Hiding Sensitive Frequent Itemsets over Privacy Preserving Distributed Data Mining

Alaa Kh. Juma'a
Computer Science Institute
University of Polytechnic /Sulaimaniya, Iraq

Sufyan T. F. Al-Janabi
College of Computer
University of Anbar, Anbar, Iraq

Nazar A. Ali
College of Administration
University of Sulaimaniya, Iraq

Received on: 08/10/2012 **Accepted on: 30/01/2013**

ABSTRACT

Data mining is the process of extracting hidden patterns from data. One of the most important activities in data mining is the association rule mining and the new head for data mining research area is privacy of mining. Privacy preserving data mining is a new research trend in privacy data for data mining and statistical database. Data mining can be applied on centered or distributed databases. Most efficient approaches for mining distributed databases suppose that all of the data at each site can be shared. Privacy concerns may prevent the sites from directly sharing the data, and some types of information about the data. Privacy Preserving Data Mining (PPDM) has become increasingly popular because it allows sharing of privacy sensitive data for analysis purposes.

In this paper, the problem of privacy preserving association rule mining in horizontally distributed database is addressed by proposing a system to compute a global frequent itemsets or association rules from different sites without disclosing individual transactions. Indeed, a new algorithm is proposed to hide sensitive frequent itemsets or sensitive association rules from the global *frequent itemsets* by hiding them from each site individually. This can be done by modifying the original database for each site in order to decrease the support for each sensitive itemset or association rule. Experimental results show that the proposed algorithm hides rules in a distributed system with the good execution time, and with limited side effects. Also, the proposed system has the capability to calculate the global frequent itemsets from different sites and preserves the privacy for each site.

Keywords*-cryptography; data mining; distributed database; frequent itemsets; sensitive association rules.*

إخفاء العناصر المتكررة الحساسة من خلال التنقيب المحافظ على خصوصية البيانات الموزعة

علاء جمعة
معهد علوم الحاسوب
الجامعة التقنية
السليمانية، العراق

سفيان الجنابي
كلية الحاسوب
جامعة الانبار، الانبار، العراق

نزار علي
كلية الإدارة
جامعة السليمانية، السليمانية، العراق

تاريخ استلام البحث: 2012/10/08 **تاريخ قبول البحث: 2013/01/30**

الملخص

إن عملية تنقيب البيانات هي عبارة عن استخلاص الأنماط المخفية من البيانات. وإن التنقيب عن العلاقات الرابطة يعد واحدا من أهم فعاليات تنقيب البيانات والتي أصبح التوجه الحديث للباحثين فيها هو الحفاظ على سرية تلك البيانات المنقب عنها. فالتنقيب المحافظ على خصوصية البيانات هو من أهم توجهات البحوث العلمية الجديدة في خصوصية البيانات وقواعد البيانات الإحصائية. ويمكن تطبيق فعاليات التنقيب هذه على قواعد

البيانات المركزية والموزعة. ورغم أن أكثر الأساليب فعالية لقواعد البيانات الموزعة تفترض التنقيب في البيانات التي يمكن تشاركها بين المواقع المختلفة، غير أن تلك الأساليب لم يعد بالإمكان تطبيقها في كثير من الأحيان بسبب المخاوف المتعلقة بخصوصية منع المواقع من تبادل البيانات بشكل مباشر، أو تبادل بعض أنواع المعلومات حول البيانات. لذلك أصبح للتنقيب المحافظ على خصوصية البيانات (PPDM) شعبية متزايدة لأنه يسمح بتبادل البيانات الحساسة الخصوصية لأغراض التحليل.

في هذا البحث، تم التصدي لمشكلة الحفاظ على خصوصية التنقيب للعلاقات الرابطة في قاعدة بيانات موزعة أفقيا من خلال اقتراح نظام لحساب العناصر (itemsets) العامة المتكررة أو العلاقات الرابطة من مواقع مختلفة دون الكشف عن المعاملات الفردية. كما نقدم هنا أيضا خوارزمية جديدة لإخفاء هذه العناصر المتكررة الحساسة أو قواعد الرابطة الحساسة من خلال إخفائها في كل موقع على حدة. ويمكن أن يتم ذلك عن طريق تعديل قاعدة البيانات الأصلية لكل موقع من أجل خفض الدعم لكل من العناصر الحساسة أو العلاقات الرابطة. النتائج التي تم الحصول عليها من تطبيق الخوارزمية المفترضة تشير إلى قدرتها على إخفاء العناصر المتكررة الحساسة بوقت تنفيذ جيد وبأقل تأثيرات جانبية. كما أن النظام المفترض استطاع الحصول على العناصر المتكررة العامة للبيانات (Global Frequent Itemset) الموزعة على عدة مواقع مع الحفاظ على خصوصية كل موقع.

الكلمات المفتاحية: التشفير، تنقيب البيانات، البيانات الموزعة، العناصر المكررة، العلاقات الرابطة الحساسة.

