Privacy Preserving Spatio-temporal Clustering on Horizontally Partitioned Data*

Ali İnan and Yücel Saygın

Sabancı University, Faculty of Engineering and Natural Sciences,
34956, Istanbul, Turkey
inanali@su.sabanciuniv.edu
ysaygin@sabanciuniv.edu

Abstract. Time-stamped location information is regarded as spatio-temporal data and, by its nature, such data is highly sensitive from the perspective of privacy. In this paper, we propose a privacy preserving spatio-temporal clustering method for horizontally partitioned data which, to the best of our knowledge, was not done before. Our methods are based on building the dissimilarity matrix through a series of secure multi-party trajectory comparisons managed by a third party. Our trajectory comparison protocol complies with most trajectory comparison functions and complexity analysis of our methods shows that our protocol does not introduce extra overhead when constructing dissimilarity matrix, compared to the centralized approach.

1 Introduction

Advances in wireless communication technologies resulted in a rapid increase in usage of mobile devices. PDAs, mobile phones and various other devices equipped with GPS technology are now a part of our daily life. One direct consequence of this change is that, using such devices, locations of individuals can be tracked by wireless service providers. Individuals sometimes voluntarily pay for being tracked by means of Location Based Services (LBS) such as vehicle telematics that offer vehicle tracking and satellite navigation. Tracking is also enforced by law in some countries, as in the case of the Enhanced-911 mandate, passed by U.S. Federal Communications Commission in 1996. The mandate requires that any cellular phone calling 911, the U.S. nationwide emergency service number, be located within at least 50 to 100 meters.

Time-stamped location information is regarded as spatio-temporal data due to its time and space dimensions and, by its nature, is highly vulnerable to misuse. In fact, privacy issues related to collection, use and distribution of individuals' location information is the main obstacle against extensive deployment of LBSs. Suppressing identifiers from the data does not suffice since trajectories can easily be re-bound to individuals using publicly available information such as home and work addresses. Therefore new privacy preserving knowledge discovery methods, designed specifically

* This work was funded by the Information Society Technologies programme of the European Commission, Future and Emerging Technologies under IST-014915 GeoPKDD project.

A Min Tjoa and J. Trujillo (Eds.): DaWaK 2006, LNCS 4081, pp. 459 – 468, 2006.

to handle spatio-temporal data, are required. Existing privacy preserving data mining techniques are not suitable for this purpose since time-stamped location observations of an object are not plain, independent attributes of this object.

In this work, we propose a privacy preserving clustering technique for horizontally partitioned spatio-temporal data where each horizontal partition contains trajectories of distinct moving objects collected by a separate site. Consider the following scenario where the proposed techniques are applicable: In order to solve traffic congestion, traffic control offices want to cluster trajectories of users. However, the required spatio-temporal data is not readily available but can be collected from GSM operators. GSM operators are not eager to share their data due to privacy concerns. The solution is running a privacy preserving spatio-temporal clustering algorithm for horizontally partitioned data.

Our method is based on constructing the dissimilarity matrix of object trajectories in a privacy preserving manner which can then be input to any hierarchical clustering algorithm. Main contributions are introduction of a protocol for secure multi-party computation of trajectory distances and its application to privacy preserving clustering of spatio-temporal data. We also provide complexity and privacy analysis of the proposed method.

In Section 2, we provide related work in the area and then formally define the problem in Section 3. Classification of trajectory comparison functions is provided in Section 4. Communication and computation phases of our method are explained in Sections 5 and 6 respectively. We provide complexity and privacy analysis in Section 7 and finally conclude in Section 8.

2 Related Work

Privacy preserving data mining has become a popular research area in the past 5 years. The aim of privacy preserving data mining is ensuring individual privacy while maintaining the efficacy of data mining techniques. Agrawal and Srikant initiated research on privacy preserving data mining with their seminal paper on constructing classification models while preserving privacy [7]. Saygin et al. propose methods for hiding sensitive association rules before releasing the data [14]. Privacy preserving data mining methods can be classified under two headings: data sanitization and secure multi-party computation. Data sanitization approaches sacrifice accuracy for increased privacy, while secure multi-party computation approaches try to achieve both accuracy and privacy at the expense of high communication and computation costs.

