Efficient Protocol for Privacy Preserving Association Rule Mining

Dr. P. R. Deshmukh , Miss Prajakta Jaswante.

Abstract- **Data mining is the most fast growing area today which is used to extract important knowledge from large data collections but often these collections are divided among several parties. This paper addresses secure mining of association rules over horizontally partitioned data. This method incorporates a protocol is that of Kantarcioglu and Clifton well known as K&C protocol. This protocol is based on an unsecured distributed version of the Apriori algorithm named as Fast Distributed Mining (FDM) algorithm of Cheung et al. The main ingredients in our protocol are two novel secure multi-party algorithms one that computes the union of private subsets that each of the interacting players hold and another that tests an element is secured or not.This protocol offers enhanced privacy with respect to the earlier protocols.**

Index Terms - **Privacy Preserving Data Mining; Distributed Computation; Frequent Itemsets; Association Rules.**

I. INTRODUCTION

Data mining has been viewed as a threat to privacy because of the widespread proliferation of electronic data maintained by corporations. This has lead to increased concerns about the privacy of the underlying data. Data mining techniques find hidden information from large database while secret data is preserved safely when data is allowed to access by single person. Now a days many people want to access data or hidden information using data mining technique even they are not fully authorized to access. For getting mutual benefits, many organizations wish to share their data to many legitimate people but without revealing their secret data.

In large applications the whole data may be in single place called centralized or multiple sites called distributed database. Methodologies are proposed by many authors for both centralized as well as distributed database to protect private data. This paper deals with privacy preserving in distributed database environment while sharing discovered knowledge/hidden information to many legitimate people.

In distributed environment, database is a collection of multiple, logically interrelated databases distributed over a computer network and are distributed among number of sites. As the database is distributed, different users can access it without interfering with one another. In distributed environment, database is partitioned into disjoint fragments and each site consists of only one fragment. Data can be partitioned in different ways such as horizontal, vertical and mixed. In horizontal partitioning of data, each fragment consists of a subset of the records of a relation R.

There are several sites (or players) that hold homogeneous databases, i.e., databases that share the same schema but hold information on different entities. The goal is to find all association rules with support at least s and confidence at least c, for some given minimal support size s and confidence level c, that hold in the unified database, while minimizing the information disclosed about the private databases held by those players. The information that we would like to protect in this context is not only individual transactions in the different databases, but also more global information such as what association rules are supported locally in each of those databases.

That goal defines a problem of secure multi-party computation. In such problems, there are M players that hold private inputs, $x1, . . . , x_M$, and they wish to securely compute $y = f(x1, . . . , x_M)$ for some public function f. If there existed a trusted third party, the players could surrender to him their inputs and he would perform the function evaluation and send to them the resulting output. In the absence of such a trusted third party, it is needed to devise a protocol that the players can run on their own in order to arrive at the required output y. Such a protocol is considered perfectly secure if no player can learn from his view of the protocol more than what he would have learnt in the idealized setting where the computation is carried out by a trusted third party. Yao [1] was the first to propose a generic solution for this problem in the case of two players. Other generic solutions, for the multi-party case, were later proposed [2, 3, 4].

Kantarcioglu and Clifton studied that problem in [5] and devised a protocol for its solution. The main part of the protocol is a sub-protocol for the secure computation of the union of private subsets that are held by the different players. (The private subset of a given player, as we explain below, includes the itemsets that are s-frequent in his partial database.) That is the most costly part of the protocol and its implementation relies upon cryptographic primitives such as commutative encryption, oblivious transfer, and hash functions. This is also the only part in the protocol in which the players may extract from their view of the protocol information on other databases, beyond what is implied by the final output and their own input. While such leakage of information renders the protocol not perfectly secure, the perimeter of the excess information is explicitly bounded in [5] and it is argued there that such information leakage is innocuous, whence acceptable from a practical point of view. Herein we propose an alternative protocol for the secure computation of the union of private subsets. The proposed protocol improves upon that in [5] in terms of simplicity and efficiency as well as privacy. In particular, our protocol does not depend on commutative encryption and oblivious transfer. While our solution may leak excess information only to a small number (three) of possible coalitions, unlike the protocol of [5] that discloses information also to some

