# Privacy-preserving in Distributed Mining of Horizontal Partitioned Data using DES Algorithm

G.Padma
M.Tech (SE)
Kakatiya Institute of Tech & Sciences, India

G.K.Shailaja
Assoc. Prof (IT)
Kakatiya Institute of Tech & Sciences, India

Rajesham Gajula
Asst.Prof (CSE)
Jayamukhi Institute of Technogical Sciences, India

## ABSTRACT
Data mining can extract important knowledge from large data collections – but sometimes these collections are split among various parties. Privacy concerns may prevent the parties from directly sharing the data, and some types of information about the data. This paper addresses secure mining of association rules over horizontally partitioned data. The methods incorporate cryptographic techniques to minimize the information shared, while adding little overhead to the mining task.

**Index Terms—**Data Mining, Security, Privacy

## 1. INTRODUCTION
Data mining technology has emerged as a means of identifying patterns and trends from large quantities of data. Data mining and data warehousing go hand-in-hand: most tools operate by gathering all data into a central site, then running an algorithm against that data. However, privacy concerns can prevent building a centralized warehouse – data may be distributed among several custodians, none of which are allowed to transfer their data to another site. This paper addresses the problem of computing association rules within such a scenario. We assume homogeneous databases: All sites have the same schema, but each site has information on different entities. The goal is to produce association rules that hold globally, while limiting the information shared about each site. Computing association rules without disclosing individual transactions is straightforward. We can compute the global support and confidence of an association rule AB ) C knowing only the local supports of AB and ABC, and the size of each database:

$$\text{Support}_{AB=>C} = \frac{\sum_{i=1}^{s} \text{support\_countABC(i)}}{\sum_{i=1}^{s} \text{database\_size(i)}}$$

$$\text{Support}_{AB} = \frac{\sum_{i=1}^{s} \text{support\_countAB (i)}}{\sum_{i=1}^{s} \text{database\_size (i)}}$$

Where s is Sites

Note that this doesn't require sharing any individual transactions. We can easily extend an algorithm such as a-priori [1] to the distributed case using the following lemma: If a rule has support > k% globally, it must have support > k% on at least one of the individual sites. A distributed algorithm for this would work as follows: Request that each site send all rules with support at least k. For each rule returned, request that all sites send the count of their transactions that support the rule, and the total count of all transactions at the site. From this, we can compute the global support of each rule and from the lemma) be certain that all rules with support at least k have been found. More thorough studies of distributed association rule mining can be found in [2], [3].The above approach protects individual data privacy, but it does require that each site disclose what rules it supports, and how much it supports each potential global rule. What if this information is sensitive? For example, suppose the Centers for Disease Control (CDC), a public agency, would like to mine health records to try to find ways to reduce the proliferation of antibiotic resistant bacteria. Insurance companies have data on patient diseases and prescriptions.

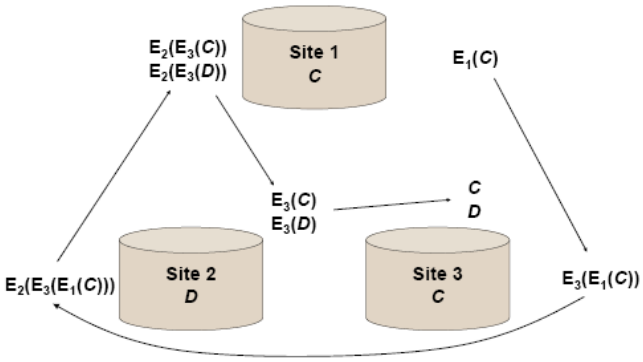


Fig. 1. Determining global candidate itemsets

Mining this data would allow the discovery of rules such as Augmenting & Summer) Infection Fall, i.e., people taking Augmenting in the summer seems to have recurring infections. The problem is that insurance companies will be concerned about sharing this data. Not only must the privacy of patient records be maintained, but insurers will be unwilling to release rules pertaining only to them. Imagine a rule indicating a high rate of complications with a particular medical procedure. If this rule doesn't hold globally, the insurer would like to know this – they can then try to pinpoint the problem with their policies and improve patient care. If the fact that the insurer's data supports this rule is revealed (say, under a Freedom of Information Act request to the CDC), the insurerer could be exposed to significant public relations or liability problems. This potential risk could exceed their own perception of the benefit of participating in the CDC study. This paper presents a

solution that preserves such secrets – the parties learn (almost) nothing beyond the global results. The solution is efficient: The additional cost relative to previous non-secure techniques is O(number of candidate itemsets _ sites) encryptions, and a constant increase in the number of