Researchers developed methods for privacy preserving clustering. Most of these methods are based on sanitizing the input and they address only centralized data. Merugu and Ghosh propose methods for constructing data mining models from the input data. These models are not considered private information. The overall clustering schema is constructed by merging these models coming from vertically or horizontally distributed data sources [9]. Oliveira and Zaiane propose methods for preserving privacy by reducing the dimensionality of the data [5]. Their method is not applicable to horizontally partitioned data and moreover, results in loss of accuracy. Vaidya and Clifton propose a secure multi-party computation protocol for k-means clustering on vertically partitioned data [10]. Jha et al. [8] propose a privacy preserving, distributed k-means protocol on horizontally partitioned data through secure multi-party computation of cluster means. Inan et al. propose another privacy preserving clustering algorithm over horizontally partitioned data that can handle numeric, categorical and alphanumeric data [6].

Privacy of spatio-temporal data is of utmost importance for individuals since such data is highly vulnerable to misuse. In this work, we focus on spatio-temporal data and propose a secure multi-party comparison protocol that is applicable to most trajectory comparison functions. Previous work on ensuring individual privacy for spatio-temporal data is limited to sanitization approaches and access control mechanisms. Gruteser and Hoh propose confusing paths to garble trajectories of individuals [11]. Beresord and Stajano introduce "mix zones", in which identification of users is blocked and pseudonyms of incoming user trajectories are mixed up while leaving these mixed zones [12]. A detailed discussion on privacy mechanisms through access control and anonymization can be found in [13]. To the best of our knowledge, this work is the first to introduce a secure multi-party solution to privacy problems in spatio-temporal data without any loss of accuracy.

3 Problem Formulation

Spatio-temporal knowledge discovery deals with time-stamped location observations of moving objects. In some applications spatial component may interpreted in a different way. For example, in stock market analysis, trajectory of a stock is the one-dimensional vector of price fluctuations in time. In weather forecasting, observations are two dimensional measurements of atmospheric pressure and temperature at weather stations. In this paper, we primarily focus on moving objects and assume that location information is two dimensional as in the case of GPS, neglecting the altitude.

Trajectory T of a moving object X is a set of location observations in the form O = (t, d) where t represents the time dimension and d represents the two dimensional location information. Number of observations for this trajectory is denoted as *length(X)* and i^{th} element of T_X is denoted by $T_X(i)$. Figure 1 depicts these notions for the sample one dimensional spatio-temporal data provided in Table 1.

Table 1. Spatio-temporal data for trajectories *X* and *Y*

X	Time				
	1	4	7	10	16
Location	2,3	4,5	6,7	3	2
Y	Time				
	2	4	6	8	
Location	4,3	3,6	7	3	

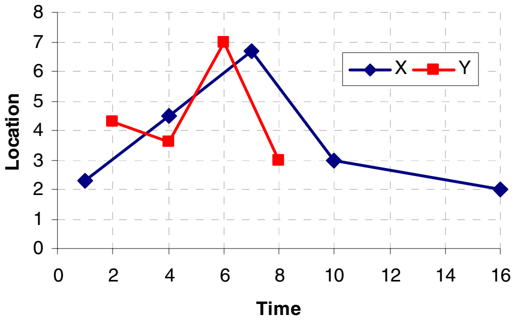

Fig. 1. Trajectories *X* and *Y*. *length(X)* = 5 and *length(Y)* = 4

Suppose that there are *K* data holders, such that K ≥ 2, which track locations of with unique object id's. The number of objects in data holder *k*'s database is denoted as $size_K$. Data holders want to cluster the trajectories of moving objects without publishing sensitive location information so that clustering results will be public to each data holder at the end of the protocol. There is a distinct third party, denoted as TP, who serves as a means of computation power and storage space. TP's role in the protocol is: (1) managing the communication between data holders, (2) privately constructing the global dissimilarity matrix, (3) clustering the trajectories using the dissimilarity matrix, and (4) publishing the results to the data holders.

Involved parties, including the third party, are assumed to be semi-honest which means that they follow the protocol as they are expected to, but may store any information that is available in order to infer private data in the future. Semi-trusted behavior is also called honest-but-curious behavior. Another assumption is that, all parties are non-colluding, i.e. they do not share private information with each other.

