Privacy Preserving Clustering on Horizontally Partitioned Data

Ali İnan, Yücel Saygın, Erkay Savaş, Ayça Azgın Hintoğlu, Albert Levi
Faculty of Engineering and Natural Sciences, Sabancı University, Istanbul, Turkey
{inanali, aycah}@su.sabanciuniv.edu
{ysaygin, erkays, levi}@sabanciuniv.edu

Abstract

Data mining has been a popular research area for more than a decade due to its vast spectrum of applications. The power of data mining tools to extract hidden information that cannot be otherwise seen by simple querying proved to be useful. However, the popularity and wide availability of data mining tools also raised concerns about the privacy of individuals. The aim of privacy preserving data mining researchers is to develop data mining techniques that could be applied on databases without violating the privacy of individuals. Privacy preserving techniques for various data mining models have been proposed, initially for classification on centralized data then for association rules in distributed environments. In this work, we propose methods for constructing the dissimilarity matrix of objects from different sites in a privacy preserving manner which can be used for privacy preserving clustering as well as database joins, record linkage and other operations that require pair-wise comparison of individual private data objects horizontally distributed to multiple sites.

1. Introduction

Data mining research deals with the extraction of potentially useful information from large collections of data with a variety of application areas such as customer relationship management, market basket analysis, and bioinformatics. The extracted information could be in the form of patterns, clusters or classification models. Association rules in a supermarket for example could describe the relationship among items bought together. Customers could be clustered in segments for better customer relationship management. Classification models could be built on customer profiles and shopping behavior to do targeted marketing.

The power of data mining tools to extract hidden information from large collections of data lead to increased data collection efforts by companies and government agencies. Naturally this raised privacy concerns about collected data. In response to that, data mining researchers started to address privacy concerns by developing special data mining techniques under the framework of "privacy preserving data mining". Opposed to regular data mining techniques, privacy preserving data mining can be applied to databases without violating the privacy of individuals. Privacy preserving techniques for many data mining models have been proposed in the past 5 years. Agrawal and Srikant from IBM Almaden proposed data perturbation techniques for privacy preserving classification model construction on centralized data [10]. Techniques for privacy preserving association rule mining in distributed environments were proposed by Kantarcioglu and Clifton [11]. Privacy preserving techniques for clustering over vertically partitioned data was proposed by Vaidya and Clifton [12]. In this paper, we propose a privacy preserving clustering technique on horizontally partitioned data. Our method is based on constructing the dissimilarity matrix of objects from different sites in a privacy preserving manner which can be used for privacy preserving clustering, as well as database joins, record linkage and other operations that require pair-wise comparison of individual private data objects distributed over multiple sites. We handle both numeric and alphanumeric valued attributes.

Most clustering algorithms handle only numerical and categorical data. However, especially in the area of bioinformatics, researchers may be interested in clustering alphanumeric data as well. Consider the following real-world problem: Several institutions are gathering DNA data of individuals infected with bird flu and want to cluster this data in order to diagnose the disease. Since DNA data is private, these institutions can not simply aggregate their data for processing but should run a privacy preserving clustering protocol.

Our main contributions can be summarized as follows:

- Introduction of a protocol for secure multi-party computation of a dissimilarity matrix over horizontally partitioned data,
- Proof of security of the protocol, analysis of communication costs
- Secure comparison methods for numeric, alphanumeric and categorical attributes,
- Application to privacy preserving clustering.

In the following sections, we provide the background and related work together with the problem formulation. We then provide privacy preserving comparison protocols for numeric, alphanumeric and categorical attributes that will later be used for dissimilarity matrix construction.

The global dissimilarity matrix is a generic data structure that describes the pair-wise distances among objects and it can be used by any standard clustering algorithm, therefore we only provide the communication cost and security analysis of our privacy preserving comparison protocols.

2. Related work and background

Researchers developed methods to enable data mining techniques to be applied while preserving the privacy of individuals. Mainly two approaches are employed in these methods: data sanitization and secure multi-party computation. Data mining on sanitized data results in loss of accuracy, while secure multi-party computation protocols give accurate results at the expense of high computation or communication costs. Most data mining techniques, i.e. association rule mining and classification, are well studied by followers of both approaches. [9] and [10] are data sanitization techniques; [11], [12] and [13] are based on secure multi-party computation techniques.