I. Introduction

Privacy preserving data mining is a new research area that investigates the side-effects of data mining methods that originate from the penetration into the privacy of individuals and organizations. Most of information systems contain private information, such as social security numbers, income, disease type, etc. Therefore, this information should be correctly protected and hidden from unauthorized access. Although, the security of data has been permanent goal in database management systems, mining of knowledge and preventing of sensitive knowledge disclosure become the most important and highest priority goal in data mining process. Basically, the sharing of data between businesses in purpose of reaching valuable information is useful, but it can bring a lot of disadvantages [1].

Recent advances in data mining algorithms increased the risk of information leakage and its confidence issue. Because of this progress, the parallel research area has been started to overcome the information leakage risks and immunization of mining environment. Privacy preserving against mining algorithms is a new research area that investigates the side-effects of data mining methods that can be derived from the privacy diffusion of persons and organizations [9].

Two problems are addressed in (PPDM); one is the protection of private data; another is the protection of sensitive rules (knowledge) contained in the data. The former settles how to get normal mining results when private data cannot be accessed accurately; the latter settles how to protect sensitive rules contained in the data from being discovered, while non-sensitive rules can still be mined normally. The latter problem is called knowledge hiding in database in (KHD) which is opposite to knowledge discovery in database (KDD) [7].

Recent studies in preserving association rule privacy have proposed many techniques like k-Anonymity methods, randomization methods, and cryptographic-based PPDM that includes Secure Multiparty Computation (SMC), homomorphic encryption, and other cryptographic techniques [2, 10 and 11].

Also, a number of techniques like perturbation and anonymization have been developed to hide association rules from being discovered from published data. In practically for a single data set, given specific rules or patterns to be hidden, many data altering techniques for hiding association rules have been proposed. They can be categorized into three basic approaches. The first approach hides one rule at a time. It first selects transactions that contain the items in a given rule. It then tries to modify items, transaction by transaction, until the confidence or support of the rule falls below minimum confidence or minimum support. The modification is done by either removing items from the transaction or inserting new items to the transactions. The second approach deals with groups of restricted patterns or sensitive association rules at a time. It first selects the transactions that contain the intersecting patterns of a group of restricted patterns. Depending on the disclosure threshold given by users, it sanitizes a percentage of the selected transactions in order to hide the restricted patterns. The third approach deals with hiding certain constrained classes of association rules [5].

C.-C. Weng et al. presented algorithm to hide frequent sensitive rule by evaluating the weight of each transaction that supports these rules and hiding these rules according to this weigh [14]. V. S. Verykios et al. presented algorithms to hide sensitive association rules, but they generated high side effects and required multiple database scans [12]. S.-L. Wang proposed an algorithm to hide sensitive items. The algorithm needs less number of database scans, but the side effects generated were higher [13].

This paper deals with the Distributed Data Mining (DDM), or more specifically, with (PPDDM). This work proposes a two-phase PPDDM system. The first phase is dedicated for privacy-preserving distributed mining by encrypting local association rule mining in each site with commutative algorithm and sending the results to all sites. In the second phase, a new algorithm is proposed to hide sensitive frequent itemsets from global support items by hiding these frequent items in each site according to the proposed algorithm. Each site will modify its original database in order to prevent any external miner from detecting any sensitive frequent itemsets. The remaining of this paper is organized as follows: Section 2 briefly discusses classes of association rule algorithms. Section 3 presents the problem description. Next, Section 4 introduces the proposed approaches and algorithm. Then, Section 5; discusses the results and performance evaluation. Finally, the paper is concluded in Section 6.

II. Classes of Association Rule Algorithms

The various approaches proposed by researchers hide sensitive information efficiently and accurately but also face the problem of side effects. The side effects occur due to correlations existing between items in the database. Side effects may decrease the informational accuracy to the users. Due to the property of correlation, association rules may possess spurious or wrong information, hide non sensitive rules unnecessarily, and accidentally disclose some sensitive rules. So the challenging task is how to protect sensitive rules from users without effecting informational accuracy to the users that is avoiding side effects as far as possible.

The algorithms which have been already developed for hiding association rules can be classified into three distinct classes, namely *heuristic* approaches, *border-based* approaches and *exact* approaches. The first class of approaches involves efficient and fast algorithms that selectively sanitize a set of transactions from the database to hide the sensitive knowledge [8]. Due to their efficiency and scalability, the heuristic approaches have been the focus of attention for the vast majority of researchers in the knowledge hiding field. However, there are several circumstances in which they suffer from undesirable side-effects that lead them to suboptimal solutions.

The second set of approaches considers the task of sensitive rule hiding through modification of the original borders in the lattice of the frequent and the infrequent patterns in the dataset. In these schemes, the sensitive knowledge is hidden by enforcing the revised borders (which accommodate the hiding of the sensitive itemsets) in the sanitized database. The algorithms in this class differ both in the borders that they track, use of the hiding strategy, and in the methodology that they follow to enforce the revised borders in the sanitized dataset. Finally, the third class of approaches contains non-heuristic algorithms which conceive the hiding process as a constraint satisfaction problem that they solve by using integer or linear programming. The main difference of these approaches, compared to the previous ones, is the fact that the sanitization process guarantees optimality in the hiding solution, provided that an optimal solution exists. On the other hand, these approaches are usually several orders of magnitude slower than the heuristic ones, especially due to the runtime of the integer/linear programming solver [1].