4 Trajectory Comparison Functions

Clustering is the process of grouping similar objects together. In order to measure the similarity between object trajectories, robust comparison functions are needed. However, trajectory comparison is not an easy task since spatio-temporal data is usually collected through sensors and therefore is subject to diverse sources of noise. Under ideal circumstances, object trajectories would be of the same length and time-stamps of their corresponding elements would be equal. The distance between two trajectories satisfying these conditions could be computed using Euclidean distance, simply by summing the distance over all elements with equal time-stamps. In real world, on the other hand, non-overlapping observation intervals, time shifts and different sampling rates are common. Although various trajectory comparison functions have been proposed to cope with these difficulties, this topic is still an ongoing research area.

Most trajectory comparison functions stem from four basic algorithms: (1) Euclidean distance, (2) Longest Common Subsequence (LCSS), (3) Dynamic Time Warping

(DTW), and (4) Edit distance. We classify these algorithms into two groups with respect to penalties added per pair-wise element comparisons: real penalty functions and quantized penalty functions. Real penalty functions measure the distance in terms of the Euclidean distance between observations while quantized penalty functions increment the distance by values 0 or 1 at each step depending on spatial proximity of the compared observations. In the following subsections we explain crucial trajectory comparison functions briefly and provide the reasoning behind this classification. For a detailed discussion on characteristics of these algorithms, please refer to [1].

Significance of our privacy preserving trajectory comparison protocol is due to the fact that it is applicable to all comparison functions explained below. Furthermore, the protocol does not trade accuracy against privacy unlike previous work.

4.1 Comparison Functions with Real Penalty

Euclidean distance, Edit distance with Real Penalty (ERP) and DTW are the comparison functions with real penalty. Euclidean distance is a naïve method based on comparing the corresponding observations of trajectories with the same length. The algorithm terminates in O(n) time, returning the sum of real penalties. Euclidean distance function is sensitive to time shifts and noise but the output is a metric value.

ERP [4] measures the minimum cost of transforming the compared trajectory to the source trajectory using insertion, deletion and replacement operations. Cost of each operation is calculated using real spatial distance values. Cost of replacing observation *i* with observation *j* is *dist(i, j)*, where *dist* is the Euclidean distance. However in case of insertion (or deletion), added cost is the distance between the inserted (or deleted) observation and the constant observation value *g*, defined by the user. ERP compares all pairs of elements in the trajectories, returning a metric value in $O(n^2)$ time. The algorithm is resistant to time shifts but not to noise.

DTW was initially proposed for approximate sequence matching in speech recognition but is generalized to similarity search in time series by authors of [3]. The algorithm is very similar to Edit distance but instead of insertions and deletions, stutters are used. The i^{th} stutter on *x* dimension, denoted as $stutter_i(x)$, repeats the i^{th} element and shifts following elements to the right. Computation cost is $O(n^2)$ as expected and resultant distance value is non-metric. Allowing repetitions strengthens the algorithm against time shifts but does not help with noise.

4.2 Comparison Functions with Quantized Penalty

Trajectory comparison functions with quantized penalty are LCSS [2] and Edit distance on Real Sequence (EDR) [1]. Both algorithms try to match all pairs of elements in the compared trajectories and therefore have a computation cost of $O(n^2)$. A pair of observations is considered a match if they are close to each other in space by less then a threshold, ε. LCSS returns the length of the longest matched sequence of observations while EDR returns the minimum number of insertion, deletion or replacement operations required to transform one trajectory to the other. Although these algorithms are resistant to time shifts and noise, distance values are not metric.

5 Communication Phase

As explained before, the protocol for privacy preserving comparison of trajectories consists of two phases: communication phase and computation phase. In the communication phase, data holders exchange data among themselves and the third party (TP), who will carry out the computation phase and publish the clustering results.

Prior to the communication phase we assume that every involved party, including the third party, has already generated pair-wise keys. These keys are used as seeds to pseudo-random number generators which disguise the exchanged messages. Diffie-Hellman key exchange protocol is perfectly suitable for key generation [15].

