Efficient Privacy Preserving Distributed Clustering Based on Secret Sharing[⋆]

Selim V. Kaya, Thomas B. Pedersen, Erkay Savaş, and Yücel Saygın

Sabanci University
Istanbul, 34956, Turkey
selimvolkan@su.sabanciuniv.edu, {pedersen,erkays,ysaygin}@sabanciuniv.edu

Abstract. In this paper, we propose a privacy preserving distributed clustering protocol for horizontally partitioned data based on a very efficient homomorphic additive secret sharing scheme. The model we use for the protocol is novel in the sense that it utilizes two non-colluding third parties. We provide a brief security analysis of our protocol from information theoretic point of view, which is a stronger security model. We show communication and computation complexity analysis of our protocol along with another protocol previously proposed for the same problem. We also include experimental results for computation and communication overhead of these two protocols. Our protocol not only outperforms the others in execution time and communication overhead on data holders, but also uses a more efficient model for many data mining applications.

1 Introduction

Recent advances in data collection and storage technologies enabled organizations to handle vast amounts of data related to their customers or users. However, this vast amount of plain data needs to be converted into useful knowledge through data mining techniques for high level decision making.

In many cases, databases are distributed among several organizations, which need to collaborate to achieve a more significant and accurate data mining model. Simply sharing the databases is not a feasible approach due to privacy concerns. As a result, Privacy Preserving Distributed Data Mining(PPDDM) techniques are developed for constructing a global data mining model over distributed databases without actually sharing the confidential data.

Efficiency in communication and computation is crucial in PPDDM since databases are often of considerable size. Sample scenarios are sensor networks or RFID applications, where the sensor nodes or RFID readers that contain the data (data holders) have very limited computation and communication capacity. In such scenarios, reducing the communication and computation costs is of utmost importance.

[⋆] This work was partially funded by the Information Society Technologies Programme of the European Commission, Future and Emerging Technologies under IST-014915 GeoPKDD project.

In this paper we propose a new protocol for privacy preserving clustering over horizontally partitioned data with only a small constant communication and computation overhead for data holders with no loss of accuracy. We reduce the privacy preserving clustering problem to privacy preserving dissimilarity matrix construction which was proposed by Inan et al. [1]. After the dissimilarity matrix is computed privately, it can be fed into any hierarchical clustering algorithm. Our protocol uses two non-colluding third parties, which receive secret shares of inputs and compute intermediary results, while a data miner performs the actual clustering.

2 Related Work and Background

The pioneering research on privacy preserving data mining was conducted by Agrawal and Srikant[2], and Lindell and Pinkas[3] in 2000. In [2], Agrawal and Srikant use data perturbation for construction of a classification model privately. The basic idea is that original data values can be perturbed in such a way that original distribution of the aggregated data can be recovered but not the individual data values. Perturbation technique is efficient to implement however it has several problems. First of all, even though the distribution of original values can be predicted with a certain confidence level, some accuracy is lost. Secondly, modification of data does not fully preserve privacy of individual values, and may cause privacy breaches as shown in [4,5]. Finally, perturbation has a predictable structure for certain cases and hence may not fully preserve privacy [6]. A different perturbation method is proposed by Saygin et al.[7] in 2001 for association rule hiding, where unknown values are introduced to hide sensitive association rules.

Authors in [3] employ cryptography as its main tool and implements a decision tree learning protocol. However oblivious transfer, which is the main building block of this protocol, causes huge computation and communication overhead due to exponentiation operation for each bit of private data and expansion of each bit of private data as a result of exponentiation respectively. Authors in [8] propose a privacy preserving association rule mining protocol over horizontally partitioned data taking advantage of commutative encryption. Nevertheless the protocol requires encryption and decryption operations to be performed over each private input by all of the participants resulting in a large communication and computation cost.

Several protocols are proposed for privacy preserving clustering. Oliveira and Zaiane [9] introduce geometric data transformation methods(GDTMs) to distort confidential data values. The protocol tries to preserve main features of the confidential data for clustering while perturbing the data to meet privacy requirements. However, perturbation causes accuracy loss in clustering, and privacy of the data is not fully guaranteed. Consequently, Oliveira and Zaiane [10] introduce the notion of Rotation-Based Transformation(RBT). RBT provides confidentiality of attribute values while completely preserving the original clustering results. However RBT method has a computation overhead since attribute values are transformed pairwise, and selection of attribute pairs should be done in such a way that variance between the original and transformed attributes are maximum. In [11], Oliveira and Zaiane propose Object Similarity-Based Representation(OSBR) and Dimensionality Reduction-Based Representation(DRBT) methods for clustering over centralized and vertically partitioned databases. Therefore, OSBR has high computation cost since each data owner sends a dissimilarity matrix to a central party yielding a communication complexity of $O(n^2)$, while DRBT can cause loss of accuracy due to dimensionality reduction in the original data.

