitigation of this problem is extremely difficult if not impossible.

5.2 Security Comparison

In both our algorithm and the algorithm by Clifton and Vaidya [24], privacy breaches may occur when two or more parties collude. Therefore, in both of the algorithms, some non-collusion assumptions have to be made for some specific parties. However, the gained information in Clifton's and Vaidya's algorithm as a result of collusion is much more severe. In the algorithm by Clifton and Vaidya, each party, upon computing local distances X_{ic}, sends X_{ic} to party 1. Party 1 adds random values α_{ic} to X_{ic} of each party. After this phase, each party sends $X_{ic} + \alpha_{ic}$ to party r. Here, collusion between parties 1 and r allows them to learn the value of X_{ic} of all other parties, which is no doubt a severe privacy breach. In our algorithm such a privacy breach is mitigated by the use of secret sharing. Each party secret-shares X_{ic} with every other party, so the value of X_{ic} can be recovered only if all parties come together. As a solution to this problem, Clifton and Vaidya[24] propose an extension to their algorithm that increases the number of colluding parties needed to reveal X_{ic} of each party. In essence, they apply their permutation steps more than once according to a chosen anti-collusion parameter. If this parameter is denoted as p, they repeat the permutation algorithm $p - 1$ times by choosing a different party at each time to play the role of party 1. This method increases security of the algorithm; however, it also increases computation and communication cost considerably.

In our algorithm, we have non-collusion assumptions for parties $1, 2, 3$ and r. If the permuter parties (parties 2 or 3) collude, they can reveal each entity's distance to each cluster mean μ_c. Also, if parties $1, r$ and one of the permuter parties(2 or 3) collude, they can reveal each entity's distance to each

cluster mean μ_c. We can say that collusion between 2 specific parties(parties 2 and 3) leads to a privacy breach in our algorithm in the worst case. We can also do the same trick as in the algorithm of [24] and increase the non-collusion threshold by applying the permutation algorithm more than once, at each iteration picking different two parties to play the roles of party 2 and 3. Since our permutation algorithm does not contain any encryption or similar expensive operations, it can be applied more than once without bringing too much computation and communication overhead.

The comparison of the security of the two algorithms reveal that collusion between 2 specific parties leads to some privacy breach in both algorithms. However, in [24] the leaked information is more severe in case of a collusion. Some information specific to a party (X_{ic} for party i) can be learned in [24], whereas in our algorithm leaked information is global information, not bound to a specific party.

6 Cost Analysis

A privacy preserving distributed data mining algorithm aiming to be used in real life applications should not bring too much communication and computation overhead. In the following two subsections we analyze the communication and computation overheads of our algorithm. The overheads mainly occur in the "Securely Finding the Closest Cluster" part of the algorithm. Thus, we only analyze this portion of our algorithm. It should be noted that both communication and computation overheads of the k-means clustering algorithm depend on the dataset. The number of iterations required before the termination criteria is met depends on the data and the initial cluster means. Therefore, in the communication and computation cost analysis, we only consider one iteration of the k-means algorithm. We let r be the number of parties, n the number of entities in the database and k the number of clusters.

6.1 Communication Cost Analysis

Most of the communication overhead in our algorithm is created in Phase 1 of the "Securely Finding the Closest Cluster" algorithm. In this phase, for each entity, one party sends secret shares of its portion of the distance to every other party for each of the k clusters. This is equal to sending each party a vector of length k. Each entry of this vector is a secret share of 32 bits. Since there are r parties and each of them send a shared secret vector of length k to every other party, the communication cost of this step of our algorithm is $32r(r-1)kn$ bits. Therefore, Phase 1 of our algorithm has a

communication complexity of $\theta(r^2kn)$.