Dissimilarity matrix is an object by object structure. In case of spatio-temporal data, an entry *D[i][j]* of the dissimilarity matrix *D* is the distance between trajectories of objects *i* and *j* calculated using any comparison function. In Section 6, we show that our privacy preserving comparison protocol is suitable for all comparison functions explained in Section 4. If trajectories of both *i* and *j* are held by the same site, this site can calculate their distance locally and send it to the third party. However, if trajectories of *i* and *j* are at separate sites, these sites should run the protocol explained below. Assuming *K* data holders, *C(K,2)* runs are required, one for each pair of data holders.

Suppose that two data holders, DH_A and DH_B, with *size(A)* and *size(B)* trajectories respectively, want to compare their data. Assume that the protocol starts with DH_A. For each trajectory *T* in DH_A's database, two pseudo-random number generators are initialized, rng_{AB} and rng_{AT}. The seed for rng_{AB} is the key shared with DH_B and the seed for rng_{AT} is the key shared with *TP*. Then, for each dimension of spatial component of *T*'s elements (i.e. x and y), DH_A disguises its input as follows: if the pseudo-random number generated by rng_{AB} is odd, DH_A negates its input and increments it by the pseudo-random number generated by rng_{AT}. Finally, DH_A sends the disguised values to DH_B.

```
Begin
  For j=0 to size($DH_A$)
    Initialize $rng_{AB}$ with the key $K_{AB}$
    Initialize $rng_{AT}$ with the key $K_{AT}$
    For m=0 to length($DH_A$[j])
      $DH_A[j][m].x = rng_{AT} + DH_A[j][m].x * -1^{rng}AB^{\%2}$
      $DH_A[j][m].y = rng_{AT} + DH_A[j][m].y * -1^{rng}AB^{\%2}$
 Send $DH_A$ to $DH_B$
End
```

Fig. 2. Pseudo code of trajectory comparison protocol at site DH_A

Upon receiving data from DH_A, DH_B initializes a matrix *M* of size *size(B)*×*size(A)*, which will be DH_B's output. For each trajectory *T* in its database, DH_B initializes a pseudo-random number generator rng_{AB} with the key shared with DH_A and negates its inputs in a similar fashion. This time negation is done when the generated number is even. DH_B then starts filling values into *M*. An entry *M[i][j][m][n]* of *M* is DH_A's j^{th} trajectory's n^{th} observation compared to DH_B's i^{th} trajectories m^{th} observation. DH_B simply adds its input to the input received from DH_A. At the end, *M* is sent to *TP* by DH_B.

```
Begin
  For i=0 to size($DH_B$)
    For j=0 to size($DH_A$)
      For n=0 to length($DH_B$[i])
        Initialize $rng_{AB}$ with the key $K_{AB}$
        For m=0 to length($DH_A$[j])
          M[i][j][n][m].x +=$DH_B$[i][n].x * $-1^{(rng_{AB}+1)\%2}$
          M[i][j][n][m].y +=$DH_B$[i][n].y * $-1^{(rng_{AB}+1)\%2}$
  Send M to TP
End
```

Fig. 3. Pseudo code of trajectory comparison protocol at site DH_B

TP subtracts the random numbers added by DH_A using a pseudo-random number generator, rng_{AT}, initialized with the key shared with DH_A. Now, absolute value of any entry *M[i][j][m][n]* is $\mid DH_A[j][n] - DH_B[i][m] \mid$. These values are all that is needed by any comparison function to compute the distance between trajectories *i* and *j*.

Pseudo codes for the roles described above are given in Figures 2, 3 and 4. Discussion on the necessity of each pseudo-random number generator used in the protocol is provided in Section 7.

```
Begin
  For i=0 to size($DH_B$)
    For j=0 to size($DH_A$)
      For n=0 to length($DH_B$[i])
        Initialize $rng_{AT}$ with the key $K_{AT}$
        For m=0 to length($DH_A$[j])
          M[i][j][n][m].x = $|$M[i][j][n][m].x - $rng_{AT}|$
          M[i][j][n][m].y = $|$M[i][j][n][m].y - $rng_{AT}|$
End
```

Fig. 4. Pseudo code of trajectory comparison protocol at site TP

6 Computation/Aggregation Phase

The third party can compute pair-wise trajectory distances for data holder sites *A* and *B*, once the comparison matrix *M* is built through the protocol in Section 5. If the comparison function measures distances using real penalty, then *M[i][j][m][n]* is the cost for *A*'s j^{th} trajectory's n^{th} observation with respect to *B*'s i^{th} trajectory's m^{th} observation. Otherwise, if a quantized penalty comparison function is to be employed, *TP* simply checks whether *M[i][j][m][n]* $< \varepsilon$ to match these two observations.

