Privacy Preserving Distributed K-Means Clustering in Malicious Model Using Zero Knowledge Proof

Sankita Patel[1], Viren Patel[2], and Devesh Jinwala[1]

[1] S.V. National Institute of Technology, Surat, Gujarat, India
[2] Government Engineering College, Dahod, Gujarat, India
{sankitapatel,virenjpatel,dcjinwala}@gmail.com

Abstract. Preserving Privacy is crucial in distributed environments wherein data mining becomes a collaborative task among participants. Solutions proposed on the lines of cryptography involve use of classical cryptographic constructs in data mining algorithms. Applicability of solutions proposed depends on the adversary model in which it is able to preserve privacy. Existing cryptography based solutions for privacy preserving clustering aim to achieve privacy in presence of semi honest adversary model. For the practical applicability of the solutions in real world settings, support of malicious adversary model is desirable. As per our literature survey, the existing research lacks any fool proof solution for privacy preserving distributed clustering in malicious adversary model. In this paper, we propose privacy preserving distributed K-Means clustering of horizontally partitioned data that supports privacy in malicious adversarial model. The basic construct involves use of secret sharing mechanism clubbed with code based zero knowledge identification scheme. We use secret sharing for privately sharing the information and code based identification scheme to add support against malicious adversaries.

Keywords: Privacy Preservation in Distributed Data Mining (PPDDM), Secure Multiparty Computation, Secret Sharing, Zero Knowledge Proof.

1 Introduction

Data mining research deals with the investigation of efficient techniques for the extraction of potentially useful information from large collections of data.Recent studies[1] have thrown light on some of the major challenges for data mining. One of the necessities identified is the increased user-friendliness of data mining results. This in turn, poses a threat to privacy concerns of individuals. Hence, there is a need to add privacy preserving mechanisms in data mining; yielding Privacy Preserving Data Mining (PPDM) [2].

In past one decade researchers have shown good interests in PPDM field. Soon after the field was introduced, many research groups started working on solutions for privacy preserving data mining (PPDM)[3][4]. The cryptographic approach is one of those directions [5] where researchers have come up incrementally with better and efficient results. The only drawback of cryptography based approach is its high overhead[6]. Hence, for this approach, one of the chief concerns is to minimize the over-heads incurred in implementing the protocols.

C. Hota and P.K. Srimani (Eds.): ICDCIT 2013, LNCS 7753, pp. 420–431, 2013.

However, there are other vital issues also associated with the existing solutions. In this paper, we concentrate on the clustering application of data mining and especially the K-Means clustering algorithm [7]. Existing solutions proposed for privacy preserving K-Means clustering are [2][8-14]. To the best of our knowledge, all of these solutions provide security in presence of the semi honest adversary model in which participating parties follow the prescribed protocol but try to infer private information using the messages they receive during protocol. Increased use of this model is also because of its less strict definition of security and privacy. Another model that can be considered is the malicious model in which parties arbitrarily deviate from the protocol run in order to infer private information. Although semi-honest model is realistic, solutions devised in malicious model provide higher security in real world settings [15].

In PPDM research, the assumption that no other party should be trusted is closest to reality, where individual never knows who to trust. More than that, even after the successful exchange of knowledge, an assurance about the integrity and trustworthiness of the shared information is required. Fulfillment of such conditions strengthens the protocol and can make it worthy of use in malicious models.

In literature, mechanisms to handle malicious environments are proposed [15-25]. Solutions for implementing basic building blocks in malicious model are proposed in [15-19][21][22][24]. [20][25] give solution for association rule mining in malicious model while [23] gives solution for k-nn classifier. As we further discuss in section 2.3, none of these mechanisms attempts to give solution for the data mining application like clustering. In this paper, we attempt to do so by adding malicious adversary support inthe privacy preserving distributed K-Means clustering algorithm proposed in [2].

1.1 Organization

The remainder of this paper is organized as follows: In section 2, we discuss background and related work. Section 3 describes our proposed approach. In section 4 and 5, we show theoretical and experimental analysis respectively.

