A New Algorithm to Preserve Sensitive Frequents Itemsets (APSFI) in Horizontal or Vertical Database

Dr. Hussein K. AL-Khafaji
Computer science, College of Al Rafidian/Baghdad
Noora A. Al-Saidi
Computer science, College of Al Rafidian/Baghdad
Email:Noora-ahmed9@yahoo.com

Revised on: 17/12/2012 & Accepted on: 9/5/2013

ABSTRACT

This research aimed to preserve on privacy of sensitive information from adversaries. We propose an Algorithm to Preserve Sensitive Frequents Itemsets (APSFI) with two ramifications to hides sensitive frequents itemsets in horizontal or vertical databases which minimize the number of database scanning processes during hiding operation. The main approach to hide sensitive frequent itemsets is to reduce the support of each given frequents sensitive 1-itemsets to be insensitive and convert another insensitive to be sensitive in the same transaction to avoid the change of database size and transaction's nature to avoid adversaries' doubt. The experiments of APSFI showed very encouraging results; it excluded 91% of database scan operations in vertical databases and 41% in horizontal layout databases in comparison with the well-known FHSFI algorithm. The experiments depict the APSFI tolerance for database size scalability, and its linear outperformance, from execution time aspect, in contrast with FHSFI.

Keyword: - Privacy Preserve, Sensitive Itemsets, Frequents Itemsets, Horizontal Database, Vertical Database.

خوارزمية جديدة لحماية العناصر المتكررة الحساسة في قواعد البيانات الأفقية او العمودية

الخلاصة

هذا البحث يهدف الى حماية سرية المعلومات الحساسة من الخصوم. هذا البحث يقدم خوارزمية لحماية العناصر المتكررة الحساسة بتفرعين احدهما لأخفاء العناصر المتكررة الحساسة في قواعد بيانات أفقية و ألأخرى قواعد بيانات عمودية.

الخوارزمية المقترحة تقلل من عدد عمليات المسح على قاعدة البيانات خلال عملية الأخفاء. ان عملية أخفاء العناصر المتكررة الحساسة تتم من خلال تقليل تكرار العناصر المتكررة الحساسة لتصبح غير حساسة وتحول عناصر اخرى غير حساسة لتصبح حساسة في نفس الصفقة لتفادي تغير حجم قاعدة البيانات و الصفقات لتجنب شك الخصوم. ان الأختبارات التي اجريت على الخوارزمية تعطي نتائج مشجعة، فهي تتجنب 91% من عمليات المسح على قاعدة البيانات العمودية و 41% على

https://doi.org/10.30684/etj.31.6B.7
2412-0758/University of Technology-Iraq, Baghdad, Iraq

قاعدة البيانات الأفقية بالمقارنة مع الخوارزمية المعروفة(FHSFI). تصف لنا هذه الأختبارات قدرة APSFI على التعامل مع قواعد بيانات ذات حجم قابل للزيادة، وخط انجاز عالي من خلال وقت التنفيذ مقارنة بخوارزمية FHSFI.

INTRODUCTION

Computers have promised us a fountain of wisdom but delivered a deluge of information. This huge amount of data makes it crucial to develop tools to discover what is called hidden knowledge. These tools are called data mining tools. So, data mining promises to discover what is hidden, but what if that hidden knowledge is sensitive and owners would not be happy if this knowledge were exposed to the public or to adversaries?

This problem motivates researcher to develop algorithms to assure data owners that privacy is protected while satisfying their need to share data for a common good.

Privacy preserving data mining (PPDM) come up with the idea of protecting sensitive data or knowledge to conserve privacy while data mining techniques can still be applied efficiently [1].

PPDM can be categorized as *data hiding* and *rule hiding*. In *data hiding* the database is modified in order to protect sensitive data of individuals. In *rule hiding* this modification is done to protect sensitive knowledge which can be mined from the database. In other words data hiding is related to input privacy while rule hiding is related to output privacy where frequent itemsets, association rules or classification rules are considered as outputs. Association rule or frequent itemset hiding is most popular method to provide output privacy [1].

There may be some situations where knowledge extracted by rule mining algorithms includes rules or itemsets that should stay unrevealed.

