An Efficient Approach for Privacy Preserving Distributed K-Means Clustering Based on Shamir's Secret Sharing Scheme

Sankita Patel, Sweta Garasia, and Devesh Jinwala

S.V. National Institute of Technology, Surat, Gujarat, India
{sjp,g.sweta,dcj}@coed.svnit.ac.in

Abstract. Privacy preserving data mining has gained considerable attention because of the increased concerns to ensure privacy of sensitive information. Amongst the two basic approaches for privacy preserving data mining, viz. Randomization based and Cryptography based, the later provides high level of privacy but incurs higher computational as well as communication overhead. Hence, it is necessary to explore alternative techniques that improve the overheads. In this work, we propose an efficient, collusion-resistant cryptography based approach for distributed K-Means clustering using Shamir's secret sharing scheme. As we show from theoretical and practical analysis, our approach is provably secure and does not require a trusted third party. In addition, it has negligible computational overhead as compared to the existing approaches.

Keywords: Privacy Preservation in Data Mining (PPDM), Secret sharing, Secure Multiparty Computation (SMC).

1 Introduction

Emerging knowledge based systems gather large amount of sensitive information from their customers. Availability of high speed Internet and sophisticated data mining tools has made sharing of this information across the organizations possible. These technologies when combined pose a threat to privacy concerns of individuals. Hence, there is a need to view data mining tools from different perspective i.e. adding privacy preserving mechanism yielding Privacy Preserving Data Mining (PPDM). Privacy preserving data mining aims to achieve data mining, while hiding sensitive data from disclosure or inference.

In general, for knowledge based systems, data is located at different sites and bringing data together at one place for analysis is not possible due to privacy laws or policies [1]. Hence incorporating privacy preserving mechanisms for distributed databases is necessary for such applications. For the distributed databases, data may be horizontally partitioned or vertical partitioned [1]. In horizontal partitioning, different sites collect the same feature set about different entities while in vertical partitioning, different sites collect different feature sets for the same set of entities. These partitioning models are formally defined in [2]. In this paper, we refer the horizontal partitioning model.

T. Dimitrakos et al. (Eds.): IFIPTM 2012, IFIP AICT 374, pp. 129–141, 2012.

Among the two main categories of PPDM approaches viz. *Randomization based* and *Cryptography based*, later provides higher level of privacy but poor scalability [3]. Amongst the two main Cryptography based approaches, the Secure Multiparty Computation (SMC) [4] provides higher level of privacy but incurs higher computational and communication overhead. As compared, homomorphic encryption based approach provides high level of privacy but incurs higher computational cost. This issue requires critical investigation when applied to data mining. This is so, since data mining requires huge databases as input; hence scalable techniques for privacy preserving data mining are needed to handle them. Therefore, in this paper, we mainly focus on reducing the computational cost of privacy preserving data mining algorithm. The secret sharing based approach is an attractive solution for PPDM which greatly reduces the computational and communication cost of SMC and provides high level of privacy [5].

In this paper, we focus on clustering application of data mining in distributed scenario. As discussed, Cryptography based approaches achieve high level of privacy but the resultant protocols are inefficient in terms of computation and communication overhead. As discussed further in section 2, the oblivious transfer based approaches proposed in [7-9] are not scalable due to their high computational and communicational overhead. Homomorphic encryption based approaches proposed in [9-11] are computationally expensive due to their complex public key operations. Hence, the scope of above two approaches is limited to small datasets and it is necessary to explore alternative technique that is scalable in terms of dataset size. Secret sharing based approaches proposed in [12] [13] aim to achieve this. However, approaches proposed in [12] [13] use either a dedicated server or Trusted Third Party (TTP) to achieve privacy. In practical scenario, the assumption about TTP cannot always be ensured and if ensured, compromise in TTP will jeopardize the privacy.

In this paper, we propose an algorithm for privacy preserving distributed clustering based on the paradigm of Shamir's secret sharing [14]. We modify the widely used K-means clustering algorithm [15-17] to run it in the distributed scenario and incorporate privacy preserving feature in it. We allow parties to collaboratively perform clustering and thus avoiding trusted third party. We compare our protocol with oblivious polynomial based and homomorphic encryption based protocols proposed in [11]. Our approach is more relevant in reducing computational cost as compared to communication cost (that does not constitute our major focus as of now, as mentioned earlier). It outperforms all the existing approaches in presence of very large datasets. Our theoretical and practical simulation supports the above argument. Further, our approach is collusion-resistant and avoids trusted third party.