2 Back Ground and Related Work

2.1 Zero Knowledge Proof Systems

In cryptography, a zero knowledge proof or zero knowledge protocol is an interactive method for one party to prove to another that a statement is true, without revealing anything other than the veracity of the statement. An effective definition of zero knowledge proofs of knowledge is given in [26] along with its relevance to identification scheme that we use in this paper.

2.2 Code Based Cryptography and Linear Codes

The identification scheme that we use in this paper is based on error-correcting code theory. We use definitions of linear codes and q-ary syndrome decoding problem from [27].

A linear code is an error-correcting code for which any linear combination of code words is also a code word. We can define it more precisely as:

"A linear code of length n and rank k is a linear subspace C with dimension k of the vector space F_q where F_q is the finite field with q elements, q a prime power and k<n. Such a code is called q-ary code. The vectors in C are called code words. The size of the code is the number of code words and equals q_k."

The weight, ω, of a codeword is the number of its elements that are nonzero and the distance between two codewords is the hamming distance between them.

The distance d of a code is minimum weight of its nonzero codewords, or equivalently, the minimum distance between distinct codewords. The error-correcting capability of such a code is the maximum number of d errors that the code is able to decode. A linear code of length n, dimension k, and distance d is called an [n,k,d] code.

Q-ary Syndrome Decoding Problem

Given H ϵR q–ary(n, r), y ϵR F_q^n, and an integer ω > 0, output a word x ϵ F_q^n, such that w(s) <= ω, H.sT= y. This problem remains NP-complete [28].

2.3 Related Work

PPDM approaches are classified into two categories [6]: 1. *Randomization Based* and 2.*Cryptography Based*. The randomization based approach for privacy preserving clustering has been addressed in [29] and approaches in this category incur low computation and communication cost but compromise with the level of privacy. The cryptography based approaches provide high level of privacy but at the cost of high computation and communication cost [6]. In this category, the privacy preservation in clustering has been achieved using the Secure Multiparty Computation (SMC) [8-10], the Homomorphic encryption[10-12] and the secret sharing [2][13][14] based techniques.

However, to the best of our knowledge, all of the above privacy preserving clustering protocols were proven or claimed to be secure only in the semi-honest model. In [30], a systematic method is described for converting protocols that are secure in the semi-honest model to ones that are equally secure and privacy-preserving in the malicious model with the use of commitment schemes and zero-knowledge proofs. The first attempt towards adding malicious adversary support in distributed association rule mining in data grids is proposed in [20]. In [15], authors show several constructions on equality, dot product and set-intersection operations in malicious model for the first time. In [19], authors proposed efficient and secure dot product and set-intersection protocols in the malicious model while reducing the computational and communication complexity of the proof of knowledge of [15] drastically.Recently, [21] proposed efficient set operations against the malicious adversaries. They assume no trusted set up or trusted third party for the computation, thus increasing the communication overhead. In [22], authors propose private vector addition protocol using probabilistic zero-knowledge protocol. In [23], authors add malicious adversary support to already existing privacy-preserving secure scalar product for private distributed k-nn classifier. In [24], authors present an implementation of Yao's protocol with the cut-and-choose methodology, which is secure in the presence of malicious adversaries.

Relation between secret sharing and Zero knowledge proof has been discussed in [31]. Authors in [32] suggest verifiable secret sharing as one of the ways to add malicious adversary support. Authors in [25] proposed Peer for privacy(P4P) framework for privacy preserving data mining using Verifiable secret sharing to add malicious adversary support.

It is apparent however, that none of the paper gives solution for clustering in distributed environment in malicious model.

3 The Proposed Approach

3.1 Privacy Preserving Distributed K-Means Clustering Using Shamir's Secret Sharing Scheme [2]

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 Weighted Average Problem to compute intermediate cluster means in distributed scenario. To collaboratively compute cluster means, all parties need to send their local clusters to every other party. For example, if two parties want to jointly perform clustering, then each party needs to send its sum of samples and number of samples values (of each cluster) to other party. We shall denote (a_i,b_i) and (d_i,e_i) as sum of samples and number of samples pairs for party A and party B respectively. Joint computation of the clusters can be performed using $(a_i+d_i)/(b_i+e_i)$. If parties send (a_i,b_i) and (d_i,e_i) pairs in clear then there is a threat to privacy violations. In [2], authors proposed new and efficient privacy preserving computation of $(a_i+d_i)/(b_i+e_i)$ using Shamir's secret sharing scheme [33]. The approach is shown in figure 1.