Privacy preserving clustering is not studied as intensively as other data mining techniques. In [1] and [2], Oliveira and Zaïane focus on different transformation techniques that enable the data owner to share the mining data with another party who will cluster it. In [3], they propose new methods for clustering centralized data: dimensionality reduction and object similarity based representation. Methods in [3] are also applicable on vertically partitioned data, in which case each partition is transformed by its owner and joined by one of the involved parties who will construct a dissimilarity matrix to be input to hierarchical clustering algorithms. [4] and [5] propose model-based solutions for the privacy preserving clustering problem. Data holder parties build local models of their data that is subject to privacy constraints. Then a third party builds a global model from these local models and cluster the data generated by this global model. All of these works follow the sanitization approach and therefore trade-off accuracy versus privacy. Except [4], none of them address privacy preserving clustering on horizontally partitioned data.

Clifton and Vaidya propose a secure multi-party computation of k-means algorithm on vertically partitioned data in [6]. More recent work in [7] by Kruger et al. proposes a privacy preserving, distributed k-means protocol on horizontally partitioned data. We primarily focus on hierarchical clustering methods in this paper, rather than partitioning methods that tend to result in spherical clusters. Hierarchical methods can both discover clusters of arbitrary shapes and deal with different data types. For example, partitioning algorithms can not handle string data type for which a “mean” is not defined.

Private comparison of sequences of letters chosen from a finite alphabet is also related to our work since we aim at clustering alphanumeric attributes as well. Atallah et al. proposes an edit distance computation protocol in [8]. The algorithm is not feasible for clustering private data due to high communication costs. We propose a new protocol for comparing alphanumeric attributes.

Our work is closest to [3] and [7] since we consider the problem of privacy preserving clustering on horizontally partitioned data by means of secure multi-party computation of the global dissimilarity matrix which can then be input to hierarchical clustering methods. There is no loss of accuracy as is the case in [3], clustering method is independent of partitioning methods and alphanumeric attributes can be used in clustering with the proposed protocols. Our dissimilarity matrix construction algorithm is also applicable to privacy preserving record linkage and outlier detection problems.

2.1. Data matrix

A data matrix is an object-by-variable structure. Each row in the matrix represents an entity through values of its attributes stored in columns. An m×n data matrix is the data of m objects on n attributes as depicted in Figure 1.

$$D = \begin{bmatrix} a_{11} & \cdots & a_{1j} & \cdots & a_{1n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{i1} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} & \cdots & a_{mj} & \cdots & a_{mn} \end{bmatrix}$$

Figure 1. Data matrix

The attributes of objects are not necessarily chosen from the same domain. In this paper, we focus on categorical, numerical and alphanumerical attributes. Each data type requires different comparison functions and corresponding protocols. Therefore local data matrices are usually accessed in columns, denoted as D_i for the i^{th} attribute of a data matrix D. Notice that D_i is a vector of size m. Size of D_i is denoted as D_i.Length.

Data matrix D is said to be horizontally partitioned if rows of D are distributed among different parties. In that case each party j has its own data matrix, denoted as D_j. D_{ji} denotes the i^{th} column of party j’s data matrix, D_j.

Data matrix is not normalized in our protocol. We rather choose to normalize the dissimilarity matrix. The reason is that each horizontal partition may contain values from a different range in which case another privacy preserving protocol for finding the global minimum and maximum of each attribute would be required. Normalization on the dissimilarity matrix yields the same effect, without loss of accuracy and the need for another protocol.

2.2. Dissimilarity matrix

A dissimilarity matrix is an object-by-object structure. An m×m dissimilarity matrix stores the distance or dissimilarity between each pair of objects as depicted in Figure 2. Intuitively, the distance of an object to itself is 0. Also notice that only the entries below the diagonal are filled, since d[i][j] = d[j][i], due to symmetric nature of the comparison functions.

$$d = \begin{bmatrix} 0 & & & & \\ dist(2,1) & 0 & & & \\ dist(3,1) & dist(3,2) & 0 & & \\ \vdots & \vdots & \vdots & 0 & \\ dist(n,1) & dist(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

Figure 2. Dissimilarity matrix

Each party that holds a horizontal partition of a data matrix D can construct its local dissimilarity matrix as long as comparison functions for object attributes are public. However privacy preserving comparison protocols need to be employed in order to calculate an entry d[i][j] of dissimilarity matrix d, if objects i and j are not held by the same party.

Involved parties construct separate dissimilarity matrices for each attribute in our protocol. Then these matrices are merged into a single matrix using a weight function on the attributes. The dissimilarity matrix for attribute i is denoted as d_i.

2.3. Comparison functions

The distance between two numeric attributes is simply the absolute value of the difference between them. Categorical attributes are only compared for equality so that any categorical value is equally distant to all other values but itself. The distance between alphanumeric attributes is measured in terms of the edit distance that is heavily used in bioinformatics.

