Fast Distributed Mining of Technologies for Privacy Enhancement

T.Santhiya, N.Vigneshkumar

Abstract— Data mining can extract a pattern of important knowledge from large data collections but sometimes these collections are splited into various parties. Privacy concerns which may prevent the parties from directly sharing of a data which matches to the pattern information, and some types of information about the data. This paper point out secure mining of association rules over horizontally partitioned database. The method incorporate cryptographic techniques to minimize the information or data shared, while adding little overhead to the mining task. This proposal contains a protocol for secure mining of association rules in horizontally distributed databases. The current leading protocol which used for secure mining is UNIFI-KC. The protocol based on the Fast Distributed Mining algorithm, and is a portion of an unsecured distributed version of the Apriori algorithm. The main integrant in this protocol are two novel secure multi-party algorithms that is one which computes the union of private subsets that each of the interacting players hold, and another that tests the inclusion of an element held by one player in a subset held by another. This protocol offers enhanced privacy combined to the protocol UNIFI-KC. In addition, it is simpler and is significantly more efficient in terms of communication rounds, communication cost and computational cost.

Index Terms— Data Mining, Frequent Item sets, Association Rules, multi-party, Privacy Preserving Data Mining; Distributed Computation, Unifying lists of locally Frequent Itemsets —Kantarcioglu and Clifton

1. Introduction

In secure mining of association rules in horizontally partitioned databases setting, thereare several sites (or players) that hold the structure of homogeneous databases, i.e., databases that share the same structure of the databases but hold information and other contents on different entities. The goal is to find out 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 reducing the information disclosed about the private databases held by those players. The information that we would like to protects the context that is not only individual transactions between the different databases, but also the database contain more global information such as what association rules are supported locally in each of those databases. That goal identifies several problems of secure multi-party computation. In such problems, there are *M* players hold private inputs, *x*1,…., *xM*, and they wish to securely compute *y* = *f*(*x*1, . . . , *xM*) 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 requiredoutput *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. Other generic solutions, for the multi-party case, were later proposed in [3], [5], [8].

In our problem, the inputs are the partial databases, and the required output is the list of association rules that hold in the unified database with support and confidence no smaller than the given thresholds *s* and *c*, respectively. As the above mentioned generic solutions rely upon a description of thefunction *f* as a Boolean circuit, they can be applied only to small inputs and functions which are realizable by simple circuits. In more complex settings, such as ours, other methods are required for carrying out this computation. In such cases, some relaxations of the notion of perfect security might beinevitable when looking for practical protocols, provided that the excess information is deemed benign (see examples of such protocols in e.g. [5], [6], [7], [8], [9]). 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. (Theprivate 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 [18] in terms of simplicity and efficiency as well as privacy. In particular, our protocol does not depend on commutative encryption and oblivious transfer (what simplifies it significantly

International Journal of Scientific & Engineering Research Volume 6, Issue 5, May-2015
ISSN 2229-5518

and contributes towards much reduced communication and computational costs). While oursolution is still not perfectly secure, it leaks excess informationonly to a small number (three) of possible coalitions, unlike theprotocol of [5] that discloses information also to some singleplayers. In addition, we claim that the excess information that our protocol may leak is less sensitive than the excess information leaked by the protocol of [5].The protocol that we propose here computes a parameterized family of functions, which we call threshold functions, in which the two extreme cases correspond to the problemsof 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 solve here as part of our discussion is the set inclusion problem; namely, the problem where Aliceholds 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.

In previous year various techniques are applied for secure mining of association rules in horizontally partitioned database. These approaches use various techniques such as data perturbation, homo-morphic encryption, keyword search and oblivious pseudorandom functions etc. These privacy preserving approaches are inefficient due to

- Homo-morphic encryption
- Higher computational cost
- In some of the techniques data owner tries to hide datafrom data miner.

Our proposed technologies based on two novel secure multiparty algorithm using these algorithms the protocol provides enhanced privacy, security and efficiency as it uses commutative encryption. Here a protocol for secure mining of association rules in horizontally distributed database. This protocol is based on: FDM Algorithm which is an unsecured distributed version of the Apriori algorithm. In our protocol two secure multiparty algorithms areinvolved:

1. Computes the union of private subsets that each interact ing players hold.
2. Tests the inclus ion of an element held by one player in subset held by another.