Merugu and Ghosh [12], and Klusch, Lodi and Moro [13] propose privacy preserving clustering methods based on sharing models representing the original data instead of sharing the original data itself. Accordingly, clustering can be performed over the model without revealing the original data points. However clustering over low quality representatives of the original data causes loss of accuracy while efforts for high quality representatives means loss of privacy.

Vaidya and Clifton [14] propose a privacy preserving k-means clustering protocol based on secure multi-party(SMC) computation. Nevertheless there is a huge communication and computation cost due to iterative execution of several SMC protocols till a convergence point for the clusters is obtained. Jha, Kruger and McDaniel propose two privacy preserving k-means clustering protocols for horizontally partitioned data in [15]. The protocols use homomorphic encryption and oblivious polynomial evaluation as their building block which are inefficient to be applied over large databases due to cost of modular exponentiation and oblivious transfer respectively.

The most recent study for privacy preserving clustering is proposed by Inan et al. [1] over horizontally partitioned data and the problem is reduced to secure computation of dissimilarity matrix. Each entry of the dissimilarity matrix is computed by a secure difference protocol where confidential data points are disguised by pseudo-random values and the disguise is removed by a trusted third party revealing the final difference. However secure difference protocol leads to privacy breaches because of the way pseudo-random values are used. According to the secure difference protocol, initiator of the protocol creates two disguise factors; one for the follower of the protocol to disguise initiators value and the other for the trusted third party to disguise which participant's input is subtracted from the other. Nevertheless, the latter disguise factor is the same for each entry point within a row of dissimilarity matrix. In other words, trusted third party can guess which site's input is subtracted from the other with a probability of $\frac{1}{2}$ for each row. On the other hand, quadratic communication cost for dissimilarity matrix computation is a huge burden for data holders.

3 Preliminaries

In our scenario we have ℓ *data holders*: $DH_1, \ldots, DH_\ell$ where DH_i has a database with n_i objects (tuples): $o^i_1, \ldots, o^i_{n_i}$. The databases all have the same schema with m numeric attributes (from an algebraic field). Since all databases have the same schema, we can write the union of the databases as $o_1, o_2, \ldots, o_N$, where $N = \sum_{i=1}^{\ell} n_i$, and where object o_i has attribute values $a_1^i, \ldots, a_m^i$. We say that the collective database is *horizontally partitioned* among the ℓ data holders.

The goal of our protocol is to compute the dissimilarity matrix of all objects in all databases, while keeping the actual values secret. At the end, each entry of the dissimilarity matrix will contain the weighted Manhattan distance between two elements from the collective database.

$$D_{ij} = \sum_{k=1}^{m} w_k |a_k^i - a_k^j|, \tag{1}$$

where $i, j = 1, \ldots, N$, and $w_1, \ldots, w_m$ are predefined weights. We introduce the notion of *partial dissimilarity matrices* which contain the numerical distance for a single attribute, so that the dissimilarity matrix can be written

$$D = \sum_{k=1}^{m} w_k D^k, \tag{2}$$

where D^k is the dissimilarity matrix with entries $D^k[i,j] = |a_k^i - a_k^j|$ which results from considering only the kth attribute.

3.1 Homomorphic Secret Sharing

Informally speaking, $(m,\, t)$-secret sharing is a method to share a secret among m parties in such a way that $t-1$ or less colluding parties cannot compute any information about the secret; but t arbitrary parties can recover the secret. A data holder that wishes to share his secret s will create m *secret-shares* $s_1, \ldots, s_m$ and send one share to each party [16,17].

The protocols we present in this paper rely on additive secret sharing. To share a secret integer[1] s between two parties, we choose a random integer r and give the share r to the first party and the share $s-r$ to the second party. Clearly both shares are random when observed alone, so no single party can compute any information about the secret. The secret is revealed by simply two parties adding their shares, hence they can recover the secret together.