III. Problem Description

Association rule mining was first introduced by Agrawal, R. et al. Let $I=\{i1, i2, \ldots, im\}$ be a set of literals, called items. Given a set of transactions *D*, where each transaction *T* in *D* is a set of items such that $T \subseteq I$, an association rule is an expression $X \Rightarrow Y$ where $X \subseteq I, Y \subseteq I, and\ X \cap Y = \varphi$. As an example, for a given database in Table 1, for a minimum support of 33% and a minimum confidence of 70%, nine association rules can be found as follows: *B*=>*A* (66%, 100%), *C*=>*A* (66%, 100%), *B*=>*C* (50%, 75%), *C*=>*B* (50%, 75%), *AB*=>*C* (50%, 75%), *AC*=>*B* (50%, 75%), *BC*=>*A*(50%, 100%), *C*=>*AB*(50%, 75%), *B*=>*AC*(50%, 75%), where the percentages inside the parentheses are supports and confidences respectively [3].

Distributed system assumed that that there are *n* sites S_0, S_1, …, S_{n-1} and the transaction database *DB* is horizontally divided into *n* non-overlapping partitions *db*0, db_1, …, db_{n-1}, where $DB = db_0 \cup db_1 \cup \ldots \cup db_{n-1}, db_i \cap db_j = \varphi$, $0 \le i \ne j \le n-1$. Each partition db_i is assigned to site S_i, and *DB* is horizontally distributed. Clearly, $|DB| = |db_0| + |db_1| + \ldots + |db_{n-1}|$. $X.sup_i$ is the local support counts of itemset *X* at site *Si*, for $0 \le i \le n-1$. The global support count of *X* in *DB* is given as $X.sup = \sum_{i=0}^{n-1} X.\mathrm{sup}_i$. *X* is globally frequent if $X.sup \ge minSup \times |DB|$. Similarly, *X* is locally frequent if $X.sup_i \ge minSup \times |db_i|$. [4]

Table I. Data Set Example [3]

TID	Items
T1	ABC
T2	ABC
T3	ABC
T4	AB
T5	A
T6	AC

Two problems are addressed here: one for protection the privacy for each site when we evaluate the global support itemsets; the other is to hide a sensitive frequent itemsets from global support items. In the first problem, the commutative encryption should be applied to preserve the global candidate itemsets in each site. Each site encrypts its own local frequent itemsets and some fake itemsets, and then sends the encrypted itemsets to the next site until all sites have encrypted all itemsets. Next, it merges all encrypted itemsets, and all encrypted itemsets are then decrypted site by site.

The second problem is to hide the sensitive frequent itemsets or rules and minimize the loss items. When the global frequent for the sensitive rules satisfies these two conditions:-

i. *Support(X=>Y)= P(X and Y) >= Min_sup ;*
ii. *Confidence(X=>Y)= P(X/Y)= [Support($X \cup Y$)/Support(X)] >= Min_conf.*

where X and Y represent the candidate attributes. That means that this rule is frequent and it should be hidden. However, this rule can be hidden by:

- Reducing the support of confidential rules (by decreasing the support of the corresponding large XY).
- Reducing the confidence of rules (by Increasing the support of X in transactions not supporting Y or decreasing the support of Y in transactions supporting both X and Y)

This can be done by deleting or adding a new data to the original database. This way prevents tools from discovering these rules, but the challenge is the data quality. When a support of items is changed, some other insensitive rules will also be affected either by hiding it or supporting another frequent rule. Therefore, we need to define good ways to reduce the negative side effects on data quality.

IV. Proposed Approache and Algorithms

The main aim of our proposed system is to securely and efficiently preserve the privacy of distributed data mining of association rules on horizontally partitioned database. This section is intended to serve as a work-in-progress report on our proposed work. This work generally can be divided into two phases: The first phase is responsible for protection of the privacy for each site when we evaluate the global support itemsets. This can be done by using SMC protocol. In this proposal, we use a commutative encryption tool. Each site encrypts its own frequent itemsets or rules, and then passes it to other sites until all sites have encrypted all frequent itemsets. These are next passed to a common site to eliminate duplicates and to begin decryption. This set is then passed to each site that decrypts each frequent itemsets. The final result represents the global support frequent itemsets.

In the second phase, we need to hide sensitive frequent itemsets from global support items. This can be done when we reduce the support of confidential rules by decreasing the number of items that support these rules. This can be done by removing these items from original database in each site until the frequent itemsets become less than *min_support* threshold. Figure 1 represents the proposed system.

The major steps for *phase one* can be explained as follows (Assuming that we have three sites S1, S2 and S3):

1. *Determination of the local frequent itemset:*
   - Each site determines local frequent itemset for the sensitive rule (R) by using the following prior algorithm pseudo code:

```
Ck: Candidate itemset of size k
Lk: frequent itemset of size k
L1= {frequent items};
```

```
for(k= 1; Lk!=Ø; k++) do begin
Ck+1= candidates generated from Lk;
for each transaction tin database do
increment the count of all candidates in Ck+1that are contained in t
Lk+1= candidates in Ck+1with min_support
end
return ⋃k Lk;
```

- Join Step: Ck is generated by joining Lk-1with itself.
- Prune Step: Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset.