2 Related Work

The review of state of the art methods for PPDM may be found in [3] [18-20]. Based on this review, PPDM approaches are classified into two categories: 1. *Randomization Based* and 2. *Cryptography Based*. The randomization based approach for privacy preserving clustering has been addressed in [6]. In this, the data being clustered is randomly modified first and then clustering is performed on the modified data. This

results in approximately correct clusters. Approaches in the first category incur low computation and communication cost but compromise with the level of privacy.

The second category of approaches i.e. cryptography based approaches provide high level of privacy but at the cost of high computation and communication cost [5]. A broad overview of the intersection between the fields of cryptography and privacy-preserving data mining may be found in [21]. The Secure Multiparty Computation has been applied for clustering in [7-9]. The limitation of these approaches is that they are computationally expensive and hence their scope is limited to small datasets only.

The second category in cryptography based approach is the homomorphic encryption. A homomorphic encryption scheme allows certain algebraic operations to be carried out on the encrypted plaintext, by applying an efficient operation to the corresponding cipher text [22]. Privacy preserving clustering based on homomorphic encryption is proposed in [9-11]. Authors in [9] and [10] address privacy preserving clustering for arbitrarily-partitioned data for semi honest two party case models. However, the public key encryption schemes used in above techniques are computationally expensive and their scope is limited to small datasets. Authors in [11] address design and analysis of privacy-preserving k-means clustering algorithm for horizontally partitioned data using oblivious polynomial evaluation and homomorphic encryption. They only present the two party case for semi-honest model. Further, the scope of algorithms is limited to small datasets.

An attractive approach for privacy preserving data mining which is recently being introduced is based on the paradigm of secret sharing [14][23]. Detailed study of comparison of encryption-based techniques and secret sharing is given in [5]. According to [5], secret sharing for privacy preserving data mining achieves best of both worlds i.e. privacy at the level of SMC based approach and efficiency at the level of randomization based approach. Privacy preserving clustering based on secret sharing has been addressed in [12] [13]. Authors in [12] propose cloud computing based solution using Chinese remainder theorem based method of secret sharing. They rely on cloud computing servers to compute clusters. Authors in [13] propose solution based on additive secret sharing for vertically partitioned data using two non colluding third parties to compute cluster means. In this solution, collusion between two specific parties reveals each entity's distance to each cluster mean. This results in privacy violations.

In this paper, we use paradigm of secret sharing and specifically Shamir's secret sharing scheme [14] to achieve privacy preserving in K-means clustering. Our approach is similar to the one proposed in [24] for association rule mining. We give theoretical and practical analysis of our approach and show that our approach is collusion-resistant and suitable for large datasets due to its low computational overhead. Further it does not require any trusted third party/servers to compute results and does not reveal intermediate private information. To the best of our knowledge, ours is the first approach to privacy preserving clustering based on Shamir's secret sharing.

3 The Proposed Algorithm

We assume here the distributed database scenario in which the data is horizontally partitioned across n parties. We modify widely used K-means clustering algorithm to

execute it for distributed scenario and then to incorporate privacy preserving feature in it. We utilize paradigm of Shamir's secret sharing to incorporate privacy preservation in K-means clustering.

3.1 Building Blocks

In this section, we review Shamir's secret sharing method [14] and distributed K-Means clustering approach without any privacy preserving mechanism [11].

Shamir's Secret Sharing

Shamir's secret sharing proposed in [14], is a form of secret sharing where a secret is divided into parts, giving each participant its own unique part, where some of the parts or all of them are needed in order to reconstruct the secret. The scheme is formally described as follows [14]:

The secret is some data D. The goal is to divide D into n pieces D1… D_n in such a way that:

1. Knowledge of any k or more D_i pieces makes D easily computable;
2. Knowledge of any k-1 or fewer D_i pieces leaves D completely undetermined i.e. all its possible values are equally likely.