Additive secret sharing we used in our protocol is homomorphic with respect to addition since adding shares pairwise gives an additive sharing of the sum of the secrets.

4 Our Protocol

There are two challenges for designing a protocol for computing Manhattan distance: (i)not to reveal private inputs, (ii) to hide which input is the largest. We employ additive homomorphic secret sharing to fulfill the first challenge, with a very small communication and computation overhead for the data holders.

[1] Or more precisely: to share an element from an additive group.

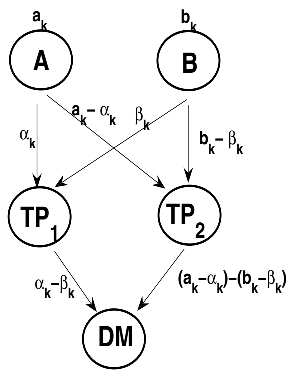

Fig. 1. Overview of the numerical distance protocol

The inputs are shared between two non-colluding third parties, TP_1 and TP_2, who can compute a secret sharing of the difference by using the homomorphic property. To avoid revealing the sign of the difference (which input is larger), TP_1 and TP_2 share a pseudo random number generator. Before the protocol starts, TP_1 and TP_2 will each prepare an $m \times N \times N$ table, whose entries are one-bit binary values from a pseudo random number generator (PRNG), which is initialized with a shared seed.

Let a_k and b_k be the private values for the kth attribute of o_i^A and o_j^B held by DH_A and DH_B, respectively. The (i, j)th entry in the D^k is $|a_k - b_k|$. To compute this Euclidean distance DH_A selects a random number α_k, and sends additive shares α_k and $a_k - \alpha_k$ to TP_1 and TP_2, respectively. Likewise, DH_B creates its own additive shares β_k and $b_k - \beta_k$ and sends them to TP_1 and TP_2, respectively. TP_1 computes $sh_1 = (-1)^{\text{PRNG(k,i,j)}}(\alpha_k - \beta_k)$ and TP_2 computes $sh_2 = (-1)^{\text{PRNG(k,i,j)}}((a_k - \alpha_k) - (b_k - \beta_k))$, and they send the results to the data miner (DM). When DM adds the two received values the result is

$$sh_1 + sh_2 = (-1)^{\text{PRNG(k,i,j)}}(a_k - b_k). \tag{3}$$

After receiving the numerical values the miner gets the result $|sh_1 + sh_2| = |a - b|$, which is the required (i, j)th entry of D^k. Overview of our Euclidean distance protocol is depicted in Figure 1.

To construct the dissimilarity matrix for the kth attribute, each data holder DH_i computes additive shares of its private values $a_k^1, a_k^2 \ldots a_k^{n_i}$, which are stored in arrays $s_1^{i,k}$ and $s_2^{i,k}$. The arrays $s_1^{i,k}$ and $s_2^{i,k}$ are, then, sent to TP_1 and TP_2, respectively. Steps of the protocol for data holders are demonstrated in Algorithm 1.

Receiving $s_1^{1,k}, s_1^{2,k}, \ldots, s_1^{\ell,k}$ from all of the data holders, TP_1 merges these arrays into s_1^k. After merge operation, s_1^k contains additive shares of the collective database for the kth attribute. Then, TP_1 initializes an $N \times N$ matrix D_1^k and

Algorithm 1. DH_i

```
Input: private values for attribute $k$: $a_k^1, a_k^2 \dots a_k^{n_i}$
Output: secret share arrays $s_1^{i,k}$ and $s_2^{i,k}$
1: Initialize secret share arrays $s_1^{i,k}$ and $s_2^{i,k}$ of size $n_i$
2: for $j = 1$ to $n_i$ do
3:     $(s_1^{i,k}[j], s_2^{i,k}[j]) = secretshare(a_k^j)$
4: end for
5: Sends $s_1^{i,k}$ to $TP_1$
6: Sends $s_2^{i,k}$ to $TP_2$
```

fills each entry (i,j) with value $(-1)^{\text{PRNG}[\text{k,i,j}]}(s_1^k[a]-s_1^k[b])$. The resulting matrix D_1^k contains additive shares of D^k. TP_1 sends D_1^k to DM. The details of the protocol for TP_1 are depicted in Algorithm 2. TP_2 performs the same steps.