single players. In addition, we claim that the excess information that our protocol may leak is less sensitive than the excess information leaked by the protocol [5].
The protocol that we propose here may computes a parameterized family of functions, which we call threshold functions, in which the two extreme cases may correspond to the problems of computing the union and intersection of private subsets. Those are in fact general-purpose protocols that can be used in other contexts as well. Another problem of secure multiparty computation that we want solve is the set inclusion problem;(namely, the problem where Alice holds a private subset of some ground set, and Bob holds an element in the ground set, and they wish to determine whether Bob's element is within Alice's subset, without revealing to either of them information about the other party's input beyond the above described inclusion.)

II. LITERATURE REVIEW AND RELATED WORK

Previous work in privacy preserving data mining has considered two related settings. One, in which the data owner and the data miner are two different entities, and another, in which the data is distributed among several parties who aim to jointly perform data mining on the unified corpus of data that they hold.

In the first setting, the goal is to protect the data records from the data miner. Hence, the data owner aims at anonymizing the data prior to its release. The main approach in this context is to apply data perturbation [6, 7]. The idea is that the perturbed data can be used to infer general trends in the data, without revealing original record information.

In the second setting, the goal is to perform data mining while protecting the data records of each of the data owners from the other data owners. This is a problem of secure multiparty computation. The usual approach here is cryptographic rather than probabilistic. Lindell and Pinkas [8] showed how to securely build an ID3 decision tree when the training set is distributed horizontally. Lin et al. [9] discussed secure clustering using the EM algorithm over horizontally distributed data. The problem of distributed association rule mining was studied in [10, 11, 12] in the vertical setting, where each party holds a different set of attributes, and in [6] in the horizontal setting. Also the work of [13] considered this problem in the horizontal setting, but they considered large-scale systems in which, on top of the parties that hold the data records (resources) there are also managers which are computers that assist the resources to decrypt messages; another assumption made in [13] that distinguishes it from [6] and the present study is that no collusions occur between the different network nodes — resources or managers.

all k-itemsets that are locally s-frequent at Dm, $1 \leq m \leq M$. Our main computational goal is to find, for a given threshold support $0 < s \leq 1$, the set of all s frequent itemsets, $Fs := \cup L$ $k=1$ $F^k{}_s$. We may then continue to find all (s, c)-association rules, i.e., all association rules of support at least sN and confidence at least c. (Recall that if X and Y are two disjoint subsets of A, the support of the corresponding association rule $X \Rightarrow Y$ is $supp(X \cup Y)$ and its confidence is $supp(X \cup Y)/supp(X)$.)