In the original work of Clifton and Vaidya[24], every party sends its local distance value *encrypted* to party 1. Party 1 adds random values to these local distance values by using the additive homomorphic property of the public key encryption scheme used, and sends the distorted values back to each data holder. Assuming that a public key encryption scheme with 1024 bits of key and block size is used, which is the minimum for security purposes, the communication cost of this phase in the work of Clifton and Vaidya is $2(r-1)1024kn = 2048(r-1)kn$ bits. This phase of Clifton and Vaidya's algorithm has a communication complexity of $\theta(nrk)$

Phase 2 of our algorithm is very similar to Clifton and Vaidya's algorithm in terms of communication cost, each party apart from party 1 will send a 32 bit integer to the r^{th} party, therefore the communication costs here are the same, $32n(r-2)k$ bits, giving a complexity of $\theta(nrk)$.

In Phase 3 of the algorithm we run an algorithm for Yao's Millionaires problem which gives us very little communication overhead. The only communication is one call to the permutation algorithm in each call to the comparison algorithm. The vectors permuted are of length 32λ, where λ is a security parameter given to the comparison algorithm. Typical values for λ are around 50. We have applied the same comparison algorithm in our implementation of the algorithm by Clifton and Vaidya, so the two algorithms have the same communication overhead in this step. This step is always executed between 2 parties, therefore complexity of this step is $\theta(nk)$

In total, our algorithm has a complexity of $\theta(nr^2k) + \theta(nrk) + \theta(rk) = \theta(nr^2k)$. Communication complexity of our algorithm is quadratic with the

number of parties. On the other hand, Clifton's[24] algorithm has a total communication complexity of $\theta(nrk) + \theta(nrk) + \theta(nk) = \theta(nrk)$, linear with the number of parties.

Although our algorithm has a higher communication complexity(quadratic vs linear), from the above analysis we observe that for values of r up to $2048/32 = 64$, our algorithm has smaller communication cost in phase 1 as compared to the work of Clifton and Vaidya[24](when using 1024-bit encryption in [24]). We confirm this bound in our experiments in Section 6.3.

6.2 Computation Cost Analysis

The computational complexity of our algorithm is $\theta(rnk)$ for each party involved in the algorithm. In Clifton's algorithm one party has the computational complexity of $\theta(nrk)$ whereas all the other parties have the computational complexity of $\theta(nk)$ However, in terms of computation overhead, our algorithm is more efficient than the algorithm of Clifton and Vaidya. Since we use secret sharing rather than encryption, our algorithm uses primitive operations only. However, the algorithm of Clifton and Vaidya uses public key encryptions which requires expensive modular exponentiation operations on very large numbers. In Phase 1 of Clifton and Vaidya's algorithm two public key encryptions and one public key decryption are needed for every entity and cluster pair. In total, $3nk$ modular exponentiations are needed in Clifton and Vaidya's algorithm. Given the primitive nature of the operations in our algorithm, our algorithm gives much lower computation overhead, which is confirmed by our experiments in Section 6.3.

6.3 Experimental Results

We implemented our algorithm and the algorithm of Clifton and Vaidya[24] and performed tests on them in order to validate the theoretical findings for both communication and computation costs given in Section 6.For testing purposes, we used two spatio-temporal datasets, consisting of moving object trajectories in the city of Milan. The datasets were provided to us by our GeoPKDD, 6^{th} Framework EU project, partners. One dataset has 150 trajectories while the second one has 600. The number of measurement points for each trajectory is over 1050. Since each measurement consists of one x and one y coordinate, each item in the dataset has over 2100 attributes. For both of the datasets, the attributes are partitioned among the parties evenly; that is, every party has equal or near equal number of attributes. However, the distribution of attributes does not affect neither of the two algorithms considerably (since the first step of each iteration is to compute local distances).

It is important to note that the initial cluster assignments of entities greatly affects the execution time of the k-means clustering algorithm. In order to make a fair comparison, we make sure that initial cluster assignments of the entities in the datasets are the same for each test for both of the protocols.

We made distributed implementations of both of the algorithms, with each data holder in both of the algorithms running as a separate process. Communication between processes is done with Message Passing Interface(MPI). The implementations are done in the C#.NET programming language and

the as an MPI library we used MPI.NET[1], developed by Indiana University. In the implementation of the protocol in [24], we use the Paillier[22] public key encryption scheme with 2048 bit cipher texts.