Algorithm 2. TP_1

```
Input: Secret share arrays $s_1^{1,k}, s_1^{2,k}, \dots, s_1^{\ell,k}$, matrix PRNG shared with $TP_2$
Output: Secret share matrix $D_1^k$
1: Initialize secret share array $s_1^k$ of size $N = \sum_{i=1}^{\ell} n_i$
2: Initialize secret share matrix $D_1^k$ of size $N \times N$
3: Merge $s_1^{1,k}, s_1^{2,k}, \dots, s_1^{\ell,k}$ into $s_1^k$
4: for $a = 1$ to $N$ do
5:     for $b = 1$ to $N$ do
6:         $D_1^k[a,b] = (-1)^{\text{PRNG}[\text{k,a,b}]}(s_1^k[a] - s_1^k[b])$
7:     end for
8: end for
9: Sends $D_1^k$ to DM
```

It is trivial for DM to construct D^k from matrices D_1^k and D_2^k by simply computing $D_1^k[i,j] + D_2^k[i,j]$ for each entry (i,j) of D^k, where $i, j = 1, 2, \dots, N$.

Security of our Protocol: Our security definition reflects that not more than a negligible amount of information is revealed about *any* object in the collective database. One must also note that information leakage is limited to whatever can be deduced from the final result. The following standard definition for security of our protocol relies on negligibly small values of ϵ.

Definition 1. *A protocol for computing partial dissimilarity matrices is* ϵ-secure *if for all parties, and for all attributes* A_j^i

$$\left|P[A_k^i = x|D^k, M] - P[A_k^i = x|D^k]\right| < \epsilon, \tag{4}$$

where M *is a transcript of all messages send to a given party, where* P *stands for probability.*

From this definition, one can conclude that the proposed protocol is ϵ-secure. Since data holders never receive any information, Equation (4) is satisfied for these parties. Since blinding factors α_i are chosen randomly and independently, Equation (4) is also satisfied for TP_1. Since attributes a_i are chosen from algebraic fields and $a_i - \alpha_i$ are also independent of the data, Equation (4) is also satisfied for TP_2. The values received by DM enables to build D, where each entry has a random sign depending on PRNG. If PRNG is secure, no additional information can be computed.

5 Complexity Analysis

In this section, we analyze computation and communication complexities of our protocol for numeric attributes [2]. Each analysis will be performed for DHs, TPs, and DM separately. We also show complexity analysis of the privacy preserving clustering protocol proposed by Inan et al. [1].

Since computation of secret shares of private inputs can be performed in parallel by each DH, computation complexity of our protocol for DHs is $O(n_{max})$, where $n_{max} = max(n_1, n_2, \ldots, n_\ell)$. On the other hand, for the protocol in [1] computation complexity of DHs is $O(N^2)$ (where N is the total number of objects) since data holders compute shared dissimilarity matrices pairwise which requires serial execution.

In our protocol, for TPs, computation of secret share of D_k yields complexity of $O(N^2)$ which is due to computation of the global dissimilarity matrix, if we assume TPs operate in parallel and pseudo random numbers PRN are generated in advance. In [1], there is only one TP and computation complexity of TP is $O(N^2)$. Complexity of our protocol for DH is $O(N^2)$, which is the cost of computing D^k.

In our protocol, each DH sends secret shares of their private inputs to TPs resulting in a total communication complexity of $O(N)$. TPs send secret shares of D^k to DM and the total communication complexity is $O(N^2)$. Since final clustering is done by DM, there is no further communication cost. On the other hand, in [1], each DH sends local and shared dissimilarity matrices to TP where global dissimilarity matrix is computed. Accordingly, communication complexity is $O(N^2)$. Summary of the computation and communication complexity analysis is depicted in Table 1.

Table 1. Computation/Communication Complexities of our Protocol and the protocol in [1]

Attribute Type	**DH**	**TP**	**DM**	**Total**
Numeric	$O(n_{max})/O(N)$	$O(N^2)/O(N^2)$	$O(N^2)/-$	$O(N^2)/O(N^2)$
Numeric for [1]	$O(N^2)/O(N^2)$	$O(N^2)$	$-/-$	$O(N^2)/O(N^2)$

[2] Due to space limitations, we did not include the extension of our protocol to other attributes types.

6 Implementation and Performance Evaluation

In this section, performance evaluation of our protocol is explained and discussed in detail in comparison with the protocol proposed in [1]. Since both protocols do not result in any loss of accuracy, we perform only two tests: communication cost analysis and computation cost analysis. The experiments are conducted on an Intel Dual-Core Centrino PC with 2 MB cache, 2 GB RAM and 1.83 GHz clock speed. We used C# programming language to implement the algorithms.