III. EXISTING SYSTEM

A. The Fast Distributed Mining Algorithm

The protocol is based on the Fast Distributed Mining (FDM) algorithm of Cheung et al. which is an unsecured distributed version of the Apriori algorithm. Its main idea is that any s-frequent itemset must be also locally s-frequent in at least one of the sites. Hence, in order to find all globally s-frequent itemsets, each player reveals his locally s-frequent itemsets and then the players check each of them to see if they are s-frequent also globally.
The FDM algorithm proceeds as follows:
(1) **Initialization:** It is assumed that the players have already jointly calculated $F^{k-1}{}_s$. The goal is to proceed and calculate $F^k{}_s$.
(2) **Candidate Sets Generation:** Each player P_m computes the set of all $(k - 1)$-itemsets that are locally frequent in his site and also globally frequent; namely, Pm computes the set $F^{k-1,m}{}_s \cap F^{k-1}{}_s$. He then applies on that set the Apriori algorithm in order to generate the set $B^{k,m}{}_s$ of candidate k-itemsets.
(3) **Local Pruning:** For each $X \in B^{k,m}{}_s$, P_m computes $suppm(X)$. He then retains only those itemsets that are locally s-frequent. We denote this collection of itemsets by $C^{km}{}_s$
(4) **Unifying the candidate itemsets:** Each player broadcasts his $C^{km}{}_s$ and then all players compute $C^k := \cup^M{}_{m=1} C^{km}{}_s$
(5) **Computing local supports:** All players compute the local supports of all itemsets in $C^k{}_s$.
(6) **Broadcast Mining Results:** Each player broadcasts the local supports that he computed. From that, everyone can compute the global support of every itemset in $C^k{}_s$. Finally, $F^k{}_s$ is the subset of $C^k{}_s$ that consists of all globally $s$frequent k-itemsets.
In the first iteration, when $k = 1$, the set $C^{1,m}{}_s$ that the mth player computes (Steps 2-3) is just $F^{1,m}{}_s$, namely, the set of single items that are s-frequent in Dm. The complete FDM algorithm starts by finding all single items that are globally s-frequent. It then proceeds to find all 2-itemsets that are globally s-frequent, and so forth, until it finds the longest globally s-frequent itemsets. If the length of such itemsets is K, then in the $(K+1)$th iteration of the FDM it will find no $(K + 1)$-itemsets that are globally s-frequent, in which case it terminates.

B. Disadvantages of Existing System

- Insufficient security
- More time consuming
- Not efficient in term of data mining and security

IV. PROPOSED SYSTEM

The FDM algorithm violates privacy in two stages: In Step 4, where the players broadcast the itemsets those are locally frequent in their private databases, and in Step 6, where they broadcast the sizes of the local supports of candidate itemsets. FDM not efficient in term of database mining as it uses Local pruning technique. Kantarcioglu and Clifton [5] proposed secure implementations of those two steps. Our improvement is with regard to the secure implementation of Step 4, which is the more costly stage of the protocol, and the one in which the protocol leaks excess

information. We proposed global pruning technique which results in improved data mining results. In this Section we describe our proposed system in term of efficient data mining and Kantarcioglu and Clifton's secure implementation of Step 4. After data mining following are the results of various scenarios.
Our proposed system works in below mention first two phases and in third phase we evaluate efficiency.

A] Efficient data mining using proposed data mining algorithm
B] Privacy Preserving using proposed unify – KC algorithm
C] Performance Evaluation

A] Efficient data mining using proposed data mining algorithm

i. **Initialization**: It is assumed that the players have already jointly calculated Fsk−1 . The goal is to proceed and calculate Fsk .
ii. **Candidate Sets Generation**: Each player Pm computes the set of all (k − 1)-itemsets that are locally frequent in his site and also globally frequent; namely, Pm computes the set Fk−1,m s ∩ Fk−1 s . He then applies on that set the Apriori algorithm in order to generate the set Bk,ms of candidate k-itemsets.
iii. **Generation of Unifying Item Set** : We do Union of all Ckms to form Unified Item Set. Which we will denote as X.
iv. **Global Frequent Item set :** For X ∈ Bk,m s , Computes global supp m(X). Compute global support and apply it on Unify Item set then it retains only those itemsets that are Globally s-Frequent
v. **Encrypting and Decrypting Global Item set**: Secure s-globally itemsets using modified Unify-KC algorithm mention in section B.

B] Privacy Preserving using proposed unify – KC algorithm (Protocol 1)

Algorithm

- Input global itemset Gi = { i1,i2,i3…in}
- Enter support and hash key
- Build a Lookup table
- Encrypt Items

```
            for i=1 to length of G
                  Count = 0
                  Item = Gi {i}
               if ( Item (freq) > support)
                   Read hask key Hk
                   Read item to array
                       for j=1 to length of item
                            Char  p = item (j)
                            Char k = hash (count)
                            p' = p + k
           Attach to encrypted Ep
           Count++
               if  count > length (Hk)
                           Count = 0
                       end
                   end
                end
              end
```