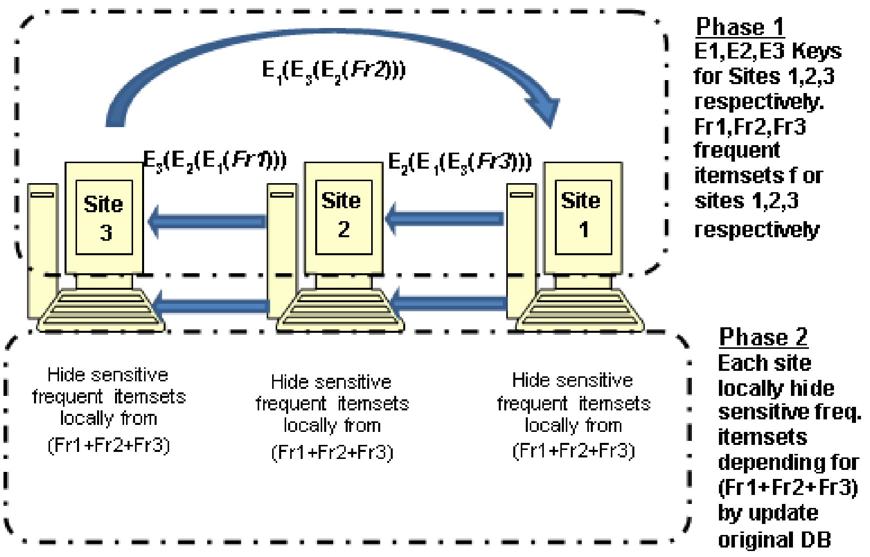
$E_1(E_3(E_2(Fr2)))$
$E_3(E_2(E_1(Fr1)))$
$E_2(E_1(E_3(Fr3)))$
Site 3
Site 2
Site 1
Phase 1
E1,E2,E3 Keys for Sites 1,2,3 respectively. Fr1,Fr2,Fr3 frequent itemsets f or sites 1,2,3 respectively
Hide sensitive frequent itemsets locally from (Fr1+Fr2+Fr3)
Hide sensitive frequent itemsets locally from (Fr1+Fr2+Fr3)
Hide sensitive frequent itemsets locally from (Fr1+Fr2+Fr3)
Phase 2
Each site locally hide sensitive freq. itemsets depending for (Fr1+Fr2+Fr3) by update original DB

Figure 1. General architecture of the proposed system

2. *Determining the global support itemsets for each site without disclosing the privacy of sites:*
   - Assume that the FI1, FI2, and FI3 represent the local support items or rules and E1, E2, and E3 represent the commutative encryption algorithm with its keys, for sites S1, S2, and S3 respectively (we use Pohlig–Hellman algorithm to perform the commutative encryption) .

     Where $\mathrm{FI1} = \sum_{i=1}^{n} \mathrm{FI}_{1i}$

     $$\mathrm{FI2} = \sum_{i=1}^{n} \mathrm{FI}_{2i} \qquad \ldots(1)$$

     $\mathrm{FI3} = \sum_{i=1}^{n} \mathrm{FI}_{3i}$
   - Each site encrypts its rule and sends it to the next site:
     - S1 determines E1 (FI1) and sends it to S2.
     - S2 determines E2 (FI2) and sends it to S3.
     - S3 determines E3 (FI3) and sends it to S1.
   - Each site encrypts the received data with its key and sends the result to the next site:
     - S1 determines E1 (E3 (FI3)) and sends it to S2.
     - S2 determines E2 (E1 (FI1)) and sends it to S3.
     - S3 determines E3 (E2 (FI2)) and sends it to S1.
   - Again, each site encrypts the received data with its key and sends the result to the next site:
     - S1 determines E1 (E3 (E2 (FI2))) and sends it to S2.

  - S2 determines E2 (E1 (E3 (FI3))) and sends it to S3.
  - S3 determines E3 (E2 (E1 (FI1))) and sends it to S1.
- Because we use a commutative algorithm the above encryption of data can be written as:
  - E1 (E2 (E3 (FI1)))
  - E1 (E2 (E3 (FI2)))
  - E1 (E2 (E3 (FI3)))
- This encrypted data is then sent to S1, and S1 decrypts it (removes E1) and sends the result to S2:
  - E2 (E3 (FI1))
  - E2 (E3 (FI2))
  - E2 (E3 (FI3))
- By the same way, S2 decrypts received data (removes E2) and sends the result to S3:
  - E3 (FI1)
  - E3 (FI2)
  - E3 (FI3)
- S3 decrypts the remained data (removes E3) and gets FI1, FI2 and FI3 without knowing from which site each of these rules have come.
- Now S3 has FI1+FI2+FI3. He can determine the global support and confidence for the rules and broadcast them to the other sites.

