An Efficient Cryptographic Privacy Preserving Algorithm for Association Rule Mining over Heterogeneous Database

Mahmoud Hussein, Ashraf El-Sisi and Nabil Ismail
Computer Science Department, Faculty of computers and Information, Menofyia University, Shebin Elkom 32511, Egypt,
ashrafelsisi@hotmail.com, fci_3mh@yahoo.com

Abstract- Recently, there are many privacy and security issues in data mining. A considerable research has focused on developing new data mining algorithms that incorporate privacy constraints. Their is a conflict between privacy and data mining. As most types of data mining produce summary results that do not reveal information about individuals. But the process of data mining may use private data, leading to the potential for privacy breaches. In this paper, we focus on privately mining association rules in vertically partitioned data. We present an efficient algorithm for this problem based on cryptographic. The proposed algorithm not only secure as other protocols for the same problem, but also much faster.

I. INTRODUCTION

Privacy preserving data mining has long been an area of active research and one of the important properties needed in any information system. So far, if we assumed that the information in each database used in mining can be freely shared. However, there is a need for computing association rules across these databases belonging to different sites in such a way that no more information than necessary is revealed from each database to the other databases. Only every site knows its input and final mining results. This need is driven by several trends like security of government agencies that need to share information within the same government and across governments. However, an agency cannot indiscriminately open up its database to all other agencies. Because privacy, privacy legislation and stated privacy policies places limits on information sharing. In the same time, it is still desirable to mine across databases while respecting privacy limits. There are many methods for privacy preserving distributed association rule mining across private databases. The method used is based on the design of the data bases used in mining. There are two main methods for partitioning data across many parties, one of them is horizontally partitioned data base and the other is vertically partitioned data base. So if we considered the data base partition horizontally then the problem of preserving the privacy is reduced to computing the union of frequent itemsets between mining parties and this problem solved as in [1], [2] and [16].

In the other hand, if we consider the data base partitioned vertically then the problem is reduced to compute scalar product between these parties as in [3-8]. Another view of the same problem is to reduce it to compute the set intersection cardinality as in [9]. So these methods try to compute the answer to the mining without revealing any additional information about user privacy. Any one can use set-intersection protocols for online recommendation services, online dating services, medical databases, and many other applications and also for mining over vertical data bases.

There are some existing techniques that one might use for solving this problem but they have some problem like increasing the running time of computing the set intersection cardinality [9]. So, we propose a fast cryptographic technique for computing set intersection cardinality and also preserve the privacy using cryptographic techniques and matrix operations. The organization of this paper is as follows. Section 2 gives an overview about the problem and related work in the area of privacy preserving data mining for association rule mining on distributed heterogeneous (vertically partitioned) databases. In section 3 the details of modified algorithm of computing the set intersection cardinality, security analysis and evaluation metrics are presented. Section 4 describes the implementation and results of modified an algorithm verse the old algorithm. Finally, some conclusions are put forward in Section 5.

II. DISTRIBUTED ASSOCIATION RULE MINING AND PROBLEM DEFINITION

Association rule mining is one of the most important data mining techniques used in many real life applications. It is used to reveal unexpected relationships in the data. By assuming heterogeneous databases: each site has different schema. The goal is to produce association rules that hold globally, while limiting the information shared about each site to preserve the privacy of data in each site.

Let there be k parties $S_1, S_2, \ldots, S_k$. The association rule mining problem can be formally stated as follows: Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of literals, called items. Let $D$ be a set of transactions, where each transaction $T$ is a set of items such that $T \subseteq I$. Associated with each transaction is a unique identifier, called its TID. We say that a transaction $T$ contains $X$, a set of some items in $I$, if $X \subseteq T$. An association rule is an implication of the form $X \Rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$, and $X \cap Y = \emptyset$. The rule $X \Rightarrow Y$ holds in the transaction set $D$ with confidence $c$ if $c\%$ of transactions in $D$ that contain $X$ also contain $Y$. The rule $X \Rightarrow Y$ has support $s$ in $D$ if $s\%$ of the transactions in $D$ contain $X \cup Y$.