Such a scheme is called a (k, n) threshold scheme. The scheme is based on polynomial interpolation: Given k points in the 2-dimensional plane $(x_1, y_1) \ldots . (x_k, y_k)$ with distinct x_i's, there is one and only one polynomial q(x) of degree k – 1 exists such that $q(x_i) = y_i$ for all i. Without loss of generality, we can assume that the data D is (or can be made) a number. To divide it into pieces D_i, we pick a random k-1 degree polynomial $q(x) = a_o + a_i x + \ldots a_{k-1} x^{k-1}$ in which a_o=D, and evaluate:

$$D_1 = q(1) \ldots . D_i = q(i) \ldots . D_n = q(n)$$

Given any subset of k of these D_i values (together with their identifying indices), we can find the coefficients of q(x) by interpolation, and then evaluate D=q(0). Knowledge of just k- 1 of these values, on the other hand, does not suffice in order to calculate D. Pseudo code for the Shamir's scheme for n parties is shown in Figure 1.

In our approach, we use (n, n) threshold scheme. We require each party to participate in the protocol. Without the cooperation of all parties, it is not possible to recover the secret.

Distributed K-Means Clustering

The K-means clustering algorithm [15-17] is a well known unsupervised learning algorithm. It is the method of cluster analysis that aims to partition the objects into k nonempty subsets (clusters), in which each object belongs to the cluster with nearest mean. Given K initial clusters, the algorithm works in two phases: In the first phase, an object is assigned to the cluster to which it is the most similar, based on the distance between the object and the cluster mean. In the second phase, new mean is computed for each cluster. The algorithm is deemed to have converged when no more new assignment are found.

In the distributed scenario, where data are located at different sites, the algorithm for K-Means clustering differs slightly. In distributed scenario, it is desirable to compute cluster means using union of data located at different parties. We use distributed

Pseudo code 1. Shamir's secret sharing

D: Secret value
P: Set of parties P_1, P_2,…, P_n to distribute the shares,
k: Number of shares required to reconstruct the secret.
Phase I: Generating and sending secret shares

1. Select a random polynomial $q(x) = a_{k-1}x_{k-1} +\ldots+ a_1x_1+a_0$ where $a_{k-1}\neq 0$ and $a_0 = D$
2. Choose n publicly known distinct random values x_1, x_2, … , x_n such that $x_i \neq 0$
3. Compute the share of each node p_i, where $share(i)=q(x_i)$
4. for i = 1 to n do
5. Send share i to node P_i.
6. end for

Phase II: Reconstruction
Require: Every party is given a point (a pair of input to the polynomial and output).

7. Given subset of these pairs, find the coefficients of the polynomial using interpolation
8. The secret is the constant term (i.e. D)

Fig. 1. Shamir's secret sharing scheme [14]

K-Means clustering in our work to add privacy preserving feature in it. We adopt Weighted Average Problem proposed in [11] to compute intermediate cluster means. One way to perform distributed K-Means clustering for two parties, namely, A and B is to use Trusted Third Party as shown in Figure 2. Here, Trusted Third Party is used for intermediate computation of cluster means. The problem with this approach is that it discloses intermediate cluster means at various locations while computing $(a_i+d_i)/(b_i+e_i)$ resulting in privacy violations; where (a_i,d_i) and (b_i,e_i) are the sum of samples and no. of samples pair in each clusters for party 1 and party2 respectively. In our approach, we propose new and efficient privacy preserving computation of $(a_i+d_i)/(b_i+e_i)$ using Shamir's secret sharing method. We allow parties to collaboratively compute cluster means and thus totally eliminate trusted third party.

3.2 The Proposed Design

We use following settings in our design. Database DB is horizontally partitioned among n parties (namely P_1, P_2… P_n), where $DB = DB_1 \cup DB_2 \ldots \cup DB_n$. In this setting, all the parties have same set of attributes, and unlabeled samples. Now all parties want to conduct distributed k means clustering on their combined data sets, in which no party wants to disclose its raw data set to others because of the concern about their data privacy. We formulate privacy-preserving distributed k means clustering to preserve privacy of each party's data while performing clustering. We assume

semi-honest model [22] here where each party correctly follows protocol run. Further, we assume that each party agrees in initial clusters before performing clustering. Now each party performs iteration locally. However, in each iteration, to find new cluster mean μi, all parties have to communicate with each other, as we are not using TTP.