We have performed two tests with each of the datasets. First test is to see how much communication overhead our protocol brings and how the communication overhead in our protocol compares to communication overhead in the protocol of Clifton and Vaidya[24]. The total amounts of transmissions caused by the protocols with respect to the number of parties are depicted in Figures 4 and 5 for two different datasets.

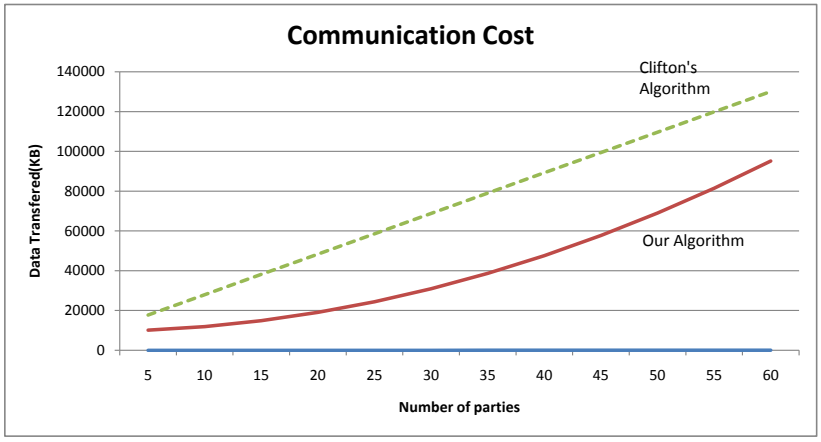

Figure 4: Communication cost(1st dataset)

[1]http://www.osl.iu.edu/research/mpi.net/

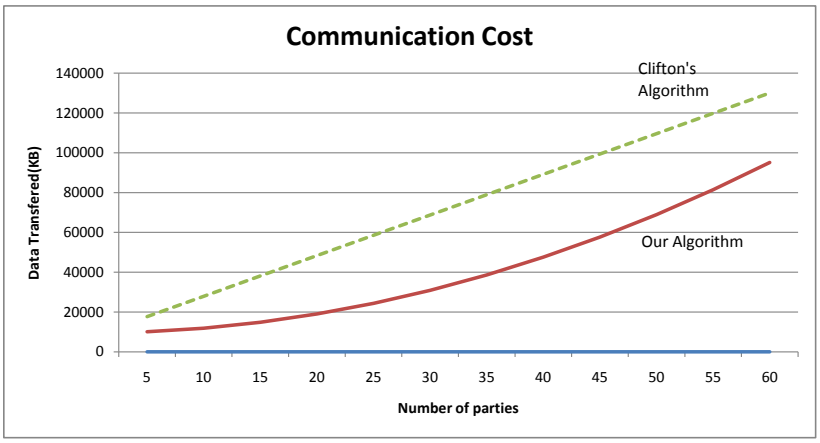

Figure 5: Communication cost(2nd dataset)

As can be seen from these figures, our protocol has lower communication cost. In the analysis section(6.1), we expected that our algorithm should have a lower communication cost up to a threshold. If 1024 bit encryption was used in the implementation of [24], this threshold was expected to be 64. Since we use Paillier encryption, which creates 2048 bit cypher texts, this threshold is higher in our figures.

The second test that we have performed is to analyze and compare the computational overheads brought by our protocol and the protocol of Clifton and Vaidya[24]. Execution times of the protocols with respect to the number of parties are shown in Figures 6 and 7 for the two datasets.