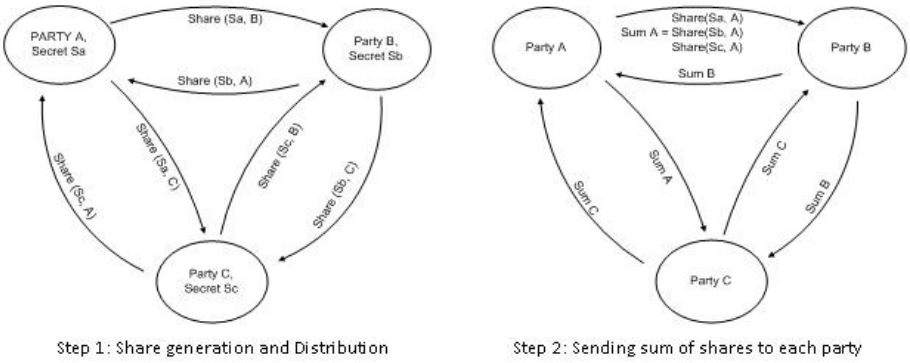

Fig. 1. Privacy Preserving Distributed K-Means clustering using Shamir's Secret Sharing scheme

As shown in figure 1, in step 1, each party generates and distributes the shares. In step 2, each party performs the addition of the shares it receives including its own share and sends the calculated sum back to every other party. By solving the linear

equations corresponding to the received sums, parties are able to calculate the sum of the secret values of all parties using Lagrange's interpolation.

Taking an example of three party $\{P_1,P_2,P_3\}$ case where they share a public constants C = {3, 4, 2} corresponding to each party and have private polynomials Q = $\{2x^2 + x + c, x^2 + 3x + c, 4x^2 + 3x + c\}$. The values they want to secretly share is let's say {3, 4, 3} respectively. Let's take an example where each party generates share for every party including self. Now when party P_1 sends its share secretly to P_3, it uses equation $Q_1(C_3)$, i.e. it sends $2(2)^2 + 2 + 3 = 13$ to P_3, similarly P_2 sends 14 to P_3 and P_3 calculates its own share to be 25 by $Q_3(C_3)$. This way P_3 receives three quadratic equations including its own; their addition gives 52. Similarly P_1 and P_2 each will receive three values each and add up to get 94 and 150 respectively. These sums are called *sum of shares*.

3.2 Attacks in Malicious Adversary Model

In this paper, we concentrate on the malicious adversary attacks that are possible for the approach proposed in [2]. Some possibilities in malicious environment like a party aborting the protocol abruptly or sending blank messages cannot be avoided even while using standard zero knowledge proofs [22].These situations can easily be handled by programming. However, in practical scenario following attacks are possible for the approach proposed in [2]:

1. *Inconsistent shares, valid sum of shares*; (here inconsistent shares are the shares which when used for reconstruction of secret, don't reveal correct secret.)
2. *Inconsistent shares, invalid sum of shares*
3. *Consistent shares, invalid sum of shares*

From the above listed attacks, in attack 1 and 2, malicious party generates and distributes inconsistent shares. This results in incorrect sum of shares and eventually incorrect cluster computation by all parties including malicious party. In practical scenario, malicious party often wants to get correct result but prevent others from getting correct result. Hence, by distributing inconsistent shares, malicious party will not achieve its actual goal.

The only way for the malicious party to get correct clusters but prevent others from getting correct clusters is using the scenario mentioned above in Attack 3. Here, all parties including malicious parties generate and distribute consistent shares. But malicious party sends invalid sum to all other parties in step 2 of algorithm shown in figure 1. If all other parties are honest, this will prevent honest parties to get correct clusters. However, malicious party gets correct clusters because the sum of shares it receives from honest parties are valid. Hence, attack 3 is important to thwart against for the approach proposed in [2].

In this paper, we attempt to provide the solution to attack 3. At the point of sharing secret sums, we use zero knowledge proof identification scheme that identifies whether the secret sum calculated by sending party is actually the one received.