In Horizontally partitioned database there are several players that hold homogeneous database. Our protocol offers enhanced privacy with respect to the current leading K and C protocol simplicity, more efficient in terms of communication rounds, communication cost and computational cost. In this problem, the inputs are the partial databases and the required output is the list of association rules that hold in the unified database with support and confidence no smaller than the given thresholds s and c, respectively.

2. PRELIMINARIES:

2.1 Process Design:

Association Rule Algorithms

An association rule is a rule which implies certain association relationships among a set of objects (such as ``occur together'' or ``one implies the other'') in a database. Given a set of transactions, where each transaction is a set of literals (called items), an **association rule** is an expression of the form X Y , where X and Y are sets of items. The intuitive meaning of such a rule is that transactions of the database which contain X tend to contain Y . An example of an association rule is: ``30% of transactions that contain beer also contain diapers; 2% of all transactions contain both of these items''. Here 30% is called the confidence of the rule, and 2% the support of the rule. The problem is to find all association rules that satisfy user-specified minimum support and minimum confidence constraints.

Apriori Algorithm

An association rule mining algorithm, **Apriori** has been developed for rule mining in large transaction databases by IBM's Quest project team[3] . A*itemset* is a non-empty set of items.

They have decomposed the problem of mining association rules into two parts

- Find all combinations of items that have transaction support above minimum support. Call those combinations frequent itemsets.
- Use the frequent itemsets to generate the desired rules. The general idea is that if, say, ABCD and AB are frequent itemsets, then we can determine if the rule AB CD holds by computing the ratio r = support(ABCD)/support(AB). The rule holds only if r >= minimum confidence. Note that the rule will have minimum support because ABCD is frequent. The Apriori algorithm used in Quest for finding all frequent itemsets is given below

```
procedure AprioriAlg()
begin
L_1 := {frequent 1-itemsets};
for ( k := 2; L_{k-1} 0; k++ ) do {
   C_k= apriori-gen(L_{k-1}) ; // new candidates
for all transactions t in the dataset do
{
  for all candidates c C_k contained in t do
        c:count++
  }
  L_k = { c C_k | c:count >= min-support}
}
Answer := _k L_k
end
```

It makes multiple passes over the database. In the first pass, the algorithm simply counts item occurrences to determine the

International Journal of Scientific & Engineering Research Volume 6, Issue 5, May-2015
ISSN 2229-5518

frequent 1-itemsets (itemsets with 1 item). A subsequent pass, say pass k, consists of two phases. First, the frequent itemsets L_{k-1} (the set of all frequent (k-1)-itemsets) found in the (k-1)th pass are used to generate the candidate itemsetsC_k, using the apriori-gen() function. This function first joins L_{k-1} with L_{k-1}, the joining condition being that the lexicographically ordered first k-2 items are the same. Next, it deletes all those itemsets from the join result that have some (k-1)-subset that is not in L_{k-1} yielding C_k. The algorithm now scans the database. For each transaction, it determines which of the candidates in C_k are contained in the transaction using a hash-tree data structure and increments the count of those candidates. At the end of the pass, C_k is examined to determine which of the candidates are frequent, yielding L_k. The algorithm terminates when $L_k$empty.

Distributed/Parallel Algorithms

Databases or data warehouses may store a huge amount of data to be mined. Mining association rules in such databases may require substantial processing power. A possible solution to this problem can be a distributed system.[5] . Moreover, many large databases are distributed in nature which may make it more feasible to use distributed algorithms.

Major cost of mining association rules is the computation of the set of large itemsets in the database. Distributed computing of large itemsets encounters some new problems. One may compute locally large itemsets easily, but a locally large itemset may not be globally large. Since it is very expensive to broadcast the whole data set to other sites, one option is to broadcast all the counts of all the itemsets, no matter locally large or small, to other sites. However, a database may contain enormous combinations of itemsets, and it will involve passing a huge number of messages.

A distributed data mining algorithm FDM (Fast Distributed Mining of association rules) has been proposed by [5], which has the following distinct features.

1. The generation of candidate sets is in the same spirit of Apriori. However, some relationships between locally large sets and globally large ones are explored to generate a smaller set of candidate sets at each iteration and thus reduce the number of messages to be passed.
2. After the candidate sets have been generated, two pruning techniques, local pruning and global pruning, are developed to prune away some candidate sets at each individual sites.
3. In order to determine whether a candidate set is large, this algorithm requires only O(n) messages for support count exchange, where n is the number of sites in the network. This is much less than a straight adaptation of Apriori, which requires O(n^2) messages.