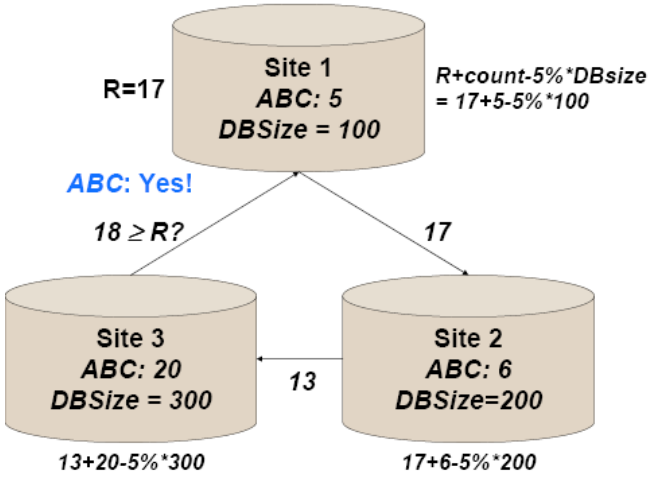


Fig. 2. Determining if itemset support exceeds 5% threshold

messages. The method presented in this paper assumes three or more parties. In the two-party case, knowing a rule is supported globally and not supported at one's own site reveals that the other site supports the rule. Thus, much of the knowledge we try to protect is revealed even with a completely secure method for computing the global results. We discuss the two party case further in Section V. By the same argument, we assume no collusion, as colluding parties can reduce this to the two-party case.

### 1.1. Private Association Rule Mining Overview

Our method follows the basic approach except that values are passed between the local data mining sites rather than to a centralized combiner. The two phases are discovering candidate itemsets (those that are frequent on one or more sites), and determining which of the candidate itemsets meet the global support/confidence thresholds. The first phase (Figure 1) uses commutative encryption. Each party encrypts its own frequent itemsets (e.g., Site 1encrypts itemset *C*). The encrypted itemsets are then passed to other parties, until all parties have encrypted all itemsets. These are passed to a common party to eliminate duplicates, and to begin decryption. (In the figure, the full set of itemsets is shown to the left of Site 1, after Site 1 decrypts.) This set is then passed to each party, and each party decrypts each itemset. The final result is the common itemsets (*C* and *D* in the figure). In the second phase (Figure 2), each of the locally supported itemsets is tested to see if it is supported globally. In the figure, the itemset *ABC* is known to be supported at one or more sites, and each computes their local support. The first site chooses a random value *R*, and adds to *R* the amount by which its support for *ABC* exceeds the minimum support threshold. This value is passed to site 2, which adds the amount by which its support exceeds the threshold (note that this may be negative, as shown in the figure.) This is passed to site three, which again adds its excess support. The resulting value (18) is tested using a secure comparison to see if it exceeds the Random value (17). If so, itemset *ABC* is supported globally. This gives a brief, oversimplified idea of how the method works. Before going into the details, we give background and definitions of relevant data mining and security techniques.

## 2. BACKGROUND AND RELATED WORK

There are several fields where related work is occurring. We first describe other work in privacy-preserving data mining, then go into detail on specific background work on which this paper builds. Previous work in privacy-preserving data mining has addressed two issues. In one, the aim is preserving customer privacy by distorting the data values [4]. The idea is that the distorted data does not reveal private information, and thus is "safe" to use for mining. The key result is that the distorted data, and information on the distribution of the random data used to distort the data, can be used to generate an pproximation to the original data *distribution*, without revealing the original data *values*. The distribution is used to improve mining results over mining the distorted data directly, primarily through selection of split points to "bin" continuous data. Later refinement of this approach tightened the bounds on what private information is disclosed, by showing that the ability to reconstruct the distribution can be used to tighten estimates of original values based on the distorted data [5]. More recently, the data distortion approach has been applied to boolean association rules [6], [7]. Again, the idea is to modify data values such that reconstruction of the values for any individual transaction is difficult, but the rules learned on the distorted data are still valid.