3.3 Verifying the Sum of Shares

A sum of shares received by one party P_1 is the sum calculated by P_3 as explained in section 3.1. In this case we consider, P_3 sends 52 as the sum of shares to P_1 and P_2. In case of P_1, its own share in 52 is 13. So the partial sum of shares that needs to be verified without P_1's own share will be 39.

In our verification scheme, we consider 39 as partial sum, the secret value that needs to be verified and should be proved by party sending it. For every such s, the prover (party P_3 in our case here) can generate a public (y, H, ω) such that $y = H.s^T$, H being a common random (r × n) q-ary parity matrix, and ω is the weight of code; where n is the size of the code, k is the dimension of q-ary parity matrix and r is equal to (n-k). Exactly following the zero knowledge identification scheme in [28], the sequence of messages between prover and verifier happens as shown in figure 2.

The scheme involves two parties; a Prover and a Verifier that communicate with each other in a five pass protocol. The Prover attempts to prove his knowledge of certain value that is known by Verifier but cannot be disclosed to him. The scheme utilizes q-ary codes over a finite field F, pseudo-random number generators and hash functions.

Once all parties verify their sum of shares from all other parties, they can solve a set of equations to get final sum of secret values of every party. If the sum of shares is not correctly verified by any party, then that party sends accusation to all other party and the protocol terminates.

P is prover and V is the verifier. Let us have q = 64, n = 6, r = 3;
H is r × n parity check matrix;
s is the integer value to be proved and verified; s is a liner code over F_q^n of weight ω,
α is randomly selected over F_q^n;
u is randomly selected over F_q^n;
γ is randomly selected q-ary code over F_q^{n*};
Σ is a permutation, $\Pi_{\gamma,\Sigma}$ is a permutation function and h is a hash function.
Step 1: P calculates $C_1 = h\ (\Sigma \parallel \gamma \parallel H.u^T)$ and $C_2 = h\ (\Pi_{\gamma,\Sigma(u)} \parallel \Pi_{\gamma,\Sigma(s)})$ and sends to V
Step 2: V replies with α
Step 3: P calculates $\beta = \Pi_{\gamma,\Sigma(u + \alpha s)}$ and sends to V
Step 4: V randomly selects b from {0,1}
Step 5: If P receives b = 0, it sends Σ, γ to V,
V verifies $C_1 = h\ (\Sigma \parallel \gamma \parallel H\ \Pi_{\gamma,\Sigma}^{-1}(\beta)^T - \alpha\ y)$
If P receives b = 1, it sends $\Pi_{\gamma,\Sigma(s)}$ to V,
V verifies $C_2 = h\ (\beta - \alpha\Pi_{\gamma,\Sigma(s)}, \Pi_{\gamma,\Sigma(s)})$ and wt $(\Pi_{\gamma,\Sigma(s)}) = \omega$

Fig. 2. Verification of integer value using Zero Knowledge Proof identification scheme [28]

The pseudo code of our proposed approach is shown in figure 3.As shown in figure 3, each party while sending the sum of shares, also prove the correctness of partial sum. Similarly, each party while receiving the sum of shares needs to verify with the party that has sent this sum too. If all sums are verified by all parties, then only the protocol can proceed. Upon getting the wrong/invalid sum by other party, party sends accusation to that party and protocol terminates.

4 Theoretical Analysis

In this section, we give security and overhead analysis of the subcomponent of our approach. We concentrate only on zero knowledge identification scheme which we use to add support of malicious adversary model.