Edit distance algorithm returns the number of operations required to transform a source string into a target string. Available operations are insertion, deletion and transformation of a character. The algorithm makes use of the dynamic programming paradigm. An (n+1) × (m+1) matrix is iteratively filled, where n and m are lengths of source and target strings respectively. Comparison result is symmetric with respect to the inputs. Details of the algorithm can be found in [8].

Input of the edit distance algorithm need not be the input strings. An n×m equality comparison matrix for all pairs of characters in source and target strings is equally expressive. We call such matrices "character comparison matrices" and denote them as CCM_{ST} for source string s and target string t. $CCM_{ST}[i][j]$ is 0 if the ith character of s is equal to the jth character of t and non-zero otherwise.

3. Problem formulation

In this section, we formally define the problem, give details on trust levels of the involved parties and the amount of preliminary information that must be known by each one.

There are k data holders, such that $k \geq 2$, each of which owns a horizontal partition of the data matrix D, denoted as D_k. These parties want to cluster their data by means of a third party so that the clustering results will be public to data holders at the end of the protocol. The third party, denoted as TP, does not have any data but serves as a means of computation power and storage space. Third party's duty in the protocol is to govern the communication between data holders, construct the dissimilarity matrix and publish clustering results to data holders.

Every party, including the third party, is semi-trusted, that is to say, they follow the protocol as they are supposed to but may store any data that is revealed to them in order to infer private information in the future. Semi-trusted behavior is also called honest-but-curious behavior. Involved parties are also assumed to be non-colluding.

Every data holder and the third party must have access to the comparison functions so that they can compute distance/dissimilarity between objects. Data holders are supposed to have previously agreed on the list of attributes that are going to be used for clustering beforehand. This attribute list is also shared with the third party so that TP can run appropriate comparison functions for different data types.

At the end of the protocol, the third party will have constructed the dissimilarity matrices for each attribute separately. These dissimilarity matrices are weighted using a weight vector sent by data holders such that the final dissimilarity matrix is built. Then the third party runs a hierarchical clustering algorithm on the final dissimilarity matrix and publishes the results. Every data holder can impose a different weight vector and clustering algorithm of his own choice.

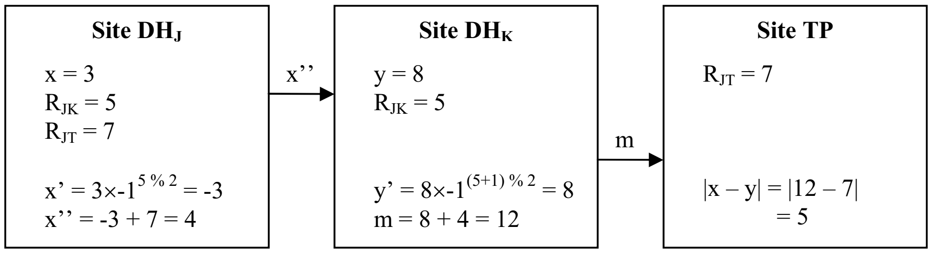

Figure 3. Sample numeric data comparison

4. Comparison protocols

As described in Section 2.2, dissimilarity matrix, d, is an object-by-object structure in which d[i][j] is the distance between objects i and j. Now consider an entry in d, d[i][j]: If both objects are held by the same data holder, the third party need not intervene in the computation of the distance between objects i and j. The data holder that is holding these objects simply computes the distance between them and sends the result to the third party. However, if objects i and j are from different parties, a privacy preserving comparison protocol must be employed between owners of these objects. It follows from this distinction that, in order to construct the dissimilarity matrix, each data holder must send its local dissimilarity matrix to the third party and also run a privacy preserving comparison protocol with every other data holder. Evidently, if there are k data holders, comparison protocol for each attribute must be repeated C(n, 2) times, for every pair of data holders.

In this section we describe the comparison protocols for numeric, alphanumeric and categorical attributes that will be the building blocks of the dissimilarity matrix construction algorithm explained in the next section. Security and communication cost analysis of the protocols are provided right after the protocols are presented.

4.1. Numeric attributes

Distance function for numeric attributes is the absolute value of the difference between the values, formally:

$$\text{distance}(x, y) = |\, x - y \,|.$$

There are three participants of the protocol: data holders DH_J and DH_K and the third party, TP. DH_J and DH_K share a secret number $r_{J\,K}$ that will be used as the seed of a pseudo-random number generator, rngJK. Similarly DH_J and TP share a secret number $r_{J\,T}$ that will be used as the seed of a pseudo-random number generator, rngJT.