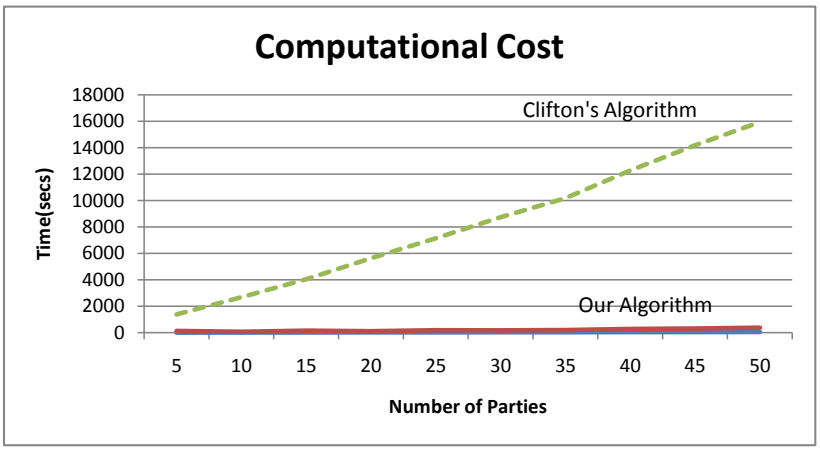

Figure 6: Computation cost(1st dataset)

The execution times of our protocol are always less than 400 seconds,for the 1st dataset and 2000 seconds for the second dataset. whereas the execution times of [24] are thousands of seconds. The reason of this significant performance difference lies in the use of public key cryptography. In order to compare each entity to each cluster mean, Clifton and Vaidya[24] have to perform expensive public key encryptions and decryptions linear in the number of parties. In our protocol, however, we do not rely on such computationally expensive public key operations. Although our algorithm has a $\theta(n^2)$ complexity, the most time consuming part in our algorithm is phase 3 of the algorithm 2. That phase is always run by 2 parties, that is the reason

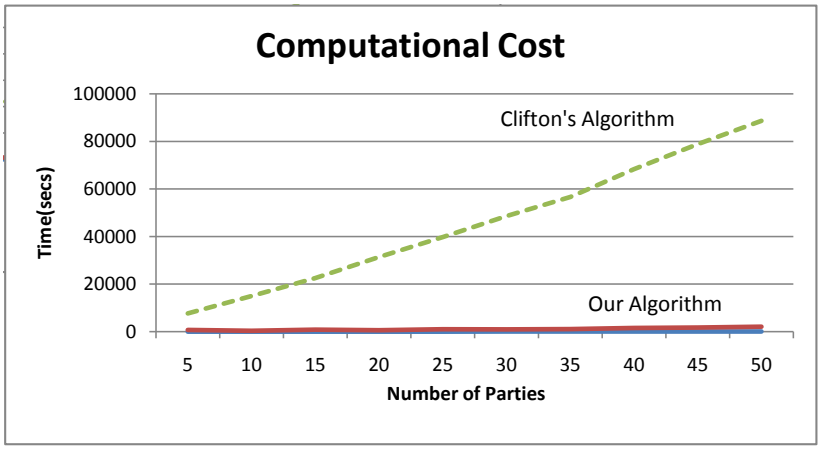

Figure 7: Computation cost(2nd dataset)

the figures do not show linear growth.

7 Conclusion

In this thesis, we studied the problem of distributed privacy preserving clustering over vertically partitioned data. We proposed algorithms for distributed k-means clustering algorithm and distributed dissimilarity matrix computation, which can be given as input to clustering algorithms such as AGNES or k-medoids. We also showed that although our algorithms are for generic numeric data, it can also handle some special properties of spatio-temporal data, i.e. moving object trajectories.

Our k-means clustering algorithm is an improvement of the work by Clifton and Vaidya[24]. By replacing computationally expensive public key encryptions with secret sharing, we showed that most of the communication and computation overhead can be avoided. Although our algorithm show worse asymptotic behavior than the work in [24], a more detailed analysis and experimental results show that our algorithm has a much better communication and computation overhead up to a reasonable amount of parties.

Secret sharing based privacy preserving data mining algorithms not only helps us archive better computation and communication overhead, but also provide better security as well. By using secret sharing, we were able to get rid of non-collusion assumptions among involved parties, or at least minimize the value information gained by colluding parties.

We believe concerns about user privacy will increase in the foreseeable future, developing new privacy preserving data mining algorithms or improving existing ones will still be an important research problem. Only secure multiparty computation provides "true" privacy among data mining algorithms. In this thesis we showed that SMC based privacy preserving data mining

algorithms do not have to suffer from high overheads, making them posses good practical value.