One interesting feature of this work is a flexible definition of privacy; e.g., the ability to correctly guess a value of '1' from the distorted data can be considered a greater threat to privacy than correctly learning a '0'. The data distortion approach addresses a different problem from our work. The assumption with distortion is that the values must be kept private from whoever is doing the mining. We instead assume that *some* parties are allowed to see *some* of the data, just that no one is allowed to see *all* the data. In return, we are able to get exact, rather than approximate, results. The other approach uses cryptographic tools to build decision trees. [8] In this work, the goal is to securely build an ID3 decision tree where the training set is distributed between two parties. The basic idea is that finding the attribute that maximizes information gain is equivalent to finding the attribute that minimizes the conditional entropy. The conditional entropy for an attribute for two parties can be written as a sum of the expression of the form $(v1 + v2) \times \log(v1 + v2)$. The authors give a way to securely calculate the expression $(v1 + v2) \times \log(v1 + v2)$ and show how to use this function for building the ID3 securely. This approach treats privacy preserving data mining as a special case of secure multi-party computation [9] and not only aims for preserving individual privacy but also tries to preserve leakage of any information other than the final result. We follow this approach, but address a different problem (association rules), and emphasize the efficiency of the resulting algorithms. A particular difference is that we recognize that some kinds of information can be exchanged without violating security policies; secure multiparty computation forbids leakage of any information other than the final result. The ability to share non-sensitive data enables highly efficient solutions.The problem of privately computing association rules in *vertically* partitioned distributed data has also been addressed [10]. The vertically partitioned problem occurs when each *transaction* is split across multiple sites, with each site having a different set of attributes for the entire set of transactions.With horizontal partitioning each site has a set of

complete transactions. In relational terms, with horizontal partioning the relation to be mined is the union of the relations at the sites. In vertical partitioning, the relations at the individual sites must be joined to get the relation to be mined. The change in the way the data is distributed makes this a much different problem from the one we address here, resulting in a very different solution.

## 2.1 Mining of Association Rules

The association rules mining problem can be defined as follows: [1] Let I = {i1, i2, . . . , in} be a set of items. Let DB be a set of transactions, where each transaction T is an itemset such that T _ I. Given an itemset X _ I, a transaction T *contains* X if and only if X _ T . An association rule is an implication of the form X ) Y where X _ I, Y _ I and X \Y = ;. The rule X ) Y has *support* s in the transaction database DB if s% of transactions in DB contain X[Y . The association rule holds in the transaction database DB with *confidence* c if c% of transactions in DB that contain X also contains Y. An itemset X with k items called k-itemset. The problem of mining association rules is to find all rules whose support and confidence are higher than certain user specified minimum support and confidence. In this simplified definition of the association rules, missing Items, negatives and quantities are not considered. In this respect, transaction database DB can be seen as 0/1 matrix where each column is an item and each row is a transaction. In this paper, we use this view of association rules.

### *2.1.1 Distributed Mining of Association Rules:*

The above problem of mining association rules can be extended to distributed environments. Let us assume that a transaction database DB is horizontally partitioned among n sites (namely S1, S2, . , Sn) where DB = DB1 [ DB2 [ . . . [ DBn and DBi resides at side Si (1 _ i _ n). The itemset X has *local* support count of X.supi at site Si if X.supi of the transactions contains X. The *global* support count of X is given as X.sup = Pn i=1 X.supi. An itemset X is *globally supported* if X.sup _ s × (Pni=1 |DBi|). Global confidence of a rule X ) Y can be given as {X [ Y } .sup/X.sup. The set of large itemsets L(k) consists of all k-itemsets that are globally supported. The set of locally large itemsets LLi(k) consists of all k-itemsets supported locally at site Si. GLi(k) = L(k) \LLi(k) is the set of globally large k-itemsets

locally supported at site Si. The aim of distributed association rule mining is to find the sets L(k) for all k > 1 and the support counts for these itemsets, and from this compute association rules with the specified minimum support and confidence.The procedure for distributed mining of association rules is summary as follows.