The association rule mining problem can be decomposed into two distinct sub problems first generate all combinations of items that have support at least minimum support then for every frequent found in the first step, generate all rules from it with minimum confidence. Most of the work done so far has focused on the first problem since generating the corresponding association rules from the frequent itemsets is a not difficult task. The most known algorithm for mining frequent itemsets is the Apriori algorithm [10]. With this algorithm, the set of transactions is viewed as a database $D$ with $n$ rows and $m$ columns, every row corresponding to a transaction and every column corresponding to an item. Each entry in the database is 0 or 1, specifying the absence or
presence of items in the set of transactions. In other words, if the i-th row in the database corresponds to transaction $t_i$ and the j-th column corresponds to item $I_j$, then the j-th entry in row i (denoted by $t_i[j]$) indicates whether or not $t_i$ contains $I_j$.

Because when studying privacy-preserving data mining it is useful to consider how data may be partitioned among the involved parties. In some cases, organizations may collect the same kind of data about different entities (for example people, traffic, etc.). From a database perspective, we may then say that the data is partitioned horizontally; that is, the same schema is used to store the data at each site. In other cases, organizations may organize data using different schemas, meaning that they collect different kinds of data, perhaps on the same entities. We then say that the data is partitioned vertically.

In this case, the database is partitioned vertically into two sets of columns, the first set (denoted by DB1) consisting of the first a columns (more precisely, the columns corresponding to items $(I_1, I_2, \ldots, I_a)$, and the second one (denoted by DB2) consisting of the remaining m-a columns (i.e., the columns corresponding to $(I_{a+1}, I_{a+2}, \ldots, I_m)$). Also, consider that we have two parties A and B, such that DB1 belongs to party A and DB2 belongs to party B. The two parties want to collaboratively find the frequent itemsets in $DB = DB1 \cup DB2$. Assume that the two parties want to find out if the itemset $(I_{a1}, I_{a2}, \ldots, I_{ap}, I_{b1}, I_{b2}, \ldots, I_{bq})$ is frequent, where $(I_{a1}, I_{a2}, \ldots, I_{ap}) \subseteq \{I_1, I_2, \ldots, I_a\}$ and $(I_{b1}, I_{b2}, \ldots, I_{bq}) \subseteq \{I_{a+1}, I_{a+2}, \ldots, I_m\}$. And suppose the data base is presented using the transaction identifier list approach not as m x n Boolean matrix [11]. In the following example we will explain this method. Consider the data base have five transaction TID1,TID2,TID3,TID4,TID5, and with two attributes X1 and X2. Figure 1 shows Boolean matrix view and TID-list view representation of data base.

<table>
<thead>
<tr>
<th>Boolean matrix view</th>
<th>TID-list view</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>TID1,TID2,TID3</td>
</tr>
<tr>
<td>X2</td>
<td>X2,TID1,TID2,TID4,TID5</td>
</tr>
<tr>
<td>TID1</td>
<td>1 1</td>
</tr>
<tr>
<td>TID2</td>
<td>1 1</td>
</tr>
<tr>
<td>TID3</td>
<td>0 0</td>
</tr>
<tr>
<td>TID4</td>
<td>1 1</td>
</tr>
<tr>
<td>TID5</td>
<td>0 1</td>
</tr>
</tbody>
</table>

**Figure 1.** Example data base in various representation

It is clear that at transaction supports an item set (set of attributes) if and only if its TID is listed in TID-list for all attributes in the itemset. So to known the number of transaction that support given itemset we need to compute the cardinality of the intersection set of the TID-lists of these attribute. In figure 2 an algorithm is given to collaboratively find the frequent itemsets in $DB = DB1 \cup DB2$ when the data base is in TID-list view.

**Input:** DB1 and DB2  
**Output:** the frequent itemsets in $DB = DB1 \cup DB2$  
$L_A$=[large l-items], $K=2$

**While** $L_{k-1} \neq \emptyset$ **do**

\[ C_j = \text{Apriori-gen}(L_{k-1}) \]

**For all candidates** $c \in C_j$ **do**

Let $A$ have items $I_{a1}, I_{a2}, \ldots, I_{ap}$ and $B$ have the remaining items $I_{b1}, I_{b2}, \ldots, I_{bq}$. And construct $S1$ at $A$ where $S1$ is TID-list of transaction contains $(I_1, I_2, \ldots, I_a)$ and construct $S2$ at $A$ where $S2$ is TID-list of transaction contains $(I_{a+1}, I_{a+2}, \ldots, I_m)$.