FDM Algorithm:

Distributed/Parallel Algorithms

Databases or data warehouses may store a huge amount of data to be mined. Mining association rules in such databases may require substantial processing power . A possible solution to this problem can be a distributed system.[5] . Moreover, many large databases are distributed in nature which may make it more feasible to use distributed algorithms.

Major cost of mining association rules is the computation of the set of large itemsets in the database. Distributed computing of large itemsets encounters some new problems. One may compute locally large itemsets easily, but a locally large itemset may not be globally large. Since it is very expensive to broadcast the whole data set to other sites, one option is to broadcast all the counts of all the itemsets, no matter locally large or small, to other sites. However, a database may contain enormous combinations of itemsets, and it will involve passing a huge number of messages. A distributed data mining algorithm FDM (Fast Distributed Mining of association rules) has been proposed by [5], which has the following distinct features.

1. The generation of candidate sets is in the same spirit of Apriori. However, some relationships between locally large sets and globally large ones are explored to generate a smaller set of candidate sets at each iteration and thus reduce the number of messages to be passed. After the candidate sets have been generated, two pruning techniques, local pruning and global pruning, are developed to prune away some candidate sets at each individual site. In order to determine whether a candidate set is large, this algorithm requires only O(n) messages for support count exchange, where n is the number of sites in the network. This is much less than a straight adaptation of Apriori, which requires O(n^2) messages.

Let D be a database of N = 18 itemsets over a set of L = 5items, A = *{*1, 2, 3, 4, 5*}*. It is partitioned between M = 3players, and the corresponding partial databases are:

D1 = *{*12, 12345, 124, 1245, 14, 145, 235, 24, 24*}*

D2 = *{*1234, 134, 23, 234, 2345*}*

D3 = *{*1234, 124, 134, 23*}* .

For example, D1 includes N1 = 9 transactions, the third ofwhich (in lexicographic order) consists of 3 items — 1, 2 and4. Setting s = 1*/*3, an itemset is s-frequent in D if it is supported by at least 6 = $sN$of its transactions. In this case,

$F1^{(s)}$= {1, 2, 3, 4}

$F2^{(s)}$= {12, 14, 23, 24, 34}

$F3^{(s)}$= {124}

$F4^{(s)}$= $F_5^{(s)}$= $\emptyset$,

and $F^{(s)}$=$F_1^{(s)}$ U $F_2^{(s)}$ U $F_3^{(s)}$. For example, the itemset34 is indeed globally s-frequent since it is contained in 7 transactions of D. However, it is locally s-frequent only in D2and D3.

In the first round of the FDM algorithm, the three players compute the sets $c_{(1,m)}^{s}$of all 1-itemsets that are locallyfrequent at their partial databases:

$C^{(1,1)}_{s}$= *{*1, 2, 4, 5*}* ,

$C^{(1,2)}_{s}$= *{*1, 2, 3, 4*}* ,

$C^{(1,3)}_{s}$=*{*1, 2, 3, 4*}* .

Hence, $c^{(1)}_{s}$=*{*1, 2, 3, 4, 5*}*. Consequently, all 1-itemsetshave to be checked for being globallyfrequent; that checkreveals that the subset of globally *s*-frequent 1-itemsets is

$F^{(1)}_{s}$=*{*1, 2, 3, 4*}*.

In the second round, the candidate itemsets are:

$C^{(2,1)}_{s}$=*{*12, 14, 24*}*

$C^{(2,2)}_{s}$=*{*13, 14, 23, 24, 34*}*

$C^{(2,1)}_{s}$=*{*12, 13, 14, 23, 24, 34*}* .

(Note that 15, 25, 45 are locally *s*-frequent at *D*1 but theyare not included in $C^{(2,1)}_{s}$since 5 was already found to beg-lobally infrequent.) Hence,

$C^{(2)}_{s}$= *{*12, 13, 14, 23, 24, 34*}*.

Then, after verifying global frequency, we are left with

$F^{(2)}_{s}$=*{*12, 14, 23, 24, 34*}*.

In the third round, the candidate itemsets are:

$C^{(3,1)}_{s}$=*{*124*}* ,$C^{(3,2)}_{s}$=*{*234*}* , $C^{(3,3)}_{s}$=*{*124*}* .

So, $C^{(3)}_{s}$=*{*124, 234*}* and, then,$F^{(2)}_{s}$=*{*124*}*. There are no-more frequent itemsets.