4.1 Security Analysis

As explained in section 3, the construction we use to add the security against malicious adversary model is the zero knowledge interactive proof for P(I, s) in the random oracle model; where I is the public construct consisting y, H and w. In identification scheme,

---

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 sum $S(x_i)$ from every party P_r
14: Provethe partial sum $S(x_i)$- $shr(v_{is}, P_r)$ to every party P_r
15: Verify the partial sum $S(x_i)$- $shr(v_{is}, P_r)$ with every party P_r
16: If (partial sum from party P_r not verified) then
17: Broadcast an accusation to Party P_r
18: Terminate the algorithm.
19: endif
20: end for
21: Solve the set of equations using Lagrange's interpolation to find the
22: sum of secret values
23: end for
24: Recompute$\mu_i$using sumof samples/no. of samples
25: until termination criteria met

Fig. 3. Privacy preserving distributed K-means clustering in malicious adversary model using Zero Knowledge Proof

the secret key holder can prove his knowledge of s by using two blending factors: the transformation by means of permutation and a random vector. A dishonest prover not knowing s; can cheat the verifier with probability of q/2(q-1); where q is the power of prime number (in our case 64). Thus the protocol has to be run several times to detect cheating prover. In our case, we run the protocol 16 times to re-duce the probability of cheating the verifier to 2^{-16}. Further, the security of the identification scheme relies on the properties of random linear q-ary codes.

4.2 Computational Cost

Adding support of malicious adversary in [2] adds extra computational cost. This includes the matrix vector multiplication and generation of pseudo-random numbers in the code based identification scheme. Apart from this, as the identification scheme requires parties to compute random on their own, this also adds up to computational overhead.

In order to find computation cost, we use following parameters from the identification scheme:

N: the number of bits needed to encode an element of F_q = 8 bits
ℓ_h: the output size of the hash function h = 160 bits
δ: the number of rounds = 16

The size of the matrix in bits would be (r × n) N = 144 bits

As each party acts as a prover in our scheme, the computational complexity for ZKP over F_q for each party is given as:

δ((n^2)multiplications + (n^2)additions) i.e. O(n^2)

4.3 Communication Cost

In [2], the number of messages exchanged after single round of clustering algorithm for a single party is O(akn) and for n parties it reaches to O(akn^2); where a is the number of attributes in dataset, k is the numbers of cluster centers and n is the number of parties.

However, the ZKP scheme used to support the malicious behavior will also need some amount of message exchange among parties. The cost in bytes will depend on the size of q-ary codes used. Let N be the number of bits needed to encode an element of F_q, in our case q is 64, i.e. 2^6, our n is 6 in this case. Bits needed to represent each element of F_q will be thus 4. So N = 4. But in practical programming, we simply initialize a two-dimensional byte array for linear code and parity matrix. This will make our N = 8. The output size of our hash function is 160. Number of rounds δ is 16.

Size of hash: l_h = 160 bits
Size of α: l_α = 6•8 = 48 bits

Size of γ: $l_\gamma = 6 \bullet 8 = 48$ bits
Size of β: $l_\beta = 6 \bullet 8 = 48$ bits
Size of Σ: $l_\Sigma = 6 \bullet 8 = 48$ bits

So the total number of bits exchanged per attribute:

$=\delta(2 \bullet l_h + l_\alpha + l_\beta + 1 + (l_\gamma + l_\Sigma + nN)/2)$
$=16(2 \bullet 160 + 6 \bullet 8 + 6 \bullet 8 + 1 + (6 \bullet 8 + 6 \bullet 8 + 6 \bullet 8)/2)$
$=16 (320 + 48 + 48 + 1 + (48 + 48 + 48)/2)$
$=7824$ bits $= 978$ bytes

Hence, the communication cost for verification of value by one party will be 0.98KB and for n parties it reaches to $0.98 \times n \times (n-1)$ KB and hence $O(n^2)$.

5 Experimental Results and Analysis

We implemented our algorithm in JAVA. The experiments are conducted on Intel Core 2 Duo CPU with 4GB RAM and 2.3GHz speed. The datasets are taken from UCI machine learning repository. We provide brief outline of datasets here, however interested readers may find details at UCI machine learning repository. Dataset1 is Mammal's Milk with 2KB size, dataset2 is the river dataset with 25KB size and dataset3 is a water treatment dataset with 82KB sizes. For our experiment, we select initial cluster centers randomly. To test the distributed application on real time data,we divided all the three datasets randomly into three sets. These three data sets were then placed on three different machines to perform real time distributed clustering. Our test application successfully shows fully functional distributed clustering over real network.