Pseudo Code 2. Distributed K-means clustering [11]

n_A, $n_{B:}$ no. of samples at party A and B
c: total no. of clusters
$u_1 \ldots u_{c:}$ initial clusters

1. do in parallel for each party i ∈ {A,B}
2. begin initialize n_A,n_B,c, μ_1,. . . , μ_c
3. do classify n_A and n_B samples according to nearest μ
4. for i := 1 to c step 1 do
5. Let Ci_A and Ci_B be the i-th cluster for Party A and Party B
6. Party A:Compute $a_i = \Sigma_{xj \in CiA}$ xj and bi=|Ci_A|
7. Party B:Compute di = $\Sigma_{xj \in CiB}$ xj and ei=|Ci_B|
8. Send (a_i, b_i) and (d_i, e_i) to TTP
9. end for
10. end parallel
11. TTP recompute μ_i by (a_i+d_i/b_i+e_i)
12. Send u_i to each party i ∈{A, B }
13. until no change in μ_i
14. return μ_1,. . . , μ_c
15. end

Fig. 2. Distributed K-means clustering with Trusted Third Party[11]

Let the number of clusters is c. Each party finds two values (a_i,b_i) for cluster i using pseudo code shown in Figure 2, where a_i is the sum of samples in cluster i and b_i is the number of samples in cluster i. Now each party has to send pairs ((a_i,b_i),….,(.a_c,b_c)) to each other to find new cluster mean u_i. If these pairs are sent in clear then there is threat to privacy violation of these data. Hence, we consider this pair (a_i,b_i) as a secret in our proposed algorithm. We share these values among the parties using the secure protocol of Shamir's secret sharing. The pseudo code of our approach for n party case is shown in Figure 3.

As shown in Figure 3, each party first decides a polynomial of degree k where k = n-1, and x publicly known distinct random values x_1, x_2,…, x_n. In the first phase, each party wants to send the value $v_s = (a_i,b_i)$ secretly. Each party selects a random polynomial $q(x) = a_{n-1}x_{n-1} + \ldots + a_1x_1 + v_s$, in which the constant term is the secret. Then it computes the shares for other parties such that the share of party P_r, is $shr(v_s,P_r) = q_i(x_r)$, where x_r is the r^{th} element of X. During the second phase, each party adds all the shares received from other parties and then sends this result to all the other parties. That is, party P_i computes $S(x_i) = q_1(x_i) + q_2(x_i) + \ldots + q_n(x_i)$ and sends to all other parties. At the third computation phase, each party P_i will have the n values of polynomial $S(x_i) = q_1(x_i) + q_2(x_i) + \ldots + q_n(x_i)$ at X with the constant term equal to the sum of all

secret values. The linear equation has a unique solution, and each party P_i can solve the set of equations and determine the value. It is the Vandermonde determinant, which gives the solution.

However it cannot determine the secret values of the other parties since the individual polynomial coefficients selected by other parties are not known to P_i.

Pseudo code 3. The proposed approach

P: Set of parties $P_1,P_2,\ldots,P_n$
$v_{is}=(a_i,b_i)$: Secret value of party P_i, where a_i is sum of samples and b_i is no. of samples in cluster
X: A set of n publicly known random values $x_1, x_2,\ldots, x_n$
k: Degree of the random polynomial, here k = n – 1
c: no. of clusters
1: do in parallel for each party $P_i \in \{1...n\}$
find $((a_i, b_i), \ldots, (a_c, b_c))$ using pseudo code described in Figure 2
2: for each secret value $v_{is} \in \{a_i,b_i\}$
3: Select a random polynomial $q_i(x) = a_{n-1}x_{n-1} + \ldots + a_1x_1 + v_{is}$
5: for r = 1 to n do
6: Compute share of party P_r, where $shr(v_{is},P_r) = q_i(x_r)$
7: send $shr(v_{is}, P_r)$ to party P_r
8: receive the shares $shr(v_{rs}, P_i)$ from every party P_r.
9: end for
10: compute $S(x_i) = q_1(x_i) + q_2(x_i) +\ldots+q_n(x_i)$
11: for r = 1 to n do
12: Send $S(x_i)$ to party P_r
13: Receive the results $S(x_i)$ from every party P_r
14: end for
15: Solve the set of equations using Lagrange's interpolation to find the
16: sum of secret values
17: end for
18: Recompute μ_i using sum of samples/no. of samples
19: until termination criteria met

Fig. 3. Privacy preserving distributed K-means clustering using Shamir's secret sharing

4 Theoretical Analysis

Several metrics for evaluating privacy preserving data mining techniques are discussed in [5] [8]. Based on this, we analyze our approach for privacy, correctness, computation cost and communication cost.