These itemsets are called sensitive itemsets. Itemset hiding intends to modify database in such a way that sensitive itemsets are hidden with minimum side effects on non-sensitive ones. Sanitization of the database by placing false or unknown values is NP-Hard problem so heuristic approaches are needed where the idea is to reduce the support and confidence of sensitive itemsets [2].

We have review algorithms that hide sensitive patterns by sanitizing datasets and we have shown their drawbacks.

The data-sharing techniques and pattern-sharing techniques face a challenge. That is, blocking as much inference channels to sensitive patterns as possible. Inference is defined as “the reasoning involved in drawing a conclusion or making a logical judgment on the basis of circumstantial evidence and prior conclusions rather than on the basis of direct observation” [3]. Farkas et al. [4] offer a good inference survey paper for more information. Frequent itemsets have an anti-monotonic property; that is, if X is frequent, all its subsets are frequent, and if X is not frequent, none of its supersets are frequent. Therefore, it is sound inference to conclude that XZ is frequent because XYZ is frequent. This has been called the “backward inference attack” [5].

On the other hand, the “forward inference attack” consists of concluding that XYZ is frequent from knowledge like “XY, Y Z, and XZ are frequent”. This is not sound inference. It has been suggested [5] one must hide one of XY, YZ, or XZ in order to hide XYZ; but, this is unnecessary (it is usually possible to hide XYZ while all of XY, YZ, and XZ remain frequent).

In huge databases, forcing to hide at least one subset among the k-1 subsets of a k-itemset results in hiding many non-sensitive itemsets unnecessarily. This

demonstrates that the method by Stanley et al. [5] removes more itemsets than necessary for unjustified reasons.

An adversary that systematically uses the "forward inference attack" in huge databases will find unlimited possibilities and reach many wrong conclusions.

Nevertheless, we argue that an adversary with knowledge that (in a sanitized database) XY, YZ and XZ are frequent may use the "forward inference attack" with much better success if XYZ is just below the privacy support threshold.

This is the drawback of the method by Atallah et al. [6] that heuristically attempts to remove the minimum non-sensitive itemsets but leaves open this inference channel. Atallah et al. proposed an (item restriction)-based algorithm to sanitize data in order to hide sensitive itemsets. The algorithm works on a set of sensitive itemsets in a one-by-one fashion.

Most of the association rule hiding algorithms are Apriori [7] based and needs multiple database scans to find support of sensitive itemsets because these techniques require data mining done prior to the hiding process.

In this research we produced two techniques to hiding sensitive itemsets in database. These techniques are based on looking for the child itemset, (1-itemset), with the highest support and reducing it. This leads to elimination of the need of pre-mining, avoidance of multiple scans of database, and heavy computational cost during hiding process. The proposed Algorithms to Preserve Sensitive Frequent Itemsets (APSFI), a solution to the forward and backward attack inference problems.

Indeed, APSFI consists of two sub algorithms, the first deal with horizontal layout and the second algorithm deal with vertical layout where horizontal database consist of two filed, transaction identifier TIDS, transaction items, while vertical database consists of an item filed and a list of the TIDS contain the item.

RELATED WORK

The advances in data mining have been proven beneficial for business, but also have introduced new problems in the preservation of privacy [8]. The privacy preserving data mining PPDM has been proposed as a solution to the problem of violating privacy while sharing data for knowledge extraction. The aim of PPDM is to develop algorithms to modify original data or mining techniques in such a way that useful knowledge can still be extracted while private data or knowledge is hidden. PPDM is mainly categorized as input and output privacy [9]. Input privacy is known as data hiding and output privacy is mostly known as rule or itemset hiding. Dasseni *et al*. then proposed a hiding algorithm based on the hamming-distance approach to reduce the confidence or support values of association rules [10]. Oliveira and Za¨ıane [11] introduced the multiple-rule hiding approach to efficiently hide sensitive itemsets. Amiri then proposed three heuristic algorithms to hide multiple sensitive rules [12]. Pontikakis *et al*. [13] proposed two heuristics approaches based on data distortion. Hong *et al.* proposed two approaches to partially delete the items within the transactions or the whole transactions from the original database for hiding sensitive itemsets [14].

Tzung-Pei.[15] then proposed a greedy-based algorithm for inserting newly transactions into the original database for efficiently hiding the sensitive itemsets.