**If** all the items in $c$ are entirely at $A$ or $B$ **Then**

that party independently computes $c$.count

**Else**

$c$.count = $|S1 \cap S2|

**End if**

**End for**

$L_k = L_{k-1} \cup \{ c | c$.count $\geq$ minimum support $\}, K= k + 1$

**End while**

**Return** $L_1 \cup \ldots \cup L_{k-2}$

**Figure 2.** Apriori Algorithm for Vertically Partitioned Data

The set intersection cardinality between $S1$ and $S2$ defined as the number of TID elements that found in $S1$ and also in $S2$. Determining if the itemset $(I_{a1}, I_{a2}, \ldots, I_{ap}, I_{b1}, I_{b2}, \ldots, I_{bq})$ is frequent reduces to testing if $|S1 \cap S2| \geq$ minimum support. Thus, the two parties want to compute the set intersection cardinality $|S1 \cap S2|$ without any party revealing its own vector. This idea can be easily incorporated into the Apriori algorithm, as shown in figure 2, this can be easily extended to more than two parties.

**A. Related Work**

Within the context of privacy-preserving data mining for vertically partitioned database[12], several protocols have been proposed to preserve the privacy. The goal is that one of the participants obtains the scalar product or the set intersection cardinality of the private vectors of all parties. Additionally, it is often required that no information about the private vectors, except what can be deduced from the cardinality product or the set intersection cardinality, will be revealed during the protocol. Moreover, since data mining applications work with a huge amount of data, it is desirable that the scalar product protocol is also very efficient. A secure scalar product protocol has various applications in privacy preserving data mining, starting with privacy-preserving frequent pattern mining on vertically distributed database and ending with privacy-preserving cooperative statistical analysis.

In [9] proposed an algorithm that makes privacy preserving association rule mining over vertical data bases using cryptography. The main idea is to compute set intersection cardinality securely cryptography to make strong privacy preserving protocol, results showed that this protocol is efficient in preserving the privacy but it has a high computation cost.

The key insight behind this protocol is to overlap the hashing and intersection phases. The protocol simply reduces the problem of intersecting $S0 \cap S1 \cap \ldots \cap Sk$ to be $(S0 \cap S1) \cap (S2 \cap S3) \cap \ldots \cap (Sk-1 \cap Sk)$. So it can be represent by a binary tree. We can reformulating the intersection as $(\ldots ((S0 \cap S1) \cap (S2 \cap S3)) \cap \ldots \cap (Sk-1 \cap Sk)) \ldots \cap k \ldots$ . The difficulty is with carrying out intersections of two sets. The solution requiring at least three parties to perform the intersection. The solution is to use a
party from the opposite side of the tree as this third party. Each party hashes its set and sends it to its “intersection partner”. The partner hashes it, and both send them to the parent third party. The third party performs the intersection. An example of this is given in figure 3.

Figure 3. Intersection between two sites

The protocol uses a commutative public key encryption scheme such as Pohlig-Hellman can be used to generate a hash function satisfying all requirements needed in [13]. Each party generates its own key pair (Ek, Dk). The length in bits for the key pair is commonly agreed upon and known to all parties. The hash function h is simply encryption with Ek. The decryption keys are not needed.

In the same area of privacy preserving association rule mining over vertical data base but using scalar product to compete the mining process not set intersection cardinality. There is a lot of work done like in [8] describe a provably private scalar product protocol that is based on homomorphic encryption and improve its efficiency so that it can also be used on massive datasets. And other like [3-7].

III. PROPOSED NEW ALGORITHM AND ILLUSTRATIVE EXAMPLE

In [9] due to employing commutative encryption algorithm, adds large overhead to compute the set intersection cardinality between the sites. We improve that by instead of use binary tree to compute the set intersection cardinality we use circle arrangement, replace the communicative encryption by applying a public-key cryptosystem and all encryption operations of all sites in the same time. Suppose we have K site so the arrangement will be as shown in figure 4 for the protocol in [9]. The new arrangement of sites used for making intersection between all sites as shown in figure 5, this new arrangement according to the description of our new algorithm shown in figure 6. The improvement is in the using of cryptography with circle arrangement will reduce the number of encryption operations needed to compute the set intersection cardinality. In our case we use one site as protocol initiator that have public key (k1) and private key (k2) and other sites have the public key only, each sites encrypt it’s data and send it to next site until reach the initiator, so this reduce the time required for encryption. We assume negligible collision probability.