The Zero Knowledge Identification Scheme was simulated by creating Prover and Verifier programs communicating through java sockets for the verification of each "partial sum". As discussed in section 3, the Prover and Verifier follows 5-pass 16-round identification scheme. Thus for one attribute/iteration, number of messages exchanged will be $(5 \cdot n \cdot n-1)$ as Prover + $(5 \cdot n \cdot n-1)$ as Verifier through communication channel. i.e. $2 \cdot (5 \cdot n \cdot n-1)$ messages.

A 16-round of zero knowledge identification protocol practically takes 185 milliseconds on local computer and 3011ms through LAN. The practical cost of this tunes out to be $2 \cdot 185$ms = 370ms between two parties and $370 \cdot (n \cdot n-1)/2$ for overall exchange over local computer and $2 \cdot 3011$ ms = 6022 ms between two parties making it $6022 \cdot (n \cdot n-1)/2$ through network communication channel.

Table 1 shows the percentage increase in overhead in terms of computational cost and communication cost for our approach. The observation differs here from those shown in [2] as in [2], the experiments were performed on local machine while in our experiment we show true distributed emulation of our algorithm.

Table 1. Comparison of Privacy Preserving Distributed K-Means clustering in presence of semi honest adversary and malicious adversary model

Test	Communication Overhead * Percentage increase in bytes attributes/iteration	Computation Overhead *Percentage increase in time(ms) attributes/iteration
Mammal Dataset		
Privacy preserving K-Means Clustering in semi honest model	168%	77.26 %
Privacy preserving K-Means clustering in malicious model	19152%	9988.95%
River Dataset		
Privacy preserving K-Means Clustering in semi honest model	336%	226.23 %
Privacy preserving K-Means clustering in malicious model	19320 %	18462.68%
Water treatment Dataset		
Privacy preserving K-Means Clustering in semi honest model	504%	229.05%
Privacy preserving K-Means clustering in malicious model	19824%	27231.71%
*Percentage increased in resources is calculated with respect to distributed K-Means clustering algorithm without privacy preserving mechanism.		

6 Conclusion and Future Work

In this paper, we attempted to extend the approach proposed in [2] to add the support of malicious adversary model using code based zero knowledge identification scheme. We give theoretical and practical analysis of our proposed approach by considering 16 round ZKP scheme. In our proposed approach, the probability of cheating the verifier is 2^{-16} which is reasonably small. As we verify only integer values, the size of matrix used is small and it results in acceptable computation and communication cost. Experimental results show that adding malicious adversary support slows down the performance of the algorithm. But at the same time we achieve fair conduction of protocol in presence of malicious adversary model.

In future, we intend to consider the of inconsistent share distribution and provide the solution for the same. We also intend to reduce the overheads incurred by the algorithm.

References

1. Kriegel, H.P., Borgwardt, K.M., Kröger, P., Pryakhin, A., Schubert, M., Zimek, A.: Future trends in data mining. Data Mining and Knowledge Discovery 15(1), 87–97 (2007)