FORMULATION AND NOTATIONS

Let DB be the database of transactions and J = {J1, ..., Jn} be the set of items. A transaction T includes one or more items in J (i.e., T$\subseteq$ J). For a given itemset X $\subseteq$ J

and a given transaction T, we say that T contains X if and only if $X \subseteq T$. The support count of an itemset X is defined to be X_{sup}, The support of itemset X can be computed by the equation (*support*(X) = ||X|| / |DB|). We say that an itemset X is large, with respect to a support threshold of s%, if $X_{sup} \geq |DB| \times s\%$ where |DB| is the number of transactions in the database DB. An association rule is an implication of the form "X => Y", where $X \subseteq J$, $Y \subseteq J$ and $X \cap Y = \emptyset$.

The association rule X =>Y is said to hold in the database DB with confidence c% if no less than c% of the transactions in DB that contain X also contain Y. The rule X => Y has support s% in DB if $X_{sup} \cup Y_{sup} = |DB| \times s\%$. For a given pair of confidence and support thresholds, the problem of mining association rules is to find out all the association rules that have confidence and support greater than the corresponding thresholds. This problem can be reduced to the problem of finding all large itemsets for the same support threshold. Thus, if s% is the given support threshold, the mining problem is reduced to the problem of finding the set $L = \{X \backslash X \subseteq J \wedge X_{sup} > |DB| \times s\%\}$ [16]. For the convenience of subsequent discussions, we call an itemset that contains exactly k items a k-itemset. We use the symbol L_k to denote the set of all k-itemsets in L.

THE PROPOSED ALGORITHMS

This section focuses on elucidating the steps of the proposed ***Algorithms to Preserve Sensitive Frequents Itemsets (APSFI),*** of hiding association rules via the hiding frequent itemsets in transactional database (DB). **APSFI** depends on the altering the support(s) of one or more items. It partially hides a frequent item to change its support to be infrequent and vice versa. In this research we propose two algorithms, the first one (**APSFI_HDB**) which deals with horizontal database (HDB) and the other (**APSFI_VDB**) which deals with vertical database (VDB). These algorithms manipulates two cases that are:

i) Hide Sensitive frequent itemsets (SFI) in transactional database.
ii) Reduce the re-mining operation over the DB to avoid a heavy computational cost during hiding SFI.

These algorithms will be explained bellow. Table (1) elucidates the notation used in the proposed algorithms.

Table (1) Notation Used in the Proposed Algorithms.

Notation	Description
DB	Original database
SL	Set of sensitive 1-itemset in a set of transactions A
S_{DB}	Set of infrequent 1-itemsets in a set of transactions A
Minsup	User defined minimum support threshold
|A|	The number of transactions in the transaction database A
X	A set of items (i.e., one itemset)
X_{sup}	Support of X in the set of transactions A
X_{TIDS}	Transaction list of X in the set of transactions A
M	Number of transactions we must be hide X from it
L^k_A	Set of large k-itemsets in a set of transactions A
SFI	Sensitive frequent itemset
HDB	Horizontal Database
VDB	Vertical Database

AR	Association Rules

The APSFI_HDB algorithm.

In this section, we introduce the "Algorithm to Preserve Sensitive Frequents Itemsets in Horizontal Database **"**, **APSFI_HDB**, step by step. Figure (1) depicts a high level code of the proposed algorithm. Firstly, we'll focus on the special case of hiding Sensitive Frequents Itemsets.

1- Select all frequent 1-itemsets into sensitive frequent itemset, SL.
2- Select infrequent 1-itemsets into small database list, $S_{DB.}$
3- For all 1-itemsets X in SL Do
reduce the support of X in **DB** to be lower than minsup threshold according to following
3.1 Compute $M=X_{sup} – (minsup-1)$
Where M is the number of deletion operation of X from DB, i.e., X will be deleted from M_X of transactions.
3.2 Remove X from SL.
3.3 Pick a suitable 1-itemset from $S_{DB,}$ Y.
Substitute Y in the updated transactions in 3.1. if that transactions dose not include Y.
3.4 Add Y to the L_{newDB} when Y become large.
3.5 Remove Y from S_{DB}.
4- Replace L_{DB} with L_{newDB}.