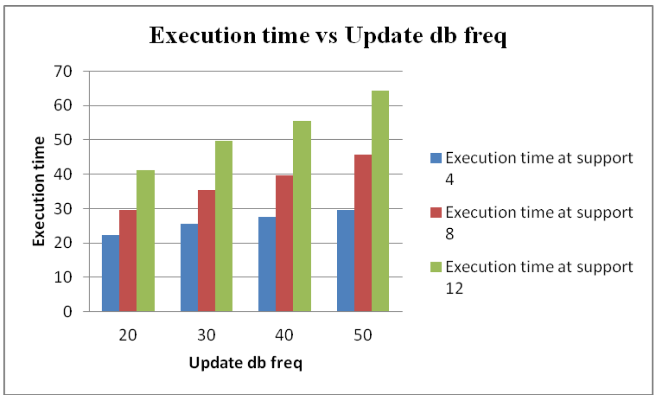
- Check in lookup table for duplicates. Discard duplicate items.
- Decrypt items
- Update lookup table

C] Performance Evaluation

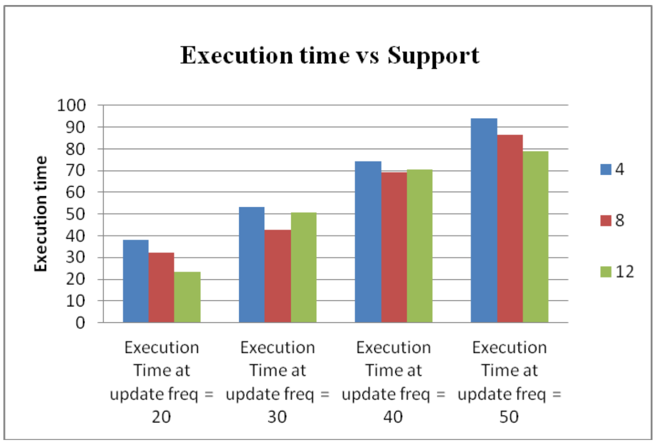
Performance can be evaluated on no of parameters for a system. The parameters we have considered are noted against others when data is mined by our improved protocol. Consider a system of three databases in which any item is searched. Following are the steps followed **-**

- For example a particular name is searched in all the databases.
- It shows the frequency at each databases for which each item occurs.
- Union of the local items takes place and they are passed onto support application.
- Support and hash key is applied thus we get set of items which we call as global itemset.
- All the global itemsets are stored in cache as it helps for fast retrieval.
- Above hash key is used to encrypt the items and it is stored in lookup table for further reference.
- Same lookup table is used to decrypt the items using respective hash key.
- So when the item is searched again it first checks whether it is found in cache. If present it gets the items from cache else it goes to search in local database.
- Thus by mining data by our protocol and securing it We get various parameters which shows relations between them.

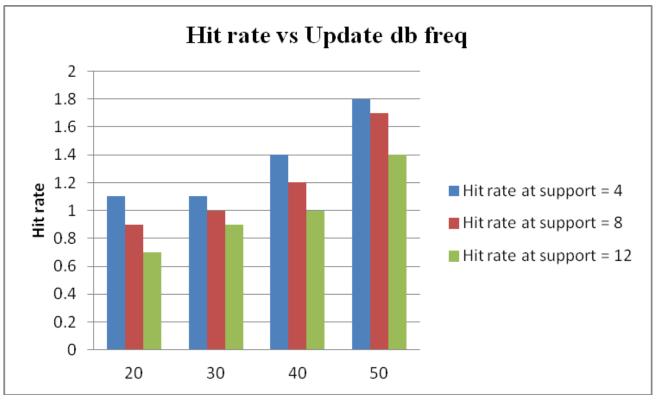
C 1] Efficiency in Mining process
[When support = 4 , 8 , 12; update db freq and execution time are noted.]