What remains is performing comparisons of the form *M[i][j]*, where both *i* and *j* are trajectories of the same data holder site. In such cases, another privacy preserving protocol is not required to compute these values, since conveying local dissimilarity matrices to *TP* does not leak any private information, proven in [5].

In order to build the dissimilarity matrix, *TP* must ensure that every data holder site has sent its local dissimilarity matrix and run the pair-wise comparison protocol with every other data holder. Figure 5 is the pseudo-code for constructing local dissimilarity matrices where *distance* denotes the comparison function.

```
Begin
  For m=0 to size(DH)
    For n=0 to m
      D[m][n]= distance(DH[m], DH[n])
End
```

Fig. 5. Pseudo code for local dissimilarity matrix construction

After gathering comparison results for all pairs of trajectories, *TP* normalizes the values in the dissimilarity matrix. These normalized distances are the only required input for most clustering algorithms, such as k-medoids, hierarchical and density based clustering algorithms. Another key observation here is that using our protocol, *TP* may use any clustering algorithm depending on requirements of the data holders.

At the end of the clustering process, the third party sends the clustering results to the data holders. The results are in the form of lists of objects identifiers, since publishing the dissimilarity matrix itself would cause private information leakage. The third party can also publish clustering quality parameters, if requested by the data holders.

7 Complexity and Privacy Analysis

In this section, we analyze the communication and computation costs of the pair-wise comparison protocol and local dissimilarity matrix construction. An analysis of the privacy offered by the protocol follows.

Every data holder has to send its local dissimilarity matrix to the third party. Computation cost of constructing the matrix is $O(n^2 * distance)$ where n is the number of trajectories and *distance* denotes the complexity of the comparison function. For Euclidean, the cost becomes $O(n^2*p)$ and for the other comparison functions it is $O(n^2*p^2)$ where p is the maximum number of observations in a trajectory.

The initiator of the comparison protocol, DH_A in Section 5, has a computation cost of $O(n*p)$. The follower, DH_B, on the other hand makes $O(n*m*p^2)$ computations where m is the number of trajectories at site DH_B. Communication costs are parallel to computation costs since every party sends the result of the computation without any further operation.

There is an apparent imbalance in the computation and communication costs of the follower and initiator parties. *TP* can easily solve this problem by arranging the sequence that pair-wise comparison protocols are carried out such that every party will be the initiator at least $\lfloor (K - 1)/2 \rfloor$ times in a setting of *K* data holders.

Sharing dissimilarity matrices does not leak any private information according to [5], as long as the private data is kept secret. The proof of the theorem relies on the fact that given the distance between two data points, there are infinitely many pairs of points that are equally distant. Since we assume that involved parties do not collude with each other and honestly follow the protocol, *TP* can not collude with a data holder site to infer private information of another data holder. Therefore sharing local dissimilarity matrices does not harm privacy unless the comparison protocol introduces inference channels that may leak private information.

In the comparison protocol, the message sent by the initiator is a matrix containing values of the form (n + r) or (-n + r) where n is initiator's input and r is a random number. In either case, these values are completely random to the follower. On the other hand, follower sends *TP* a matrix of values of the form (n – m + r) or (m – n + r). Although *TP* knows r, (n – m) or (m – n) does not help inferring either n or m, since there are infinitely many pairs (m, n) whose distance is | m – n |.

Purpose of the pseudo-random number generator shared between the initiator and the follower is preventing *TP* from inferring whose input is larger. Suppose that always the follower subtracts its input from the initiator's input. If $m > n$, (n + r – m – r) = (n – m) would be negative, pointing out that follower's input is greater. Shared pseudo-random number generator garbles the negation sequence and prevents such inferences.