In *Phase two*, a proposed algorithm is used to hide sensitive frequent itemsets or rules by decreasing the support of their generating itemsets until their support is below the minimum support threshold. The steps for hiding sensitive frequent itemsets or rules for each site in the system can be explained as follows:

1. Each site has Local frequent itemset, global frequent itemsets, local database ($|Td|$) and minimum support threshold.
2. Input is the sensitive rule or frequent itemsets.
3. Extract all transactions that support this sensitive frequent items (Tf).
4. Evaluate the Site Remove Frequent Itemset (SRFI) that is representing how many frequent itemsets are needed to remove from each site as seen in eq. (2).

   $$SRFI = ((LSF/GSF) * (GSF - min\_supp))/100 * |Td| \quad \dots(2)$$

   where
   LSF = Local support frequent,
   GSF =Global support frequent, and
   min_supp =Minimum Support Threshold.
   $|Td|$ = number of transections in local site
5. Evaluation of the number of each item in sensitive itemsets needs to be removed from $|Tf|$ by eq. (3):

   $$|Iri| = |IDi|/|IDt| * SRFI \quad \dots(3)$$

   where
   |Iri| = number of item i needed to be removed,
   |IDi| = count for item i in database,
   |IDt| = summation for all items count in database;

   $$|IDt| = \sum_{i=1}^{n} ID_i \quad \dots(4)$$

This evaluation performed according to the ratio of each item in sensitive itemsets for all database transaction in order to reduce the side effects that may result from modifying the database.

6. Sort (Tf) in ascending order to minimize the impact that applied changes will have in database.
7. Sort items in itemsets in descending order according to the |Ir|. This also will minimize the side effects that can happen when modifying the database.
8. Remove items from (Tf) according to the above sorting, by setting the value for this item to “0” instead of “1”.

These steps will be done at all sites in the system in order to recalculate the global support itemsets. After recalculation, the sensitive frequent itemsets or rules have been hidden from the new global support itemsets. The pseudo code for the proposed algorithm is shown in Figure 2.

Let us take an example to clarify the operation of the proposed algorithm. Consider three local sites S1, S2 and S3, which have DB1, DB2 and DB3 respectively. We need to hide Sensitive Frequent Itemsets or rules “1, 15 and 16“. The local frequent itemset (LSF) in each site and global frequent itemset (GSF) for these rules are explained in Table II. Let the minimum support threshold be 5%. The size of database is 30000 transactions (10000 transactions in each site). To hide this Sensitive Frequent Itemsets we need to modify the database in each site. The following steps perform the hiding operation in each site.

A. Steps in Site 1

Site 1 has the following parameters:
|Tf|=692, |ID1|=4218, |ID2|=4176, |ID3|=3975, |IDt|=12360 and $|Td1| = 10000$. Now, we modify DB1 according to the following calculations, and evaluate the New LSF for site1:

SRFI = ((6.92/7.38)* (7.38-5)) /100 * 10000=223.64

$|\text{Ir1}|$= (4218/12360)*223.64 = 77 (number of transactions that contains item 1 needed to be modified in DB1)

$|\text{Ir2}|$= (4176/12360)*223.64 = 76 (number of transactions that contains item 15 needed to be modified in DB1)

$|\text{Ir3}|$= (3975/12360)*223.64 = 72 (number of transactions that contains item 16 needed to be modified in DB1)

Thus, the new LSF S1=692-(77+76+72) = 467. The percentage of the new LSF is S1=467/100 = 4.67%.

```
Begin
1. Extract |Tf| from database.
2. Evaluate SRFI= (LSF/GSF)*(GSF-min_supp).
3. Calculate |IDi| and |IDt,| where |IDt| = $\sum_{i=1}^{n} ID_i$.
4. Evaluate |Iri|=|IDi|/|IDt|* SRFI.
5. Sort (Tf) in ascending order.
6. Sort items in item set in descending order.
7. Let N= number of items in sensitive itemset
8. For i=1 to N {
     9. For j=1 to |Iri|
      // choose the transaction in Tf
     // with the lowest size
   10. Choose item i
   11. set to zero (i, t values of items).
12. }   end loop 1
13. }   end loop 2
```

Table II. LSF and GSF before Hiding Sensitive Frequent Itemset

Figure 2. The pseudo code of the proposed hiding algorithm

Sensitive Freq. Itemsets or Rule	LSF S1 (%)	LSF S2 (%)	LSF S3 (%)	GSF (%)
1 15 16	6.92	7.57	7.53	7.38

B. *Steps in Site 2*

Site 2 has the following parameters:
|Tf|=757, |ID1|=4310, |ID2|=4230, |ID3|=3996, |IDt|=12536 and $|Td2| = 10000$. Now, we modify DB2 according to the following calculations, and evaluate the New LSF for site2:

SRFI = (757/7.38)* (7.38-5) /100 * 10000=244.648

$|Ir1|$= (4310/12536)* 244.648= 85 (number of transactions that contains item 1 needed to be modified in DB2)

$|Ir2|$= (4230/12536)* 244.648= 83 (number of transactions that contains item 15 needed to be modified in DB2)

$|Ir3|$= (3996/12536)* 244.648= 78 (number of transactions that contains item 16 needed to be modified in DB2)

Thus, the new LSF S1=757-(85+83+78) = 511. The percentage of the new LSF is S1=511/100 = 5.11%.