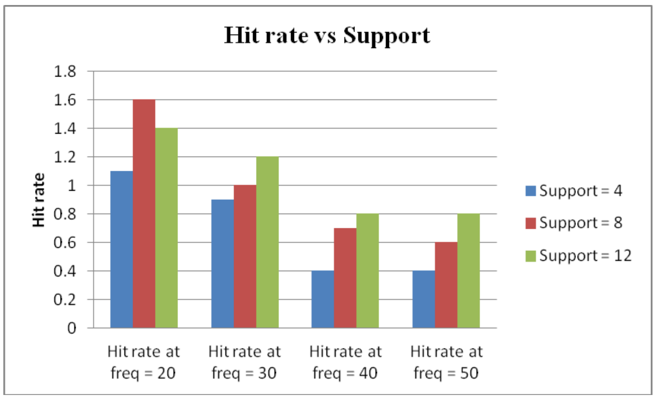
C 2] Efficiency in Mining process
[When update db frequency = 20 , 30 , 40 , 50 ; support and execution time are noted.]



C 3] Efficiency in Mining process
[When support = 4 , 8 , 12 update db frequency and hit rate are noted.]

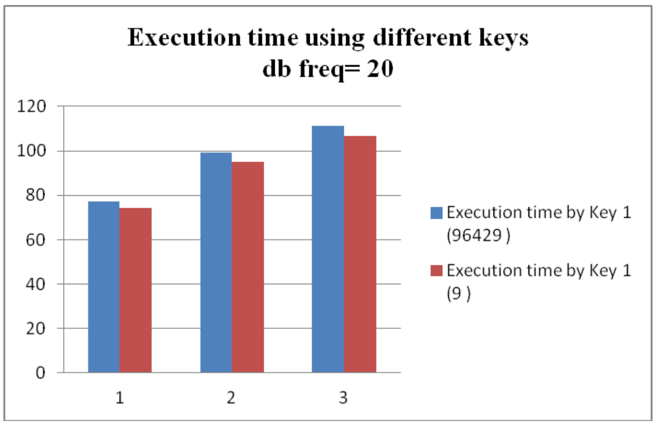


C 4] Efficiency in Mining process
[When db freq = 20 , 30 , 40 , 50 support and hit rate are noted.]

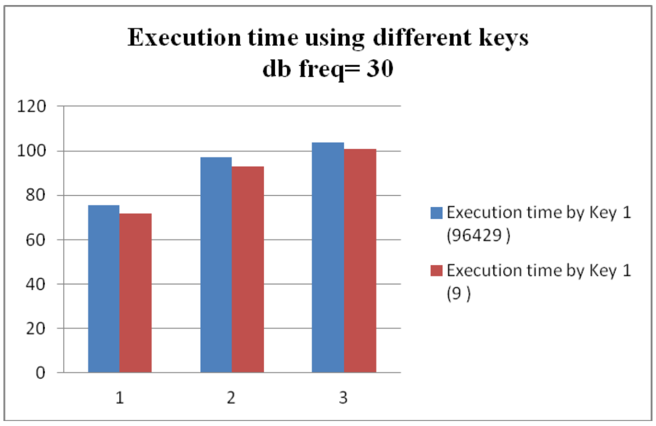


C 5] Efficiency in Security
During encryption when different hash key values are used below are the execution time variations at different database frequency.

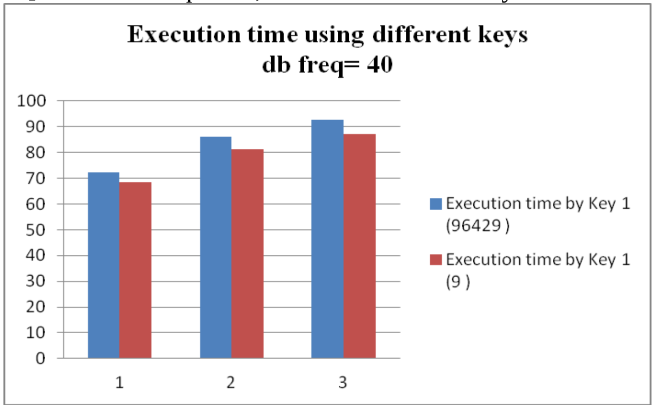
A] When db freq = 20 , two values of hash keys are noted.