1) **Candidate Sets Generation**: Generate candidate sets CGi(k) based on GLi(k−1), itemsets that are supported by the Si at the (k-1)-th iteration, using the classic apriori candidate generation algorithm. Each site generates candidates based on the intersection of globally large (k-1) itemsets and locally large (k-1) itemsets.

2) **Local Pruning**: For each X 2 CGi (k), scan the database DBi at Si to compute X.supi. If X is locally large Si, it is included in the LLi (k) set. It is clear that if X is supported globally, it will be supported in one site.

3) **Support Count Exchange**: LLi (k) are broadcast, and each site computes the local support for the items in [iLLi (k).

4) **Broadcast Mining Results**: Each site broadcasts the local support for itemsets in [iLLi(k). From this, each site is able to compute L(k).

The details of the above algorithm can be found in [2].

## 2.2 Secure Multi-party Computation

Substantial work has been done on secure multi-party computation. The key result is that a wide class of computations can be computed securely under reasonable assumptions. We give a brief overview of this work, concentrating on material that is used later in the paper. The definitions given here are from Goldreich. For simplicity, we concentrate on the two-party case. Extending the definitions to the multi-party case is straightforward.

### *2.2.1 Security in semi-honest model*

A semi-honest party follows the rules of the protocol using its correct input, but is free to later use what it sees during execution of the protocol to compromise security. This is somewhat realistic in the real world because parties who want to mine data for their mutual benefit will follow the protocol to get correct results. Also, a protocol that is buried in large, complex software cannot be easily altered. A formal definition of private two-party computation in the semi-honest model is given below. Computing a function privately is equivalent to computing it securely.

# 3. SECURE ASSOCIATION RULE MINING

We will now use the tools described above to construct a distributed association rule mining algorithm that preserves the privacy of individual site results. The algorithm given is for three or more parties – the difficulty with the two-party case is discussed in Section V.

## 3.1 Problem Definition

Let 3 be the number of sites. Each site has a private transaction database DBi. We are given support threshold s. The goal is to discover all association rules satisfying the thresholds. We further desire that disclosure be limited: No site should be able to know contents of a transaction at any other site, what rules are supported by any other site, or the specific value of support for any rule at any other site, unless that information is revealed by knowledge of one's own data and the final result. E.g., if a rule is supported globally but not at one's own site, we can deduce that at least one other site supports the rule.

## 3.2 Method

Our method follows the general approach of the FDM algorithm [2], with special protocols replacing the broadcasts of LLi(k) and the support count of items in LL(k). We first give a method for finding the union of locally supported itemsets without revealing the originator of the particular itemset. We then provide a method for securely testing if the support count exceeds the threshold.

### *3.2.1 Secure union of locally large itemset*

In the FDM algorithm, step 3 reveals the large itemsets supported by each site. To accomplish this without revealing what each site supports, we instead exchange locally large itemsets in a way that obscures the source of each itemset.We assume a secure commutative encryption algorithm with negligible collision probability.

The main idea is that each site encrypts the locally supported itemsets, along with enough "fake" itemsets to hide the actual number supported. Each site then encrypts the itemsets from other sites. In Phases 2 and 3, the sets of encrypted itemsets are merged. Since Equation 3 holds, duplicates in the locally supported itemsets will be duplicates in the encrypted itemsets, and can be deleted. The reason this occurs in two phases is that if a site knows which fully encrypted itemsets come from which sites, it can compute the size of the intersection between any set of sites. While generally innocuous, if it has this information for itself, it can guess at the itemsets supported by other sites. Permuting the order after encryption in Phase 1 prevents knowing exactly which itemsets match, however separately merging itemsets from odd and even sites in Phase 2 prevents any site from knowing the fully encrypted values of its own itemsets.1 Phase 4 decrypts the merged frequent itemsets. Commutativity of encryption allows us to decrypt all itemsets in the same order regardless of the order they were encrypted in, preventing sites from tracking the source of each itemset. The detailed algorithm is given in Protocol 1. In the protocol F represents the data that can be used as fake itemsets. |LLei(k)| represents the set of the encrypted k itemsets at site i. Ei is the encryption and Di is the decryption by site i. 1An alternative would be to use an anonymizing protocol [16] to send all fully encrypted itemsets to Site 0, thus preventing Site 0 from knowing which were its own itemsets. The separate odd/even merging is lower cost and achieves sufficient security for practical purposes.