C. *Steps in Site 3*

Site 3 has the following parameters:
|Tf|=753, |ID1|=4305, |ID2|=4294, |ID3|=4010, |IDt|=12609 and $|Td3| = 10000$. Now, we modify DB3 according to the following calculations, and evaluate the New LSF for site3:

SRFI = ((7.53/7.3875)* (7.3875-5)) /100 *10000=243.35

$|Ir1|$= (4305/12609) * 243.35= 84 (number of transactions that contains item 1 needed to be modified in DB3)

$|Ir2|$= (4294/12609) * 243.35= 83 (number of transactions that contains item 15 needed to be modified in DB3)

$|Ir3|$= (4010/12609)* 243.35= 78 (number of transactions that contains item 16 needed to be modified in DB3)

Thus, the new LSF S1=757-(84+83+78) = 510. The percentage of the new LSF is S1=510/100 = 5.10 %.

Therefore, the new GSF is calculated as:

New GSF = (new LSF S1+new LSF S2+new LSF S3)/3 …(5)

= (4.67+5.11+5.10)/3 = 4.96%

The new GSF (4.96%) is less than the *minimum-support* threshold (5%). That means that these frequent itemsets or rules have been successfully hidden. Table III represents the local frequent itemset in each site and global frequent itemset for Sensitive Frequent Itemsets or rule "1 15 16" after the application of the proposed algorithm and hiding this frequent itemset.

Table III. LSF and GSF after Hiding Sensitive Frequent Itemset

Sensitive Freq. Itemsets or Rule	LSF S1(%)	LSF S2(%)	LSF S3(%)	GSF (%)
1 15 16	4.67	5.11	5.10	4.96

V. Results Analysis and Performance Evaluation

The experiments for the proposal algorithm performed on a notebook with 2G MHz processor and 2 GB memory, under Windows XP operating system (in a distributed system setting there are three notebooks with the same properties). The sequence database generated for the experiments can be generated by using a Sequence Database Generator "SeqDBGen" [15] that works like IBM data generator [16]. Databases with sizes 30000, 60000, and 90000 transactions are generated for the series of experiments. The average length of transactions of each database (ATL) is 10 and 30 items in the generated database. The range of minimum support threshold given is 5-8%. The experimental results are obtained by averaging from 4 independent trials for each size of transaction with different sensitive frequent itemsets. Figure 3 represents the percentage of the difference between the evaluated global frequent itemset in central and distributed system with privacy. The result obtained a ratio of the global frequent itemset in the distributed system of about 99.99 % when it is compared with the global frequent itemset in the centralized system.

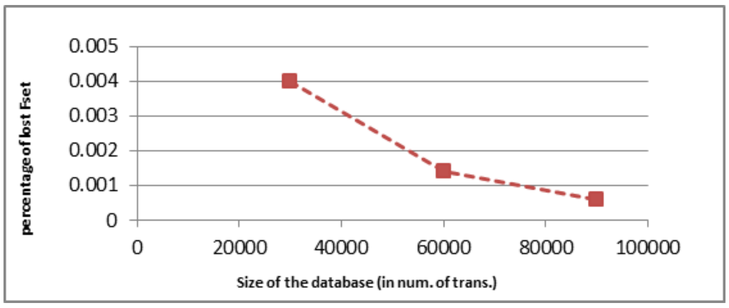

Figure 3. Percentage of Lost Rules in the Distributed System over Centralized System

The performance of the hiding algorithms has been measured according to two criteria: time requirements and side effects produced. Concerning the time requirements, we have considered the time needed by each algorithm to hide a specified set of rules. Concerning the side effects, we have considered the number of "lost" rules introduced by the hiding process. Hiding rules produce some changes in the original database. Such changes affect the set of rules mined. In particular, not all the rules that can be mined from the source DB can still is retrieved in the released DB. We call the former rules "lost rules".

To assess the performance of the proposed algorithms, this algorithm is used to hide sets of 5 and 10 frequent itemsets or rules mined from the datasets. In each time, the time required for hiding process will be measured. There is a file that contains the global frequent itemsets with the minimum support threshold. After the completion of the hiding process, we mine the released database and then compare the frequent itemsets or rules generated by the two databases. In order to do that, we check if the non-sensitive rules mined from the source database could still be mined in the released database. To do so, we compare each frequent itemsets or rules mined from the original database with each frequent itemsets rule mined form the released DB. If the rule is not found, it is considered as “lost”. Note that this process of rule checking tends to consider the rules selected for hiding as “lost”, since they cannot be retrieved from the released database. However, since they have been hidden on purpose, we excluded them from the set of “lost rules”.

The proposed algorithm applied on both centralized and distributed database. Tables IV-VII below represent the results for time measurements and the lost frequent itemsets in both cases. As shown in Tables IV and VI and Figures 4, 5, 6 and 7, there is no clear change for the number of lost itemsets in the distributed system when it is compared with the centralized database system. That means the proposal algorithm not more effected when it works on the distributed database system. However, the lost frequent itemsets do not have a significant effect on the results, because they are of very low percentage. Also, from the mentioned Figures, we can observe that the number of the lost itemsets is independent of the size of database, but it is linearly-related with the size of sensitive frequent itemsets.