2. Patel, S., Garasia, S., Jinwala, D.: An Efficient Approach for Privacy Preserving Distributed K-Means Clustering Based on Shamir's Secret Sharing Scheme. In: Dimitrakos, T., Moona, R., Patel, D., McKnight, D.H. (eds.) Trust Management VI. IFIP AICT, vol. 374, pp. 129–141. Springer, Heidelberg (2012)
3. Agrawal, R., Srikant, R.: Privacy-preserving data mining. ACM SIGMOD 29(2), 439–450 (2000)
4. Lindell, Y.: Privacy Preserving Data Mining. J. Cryptology, IACR, 177–206 (2002)
5. Pinkas, B.: Cryptographic Techniques for Privacy Preserving Data Mining. SIGKDD Explorations 4(2), 12–19 (2002)
6. Wu, X., Chu, C.-H., Wang, Y., Liu, F., Yue, D.: Privacy Preserving Data Mining Research: Current Status and Key Issues. In: Shi, Y., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2007, Part III. LNCS, vol. 4489, pp. 762–772. Springer, Heidelberg (2007)
7. Lloyd, S.P.: Least squares quantization in PCM. IEEE Transactions on Information Theory 28, 129–137 (1982)
8. Vaidya, J., Clifton, C.: Privacy-preserving k-means clustering over vertically partitioned data. In: Proc. 9th ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining. ACM Press (2003)
9. 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)
10. Jagannathan, G., Wright, R.N.: Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In: Proc. KDD, pp. 593–599 (2005)
11. Bunn, P., Ostrovsky, R.: Secure two-party k-means clustering. In: Proc. ACM Conference on Computer and Communications Security, pp. 486–497 (2007)
12. Jha, S., Kruger, L., McDaniel, P.: Privacy Preserving Clustering. In: de Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 397–417. Springer, Heidelberg (2005)
13. 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)
14. Doganay, M.C., Pedersen, T.B., Saygin, Y., Savas, E., Levi, A.: Distributed privacy preserving k-means clustering with additive secret sharing. In: Proc. 2008 International Workshop on Privacy and Anonymity in Information Society, Nantes, France, pp. 3–11 (2008)
15. Kantarcioglu, M., Kardes, O.: Privacy-preserving data mining in the malicious model. International Journal of Information and Computer Security 2(4), 353–375 (2008)
16. Lindell, Y., Pinkas, B.: An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007)
17. Lindell, Y., Pinkas, B., Smart, N.: Implementing Two-Party Computation Efficiently with Security Against Malicious Adversaries. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 2–20. Springer, Heidelberg (2008)
18. Zhan, J., Chang, L., Matwin, S.: How to Prevent Private Data from being Disclosed to a Malicious Attacker, Learning. In: Lin, T.Y., Xie, Y., Wasilewska, A., Liau, C.-J. (eds.) Data Mining: Foundations and Practice. SCI, vol. 118, pp. 517–528. Springer, Heidelberg (2008)
19. Emura, K., Miyaji, A., Rahman, M.S.: Efficient Privacy-Preserving Data Mining in Malicious Model. In: Cao, L., Feng, Y., Zhong, J. (eds.) ADMA 2010, Part I. LNCS, vol. 6440, pp. 370–382. Springer, Heidelberg (2010)

20. Gilburd, B., Schuster, A., Wolff, R.: Privacy-preserving data mining on data grids in the presence of malicious participants. In: Proc. of HPDC 2004, Honolulu, Hawaii (June 2004)
21. Hazay, C., Nissim, K.: Efficient Set Operations in the Presence of Malicious Adversaries. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 312–331. Springer, Heidelberg (2010)
22. Duan, Y., Canny, J.: Practical private computation and zero-knowledge tools for privacy-preserving distributed data mining, In: SDM 2008 (2008)
23. Shah, D., Zhong, S.: Two methods for privacy preserving data mining with malicious participants. Information Sciences 177(23), 5468–5483 (2008)
24. Lindell, Y., Pinkas, B.: Secure Two-Party Computation via Cut-n-Choose Oblivious Transfer. International Association for Cryptologic Research (2011)
25. Duan, Y., Canny, J.F., Zhan, J.Z.: Efficient Privacy-Preserving Association Rule Mining: P4P Style. In: Proc. CIDM, pp. 654–660 (2007)
26. Feige, U., Fiat, A., Shamir, A.: Zero Knowledge Proofs of Identity. Computing, 210-217 (1987)
27. Bernstein, D.J., Buchman, J., Dahmen, E.: Post-Quantum Cryptography. Springer, Heidelberg (2008)
28. Cayrel, P.-L., Véron, P., El Yousfi Alaoui, S.M.: A Zero-Knowledge Identification Scheme Based on the q-ary Syndrome Decoding Problem. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 171–186. Springer, Heidelberg (2011)
29. Oliveira, S.R.M.: Privacy preserving clustering by data transformation. In: Proc.18th Brazilian Symposium on Databases, pp. 304–318 (2003)
30. Goldreich, O.: Foundations of Cryptography. Cambridge University Press (2001)
31. De Santis, A., Di Crescenzo, G., Persiano, G.: Secret Sharing and Perfect Zero-Knowledge. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 73–84. Springer, Heidelberg (1994)
32. Pedersen, T.B., Saygin, Y., Savas, E.: Secret sharing vs. encryption-based techniques for privacy preserving data mining. In: Proc. UNECE/Eurostat Work Session on SDC (2007)
33. Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)