4.1 Privacy

In our proposed approach, the secret value v_i of a party P_i cannot be revealed even if all the remaining parties exchange their shares. Since each party P_i executes Shamir's

secret sharing algorithm with a random polynomial of degree n-1, the value of that polynomial at n different points are needed in order to compute the coefficients of the corresponding polynomial, i.e., the secret value of party P_i. P_i computes the value of its polynomial at n points as shares, and then keeps one of these shares for itself and sends the remaining n-1 shares to other parties. Since all n shares are needed to reveal the secret, other parties cannot compute secret even if they combine their shares.

Further, no party learns anything more than its prescribed output. This is so, because as per the approach followed (explained in section 3.2), every party shares its local cluster means as the secret; for which it chooses different polynomial randomly. Hence, it is not possible for a party to determine the secret values of other parties, since the individual polynomial coefficient selected by each party is not known to other parties. In addition, disclosure of intermediate cluster means during the program execution is prevented as intermediate cluster means are calculated at each site and there is no need to communicate them.

4.2 Correctness

Each party is guaranteed that the output that it receives is correct. Assuming that party P_i has private vector A_i. According to method, they have to perform addition of all shares to get the secret value. The secret value is the constant term of the sum polynomial $S(x) = q_1(x) + q_2(x) + \ldots + q_n(x)$, so we need to solve the linear equations, noting there are n unknown coefficients and n equations.

$$D = \begin{vmatrix} x_1^{n-1} & x_1^{n-2} & \ldots & x_1 & 1 \\ x_2^{n-1} & x_2^{n-2} & \ldots & x_2 & 1 \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ x_n^{n-1} & x_n^{n-2} & \ldots & x_n & 1 \end{vmatrix}$$

It is the Vandermonde determinant. When $D = \prod_{i,j=0,i<j}^{n}(xi - xj) = 0$, that is $x_i \neq x_j$, the equations has a unique solution, and each party P_i can solve the set of equations and determine the value of $\sum_{i=1}^{n} = Aij$. However it cannot determine the secret values of the other parties since the individual polynomial coefficients selected by other parties are not known to P_i.

4.3 Computation Cost

The computation cost depends on the initial clusters and the no. of iterations required for finding final clusters. We give here the computation cost for single iteration. Assume that for every party P_i, the cost of generating random polynomial $q_i(x)$, i = 1, 2,..., n is C. In proposed approach, we have two values as a secret so we have to generate random polynomial two times. So total computation cost is $O(n(C_1+C_2))$, where C_1= cost for generating random polynomial for sum of samples, C_2= cost for generating random polynomial for no. of samples and n= no of parties. The total number of $2n\ (n-1)$ additions are calculated to find $s(x) = q_1(x) + q_2(x) + ,\ldots, + q_n(x)$.

Efficient $O(n\log^2 n)$ algorithms for polynomial evaluation are available [14]. Hence the computation cost for our proposed approach is quadratic i.e. $O(n^2)$.

4.4 Communication Cost

Assuming there are r attributes in dataset and n parties and k clusters, for one iteration, the communication cost for each party is $kr(n-1)+2k(n-1)$ messages i.e. $O(krn)$. In comparison to Trusted Third Party based approach, our approach incurs more communication cost because for collaboratively computing cluster means, communication between every party is necessary.

5 Experimental Evaluation

We have implemented our algorithm in MATLAB. The experiments are conducted on Intel Core 2 Duo CPU with 4GB RAM and 2.93GHz speed. Our experiments are performed on Small, medium, large and very large data-sets as described below. We took two datasets similar to those used in [11] in order to perform fair comparison. We provide brief outline of datasets here, however interested readers may find details in [25-28]. Dataset1 is Mammal's Milk [25] with 2KB size, 25 samples and 6 attributes per sample. Dataset2 is the river dataset [26] with 25KB size, 84 samples and 15 attributes per sample. Dataset3 is a speech dataset [27] with 650KB size, 5687 samples and 12 attributes per sample. Dataset4 is taken mainly to show the feasibility of our approach for very large dataset. For this purpose, we have experimented with forest cover dataset [28] with 73MB size, 581012 samples and 54 attributes per sample. For our experiment, we select first two samples as initial cluster centers.