Suppose that parties DH_J and DH_K want to compute the distance between two integers x and y, held by each site respectively. A naïve protocol for comparing x and y without using the random number generator rngJK would be as follows: DH_J generates a random number r using rngJT and sends (r+x) to DH_K. Since r disguises the value of x, there is no information leak. DH_K then subtracts y from (r+x) and sends (r+x-y) to TP, who can generate the same random number r, since TP and DH_J share the seed of rngJT. TP subtracts r from (r+x-y) and successfully computes |x - y|, result of the distance function. However in this protocol TP can find out whether x is larger than y or vice versa. A positive distance value implies that x > y because y is subtracted from x.

In order to prevent possible inferences based on the above information leak, the input that will be negated should be chosen arbitrarily. What's more is that both DH_J and DH_K should know which input will be negated. Shared random number generator rngJK addresses this problem. If the random number generated by rngJK is odd, DH_J negates its input. Otherwise, DH_K does. The protocol is demonstrated in Figure 3 below. x and y are private inputs of data holder sites and R_{JK} and R_{JT} are the random numbers generated by pseudo-random number generators rngJK and rngJT respectively.

Sites DH_J, DH_K and TP can privately compute the distance function using this refined protocol. However the protocol has to be repeated for each object pair. A final adjustment solves this problem and enables batch processing of inputs at all sites. Site DH_J processes each value in DH_J (data vector at site j) using both rngJK and rngJT and sends the resulting vector to site DH_K. Site DH_K initializes a matrix of size DH_K .Length × DH_J .Length to be sent to site TP. This matrix is filled row by row, each row containing intermediary comparison values for an input from DH_K against all inputs from DH_J. At the end of each row, DH_K should re-initialize rngJK using the seed $r_{J\,K}$ to be able to remember the oddness/evenness of the random numbers generated at site DH_J. Finally, site TP, upon receiving this matrix, subtracts the random numbers generated by rngJT. Similarly TP should re-initialize rngJT at each row, since all numbers in a column are disguised using the same random number by DH_J.

```
INPUT: Pseudo-random number generator seeds $r_{J\,K}$ and $r_{J\,T}$, Input data vector $DH_J$
OUTPUT: Data vector $DH'_J$
Begin
    1. Initialize rngJK with seed $r_{J\,K}$
    2. Initialize rngJT with seed $r_{J\,T}$
    3. Initialize data vector: $DH'_J = \{DH_J .Length\}$
    For m = 0 to $DH_J$ .Length-1
        4. $DH'_J [m] = rngJT.Next( ) + DH_J [m] \times -1^{rngJK.Next( )\%2}$
    5. Send $DH'_J$ to site $DH_K$
End
```

Figure 4. Pseudocode of numeric attribute comparison at site DH_J

```
INPUT: Pseudo-random number generator seed $r_{J\,K}$, Input data vectors $DH_K$ and $DH'_J$
OUTPUT: Pair-wise comparison matrix s
Begin
    1. Initialize rngJT with seed $r_{J\,T}$
    2. Initialize comparison matrix: $s = \{DH_K .Length \times DH'_J .Length\}$
    For m = 0 to $DH_K$ .Length-1
        For n = 0 to $DH'_J$ .Length-1
            3. $s[m] [n] = DH'_J [n] + DH_K [m] \times -1^{(rngJK.Next( ) + 1)\%2}$
        4. Re-initialize rngJK with seed $r_{J\,K}$
    5. Send s to size TP
End
```

Figure 5. Pseudocode of numeric attribute comparison at site DH_K

```
INPUT: Pseudo-random number generator seed $r_{J\,T}$, Pair-wise comparison matrix s
OUTPUT: Dissimilarity matrix component for $DH_J$ and $DH_K$, J_K
Begin
    1. Initialize rngJT with seed $r_{J\,T}$
    2. Initialize matrix of distance values: $J\_K = \{s.Length \times s[0].Length\}$
    For m = 0 to s.Length-1
        For n = 0 to s[0].Length-1
            3. $J\_K[m] [n] = |\, s[m] [n] - rngJT.Next( ) \,|$
        4. Re-initialize rngJT with seed $r_{J\,T}$
End
```

Figure 6. Pseudocode of numeric attribute comparison at site TP

Pseudocode for the comparison protocol of integers is given in Figures 4, 5 and 6. For other data types, i.e. real values, only data type of the vector DH'_J and the random numbers generated by rngJT need to be changed.