Furthermore, from Tables V and VII and Figures 8 and 9, it is possible to notice that the measured time is of a linear growth with the size of database. The time requirement for hiding 10 frequent itemsets is higher than time needed to hide 5. This is an expected result because in the case of hiding 10 frequent itemsets, we need more modifications in database transactions; this will consume more time. Also, we observe that the time required in the distributed system is less than time required in central system. This is because the database is divided into different sites and the hiding algorithm work independently and in parallel on each of these sites. The number of the lost frequent itemsets is dependent on the sensitive frequent itemsets or rules. If these sensitive rules have high support, they will be of more impact on the database and more items would be required to be removed from transactions. This will increase the number of lost frequent itemsets or rules.

Table IV. Lost frequent itemsets in centralized DB

Transaction / Sensitive Frequent Items	30000(1484)		60000 (1437)		90000 (1448)	
	Lost F. Items	Lost F. Items %	Lost F. Items	Lost F. Items %	Lost F. Items	Lost F. Items %
5	18	1.2	8	0.55	10	0.6
10	35	2.3	23	1.6	17	1.1

Table V. Required Time in Centralized DB

Transaction / Sensitive F. Items	30000 (1484)	60000 (1437)	90000 (1448)
	Time (second)	Time (second)	Time (second)
5	3.3	7	10
10	7	11.2	17.2

Table VI. Lost frequent itemsets in the distributed DB

Transaction / Sensitive F. Items	30000(1478)	60000 (1434)	90000 (1447)

	Lost F. Items	Lost F. Items %	Lost F. Items	Lost F. Items %	Lost F. Items	Lost F. Items %
5	15	1	9	0.6	8	0.5
10	37	2.5	28	1.9	19	1.31

Table VII. Measured Time in the distributed DB

Transaction / Sensitive F. Item	30000 (1478)	60000 (1434)	90000 (1447)
	Time (second)	Time (second)	Time (second)
5	1	1.6	3.3
10	2.1	3.9	7