We model multiparty case where the number of parties is greater than two by randomly subdividing the samples into equal sized subsets and assigning them to each party. In real environments the size of the sets may be vastly different. We show feasibility of our approach by executing our algorithm on local machine with different processes for different parties. Therefore, the execution time for the algorithm does not include the actual communication time between different parties. We take two different settings to measure the performance of the proposed scheme:

1. Executing our algorithm on four different size datasets.
2. Executing our algorithm with different number of data holders.

To analyze the results, we find computation and communication cost of our algorithm. Computation cost is measured in terms of time required for execution and communication cost is measured in terms of the number of bytes exchanged during execution.

Our first observation is to show the effect of dataset size on computation cost, we run our algorithm for 3 parties and 6 parties and with dataset1, dataset2 and dataset3. The results are shown in Figure 4. As expected, there is a linear relationship between dataset size and computation cost. Further, we also measure execution time of our algorithm on very large forest cover dataset and show that it requires 668.8 seconds to perform clustering with 3 party setting. This observation shows the feasibility of our approach in practical scenario where large datasets exists.

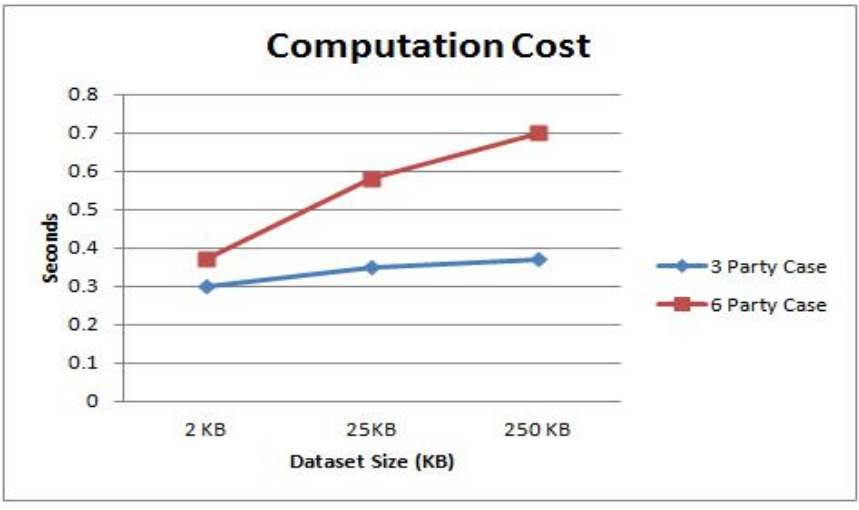

Fig. 4. Effect of dataset size on computation cost

Our next observation in Figure 5 is to show the effect of dataset size on communication cost. As discussed section 4, communication cost linearly depends on the number of attributes in dataset. We obtained similar results in our experimentation also. Dataset2 has more number of attributes as compared to dataset3; so the overall communication cost for dataset2 is more than dataset3. Further, results in Figure 5 show the effect of number of parties on communication cost. Increasing the number of parties has the effect of increasing the communication cost; simply because the number of messages required to be exchanged would be more.