Analysis of communication costs and privacy

We denote data sizes for parties DH_J and DH_K as n and m respectively. Communication cost of the protocol at the initiator site, DH_J, is $O(n^2+n)$. $O(n^2)$ is the cost of sending local dissimilarity matrix to the third party and O(n) is the cost of sending disguised values to site DH_K. $O(n^2+n)$ is practically $O(n^2)$ for large values of n. At site DH_K, communication cost of the protocol is $O(m^2+m\times n)$. Again $O(m^2)$ is the local dissimilarity matrix cost and $O(m\times n)$ is the cost of sending matrix s to the third party.

Our comparison protocol preserves privacy as long as a high quality pseudo-random number generator, that has a long period and that is not predictable, is used and the channels are secured. Proposing such a pseudo-random number generator or analyzing existing generators is out of the scope of this work. We assume that an appropriate pseudo-random number generator is available.

The message x'' is practically a random number for party DH_K since the pseudo-random number generated using rngJT is not available. Inference by TP, using message m, is not possible since given $\pm(x-y)$ there are infinitely many points that are $|\, x - y \,|$ apart of each other. The very same idea is used and explained in detail in Section 5.

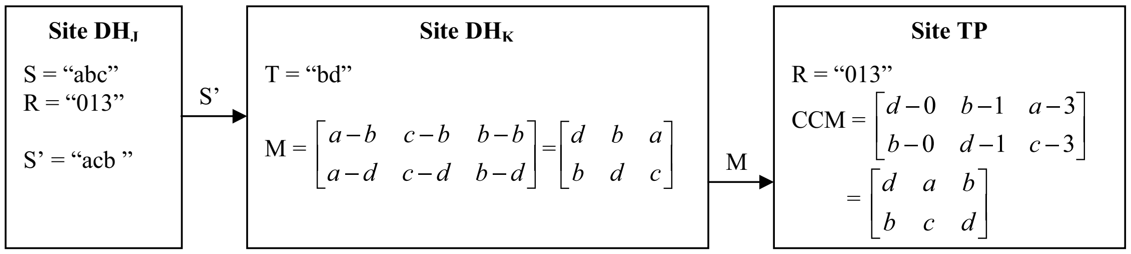
Site DH$_J$
S = "abc"
R = "013"
S' = "acb "
S'
Site DH$_K$
T = "bd"
$M = \begin{bmatrix} a-b & c-b & b-b \\ a-d & c-d & b-d \end{bmatrix} = \begin{bmatrix} d & b & a \\ b & d & c \end{bmatrix}$
M
Site TP
R = "013"
$CCM = \begin{bmatrix} d-0 & b-1 & a-3 \\ b-0 & d-1 & c-3 \end{bmatrix} = \begin{bmatrix} d & a & b \\ b & c & d \end{bmatrix}$

Figure 7. Sample alphanumeric data comparison

We now explain the reason why the channels must be secured. TP can predict the values of both x and y if he listens the channel between DH$_J$ and DH$_K$. Notice that x'' = r ± x and TP knows the value of r. Therefore he infers that the value of x is either (x''– r) or (r – x''). For each possible value of x, y can take two values: either (x – | x – y |) or (x + | x – y |). In order to prevent such inferences, the channel between DH$_J$ and DH$_K$ must be secured. Another thread is eavesdropping by DH$_J$ on the channel between DH$_K$ and TP. This channel carries the message m = r ± (x – y) and DH$_J$ knows the values of both r and x. Therefore this channel must be secured as well.

Although comparing a pair of numeric inputs using our protocol preserves privacy, as shown above, comparing many pairs at once introduces a thread of privacy breaches by means of a frequency analysis attack by the third party, TP. Notice that ith column of the pair-wise comparison matrix s, received by TP from DH$_K$, is "private data vector of DH$_K$" plus "identity vector times (ith input of DH$_J$ – ith random number of rngJT)" or negation of the expression. If the range of values for numeric attributes is limited and there is enough statistics to realize a frequency attack, TP can infer input values of site DH$_K$. In such cases, site DH$_K$ can request omitting batch processing of inputs and using unique random numbers for each object pair.

4.2. Alphanumeric attributes

Alphanumeric attributes are compared in terms of the edit distance. Similar to the comparison protocol for numeric attributes, two data holder parties and a third party are involved in the protocol. While data holders DH$_J$ and DH$_K$ build their local dissimilarity matrices using the edit distance algorithm, third party can not use this algorithm to compare inputs from distinct data sources since input strings, source and target, are private information in the latter case. Therefore we devise a new protocol below, which allows the third party to securely compute the distance between all inputs of data holders. The idea behind the protocol is privately building a character comparison matrix (CCM), as explained in Section 2.3.