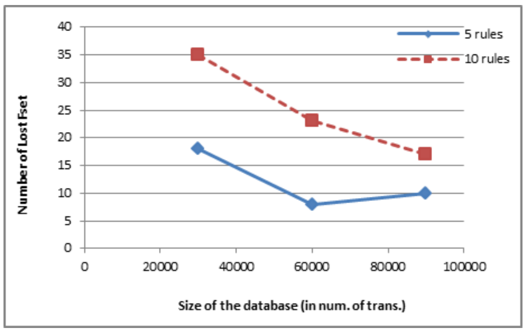

Figure 4. Number of lost frequent itemsets in the centralized system

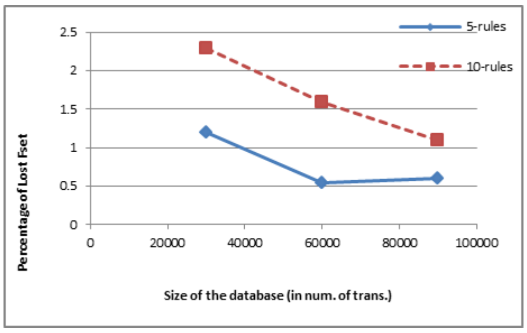

Figure 5. Percentage of lost frequent itemsets in centralized system

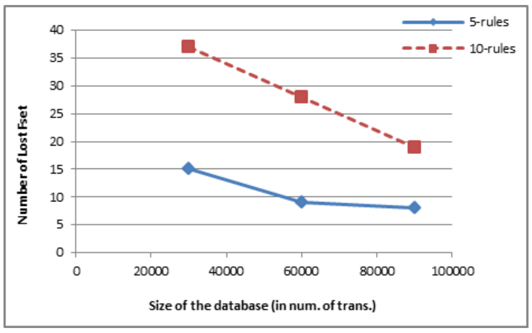

Figure 6. Number of lost frequent itemsets in the distributed system

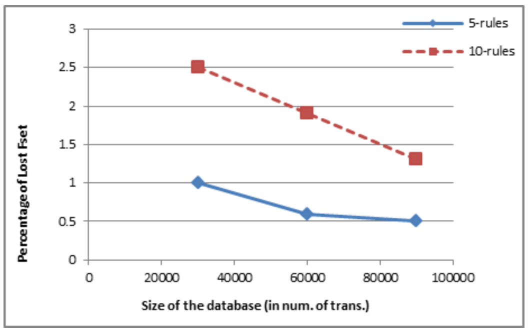

Figure 7. Percentage of lost frequent itemsets in the distributed system

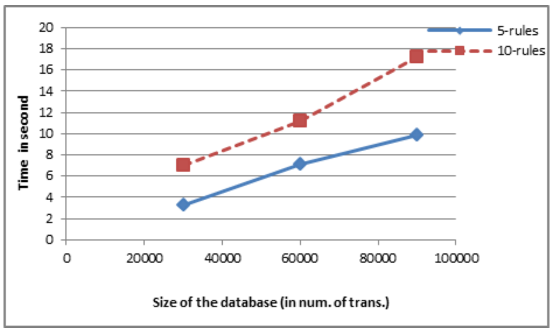

Figure 8. Required time in the centralized system

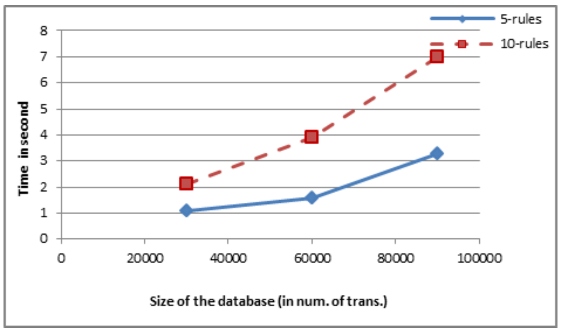

Figure 9. Required time in the distributed system

VI. Conclusion and Future Work

Privacy-preserving data mining on the distributed databases has become a hot research area. Not only in this competitive, but also cooperative business environment, companies, hospitals and other organizations need to share information with others, but not sharing the data. Also, some organizations need to hide some sensitive frequent sets or rules from any other external miner. In this paper, we proposed a system to allow sites like companies or other organizations to share knowledge while protecting at the same time the privacy of each site. We have extracted the frequent itemsets or rules from the distributed database and compared the resultant rules with the rules extracted from the same database, but in a centralized system setting. We have obtained a very small percentage error (about 0.002 %) in the distributed setting case. Also, we have proposed a new algorithm for hiding sensitive itemsets in distributed database. Its operation depends on the ratio of the frequent for sensitive itemsets in each site and the ratio of item counts for each item in the sensitive itemsets for the local database.

According to the obtained results, our proposal has resulted in a very limited side effect (lost frequent itemsets), while obtaining a significant reduction in the time requirement for the case of the distributed database system. As a future work, we will continue to improve the performance of our algorithm to hide sensitive association rules with the lowest possible level of side effects (lost rules and new rules) in a distributed system. We also try to more reduction of the algorithm time requirements.

REFERENCES

[1] Aggarwal, C.-C. and P. S. Yu, *Privacy-Preserving Data Mining Models and Algorithms*, Springer, USA , 2008. ISBN 978-0-387-70991-8.

[2] Agrawal, D. and C. Aggarwal, "The design and quantification of privacy preserving data mining algorithms," Proc. of ACM PODS Conference, 2002.

[3] Agrawal, R., T. Imielinski, and A. Swami, “Mining Association Rules between Sets of Items in Large Databases”, In Proceedings of ACM SIGMOD International Conference on Management of Data, Washington DC, May 1993.

[4] Chang, C.-C., J.-S. Yeh, and Y.-C. Li, “Privacy-Preserving Mining of Association Rules on Distributed Databases”, IJCSNS International Journal of Computer Science and Network Security, VOL.6 No.11, November 2006.

[5] Chintamani, R.D., I.F. Shaikh, and A.D. Waghmare, ”Hiding Sensitive Association Rules on Stars”, Emerging Trends in Computer Science and Information Technology -2012 (ETCSIT2012), Proceedings published in International Journal of Computer Applications (IJCA).

[6] Dasseni, E., V. Verykios, A. Elmagarmid and E. Bertino, “Hiding Association Rules by Using Confidence and Support” in Proceedings of 4th Information Hiding Workshop, 369-383, Pittsburgh, PA, 2001.

[7] Guo, Y., "Reconstruction-Based Association Rule Hiding," Proceedings of SIGMOD 2007, Ph.D. Workshop on Innovative Database Research 2007 (IDAR2007), June 10, 2007, Beijing, China.

[8] Muthu Lakshmi, N. and K. Sandhya Rani, ”An improved algorithm for hiding sensitive association rules using exact approach”, IRACST - International Journal of Computer Science and Information Technology & Security (IJCSITS), ISSN: 2249-9555Vol. 2, No. 1, 2012

[9] Naderi, M., D. K. Badie, and A. K. Zadeh, “A Novel Method for Privacy Preserving in Association Rule Mining Based on Genetic Algorithms”, Journal of Software, Vol. 4, No. 6, August 2009.

[10] Shah, K., A. Thakkar, A. Ganatra “A Study on Association Rule Hiding Approaches” International Journal of Engineering and Advanced Technology (IJEAT), ISSN: 2249 – 8958, Volume-1, Issue-3, February 2012.

[11] Shen, Y., H. Shao, and Y. Li, “Research on the Personalized Privacy Preserving Distributed Data Mining,” IEEE Second International Conference on Future Information Technology and Management Engineering, 2009.

[12] Verykios, V. S., A.K. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni, “Association Rule Hiding,” IEEE Transactions on Knowledge and Data Engineering, vol.16, no. 4, pp. 434-447, 2004.

[13] Wang, S.L., “Hiding sensitive predictive association rules”, Systems, Man and Cybernetics, 2005 IEEE International Conference on Information Reuse and Integration, vol. 1, pp. 164-169, 2005.

[14] Weng, C.-C., S.-T. Chen, and Y.-C. Chang, “A Novel Algorithm for Hiding Sensitive Frequent Itemsets”, IEEE Intelligent Systems Design and Applications, Vol. 3, pp.202-208, 2008.

[15] The Sequence Database Generator, Website at http://www.philippe-fournier-viger.com/seqdbgen/2012 , Accessed on 1/6/2012.

[16] IBM data generator, Website at http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/datamining/ mining.shtml , Accessed on 1/4/2012.