Figure (1) the high level code of the proposed algorithm.

Indeed Figure (1) is self-documented therefore; a brief description will be presented for its steps. According to step#1 all the frequent items will be listed in what so called sensitive frequent itemset (SL), while the infrequent items will be in small database (S_{DB}). Step#3 is repeated for all the items of SL and S_{DB}. Each item X of SL will be assigned a number, M_X, which represents the times of its deletions from M transactions. X will be substituted by Y element of S_{DB} if it is not exist in a specified transaction that X is deleted from. In many cases the substitution of X and Y is not one to one substitutions. The purpose of this operation is to change the support of X and Y.

```
APSFI_HDB algorithm  (input: DB, L_DB, |DB|, minsup);
                     (output: DB, L_newDB)
1.  SL= all 1-itemsets in L_DB
2.  S_DB = DB- SL.
3.  D=|DB|.
4.  if SL ≠Ø then
5.    Begin
6.       S=| S_DB|.
7.       J=1
8.       for all X ∈ SL do Begin
9.          X_sup= | X_TIDS|.
10.         if J<=S then Begin
11.            Y=transaction(J) from S_DB.
12.           Y_sup= | Y_TIDS|.
13.         end;{if}
14.         M= X_sup-(minsup-1).
15.         for i= 1 to M do Begin
16.            K=element(i) from X_TIDS.
17.            remove X from transaction K in DB.
18.            if (J<=S ) and (K ⊄ Y_TIDS) then Begin
19.               add Y to transaction K in DB
20.               Y_sup= Y_sup +1.
21.            end;{if}
22.            else if transaction(K)= Ø then  Begin
23.                  D=D-1
24.                  minsup=sth%*D /sth% is support threshold /
25.                  M= X_sup-(minsup-1).
26.               end; {if}
27.         end;{for}
28.         remove X  from SL
29.         if (J<=S ) then
30.            if  Y_sup >=minsup then Begin
31.               add Y to L_NewDB
32.               remove Y from S_DB
33.            end;{if}
34.         J=J+1
35.      end;{for}
36. end.{if}
37. remove all itemsets from L_DB
38. L_DB =L_NewDB
```

Figure (2) APSFI_HDB algorithm.

The **first step** selects all 1-itemsets of (L_{DB}) which required to be hidden in the database (DB). The selected items are stored in a list SL. In the **second step,** all the

infrequent 1-itemsets of DB are retrieved and stored in (S_{DB}) depending on eliminating SL from DB. **Step 8** selects an item represented by X from SL to hide it**. Step9** counts the support of item(X), its support represented by X_{sup}. **In the step 11**, Y stores an item of transaction#J in S_{DB}, while Y_{sup} , in step 12, contains the support of the item stored in Y. The purposes of **steps (14, 15, 16, 17, 18, and 19)** are computing the number of transactions, M, removing X from M transactions to convert X to infrequent item, adding Y, M times to the transactions that X removed from and/or to transactions that do not contain Y, to alter Y to be frequent, i.e., the addition of Y is restricted to the condition that Y is not an item in the transaction K. In many cases, deletion of one item converts a transactions to a null set, i.e. it becomes without items, therefore line #22 checks this case. Line #23 will decrease the number of transactions by one due to the eliminations. Line #24 will compute a

New value of minsup according to the elimination, anyway its percentage should be maintained, and also the value of M will be recomputed. This process will continue according to the value of M in step #25. Line #28 perform the duty of removing the sensitive itemset X from SL, i.e., sensitive frequent itemsets. **steps(29-32)** are checking the frequency of Y according to its support. Y will be deleted from S_{DB} and added to the new large database L_{newDB}. The steps (8- 35) continue iteratively until SL become empty.

After that, all itemsets in L_{DB} will be removed and substituted by the elements of L_{newDB} to be the new sensitive itemsets DB.

Example (1)

An example database is shown in Table (2), and one can find the support of each mined large k-itemset in Table (3), depending on minsup=2.

Table (2) Hypothesis Database.

TIDS	Items
1	A,C,D,E,G
2	A,B,D,E,G
3	A,D,F,G,H
4	A,E,I
5	F,G,H

Table (3) the new frequent Itemsets With minsup=2.