One possible attack against our comparison protocol could be statistical analysis. Notice that observations of every trajectory in initiator's database with the same index is disguised using the same random number. This is due to the fact that the pseudo-random number generator is re-initialized at each step. Given enough statistics on the data and assuming that the databases are large enough to contain many repetitions of spatial values, such an attack is realizable. But considering that the domain of spatial values is very large and such statistics is not publicly available, we regard these types of attacks as very unlikely to succeed.

8 Conclusion

In this paper, we proposed a protocol for privacy preserving comparison of trajectories and its application to clustering of horizontally partitioned spatio-temporal data. The main advantage of our protocol is its applicability to most trajectory comparison functions and different clustering methods such as hierarchical clustering. The data holder sites can decide the clustering algorithm of their choice and receive clustering quality parameters together with the results. Only a small share of existing privacy preserving clustering algorithms can handle horizontally partitioned data and these algorithms do not specifically address spatio-temporal attributes.

We also provided complexity and privacy analysis of our protocol and observed that communication and computation costs are parallel to the computation costs for clustering local data. Privacy analysis shows that an attack using statistics of spatial components is possible but very unlikely to succeed. A proof-of-concept implementation of the clustering algorithm in C# language is available at [17]. We used real spatio-temporal datasets from the R-Tree Portal [16] for debugging and verifying the software.

References

1. Chen, L., Özsu, M. T., Oria V.: Robust and Fast Similarity Search for Moving Object Trajectories. In: Proc. of the 2005 ACM SIGMOD. (2005) 491-502
2. Vlachos, M., Kollios, G., Gunopulos, D.: Discovering Similar Multidimensional Trajectories. In: Proc. of the 18th ICDE. (2002) 673-684

3. Yi, B-K., Jagadish, H. V., Faloutsos, C.: Efficient Retrieval of Similar Time Sequences Under Time Warping. In: Proc. of the 14th ICDE. (1998) 201-208
4. Chen, L., Ng, R.: On the Marriage of Edit Distance and Lp-Norms. In: Proc. of the 2004 VLDB. (2004) 792-803
5. Oliveira, S.R.M., Zaiane, O.R.: Privacy Preserving Clustering by Object Similarity-Based Representation. In: Proc. of the 2004 ICDM Workshop on Privacy and Security Aspects of Data Mining. (2004) 40-46
6. Inan, A., Saygin, Y., Savas, E., Hintoglu, A.A., Levi, A.: Privacy Preserving Clustering on Horizontally Partitioned Data. In: Proc. of the 22nd ICDE Workshop on Privacy Data Management. (2006)
7. Agrawal, R., Srikant, R.: Privacy Preserving Data Mining. In: Proc. of the 2000 ACM SIGMOD. (2000) 439-450
8. Jha, S., Kruger, L., Mc Daniel, P.: Privacy Preserving Clustering. In: Proc. of the 10th European Symposium on Research in Computer Security. (2005) 397-417
9. Merugu, S., Ghosh, J.: Privacy Preserving Distributed Clustering using Generative Models. In: Proc. of the 3rd ICDM. (2003) 211-218
10. Vaidya, J., Clifton, C.: Privacy Preserving K-Means Clustering over Vertically Partitioned Data. In: Proc. of the 9th ACM SIGKDD. (2003) 206-215
11. Hoh, B., Gruteser, M.: Protecting Location Privacy through Path Confusion. In: Proc. of the 2005 SecureComm. (2005)
12. Beresford, A.R., Stajano, F.: Mix Zones: User Privacy in Location-Aware Services. In: Proc. of PerCom Workshops. (2004) 127-131
13. Beresford, A.R.: Location Privacy in Ubiquitous Computing. Ph.D. Dissertation, University of Cambridge. (2004)
14. Saygin, Y., Verykios, V.S., Clifton, C.: Using Unknowns to Prevent Discovery of Association Rules. In: SIGMOD Record 30(4). (2001) 45-54
15. Diffie, W., Hellman, M.E.: New Directions in Cryptography. In: IEEE Transactions on Information Theory. (1976) IT-200, 644-654
16. The R-Tree Portal. <http://isl.cs.unipi.gr/db/projects/rtreeportal/trajectories.html>. (March 28, 2006)
17. "ppSTClusteringOnHP.zip" [3510K]. <http://students.sabanciuniv.edu/~inanali/ppSTClusteringOnHP.zip>. (March 28, 2006)