Protocol: protocol for computing size of intersection between K sites
Require: k > 3 sites numbered i: 0 . . . k, each having a local Boolean vector Xi, all Xi elements is 0 or 1, the initiator site generate the keys K1 and K2 and send K1 to all sites.
Step1: encryption by all sites with Padding
for each site i do
for each X E LV do // LV is local vector data
LVei = LV ei U {permute (E(X),E(Y))} // Y is value greater than 1
end for
end for
Step 2: exchange encrypted data between sites
construct M that every vector in M is local vector of one site and put first vector of site 1 in M
send M to site 2
for each site i do // i: 2…….k
receive M from site i-1
permute M with LVei and the permutation is in rows only
send M to site i+1
end for
Step 3: decrypt received data and make intersection
decrypt M
for all values in M if the value is greater than 1 then replace it with 1
compute P = \sum \prod
Publish the set intersection cardinality as P to all sites

Figure 4. Arrangement of sites in old algorithm

Figure 5. Arrangement of sites in new algorithm

Figure 6. Proposed new algorithm
Step 2: Begin exchange the encrypted data. Site 1 send its
encrypted data to site 2 and when site 2 receive it, all data
merged and send to the next site in round until reaching site
k.

Step 3: When reaching site K all data received are
decrypted then the site perform the intersection computation
of all data.

A. Illustrative Example

We describe the new protocol by considering a running
example involving three parties. The extension to more than
three parties follows the same idea. Let P1, P2, P3 be three
parties that want to collaboratively compute the product LV 1
× LV 2 × LV3 to get the intersection between them, where

\[
\begin{bmatrix}
1 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 \\
\end{bmatrix}
\]

are the vectors associated with P1, P2, P3, respectively.
Each of the vectors has size n = 5. Let P1 be the initiator
of the protocol. The protocol initiator generate the public (k1)
and private (k2) keys. Initiator publish the public key to all
other parties.

Step 1. First, P1 forms an (n × x1) where x1 is random
number generated by P1 (suppose x1=3). Initially we insert
LV1 in matrix first column encrypted by public key. All other
matrix values are encrypted random values of numbers
greater than 1. Finally permute each row values independently of other rows. Then, P1 sends M to P2. The
following matrix after permutation:

\[
M_{5 \times 3} =
\begin{bmatrix}
\begin{bmatrix}
 e_{i1}(R1) & e_{i1}(R2) & e_{i1}(R3) & e_{i1}(R4) & e_{i1}(R5) & e_{i1}(R6) & e_{i1}(R7) \\
e_{i2}(R1) & e_{i2}(R2) & e_{i2}(R3) & e_{i2}(R4) & e_{i2}(R5) & e_{i2}(R6) & e_{i2}(R7) \\
e_{i3}(R1) & e_{i3}(R2) & e_{i3}(R3) & e_{i3}(R4) & e_{i3}(R5) & e_{i3}(R6) & e_{i3}(R7) \\
e_{i4}(R1) & e_{i4}(R2) & e_{i4}(R3) & e_{i4}(R4) & e_{i4}(R5) & e_{i4}(R6) & e_{i4}(R7) \\
e_{i5}(R1) & e_{i5}(R2) & e_{i5}(R3) & e_{i5}(R4) & e_{i5}(R5) & e_{i5}(R6) & e_{i5}(R7) \\
\end{bmatrix}
\end{bmatrix}
\]

Step 2. Upon receiving M from P1, P2 forms an (n ×
(x1+x2)) matrix where M1[I][J] = M[I][C] for all I, C
and x2 is random number (suppose x2=3). Then insert LV2 in matrix
x2+1 column encrypted by public key. All other matrix
values are encrypted random values of numbers greater than 1. Finally permute each row values independently of other rows. Then, P2 sends M1 to P3. And when received at P3 make the same thing as in P3 and suppose x3=2. In matrices, the random numbers are donated by R followed by number without any meaning. It is only different numbers. The following two matrices after permutation:

At site 2:

\[
M_{5 \times 5} =
\begin{bmatrix}
\begin{bmatrix}
 e_{i1}(R1) & e_{i1}(R2) & e_{i1}(R3) & e_{i1}(R4) & e_{i1}(R5) & e_{i1}(R6) & e_{i1}(R7) \\
e_{i2}(R1) & e_{i2}(R2) & e_{i2}(R3) & e_{i2}(R4) & e_{i2}(R5) & e_{i2}(R6) & e_{i2}(R7) \\
e_{i3}(R1) & e_{i3}(R2) & e_{i3}(R3) & e_{i3}(R4) & e_{i3}(R5) & e_{i3}(R6) & e_{i3}(R7) \\
e_{i4}(R1) & e_{i4}(R2) & e_{i4}(R3) & e_{i4}(R4) & e_{i4}(R5) & e_{i4}(R6) & e_{i4}(R7) \\
e_{i5}(R1) & e_{i5}(R2) & e_{i5}(R3) & e_{i5}(R4) & e_{i5}(R5) & e_{i5}(R6) & e_{i5}(R7) \\
\end{bmatrix}
\end{bmatrix}
\]

At site 3:

\[
\begin{bmatrix}
\begin{bmatrix}
 e_{i1}(R3) & e_{i1}(R4) & e_{i1}(R5) & e_{i1}(R6) & e_{i1}(R7) \\
e_{i2}(R3) & e_{i2}(R4) & e_{i2}(R5) & e_{i2}(R6) & e_{i2}(R7) \\
e_{i3}(R3) & e_{i3}(R4) & e_{i3}(R5) & e_{i3}(R6) & e_{i3}(R7) \\
e_{i4}(R3) & e_{i4}(R4) & e_{i4}(R5) & e_{i4}(R6) & e_{i4}(R7) \\
e_{i5}(R3) & e_{i5}(R4) & e_{i5}(R5) & e_{i5}(R6) & e_{i5}(R7) \\
\end{bmatrix}
\end{bmatrix}
\]

Finally send M2 to P1.

Step 3. Decrypt all values in M2 then replace all values
greater than 1 by 1 and construct vector P
as: P[i]=M[i][0]*M[i][1]*…….*M[i][j]
//i =1,2,……,n and j=1,2,3,……,(x1+x2+x3)

At site1 after decrypting all values:

\[
M_{5 \times 5} =
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\end{bmatrix}
\]

Then publish the product P = \sum_P = 2. Then the
intersection cardinality between all vectors is 2.

P_{5 \times 1} =
\begin{bmatrix}
1 \\
0 \\
1 \\
0 \\
\end{bmatrix}

B. Security Analysis and evaluation metrics

It is important to proof that the new protocol is preserving
the privacy. Also performance measure is another method to
evaluate our protocol. So in this section we will proof that our
algorithm preserve the privacy and determine the
performance metrics we used.

Security Analysis

Theorem 1: Our algorithm privately computes the set
intersection cardinality between any number of vectors
present in the database without revealing any private
information about users of data base.

Proof. To show that our algorithm preserves the privacy
can be done by using the idea of simulating every thing
during the protocol running to know what data every site see
in running the protocol [14]. The proof as following:

In step 1: Every site encrypt the LV with Padding values
other than (0 and 1) there is no communication between sites
then no privacy loss.

In step 2: Every site send the encrypted LV with Padding
values to next site then permute this values and this operation
continue until reaching last site so the results of any
computation are encrypted and next site don't have the private
key then no privacy loss.

In step 3: Finally the initiator decrypt the received data and
because the data are padded and permuted so the initiator
can't know any information about any site then compute the
final intersection and publish it. Any site can't deduce from it
any information about others. So this protocol can't loss the
privacy in the semi honest model and for malicious model.
The collision found if the initiator and site 2 collude with
each other. Because we choose the initiator every time running the protocol, so we have negligible collision in our protocol.

For our proposed algorithm may be some one can say that, first party can know everything. But, in our proposed algorithm when we begin computing the intersection, we construct matrix with one column which is the input of this party (encrypted) and pad the matrix with other column. So the matrix has other column not only the input of this party. As the process of exchanging the matrix begin every site will add his input vector (encrypted) and other padding columns (encrypted) and then permute the values in rows, this process continue until reaching the last party which it is the first party. Now the last matrix has columns, some of it is the padding values and other is the vectors we need to compute it's intersection. But because we make permutation, so no column is corresponding to the vector of any site, from this we can't know what column related to which party. And also because our algorithm move the data encrypted between the sites no party know any information, because no party know the private key other than first party.

**Evaluation metrics**

To measure the performance of our method we use computation costs as performance metrics. Cost estimation for association rule mining using the method we have presented can be computed as following: The number of sites is K. Let the total number of locally vectors LV. Let t be the number of bits in the output of the encryption of an itemset. Based on current encryption standards t = 512 is a more appropriate value. Suppose $M = t \times LV$ is the encrypted message of any site. Then total computation cost for our protocol is $O(K \times M)$ and the algorithm in [9] need $O(K \times \log K \times M)$ where M in our algorithm is close to M in algorithm proposed in [9].