Detailed description

•In Phase 0 (Steps 2-4), the players select the needed cryptographic primitives: They jointly select a commutative cipher, and each player selects a corresponding private random key. In addition, they select a hash function *h* to apply on all itemsets prior to encryption. It is essential that *h* will not experiencecollisions on Ap(Fs(k-1))in order to make it invertible on Ap(Fs(k-1)). Hence, if such collusions occur (an event of a very small probability), a different hash function must be selected. At the end, the players compute a lookup table with the hash values of all candidate itemsets in Ap(Fs(k-1)); that table will be used later on to find the preimage of a given hash value.

• In Phase 1 (Steps 6-19), all players compute a composite encryption of the hashed sets $C_s^{(k,m)}$, $1 \le m \le M$. First(Steps 6-12), each player Pm hashes all itemsets in Ck,m s and then encrypts them using the key Km. (Hashing is needed in order to prevent leakage of algebraic relations between itemsets, see [18, Appendix].) Then, he adds to the resulting set faked itemsets until its size becomes |Ap(F^{k-1}s)|, in order to hide the number of locally frequent itemsets that he has.(Since $C^{k,m}$s⊆Ap(F^{k-1}s), the size of $C^{k,m}$sis bounded by|Ap(F^{k-1}s)|, for all 1 ≤ m ≤ M.) We denote the resultingset by Xm. Then (Steps 13-19), the players start a loopof M − 1 cycles, where in each cycle they perform thefollowing operation: Player Pm sends a permutation of Xmto the next player Pm+1; Player Pm receives from Pm−1 apermutation of the set Xm−1 and then computes a new Xmas Xm= EKm(Xm−1). At the end of this loop, Pm holdsan encryption of the hashed $C^{k,m+1}$s using all M keys. Dueto the commutative property of the selected cipher, Player Pmholds the set {EM(· · · (E2(E1(h(x)))) · · ·) : x ∈$C^{k,m+1}$s}.

• In Phase 2 (Steps 21-26), the players merge the lists ofencrypted itemsets. At the completion of this stage P1 holdsthe union set hashed and then encrypted byall encryption keys, together with some fake itemsets that wereused for the sake of hiding the sizes of the sets; thosefake itemsets are not needed anymore and will be removedafter decryption in the next phase.

The merging is done in two stages, where in the first stagethe odd and even lists are merged separately. As explainedin [18, Section 3.2.1], not all lists are erged at once sinceif they were, then the player who did the merging (say P1)would be able to identify all of his own encrypted itemsets(as he would get them from PM) and then learn in which ofthe other sites they are also locally frequent.

In Phase 3 (Steps 28-34), a similar round of decryptionsis initiated. At the end, the last player who performs the lastdecryption uses the lookup table T that was constructed in Step 4 in order to identify and remove the fake itemsets andthen to recover C^{k}s. Finally, he broadcasts C^{k}sto all his peers.

Going back to the running example in Section 1.1.3, the set F^{2}*s* , consisting of all 2-itemsets that are globally *s*-frequent, includes the itemsets*{*12, 14, 23, 24, 34*}*. Applying on it the Apriori algorithm, we find that Ap(F^{2}_{s}) = *{*124, 234*}*.

Therefore, each of the three players proceed to look for 3-itemsetsfrom Ap(F^{2}_{s}) that are locally *s*-frequent in his partial database.Since $C^{3,1}$s= *{*124*}*, *P*1 will hash and encrypt the itemset 124and will add to it one fake itemset, since |*Ap*(*F*2*s*)| = 2. As$C^{3,2}_{s}$= *{*234*}* and *C*3*,*3*s* = *{*124*}*, also *P*2 and *P*3 will eachuse one fake itemset.

At the completion of the protocol, thethree players will conclude that *C*3
s = *{*124*,* 234*}*. Then, byapplying the protocol in Section 3, they will find out that onlythe first of these two candidate itemsets is globally frequent,hence *F*3*s* = *{*124*}*.