Alphabet of the strings that are to be compared is assumed to be finite. This assumption enables modulo operations on alphabet size, such that addition of a random number and a character is another alphabet character.

The third party and data holder party DH$_J$ shares a secret number which will be seed of a pseudo-random number generator. In order to compare a string s at site DH$_J$ with a string t at site DH$_K$, DH$_J$ first generates a random vector of size s.Length and add this random vector to s. The outcome, denoted as s', is practically a random vector for DH$_K$. DH$_J$ sends s' to DH$_K$, who generates t.Length copies of s' and builds a matrix of size (t.Length)×(s'.Length). Then DH$_K$ subtracts its input, t, from every column of the matrix and sends the resultant matrix to the third party. The third party can generate the same pseudo-random numbers as DH$_J$ since they share the seed to the generator. Finally the third party generates the same random vector of size s'.Length as DH$_J$ did and subtracts this vector from every row of the matrix received from DH$_K$ to obtain the CCM for s and t.

A simple demonstration of the protocol is given in Figure 7. R is the random vector generated using the pseudo-random number generator. Alphabet is assumed to be A = { a, b, c, d }. Site TP can construct the CCM for the compared strings: CCM[0][1] = a = 0, which implies that s[1] = t[0], as is the case.

Comparison of multiple strings at once requires an extension to the above protocol: DH$_J$ re-initializes its pseudo-random number generator with the same seed after disguising each input string. Correspondingly, the third party should re-initialize its generator after successfully decoding each CCM. Pseudo-code for this extended comparison protocol is given in Figures 8, 9 and 10.

Analysis of communication costs and privacy

Data holder party DH$_J$ sends its local dissimilarity matrix to the third party and a vector of exactly the same size of its input to DH$_K$. If DH$_J$ has n inputs of at most p characters, its communication cost is $O(n^2+n\times p)$.

```
INPUT: Pseudo-random number generator seed $r_{JT}$, Input data vector $DH_J$
OUTPUT: Data vector $DH'_J$
Begin
    1. Initialize rngJT with seed $r_{JT}$
    2. Initialize data vector: $DH'_J = \{DH_J . Length\}$
    For m = 0 to $DH_J$ .Length-1
        For p = 0 to $DH_J$ [m] . Length-1
            3.  $DH'_J$ [m] [p] = $DH_J$ [m] [p] + rngJT.Next( )
        4. Re-initialize rngJT with seed $r_{JT}$
    5. Send $DH'_J$ to site $DH_K$
End
```

Figure 8. Pseudocode of alphanumeric attribute comparison at site DH_J

```
INPUT: Input data vectors $DH_K$ and $DH'_J$
OUTPUT: Pair-wise, intermediary CCM matrix M
Begin
    1. Initialize CCM matrix: $M = \{DH_K . Length \times DH_J . Length\}$
    For m = 0 to $DH_K$ .Length-1
        For n = 0 to $DH'_J$ .Length-1
            2. Initialize CCM: M [m] [n] = $\{DH_K [m].Length \times DH'_J [n].Length\}$
            For q = 0 to $DH_K$ [m] . Length-1
                For p = 0 to $DH'_J$ [n] . Length-1
                    3. M [m] [n] [q] [p] = $DH'_J$ [n] [p] – $DH_K$ [m] [q]
    4. Send M to site TP
End
```

Figure 9. Pseudocode of alphanumeric attribute comparison at site DH_K

```
INPUT: Pseudo-random number generator seed $r_{JT}$, Intermediary CCM matrix M
OUTPUT: Dissimilarity matrix component for $DH_J$ and $DH_K$, J_K
Begin
    1. Initialize rngJT with seed $r_{JT}$
    2. Initialize matrix of distance values: J_K = {M.Length × M[0].Length}
    For m = 0 to M.Length-1
        For n = 0 to M[0] .Length-1
            For q = 0 to M[m] [n].Length-1
                For p = 0 to M[m] [n] [q].Length-1
                    If M [m] [n] [q] [p] – rngJT.Next( ) = = 0
                        3. M [m] [n] [q] [p] = 0
                    Else
                        4. M [m] [n] [q] [p] = 1
                5. Re-initialize rngJT with seed $r_{JT}$
            6. J_K[m] [n] = EditDistance (M [m] [n])
End
```

Figure 10. Pseudocode of alphanumeric attribute comparison at site TP

Communication cost of Party DH_K is $O(m^2+m\times q\times n\times p)$ where m is the data size and q is the longest input's length. $O(m^2)$ is the transfer cost for local dissimilarity matrix and $O(m\times q\times n\times p)$ is the transfer cost for intermediary CCMs.