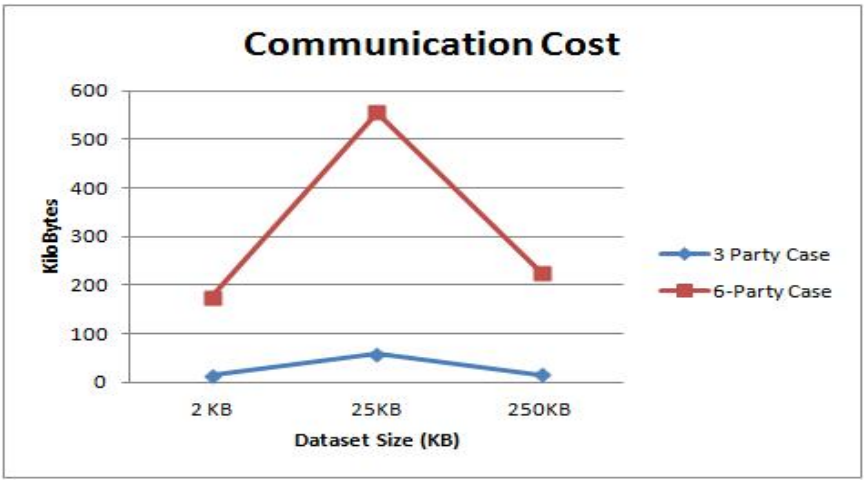

Fig. 5. Effect of dataset size on communication cost

We use results shown in [11] as a base for comparing our protocol against Oblivious Polynomial Transfer and Homomorphic encryption. In [11] authors have also taken dataset2 and dataset3 i.e. river and speech datasets respectively to conduct experiments. Experiments in [11] were conducted for a 2-party case, while here we experiment with a 3-party case. Selection of initials clusters may vary in our case and the one proposed in [11] and so is the overall cost for protocol execution. Hence, for fair comparison, we take attribute/iteration statistics i.e. cost of per attribute clustering in a single iteration of the K-Means algorithm and measure computation and communication cost for the same. We show, for our algorithm, percentage increase in resources with respect to distributed K-Means clustering algorithm without privacy preserving mechanism. Table 1 shows comparison of our protocol and the protocol proposed in [11].

In terms of computation overhead, our approach is about 200 times faster than the homomorphic encryption based approach for river dataset and about 85 times faster than the speech dataset. This is due to the fact that our approach uses only primitive

Table 1. Comparison of our approach with Oblivious Polynomial Evaluation and Homomorphic Encryption based approaches

Test	Communication Overhead * Percentage increase in bytes attributes/iteration	Computation Overhead *Percentage increase in milliseconds attributes/iteration
River Dataset		
Distributed K-Means Clustering (without privacy preserving)	0%	0%
Oblivious Polynomial Evaluation [11]	40116.47%	22715.16%
Homomorphic Encryption [11]	314.35%	4915.67%
Our Protocol	533.33%	25.26%
Speech Dataset		
Distributed K-Means Clustering (without privacy preserving)	0%	0%
Oblivious Polynomial Evaluation [11]	34402.07%	6919.87%
Homomorphic Encryption [11]	268.08%	1474.58%
Our Protocol	533.33%	17.5%
*Percentage increase in resources is calculated with respect to Distributed K-Means Clustering approach without privacy preserving mechanism		

operations to perform clustering and eliminates costly public key operations that are required in homomorphic encryption based approach. Hence, our approach is more suitable for the practical scenario where organizations own large datasets.

In terms of communication overhead, our approach incurs slightly more overhead as compared to that in homomorphic encryption based approach. It is to be noted that results in [11] are for a two party case, whereas our results are for a 3 party case (the minimum parties required in our approach is two). We believe that our approach would be more efficient in terms of communication cost as compared to corresponding homomorphic encryption based approach in case of increased number of parties.

6 Conclusion

We presented an efficient algorithm for privacy preserving distributed K-Means clustering using Shamir's secret sharing scheme. Our approach collaboratively computes cluster means and hence avoid trusted third party. We compared our approach with the oblivious polynomial evaluation and homomorphic encryptions based approaches proposed in [11] and show that in terms of computation cost, our approach is hundreds of magnitude faster than the oblivious polynomial evaluation and homomorphic encryption based approaches and hence is more suitable for large datasets in practical scenario.

Currently our algorithm supports horizontal partitioning in presence of semi honest adversary model. As a future work, we intend to extend our algorithm in vertical partitioning in presence of malicious adversary model. In addition, we intend to show the results from a realistic distributed emulation.