Experimental Setup: To measure the performance of our protocol and the one in [1], two test cases are identified for varying: i) total number of entities (total database size), and ii) number of data holders.

Each test case is performed over numeric attributes. For each experiment, we measure the communication and computation overhead of our protocol against the protocol proposed in [1]. Comparison of the protocols in the experiments are confined to the formation of the global dissimilarity matrices and clustering is not taken into consideration. For all the experiments, we denote our protocol as “our protocol” and the one in [1] as “protocol” in the figures. In order to measure the effect of the database size, we fixed the number of data holders to four, where each data holder has an equal share of database.

For the test case (i), we used total database sizes of 2K, 4K, 6K, 8K and 10K. In test case (ii), number of data holders, excluding TPs, is 2, 4, 6, 8, and 10. For each test case, we use synthetically generated datasets. Synthetic datasets are more appropriate for our experiments since we try to evaluate scalability and efficiency of our protocol for varying parameters, and synthetic datasets can be generated by controlling the number of entities, number of data holders, and average length of attributes. Data generator is developed in Eclipse Java environment. For the numeric attributes, each entity is chosen from the interval [0, 10000].

In our experiments, we use 128-bit Advanced Encryption Standard (AES) cipher to generate PRNs to disguise data holders' inputs. We use Cryptography namespace of MS .Net platform to perform AES encryption in the implementation of our protocol and [1]. For our protocol, keys and initialization vectors (IVs) are chosen by each data holder independent of the others, while for [1], seeds for pseudo-random number generator shared between data holders are used as keys and an initialization vector (IV) globally known to every data holder is used. Ciphertext as a result of encryption of IV by AES key is used as the pseudo-random number.

Computation Cost Analysis: Comparison of computation costs between our protocol and the protocol in [1] for varying database sizes from $2K$ to $10K$ is depicted in Figure 2. Figure 2 shows that both our protocol and the one in [1] behave quadratically which is due to formation of global dissimilarity matrix. However, our protocol performs better than the one in [1] since data holders operate in parallel in our protocol and the overall computation cost for each data holder is n AES encryptions for computing secret shares of the data, where n is database size of each data holder. However, the protocol in [1] performs n AES encryptions at each data holder to disguise data values and n AES encryptions at TP to remove these disguise factors. As a result, the protocol in [1] performs $k * n$ more AES encryptions than our protocol where there are k data holders. As Figure 2 shows, execution time difference between our protocol and the one in [1] gets larger as database size increases since number of AES encryptions performed at TP also increases. On the other hand, in our protocol no encryption is performed by any party other than data holders.

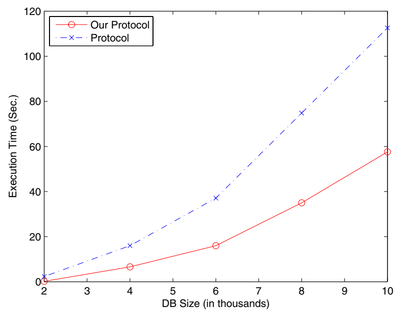

Fig. 2. Computation cost for different database sizes

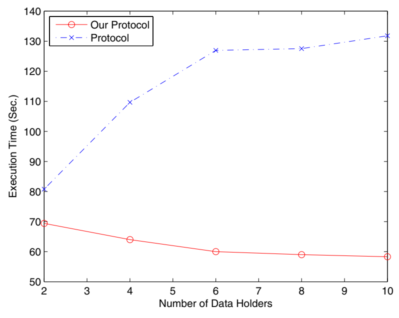

Fig. 3. Computation cost for different number of data holders

Our protocol scales better than the one in [1] with the number of data holders. For this experiment, we generate a dataset of 10K entities and then horizontally partitioned this dataset by distributing the complete dataset over the data holders so that each party holds the same number of entities. As depicted in Figure 3, execution time for the protocol in [1] increases as number of data holders increases. This is due to the fact that, C_2^k number of pairwise computation between data holders have to be performed to compute shared dissimilarity matrices where k is total number of data holders. However, increase in total execution time for [1] gets smaller as number of data holders increases since amount of data owned by each data holder gets smaller. On the other hand, increase in number of data holders reduces total execution time for our protocol since share of each data holder gets smaller which means fewer numbers of encryptions are performed by data holders in parallel. In our protocol, the computation cost of TPs and DM are not affected by the change in number of DH.