Privacy analysis of alphanumeric comparison protocol is similar to that of numeric comparison protocol. Vector s' sent to DH_K by DH_J is a random vector for DH_K since he does not have access to the random number generator rngJT. Again, we are assuming that the pseudo-random number generator has a long period and is unpredictable. The message from DH_K to TP contains the CCM for source and target strings and does not leak any private information.

4.3. Categorical attributes

Distance function for categorical attributes can be described as follows:

$$\text{distance}(a, b) = \begin{cases} 0, & \text{if } a = b; \\ 1 & \text{otherwise.} \end{cases}$$

This distance function is not adequate to measure the dissimilarity between ordered or hierarchical categorical attributes. Such categorical data requires more complex distance functions which are left as future work.

Comparison protocol for categorical attributes is quite intuitive. Data holder parties share a secret key to encrypt their data. Value of the categorical attribute is encrypted for every object at every site and these encrypted data are sent to the third party, who can easily compute the distance between categorical attributes of any pair of objects. If ciphertext of two categorical values are the same, then plaintexts must be the same. Third party merges encrypted data and runs the local dissimilarity matrix construction algorithm in Figure 12. Outcome is not a local dissimilarity matrix for the categorical attribute, since data from all parties is input to the algorithm.

Analysis of communication costs and privacy

Each party sends its data in encrypted format, therefore communication cost for a party with n objects is O(n).

The third party can not infer any private information unless the encryption key is available. However this is not possible since data holders and the third party are non-colluding and semi-honest.

5. Dissimilarity matrix construction

In this section, we explain how to build dissimilarity matrices for different data types using the comparison protocols of Section 4. The third party should run the appropriate construction algorithm for every attribute in the list of attributes chosen for clustering such that a corresponding dissimilarity matrix is built.

Construction algorithm for numeric and alphanumeric attributes are essentially the same. Every data holder first builds his local dissimilarity matrix according to Figure 12 and sends the result to the third party. Then for each pair of data holders, the comparison protocol of the attribute is carried out. Finally the third party merges all portions of the dissimilarity matrix from local matrices and comparison protocol outputs. The process ends after a normalization step that scales distance values into [0, 1] interval. Details of the algorithm are given in Figure 11.

Publishing local dissimilarity matrices does not cause private information leakage as proven in [3]. The proof relies on the fact that given the distance between two secret points, there are infinitely many pairs of points that are equally distant. Unless there is an inference channel that helps finding out one of these points, dissimilarity matrix sharing does not result in privacy breaches. For a more formal statement of the theorem and its proof, please refer to [3].

Construction algorithm for categorical data is much simpler. Data holders encrypt their local data and send them to the third party who will merge these encrypted data and run the local dissimilarity matrix construction algorithm on global data.

```
INPUT: List of data holders DH
OUTPUT: Dissimilarity matrix d
Begin
  For each Data holder $DH_J \in DH$
     1. Request local dissimilarity matrix
     For each Data holder $DH_K \in DH$ and $K > J$
        2. Run comparison protocol btw. $DH_J$ and $DH_K$
  3. Construct dissimilarity matrix d
  4. Normalize: d[m][n] = d[m][n] / maximum value in d
End
```

Figure 11. Pseudocode for dissimilarity matrix construction of numeric/alphanumeric attributes

```
INPUT: Comparison function distance(), Data vector $D_J$
OUTPUT: Local dissimilarity matrix d
Begin
  1. Initialize d = {$D_J$.Length × $D_J$.Length}
  For m = 0 to $D_J$.Length-1
     For n = 0 to m
        2. d[m] [n] = distance($D_J$ [m], $D_J$ [n])
End
```

Figure 12. Pseudocode for local dissimilarity matrix construction algorithm

Upon construction of the dissimilarity matrices for all attributes, the third party notifies the data holders, asking for their attribute weight vectors and hierarchical clustering algorithm choices. Then dissimilarity matrices of attributes are merged into a final dissimilarity matrix, which will be input to the appropriate clustering algorithm. There is no privacy concern after the dissimilarity matrices are built, since the third party can run hierarchical clustering algorithms locally, without any intervention of the data holders. Therefore we omit the discussion on how clustering is actually performed.