References

1. Shaneck, M., Kim, Y., Kumar, V.: Privacy Preserving Nearest Neighbor Search. In: ICDM Workshops, pp. 541–545 (2006)
2. Aggarwal, C.C., Yu, P.S.: Privacy-Preserving Data Mining: A Survey. In: Michael, G., Sushil, J. (eds.) Handbook of Database Security, pp. 431–460. Springer (2007)
3. Wu., X., Chu, C.H., Wang, Y., Liu, F., Yue, D.: Privacy preserving data mining research: current status and key issues. In: 7th International Conference on Computational Science ICCS 2007, pp. 762–772 (2007)
4. Goldreich, O.: The Foundations of Cryptography, vol. 2. Cambridge Univ. Press, Cambridge (2004)
5. Pedersen, T.B., Saygin, Y., Savas, E.: Secret sharing vs. encryption-based techniques for privacy preserving data mining. In: UNECE/Eurostat Work Session on SDC (2007)
6. Oliveira, S.R.M.: Privacy preserving clustering by data transformation. In: 18th Brazilian Symposium on Databases, pp. 304–318 (2003)
7. Vaidya, J., Clifton, C.: Privacy-preserving k-means clustering over vertically partitioned data. In: 9th ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining. ACM Press (2003)
8. Inan, A., Kaya, S.V., Saygin, Y., Savas, E., Hintoglu, A.A., Levi, A.: Privacy preserving clustering on horizontally partitioned data. Data Knowl. Eng., 646–666 (2007)

9. Jagannathan, G., Wright, R.N.: Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In: KDD, pp. 593–599 (2005)
10. Bunn, P., Ostrovsky, R.: Secure two-party k-means clustering. In: ACM Conference on Computer and Communications Security, pp. 486–497 (2007)
11. Jha, S., Kruger, L., McDaniel, P.: Privacy Preserving Clustering. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 397–417. Springer, Heidelberg (2005)
12. Upmanyu, M., Namboodiri, A.M., Srinathan, K., Jawahar, C.V.: Efficient Privacy Preserving K-Means Clustering. In: Chen, H., Chau, M., Li, S.-h., Urs, S., Srinivasa, S., Wang, G.A. (eds.) PAISI 2010. LNCS, vol. 6122, pp. 154–166. Springer, Heidelberg (2010)
13. Doganay, M.C., Pedersen, T.B., Saygin, Y., Savas, E., Levi, A.: Distributed privacy preserving k-means clustering with additive secret sharing. In: 2008 International Workshop on Privacy and Anonymity in Information Society, Nantes, France, pp. 3–11 (2008)
14. Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)
15. Forgey, E.: Cluster analysis of multivariate data: Efficiency vs. interpretability of classification. Biometrics 21(768) (1965)
16. Lloyd, S.P.: Least squares quantization in PCM. IEEE Transactions on Information Theory 28, 129–137 (1982)
17. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–296 (1967)
18. Kantarcioglu, M., Clifton, C.: Privacy-preserving Distributed Mining of Association Rules on Horizontally Partitioned Data. In: ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD), pp. 639–644 (2002)
19. Verykios, S., Bertino, E., Fovino, I., Provenza, L., Saygin, Y., Theodoridis, Y.: State-of-the-art in Privacy Preserving Data Mining. ACM SIGMOD Record 33(1), 50–57 (2004)
20. Bertino, E., Fovino, I., Provenza, L.: A Framework for Evaluating Privacy Preserving Data Mining Algorithms. Data Mining and Knowledge Discovery 11(2), 121–154 (2005)
21. Pinkas, B.: Cryptographic techniques for privacy-preserving data mining. SIGKDD Explor. Newslett. 4(2), 12–19 (2002), http://doi.acm.org/10.1145/772862.772865, doi:10.1145/772862.772865
22. Lindell, Y., Pinkas, B.: Secure multiparty computation for privacy-preserving data mining. Journal of Privacy and Confidentiality 1(1), 59–98 (2009)
23. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: 19th Annual ACM Conference on Theory of Computing (STOC), pp. 1–10. ACM Press (1988)
24. Ge, X., Yan, L., Zhu, J., Shi, W.: Privacy preserving distributed association rule mining based on a secret sharing technique. In: Second International Conference on Software Eng. and Data Mining, pp. 345–350 (2010)
25. http://www.uni-koeln.de/themen/statistik/data/cluster/milk.dat
26. Information and Computer Science. COIL 1999 Competition Data, The UCI KDD Archive. University of California Irvine (October 1999), http://kdd.ics.uci.edu/databases/coil/coil.html
27. Information and Computer Science. Japanese Vowels. University of California Irvine (June 2000), http://kdd.ics.uci.edu/databases/JapaneseVowels/JapaneseVowels.html
28. http://archive.ics.uci.edu/ml/datasets/Covertype