Communication Cost Analysis: Overall communication costs of our protocol and the protocol in [1] for various database sizes are depicted in Figure 4. As seen in the figure, overall communication cost of our protocol is more than the one in [1] due to the secret sharing employed in our protocol where two shares are created for each attribute value. Both protocols behave quadratically since overall communication cost is dominated by communication cost of dissimilarity matrices. On the other hand, communication cost of data holders for our

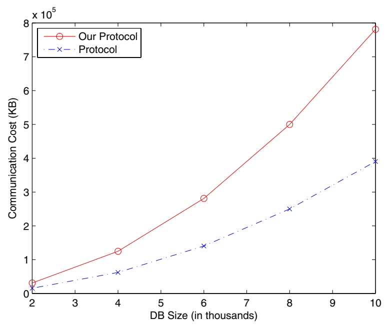

Fig. 4. Overall communication cost for different database sizes

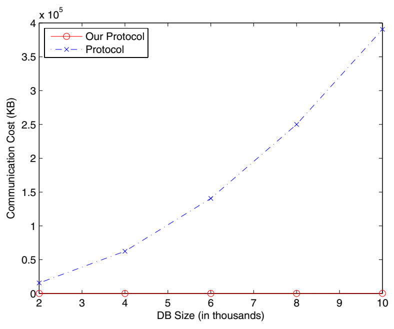

Fig. 5. Communication cost of data holders for different database sizes

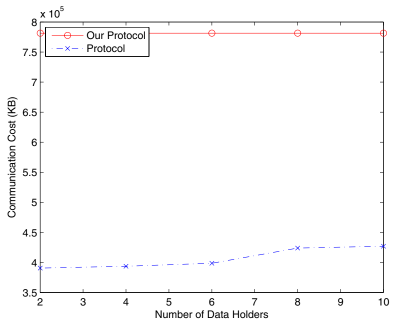

Fig. 6. Overall communication cost for different numbers of data holders

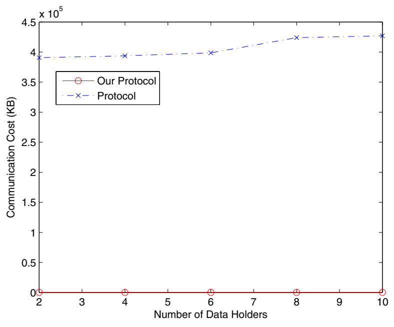

Fig. 7. Communication cost of data holders for different numbers of data holders

protocol and the one in [1] is depicted in Figure 5. Communication cost of our protocol for data holders is linear with respect to the size of each data holder's dataset while the protocol in [1] requires quadratic communication cost for data holders. For that reason, communication cost of our protocol for data holders is negligible compared to the protocol in [1]. Accordingly, our protocol shifts the communication burden to the third parties and requires negligible amount of communication from data holders which are assumed to be resource limited. On the other hand, the protocol in [1] requires all the communication to be performed by data holders, which is not appropriate for our scenario where data holders (e.g. sensor nodes and RFID readers) have limited resources.

Analysis of overall communication costs for different number of data holders are depicted in Figure 6, which shows that communication cost of our protocol remains the same for varying number of data holders. On the other hand, communication cost of the protocol in [1] increases with increasing number of data holders. However, the amount of increase in communication cost gets smaller as number of data holders increase, due to the same reasoning as in Figure 3. As

Figure 6 shows, overall communication cost of our protocol is more than the protocol in [1]. However when communication costs of data holders are compared, our protocol outperforms the protocol in [1] as shown in Figure 7.

7 Discussions and Conclusion

In this paper, we proposed a privacy preserving distributed clustering protocol for horizontally partitioned data using secret sharing scheme, which is homomorphic with respect to addition operation. The model that we adopted is unprecedented in the sense that it uses two non-colluding third parties. The idea of using two third parties that greatly alleviates the computation and communication burden of the data holders is especially useful in applications such as sensor networks and RFID where data holders are resource-limited sensor nodes and RFID readers. When compared to the most efficient former techniques, which exclusively rely on the computation and communication capabilities of the data holders, our protocol can run even on the most simple data holders, such sensor nodes or RFID readers. One can even think that there is no need for the data holders to store the actual data. It is true that the communication overhead between the two third parties is greater than the other protocols. Nevertheless, third parties can easily be equipped with high computation capability and bandwidth.