References

[1] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 439–450. ACM, 2000.

[2] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. In *SIGMOD '99: Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pages 49–60, New York, NY, USA, 1999. ACM.

[3] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10, New York, NY, USA, 1988. ACM.

[4] Paul S. Bradley and Usama M. Fayyad. Refining initial points for k-means clustering. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 91–99, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[5] L. Chen and R. Ng. The marriage of lp-norms and edit distance, 2004.

[6] Lei Chen, M. Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 491–502, New York, NY, USA, 2005. ACM.

[7] Chris Clifton, Murat Kantarcioglu, Jaideep Vaidya, Xiaodong Lin, and Michael Y. Zhu. Tools for privacy preserving distributed data mining. *SIGKDD Explor. Newsl.*, 4(2):28–34, 2002.

[8] Martin Ester, Hans peter Kriegel, Jrg S, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[9] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules, 2002.

[10] Marc Fischlin. A cost-effective pay-per-multiplication comparison method for millionaires. In *Progress in Cryptology - CT-RSA 2001: The Cryptographers' Track at RSA Conference 2001*, volume 2020 of *Lecture Notes in Computer Science*, page 457, 2001.

[11] Scott Gaffney and Padhraic Smyth. Trajectory clustering with mixtures of regression models. In *Knowledge Discovery and Data Mining*, pages 63–72, 1999.

[12] Ali Inan, Yücel Saygın, Erkay Savaş, Ayça Azgın Hintoğlu, and Albert Levi. Privacy preserving clustering on horizontally partitioned data. In *22nd International Conference on Data Engineering Workshops (ICDEW'06)*, page 95. IEEE Computer Society, 2006.

[13] Murat Kantarcioglu and Chris Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Trans. Knowl. Data Eng.*, 16(9):1026–1037, 2004.

[14] Murat Kantarcioğlu, Jiashun Jin, and Chris Clifton. When do data mining results violate privacy? In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 599–604, New York, NY, USA, 2004. ACM.

[15] S. V. Kaya, T. B. Pedersen, E. Savaş, and Y. Saygin. Efficient privacy preserving distributed clustering based on secret sharing. In *PAKDD 2007 International Workshops: Emerging Technologies in Knowledge Discovery and Data Mining*, pages 280–291. Springer, 2007.

[16] Selim Volkan Kaya. Toolbox for Privacy Preserving Data Mining. Master's thesis, Sabanci University, Istanbul, TURKEY, July 2007.

[17] Sven Laur, Helger Lipmaa, and Taneli Mielikäinen. Cryptographically private support vector machines. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 618–624. ACM, 2006.

[18] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 593–604, New York, NY, USA, 2007. ACM.

[19] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. *Lecture Notes in Computer Science*, 1880:36–??, 2000.

[20] Kun Liu, Hillol Kargupta, and Jessica Ryan. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Trans. Knowl. Data Eng.*, 18(1):92–106, 2006.

[21] Chris J. Needham and Roger D. Boyle. Performance evaluation metrics and statistics for positional tracker evaluation. In *ICVS*, pages 278–289, 2003.

[22] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology — EUROCRYPT '99. International Conference on the Theory and Application of Cryptographic Techniques*, Lecture Notes in Computer Science, pages 223–238. Springer-Verlag, May 1999.

[23] Tomas Sander, Adam Young, and Moti Yung. Non-interactive cryptocomputing for nc1. In *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 554, Washington, DC, USA, 1999. IEEE Computer Society.

[24] Jaideep Vaidya and Chris Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206–215, New York, NY, USA, 2003. ACM Press.

[25] Jaikumar Vijayan. House committee chair wants info on cancelled dhs data-mining programs. *Computer World*, September 18 2007.

[26] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories, 2002.

[27] Rebecca Wright and Zhiqiang Yang. Privacy-preserving bayesian network structure computation on distributed heterogeneous data. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on*

Knowledge discovery and data mining, pages 713–718, New York, NY, USA, 2004. ACM.

[28] Andrew C. Yao. Protocols for secure computations. In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science (FOCS '82)*, pages 160–164, 1982.