2.2 A secure multiparty protocol for computing the OR of private binary vectors

Protocol UNIFI-KC securely computes of the union of privatesubsets of some publicly known ground set.Sucha problem is equivalent to the problem of computing the OR ofprivate vectors. Indeed, if the ground set then any subset may be described by the characteristicbinary vector. Let the binary vector that characterizes theprivate subset held by player .Then theunion of the private subsets is described by the OR of thoseprivate vector, Such a simple function can be evaluated securely by thegeneric solutions suggested in [3], [5], We present herea protocol for computing that function

International Journal of Scientific & Engineering Research Volume 6, Issue 5, May-2015
ISSN 2229-5518

which is much simplerto understand and program and much more efficient than thosegeneric solutions. It is also much simpler than Protocol UNIFIKCand employs less cryptographic primitives. Our protocol(Protocol 2) computes a wider range of functions, which wecall threshold functions.

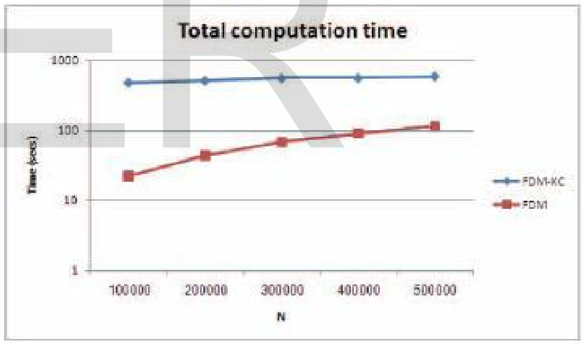
2.4 Privacy

We begin by analyzing the privacy offered by Protocol UNIFIKC.That protocol does not respect perfect privacy since itreveals to the players information that is not implied by their
own input and the final output. In Step 11 of Phase 1 of theprotocol, each player augments the set *Xm*by fake itemsets.To avoidunnecessary hash and encryption computations, thosefake itemsets are random strings in the ciphertext domainof the chosen commutative cipher. The probability of twoplayers selecting random strings that will become equal atthe end of Phase 1 is negligible; so is the probability ofPlayer *Pm* to select a random string that equals *EKm*(*h*(*x*))for a true itemsetx ∈Ap(F^{k-1}s). Hence, every encrypteditemset that appears in two different lists indicates with highprobability a true itemset that is locally *s*-frequent in bothof the corresponding sites. Therefore, Protocol UNIFI-KCreveals the following excess information:
(1) *P*1 may deduce for any subset of the odd players, thenumber of itemsets that are locally supported by all ofthem.
(2) *P*2 may deduce for any subset of the even players, thenumber of itemsets that are locally supported by all ofthem.
(3) *P*1 may deduce the number of itemsets that are supportedby at least one odd player and at least one even player.
(4) If *P*1 and *P*2 collude, they reveal for any subset of theplayers the number of itemsets that are locally supportedby all of them.
As for the privacy offered by Protocol UNIFI, we considertwo cases: If there are no collusions, then, by Theorem 2.3,Protocol UNIFI offers perfect privacy with respect to allplayers
Pm, *m* ≠ 2, and computational privacy with respectto *P*2. This is a privacy guarantee better than that offeredby Protocol UNIFI-KC, since the latter protocol does revealinformation to *P*1 and *P*2 even if they do not collude with anyother player.
If there are collusions, both Protocols UNIFI-KC andUNIFI allow the colluding parties to learn forbidden information. In both cases, the number of “suspects” is small —in Protocol UNIFI-KC only *P*1 and *P*2 may benefit from a collusion while in Protocol UNIFI only *P*1, *P*2 and *PM*can extract additional information if two of them collude (see Theorem 2.3). In Protocol UNIFI-KC, the excess informationwhich may be extracted by *P*1 and *P*2 is about the number ofcommon frequent itemsets among any subset of the players.

Namely, they may learn that, say, *P*2 and *P*3 have manyitemsets that are frequent in both of their databases (but notwhich itemsets), while *P*2 and *P*4 have very few itemsetsthat are frequent in their corresponding databases. The excessinformation in Protocol UNIFI is different: If any two outof *P*1, *P*2 and *PM* collude, they can learn the sum of allprivate vectors. That sum reveals for each specific itemsetin Ap(F^{k-1}s) the number of sites in which it is frequent,but not which sites. Hence, while the colluding players inProtocol UNIFI-KC can distinguish between the differentplayers and learn about the similarity or dissimilarity betweenthem, Protocol UNIFI leaves the partial databases totallyindistinguishable, as the excess information that it leaks iswith regard to the itemsets only.To summarize, given that Protocol UNIFI reveals no excessinformation when there are no collusions, and, in addition,when there are collusions, the excess information still leavesthe partial databases indistinguishable, it offers enhanced privacypreservation in comparison to Protocol UNIFI-KC.