Clearly, Protocol 1 finds the union without revealing which itemset belongs to which site. It is not, however, secure under the definitions of secure multi-party computation. It reveals the number of itemsets having common support between sites, e.g., sites 3, 5, and 9 all support some itemset. It does not reveal *which* itemsets these are, but a truly secure computation (as good as giving all input to a "trusted party") could not reveal even this count. Allowing innocuous information leakage (the number of itemsets having common support) allows an algorithm that is sufficiently secure with much lower cost than a fully secure approach. If we deem leakage of the number of commonly supported itemsets as acceptable, we can prove that this method is secure under the definitions of secure multi-party computation. The idea behind the proof is to show that given the result, the leaked information, and a site's own input, a site can simulate everything else seen during the protocol. Since the simulation generates everything seen during execution of the protocol, the site clearly learns nothing new from the protocol beyond the input provided to the simulator. One key is that the simulator does not need to generate exactly what is seen in any particular run of the protocol. The exact content of messages passed during the protocol is dependent on the random choice of keys; the simulator must generate an equivalent distribution, based on ran dom choices made by the simulator, to the distribution of messages seen in real executions of the protocol. A formal proof that this proof technique shows that a protocol preserves privacy can be found in [9]. We use this approach to prove that Protocol 1 reveals only the union of locally large itemsets and a clearly bounded set of innocuous information.

## 3.3 Theorem

*3.3.1:* Protocol 1 privately computes the union of the locally large itemsets assuming no collusion, revealing at most the result [N i=1LLi(k) and:

1) Size of intersection of locally supported itemsets between any subset of odd numbered sites,

2) Size of intersection of locally supported itemsets between any subset of even numbered sites, and

3) Number of itemsets supported by at least one odd and one even site.

*Proof: Phase 0:* Since no communication occurs in Phase 0, each site can simulate its view by running the algorithm on its own input.

*Phase 1:* At the first step, each site sees LLei−1(k). The size of this set is the size of the global candidate set CG(k), which is known to each site. Assuming the security of encryption, each item in this set is computationally indistinguishable from a number chosen from a uniform distribution. A site can therefore simulate the set using a uniform random number Generator. This same argument holds for each subsequent round.

*Phase 2:* In Phase 2, site 0 gets the fully encrypted sets of itemsets from the other even sites. Assuming that each site knows the source of a received message, site 0 will know which fully encrypted set LLe(k) contains encrypted itemsets from which (odd) site. Equal itemsets will now be equal in encrypted form. Thus, site 0 learns if any odd sites had locally supported itemsets in common. We can still build a simulator for this view, using the information in point 1 above. If there are k itemsets known to be common among all bN/2c odd sites (from point 1), generate k random numbers and put them into the simulated LLei(k). Repeat for each bN/2c−1 subset,etc., down to 2 subsets of the odd sites. Then fill each LLei(k) with randomly chosen values until it reaches size |CGi(k)|.The generated sets will have exactly the same combinations of common items as the real sets, and since the *values* of the items in the real sets are computationally indistinguishable from a uniform distribution, their simulation matches the real values.

The same argument holds for site 1, using information from point 2 to generate the simulator.

*Phase 3:* Site 1 eliminates duplicates from the LLei(k) to generate RuleSet1. We now demonstrate that Site 0 can simulate RuleSet1. First, the size of RuleSet1 can be simulated knowing point 2. There may be itemsets in common between RuleSet0 and RuleSet1. These can be simulated using point 3: If there are k items in common between even and odd sites, site 0 selects k random items from RuleSet0 and inserts them into RuleSet1. RuleSet1 is then filled with randomly generated values. Since the encryption guarantees that the values are computationally indistinguishable from a uniform distribution, and the set sizes |RuleSet0|, |RuleSet1|,and |RuleSet0\RuleSet1| (and thus |RuleSet|) are identical in the simulation and real execution, this phase is secure.