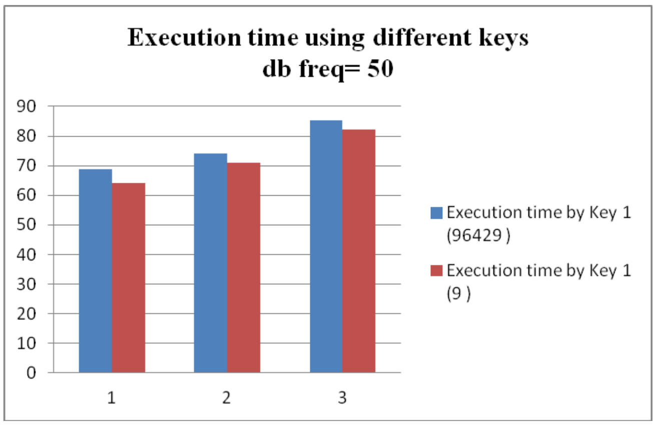
B] When db freq = 30 , two values of hash keys are noted.



C] When db freq = 40 , two values of hash keys are noted.



D] When db freq = 50 , two values of hash keys are noted.



Efficiency Evaluation

We evaluate efficiency against existing system on below parameters:

1] Efficiency in Mining process
A] No of transaction v/s Execution Time
B] Hit Rate Comparison

2) Efficiency in Security
A] Time required for Encryption and Decryption comparison
B] Memory usage comparison

1] No of transaction v/s Execution Time

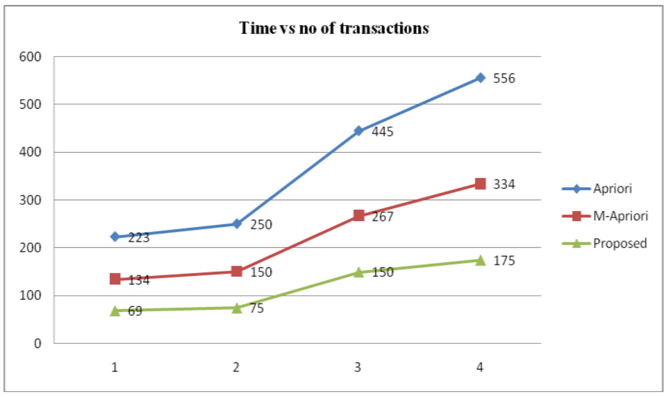
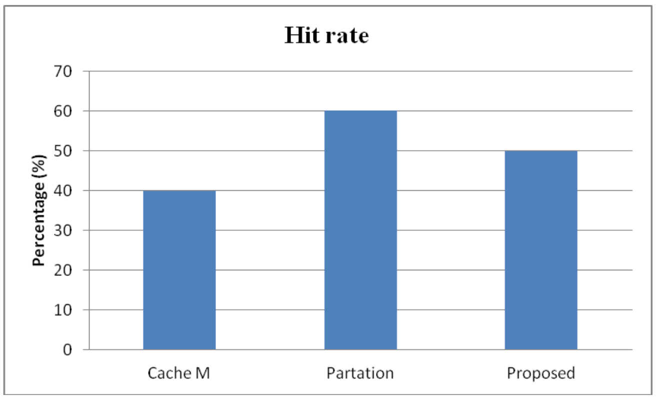


Table 1] **No of transaction v/s Execution Time**

Update db freq	Apriori algo	M-Apriori algo	Proposed algo
20	223	134	69
30	250	150	75
40	445	267	150
50	556	334	175