<table>
<thead>
<tr>
<th>K</th>
<th>N</th>
<th>algorithm in [9]</th>
<th>new algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1000</td>
<td>1.620293</td>
<td>0.30836</td>
</tr>
<tr>
<td>5</td>
<td>5000</td>
<td>6.620657</td>
<td>1.38222</td>
</tr>
<tr>
<td>5</td>
<td>10000</td>
<td>14.20426</td>
<td>2.568691</td>
</tr>
<tr>
<td>5</td>
<td>20000</td>
<td>34.22353</td>
<td>5.258866</td>
</tr>
<tr>
<td>5</td>
<td>50000</td>
<td>124.3171</td>
<td>12.49961</td>
</tr>
<tr>
<td>10</td>
<td>1000</td>
<td>7.354613</td>
<td>0.511975</td>
</tr>
<tr>
<td>10</td>
<td>5000</td>
<td>38.77106</td>
<td>2.522523</td>
</tr>
<tr>
<td>10</td>
<td>10000</td>
<td>74.26619</td>
<td>5.058596</td>
</tr>
<tr>
<td>10</td>
<td>20000</td>
<td>157.3343</td>
<td>8.747175</td>
</tr>
<tr>
<td>10</td>
<td>50000</td>
<td>412.8676</td>
<td>21.43766</td>
</tr>
</tbody>
</table>

**Table. 1 Comparison execution time of new algorithm verse old algorithm in seconds**

<table>
<thead>
<tr>
<th>encryption algorithm</th>
<th>RSA</th>
<th>Pohlig-Hellman</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>N</td>
<td>algorithm in [9]</td>
</tr>
<tr>
<td>5</td>
<td>1000</td>
<td>527.2839</td>
</tr>
<tr>
<td>5</td>
<td>5000</td>
<td>2452.084</td>
</tr>
<tr>
<td>5</td>
<td>10000</td>
<td>4445.134</td>
</tr>
<tr>
<td>5</td>
<td>20000</td>
<td>9179.806</td>
</tr>
<tr>
<td>5</td>
<td>50000</td>
<td>1321.384</td>
</tr>
<tr>
<td>10</td>
<td>1000</td>
<td>5487.498</td>
</tr>
<tr>
<td>10</td>
<td>5000</td>
<td>11338.57</td>
</tr>
<tr>
<td>10</td>
<td>10000</td>
<td>20432.51</td>
</tr>
<tr>
<td>10</td>
<td>50000</td>
<td>46936.54</td>
</tr>
</tbody>
</table>

**Fig. 6.** Percentage time of new algorithm related to old algorithm when $K=5$ and $K=10$ with RSA

**Fig. 7.** Percentage time of new algorithm related to old algorithm when $K=5$ and $K=10$ with Pohlig-Hellman

From the results we can find that new algorithm has a better performance in computations time than the algorithm in [9]. This due to the computation cost for our algorithm is function in (K) site, but the algorithm in [9] is function in (K*log K) sites. In the same time based on proofing for privacy
preserving our algorithm preserve the privacy also. New algorithm is more flexible to extend it to any number of sites without any change in arrangement of sites because it is as circle not a tree. And also any increase doesn’t add more time to algorithm because all client sites perform the encryption of local vector in the same time so the overhead in computation for new site only.

V. CONCLUSION AND FURTHER WORK

In this paper we presented privacy preserving association rule mining algorithms of have been recently introduced with the aim of preventing the discovery of sensible information. We modify an algorithm of privacy preserving association rule mining on distributed heterogeneous data by optimize the computation required for all sites, modify the algorithm and how compute the distributed association rule mining. This modification for the algorithm of computing distributed association rule mining to preserve the privacy of users. An implementation for modified algorithm is presented. From the results obtained we can say that our algorithm is good privacy preserving algorithm with better performance. And also any increase doesn’t add more time to algorithm because all client sites perform the encryption of local vector in the same time. So the overhead in computation for new site only. The total computation cost for our algorithm is function in (K) sites. Our protocol is base in public key cryptography so we use Pohlig-Hellman and RSA as example for testing of our algorithm. For future work we can replace it with other public key cryptography.

REFERENCES