*Phase 4:* Each site sees only the encrypted items after decryption by the preceding site. Some of these may be identical to items seen in Phase 2, but since all items must be in the union, this reveals nothing. The simulator for site i is built as follows: take the values generated in Phase 2 step N − 1 − i, and place them in the RuleSet. Then insert random values in RuleSet up to the proper size . The values we have not seen before are computationally indistinguishable from data from a uniform distribution, and the simulator includes the values we have seen (and knew would be there), so the simulated view is computationally indistinguishable from the real values.The simulator for site N − 1 is different, since it learns RuleSet(k). To simulate what it sees in Phase 4, site N – 1 takes each item in RuleSet(k), the final result, and encrypts it with EN−1. These are placed in RuleSet. RuleSet is then filled with items chosen from F, also encrypted with EN−1.Since the choice of

items from F is random in both the real and simulated execution, and the real items exactly match in the real and simulation, the RuleSet site N − 1 receives in Phase 4 is computationally indistinguishable from the real execution. Therefore, we can conclude that above protocol is privacy preserving in the semi-honest model with the stated assumptions. The information disclosed by points 1-3 could be relaxed to the number of itemsets support by 1 site, 2 sites... N sites if we assume anonymous message transmission. The number of jointly supported itemsets can also be masked by allowing sites to inject itemsets that are not really supported locally. These fake itemsets will simply fail to be globally supported, and will be filtered from the final result when global support is calculated as shown in the next section. The jointly supported itemsets "l eak" then becomes an upper bound rather than exact, at an increased cost in the number of candidates that must be checked for global support. While not truly zeroknowledge, it reduces the confidence (and usefulness) of the leaked knowledge of the number of jointly supported itemsets. In practical terms, revealing the size (but not content) of intersections between sites is likely to be of little concern.

*2) Testing support threshold without revealing support count:* Protocol 1 gives the full set of locally large itemsets LL(k). We still need to determine which of these itemsets are supported globally. Step 4 of the FDM algorithm forces each site to reveal its own support count for every itemset in LL(k). All we need to know is for each itemset X 2 LL(k), is X.sup _ s% × |DB|? The following allows us to reduce this to a comparison against a sum of local values (the *excess support* at each site):

X.sup _ s _ |DB| = s _ (Xni=1|DBi|)Xni=1

X.supi _ s _ (Xni=1|DBi|)Xni=1

(X.supi − s _ |DBi|) _ 0

PTherefore, checking for support is equivalent to checking if ni=1(X.supi − s _ |DBi|) _ 0. The challenge is to do this without revealing X.supi or |DBi|. An algorithm for this is given in Protocol 2.The first site generates a random number xr for each itemset X, adds that number to its (X.supi − s _ |DBi|), and sends it to the next site. (All arithmetic is modm _ 2 _ |DB|, for reasons that will become apparent later.) The random number masks the actual excess support, so the second site learns nothing about the first site's actual database size or support. The second site adds its excess support and sends the value on. The random value now hides both support counts. The last site in the change now has

Pni=1(X.supi − s _ |DBi|) + xr(mod m).

Since the total database size |DB| _ m/2, negative summation will be mapped to some number that is bigger then or equal to m/2. (−k = m − k mod m.) The last site needs to test if this sum minus xr (mod m) is less then m/2. This can be done securely using Yao's generic method [11]. Clearly this algorithm is secure as long as there is no collusion, as no site can distinguish what it receives from a random number. Alternatively, the first site can simply send xr to the last site. The last site learns the actual excess support, but does not learn the support values for any single site. In addition, if we consider the excess support to be a valid part of the global result, this method is still secure.

*Theorem 3.2:* Protocol 2 privately computes globally supported itemsets in the semi-honest model.

*Proof:* To show that Protocol 2 is secure under the semi-honest model, we have to show that a polynomial time simulator can simulate the view of the parties during the execution of the protocol, based on their local inputs and the global result. We also use the general composition theorem for semi-honest computation [9]. The theorem says that if g securely reduces to f, and f is computed securely, then the computation of f(g) is secure.