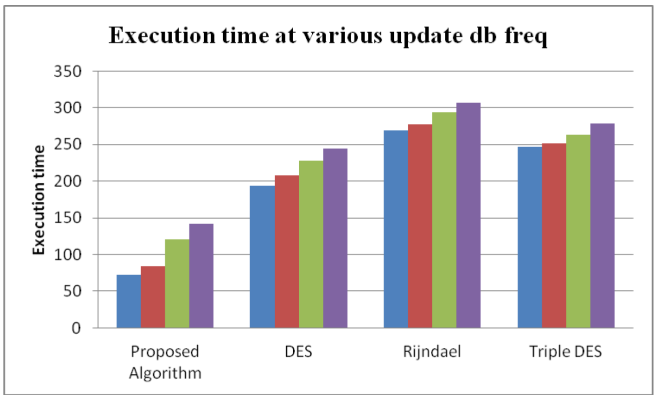
2] Hit Rate Comparison



2] **Hit Rate Comparison**

Percentage	Cache M	Partition	Proposed
%	40	60	50

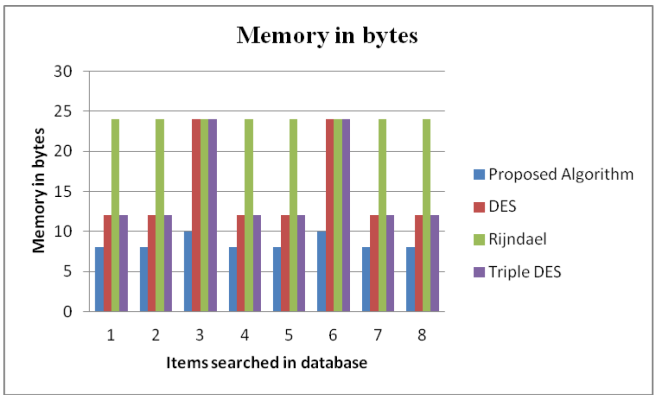
3] Time required for Encryption and Decryption comparison (millisec)



3] Time required for Encryption and Decryption comparison (millisec)

Update db frequency	Proposed Algorithm	DES	Rijndael	Triple DES
20	72	194	269	247
30	84	208	278	251
40	121	228	294	263
50	142	244	307	279

4] Memory usage comparison (in bytes)



4] Memory usage comparison (in bytes)

Original Item	Proposed Algorithm	DES	Rijndael	Triple DES
Nagpur	8	12	24	12
67	8	12	24	12
college	10	24	24	24
er	8	12	24	12
Eng	8	12	24	12
Amravati	10	24	24	24
Ti	8	12	24	12
34	8	12	24	12

V. CONCLUSION

We proposed a protocol for secure mining of association rules in horizontally distributed databases that improves significantly upon the current leading protocol in terms of privacy and efficiency. Our protocol works significantly well in terms of resource consumption as well. Thus above are the results indicates our implementation as superior than the existing system. Concept of global pruning has helped out alot to improve efficiency in terms of data mining and security. We have also provided security to our system by applying encryption algorithm. Thus data can be mined more securely.

VI. REFERENCES

[1] A.C. Yao. Protocols for secure computation. In *FOCS*, pages 160–164, 1982.

[2] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *STOC*, pages 503–513, 1990.

[3] A. Ben-David, N. Nisan, and B. Pinkas. FairplayMP - A system for secure multi-party computation. In *CCS*, pages 257–266, 2008.

[4] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

[5] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Transactions on Knowledge and Data Engineering*, 16:1026–1037, 2004.

[6] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *SIGMOD Conference*, pages 439–450, 2000.

[7] A.V. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *KDD*, pages 217–228, 2002.

[8] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Crypto*, pages 36–54, 2000.

[9] X. Lin, C. Clifton, and M.Y. Zhu. Privacy-preserving clustering with distributed EM mixture modeling. *Knowl. Inf. Syst.*, 8:68–81, 2005.

[10] M. Kantarcioglu, R. Nix, and J. Vaidya. An efficient approximate protocol for privacy-preserving association rule mining. In *PAKDD*, pages 515–524, 2009.

[11] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *KDD*, pages 639–644, 2002.

[12] J. Zhan, S. Matwin, and L. Chang. Privacy preserving collaborative association rule mining. In *Data and Applications Security*, pages 153– 165, 2005.

[13] A. Schuster, R. Wolff, and B. Gilburd. Privacy-preserving association rule mining in large-scale distributed systems. In *CCGRID*, pages 411–418, 2004.