Dissimilarity matrices must be kept secret by the third party because data holder parties can use distance scores to infer private information since they have a portion of data. That's why clustering results are published as a list of objects of each cluster. The third party can also provide clustering quality parameters such as average of square distance between members. A sample output is depicted in Figure 13. X_j denotes the object with id j at site X.

Cluster1	A_1, A_3, B_4, C_3
Cluster2	B_2, B_3, C_1, C_2
Cluster3	A_2, B_1

Figure 13. Sample Clustering Result

6. Conclusion

In this paper we proposed a method for privacy preserving clustering over horizontally partitioned data. Our method is based on the dissimilarity matrix construction using a secure comparison protocol for numerical, and alphanumeric, and categorical data. Previous work on privacy preserving clustering over horizontally partitioned data was on a specific clustering algorithm and only for numerical attributes. The main advantage of our method is its generality in applicability to different clustering methods such as hierarchical clustering. Hierarchical clustering methods can both discover clusters of arbitrary shapes and deal with diverse data types. For example, we can cluster alphanumeric data that is of outmost importance in bioinformatics researches, besides numeric and categorical data. Another major contribution is that quality of the resultant clusters can easily be measured and conveyed to data owners without any leakage of private information.

We also provided the security analysis and communication costs and observed that the communication costs of our protocols are parallel to the computation costs of the operations in case of centralized data. Although the constructed dissimilarity matrix is input to hierarchical clustering algorithms, there are various other application areas of the proposed protocols such as record linkage and outlier detection problems.

As a future work, we plan to expand our privacy analysis for the comparison protocol of alphanumeric attributes so that possible attacks using statistics of the input language are addressed as well. However such analysis depends heavily on the intrinsic properties of the language, therefore building a general model is not an easy task. Even for specific languages (i.e. DNA), proper statistics is not readily available.

7. References

[1] Oliveira, S. R. M., Zaïane, O. R. “Achieving Privacy Preservation When Sharing Data for Clustering”, In *Proc. of the International Workshop on Secure Data Management in a Connected World*, p.67-82, 2004.

[2] Oliveira, S. R. M., Zaïane, O. R. “Privacy Preserving Clustering By Data Transformation”, In *Proc. of the 18th Brazilian Symposium on Databases*, p.304-318, 2003.

[3] Oliveira, S. R. M., Zaïane, O. R. “Privacy Preserving Clustering By Object Similarity-Based Representation and Dimensionality Reduction Transformation”, In *Proc. of the 2004 ICDM Workshop on Privacy and Security Aspects of Data Mining*, p.40-46, 2004.

[4] Merugu, S., Ghosh, J. “Privacy-Preserving Distributed Clustering Using Generative Models”, In *Proc. of the Third IEEE International Conference on Data Mining*, p. 211-218, 2003.

[5] Klusch, M., Lodi, S., Moro, G. “Distributed Clustering Based on Sampling Local Density Estimates”, In *Proc. of the Eighteenth International Joint Conference on Artificial Intelligence*, p.485-490, 2003.

[6] Vaidya, J., Clifton, C. “Privacy-Preserving K-Means Clustering over Vertically Partitioned Data”, In *Proc. of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p.206-215, 2003.

[7] Jha, S., Kruger, L., McDaniel, P. “Privacy Preserving Clustering”, In *Proc. of the 10th European Symposium on Research in Computer Security*, p.397-417, 2005.

[8] Atallah, M. J., Kerschbaum, F., Du, W. “Secure and Private Sequence Comparisons”, In *Proc. of the 2003 ACM Workshop on Privacy in the Electronic Society*, p.39-44, 2003.

[9] Vassilios S. Verykios, Ahmed Elmagarmid, Elisa Bertino, Yucel Saygin, Elena Dasseni. Association Rule Hiding. *IEEE Transactions on Knowledge and Data Engineering* Vol. 16 No. 4, 2004.

[10] Agrawal, R., Srikant, R. “Privacy Preserving Data Mining”, In *Proc. of the 2000 ACM SIGMOD Conference on Management of Data*, p. 439-450, 2000.

[11] Kantarcioglu, M., Clifton, C. “Privacy-Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data”, *IEEE Transactions on Knowledge and Data Engineering* Vol. 16 No. 9, 2004.

[12] Vaidya, J., Clifton, C. “Privacy Preserving Association Rule Mining in Vertically Partitioned Data”, In *Proc. of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p.639-644, 2002.

[13] Du, W., Zhan, Z. “Building Decision Tree Classifier on Private Data”, In *Proc. of the IEEE ICDM Workshop on Privacy, Security and Data Mining*, p.1-8, 2002.