The use of two third parties and non-collusion property are realistic when they are chosen to have conflicting interests in the data mining results of the actual data. As an example, one third party can be a consumer organization who is interested in the privacy of consumers, while the other is representative of the industry - they both have interests in the right outcome, but will never collude.

We proved information theoretically that the third parties cannot gather any information about the data under the non-colluding assumption. Therefore, our protocol adopts a stronger security model than computational infeasibility, which is used by the majority of other privacy preserving data mining algorithms.

Other two benefits of the proposed protocol are that the data holders do not need to be synchronized or to share keys. Almost all previously proposed methods rely on synchronous and interactive protocols, where data holders must always be on-line during the protocol execution, while the data holders in our protocol can send the shares of their data asynchronously at their convenience. Since there is no communication between the data holders, there is no need for them to share keys; therefore there is no key distribution problem.

And finally, the model based on the use of two third parties and homomorphic secret sharing can be extended to other data types and different dissimilarity metrics.

References

1. Inan, A., Saygın, Y., Savaş, E., Hintoğlu, A.A., Levi, A.: Privacy preserving clustering on horizontally partitioned data. In: Privacy Preserving Clustering on Horizontally Partitioned Data, p. 95. IEEE Computer Society, Los Alamitos (2006)

2. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: SIGMOD 2000. Proceedings of the 2000 ACM SIGMOD international conference on Management of data, pp. 439–450. ACM Press, New York (2000)
3. Lindell, Y., Pinkas, B.: Privacy preserving data mining. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 36–54. Springer, Heidelberg (2000)
4. Evfimievski, A., Gehrke, J., Srikant, R.: Limiting privacy breaches in privacy preserving data mining. In: PODS 2003. Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp. 211–222. ACM Press, New York (2003)
5. Evfimievski, A., Srikant, R., Agarwal, R., Gehrke, J.: Privacy preserving mining of association rules. Inf. Syst. 29(4), 343–364 (2004)
6. Kargupta, H., Datta, S., Wang, Q., Sivakumar, K.: Random-data perturbation techniques and privacy-preserving data mining. Knowl. Inf. Syst. 7(4), 387–414 (2005)
7. Saygin, Y., Verykios, V.S., Clifton, C.: Using unknowns to prevent discovery of association rules. SIGMOD Rec. 30(4), 45–54 (2001)
8. Kantarcıoğlu, M., Clifton, C.: Privacy-preserving distributed mining of association rules on horizontally partitioned data. IEEE Transactions on Knowledge and Data Engineering 16(9), 1026–1037 (2004)
9. Oliveira, S., Zaiane, O.R.: Privacy preserving clustering by data transformation. In: 18th Brazilian Symposium on Databasesn, pp. 304–318 (2003)
10. Oliveira, S., Zaiane, O.R.: Achieving privacy preservation when sharing data for clustering. In: Jonker, W., Petković, M. (eds.) SDM 2004. LNCS, vol. 3178, pp. 67–82. Springer, Heidelberg (2004)
11. Oliveira, S., Zaiane, O.R.: Privacy-preserving clustering by object similarity-based representation and dimensionality reduction transformation. In: Workshop on Privacy and Security Aspects of Data Mining (PSDM 2004) in conjunction with the Fourth IEEE International Conference on Data Mining (ICDM 2004), pp. 21–30 (2004)
12. Merugu, S., Ghosh, J.: Privacy-preserving distributed clustering using generative models. In: ICDM 2003. Proceedings of the Third IEEE International Conference on Data Mining, Washington, DC, USA, pp. 211–218. IEEE Computer Society, Los Alamitos (2003)
13. Klusch, M., Lodi, S., Moro, G.: Distributed clustering based on sampling local density estimates. In: IJCAI 2003. Proceedings of the 18th International Joint Conference on Artificial Intelligence, pp. 485–490. AAAI Press (2003)
14. Vaidya, J., Clifton, C.: Privacy-preserving k-means clustering over vertically partitioned data. In: KDD 2003. Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 206–215. ACM Press, New York (2003)
15. Jha, S., Kruger, L.P.M.: Privacy preserving clustering. In: ESORICS'05:10th European Symposium On Research In Computer Security, pp. 397–417 (2005)
16. Asmuth, C., Bloom, J.: A modular approach to key safeguarding. IEEE Transactions on Information Theory 29(2) (1983)
17. Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)