## 4. EXPERIMENTAL SETUP AND IMPLEMENTATION

We have done mining on shop data bases. This consists of the binary transactions. The table in the database consists of the attributes like Mouse, Keyboard, Pen drive, Card reader, Headphones and Speakers. We have mined the database using apriori algorithm. After conducting the apriori algortithms the support counts which are generated at each individual sites are encrypted using DES Algorithm.The encrypted counts will be sent to all the other sites. The other sites will not know the count because of encryption.After sending all the counts to all the sites the sites will know the total support count but not individual support count

## 5. CONCLUSION

This paper is a implementation version of Privacy-preserving Distributed Mining of Association Rules on Horizontally Partitioned Data Using DES Algorithm. This paper mainly addressed the implementation. Privacy-preserving Distributed Mining of Association Rules on Horizontally Partitioned Data Privacy-preserving Distributed Mining of Association Rules on Horizontally Partitioned Data can be implemented using other secure multy party computation.

## 6. REFERENCES

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proceedings of the 20th International Conference on VeryLarge Data Bases*. Santiago, Chile: VLDB, Sept. 12-15 1994, pp.487–499. [Online]. Available: http://www.vldb.org/dblp/db/conf/vldb/vldb94-487.html

[2] D. W.-L. Cheung, J. Han, V. Ng, A. W.-C. Fu, and Y. Fu, "A fast distributed algorithm for mining association rules," in *Proceedings of the 1996 International Conference on Parallel and Distributed Information Systems (PDIS'96)*. Miami Beach, Florida, USA: IEEE, Dec. 1996,pp. 31–42.

[3] D. W.-L. Cheung, V. Ng, A. W.-C. Fu, and Y. Fu, "Efficient mining of association rules in distributed databases," *IEEE Trans. Knowledge Data Eng.*, vol. 8, no. 6, pp. 911–922, Dec. 1996.

[4] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proceedings of the 2000 ACM SIGMOD Conference on Management of Data*. Dallas, TX: ACM, May 14-19 2000, pp. 439–450. [Online]. Available: http://doi.acm.org/10.1145/342009.335438.

[5] D. Agrawal and C. C. Aggarwal, "On the design and quantification of privacy preserving data mining algorithms," in *Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. Santa Barbara, California, USA:ACM,

May 21-23 2001, pp. 247–255. [Online]. Available: http://doi.acm.org/ 10.1145/375551.375602.

[6] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, "Privacy preserving mining of association rules," in *The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada, July 23-26 2002, pp. 217–228. [Online]. Available: http://doi.acm.org/10.1145/775047.775080

[7] S. J. Rizvi and J. R. Haritsa, "Maintaining data privacy in association rule mining," in *Proceedings of 28th International Conference on Very Large Data Bases*. Hong Kong: VLDB, Aug. 20-23 2002, pp. 682–693. [Online]. Available: http://www.vldb.org/conf/2002/S19P03.pdf

[8] Y. Lindell and B. Pinkas, "Privacy preserving data mining," in *Advances in Cryptology – CRYPTO 2000*. Springer-Verlag, Aug. 20-24 2000, pp. 36–54. [Online]. Available: http://link.springer.de/link/ service/series/ 0558 /bibs/ 1880/18800036.htm

[9] O. Goldreich, "Secure multi-party computation," Sept. 1998, (working draft). [Online]. Available: http://www.wisdom.weizmann.ac.il/_oded/pp.html

[10] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada, July 23-26 2002, pp. 639–644. [Online]. Available: http://doi.acm.org/10.1145/775047.775142

[11] A. C. Yao, "How to generate and exchange secrets," in *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*. IEEE,1986, pp. 162–167.

[12] I. Ioannidis and A. Grama, "An efficient protocol for yao's millionaires' problem," in *Hawaii International Conference on System Sciences(HICSS-36)*, Waikoloa Village, Hawaii, Jan. 6-9 2003.

[13] O. Goldreich, "Encryption schemes," Mar. 2003, (working draft). [Online]. Available: http://www.wisdom.weizmann.ac.il/_oded/PSBookFrag/enc.ps

[14] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978. [Online]. Available: http://doi.acm.org/10.1145/359340.359342