3. DISCUSSION

Here in figure 1 two graphs shows the number oftransaction N has little effect on the runtime of FDM-KC.The second set of graph in figure 2 illustrates how theCommunicate



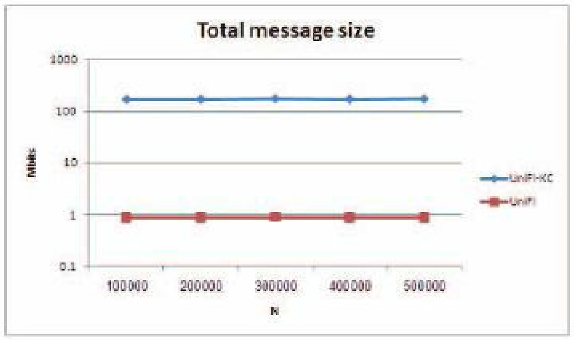
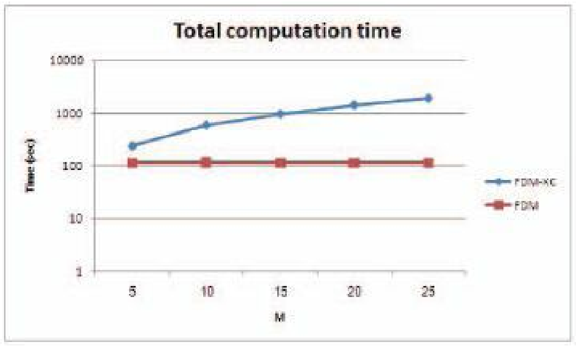


Figure 1: Computation and communication costs versusthe number of transactions NComparatively study of FDM and FDM-KC is shown inthe figure 1 and figure 2.



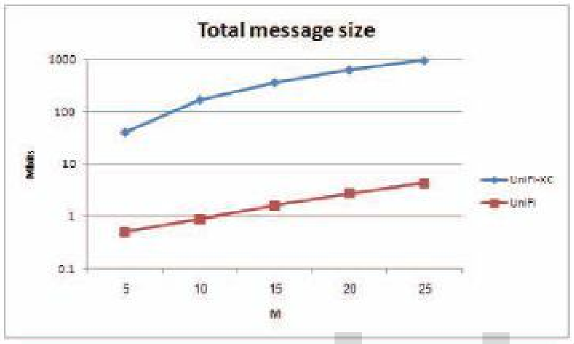


Figure 2: Computation and communication costs versusthe number of players *M*

4. CONCLUSION

In this project we devise a protocol for secure mining ofassociation rules in horizontally partitioned distributeddatabases. The protocol is more efficient than current leading K and C protocol. The main ingredients of this protocol are two novel secure multiparty algorithms in which these two main operations are union and intersection. The protocol exploits the fact that the underlying problem is of interest only if the number ofplayer is more than two. The direction to future work is to devise an efficient protocol for inequality verifications that uses theexistenceof semi-honest third party and another in Implementation of the techniques to the problem of distributed association rule mining in vertical setting.

REFERENCES

[1] Tamirtassa, "Secure Mining of Association Rules inHorizontally Distributed Databases", IEEEtransactions on knowledge and data engineering,
2013.

[2] J. Vaidya and C. Clifton, "Privacy preservingassociation rule mining in vertically partitioneddata," in *The Eighth ACM SIGKDD InternationalConference on Knowledge Discovery and DataMining*, Edmonton, Alberta, Canada, July 23-26
2002, pp. 639–644.

[3] M.Kantarcioglu and C. Cl` ifton., "Privacy-preservingdistributed mining of association rules onhorizontally partitioned data", *IEEE Transactions onKnowledge and Data Engineering*, 16:1026–1037,2004.

[4] R.Agrawal and R. Srikant.,"Privacy-preserving datamining", *SIGMOD Conference*, pages 439–450,2000.

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

[6] M. Kantarcioglu, R. Nix, and J. Vaidya,"An efficientapproximate protocol for privacy-preservingassociation rule mining", In *PAKDD*, pages 515–524, 2009.

[7] M. Freedman, Y. Ishai, B. Pinkas, and O. Reingold.,"Keyword search and oblivious pseudorandomfunctions", In *TCC*, pages 303–324, 2005.

[8] T. Tassa and E. Gudes. Secure distributed computation of anonymized views of shared databases. *Transactions on Database Systems*, 37, Article 11, 2012.

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