Support	Items	NO.
4	A,G	2
3	D,E,AD,AE,AG,ADG	6
2	F,H,DE,EG,GH,ADE	6

The first step selects all 1-itemset from L_{DB} and will be stored in a list SL = {A,D,E,F,G,H}. The infrequent 1-itemsets of DB are stored in the list S_{DB}={B,C,I}, shown in Table (4).

Table (4) Small database (S_{DB})

NO.	Items
1	B
2	C
3	I

Count the number of transactions in DB, D=5. Count the number of transactions in S_{DB}, S=3. We select one itemset X={A} from SL, and find the support of this item(X) A_{SUP}=4, after that we check the $S_{DB} \neq \emptyset$, fetch one itemset Y={B} from list S_{DB}, and compute the support of it B_{SUP}=1. The next step, computes the number of transactions required to hide A from by using equation $M= X_{SUP}-(minsup-1)$, M=3, i.e., remove A from M transactions to convert A to infrequent item i.e. A_{SUP}=1 (insensitive item). Add item(B) M times to the transactions that A removed from with the condition that these transactions do not contain item(B), to alter B to be frequent item B_{SUP}=3 (sensitive item), then insert item B to the list L_{newDB}.

After that, we have to check if the transaction (K) became empty, if so decrement D (the number of transactions in DB), compute the new minsup and the new number of transactions (M). Now recall Table (2) after hiding the item (A) and switch it by item(B).

Table (5)

TIDS	Items
1	**B**,C,D,E,G
2	B,D,E,G
3	**B**,D,F,G,H
4	A,E,I
5	F,G,H

These steps continue iteratively until SL become empty and then translate all itemsets from L_{newDB} to L_{DB} (after removing all itemsets from it) to generate new sensitive itemset through several iteration. Table (6) depicts The new database after hiding all sensitive 1-itemset from database.

Table (6) new database.

TIDS	Items
1	B,C,I
2	B,C,I
3	B,D

4	A,E,I
5	F,G,H

Table (7) The new Frequent Itemsets With minsup=2.

Support	Items	NO.
3	B,I	2
2	C,BC,BI,BCI	4

The APSFI_VDB algorithm.

This algorithm deals with vertical databases. The vertical layout has been shown to be successful for association mining. The benefits of using the vertical format have been demonstrated in Monte [17], a system for OLAP. The purpose of this algorithm (APSFI_VDB) is to preserve SFI in DB by hide frequent 1-itemsets in DB. In this approach we avoid the expensive steps of mining the database several times during the sanitization process. Figure (4) depicts this algorithm.

APSFI_VDB (input: DB, L_{DB}, minsup); (output: DB, L_{newDB})
1- SL= all 1-itemsets in L_{DB}
2- S_{DB} = DB- SL.
3- j=1
4- **For** all X ∈ SL **do Begin**
5- X_{sup}= | X_{TIDS}|.
6- M= X_{sup}-(minsup-1)
7- Y=transaction(j) from S_{DB}
8- Y_{sup}=| Y_{TIDS}|
9- **For** i= 1 to M **do Begin**
10- td= element(i) from X_{TIDS}
11- **If** (Y<>∅) and (td ⊄ Y_{TIDS}) **then Begin**
12- add td to Y_{TIDS}
13- Y_{sup}= Y_{sup}+1
14- **End;**{if}
15- Remove td from X_{TIDS}
16- X_{sup}= X_{sup}-1
17- **End;**{for}
18- j=j+1
19- Remove X from SL
20- **If** Y_{sup} >=minsup **then Begin**
21- Add Y to L_{newDB}
22- Remove Y from S_{DB}
23- **End;**{if}
24- **End;**{for}
25- remove all itemsets from L_{DB}
26- L_{DB} =L_{NewDB}
27-**End.**{Alg.}

Figure (4) APSFI_VDB algorithm.

The **first step** selects all 1-itemsets of L_{DB} which is required to be hidden in the database (DB). The selected items are stored in a list SL. In the **second step,** all the infrequent 1-itemsets of DB are retrieved and stored in (S_{DB}) depending on eliminating SL from DB. The steps (4,5) select one itemset(X) from list SL and find the support of it. Step(6) finds the number of transactions(M) that we can remove X from.

In the steps(7,8), Y stores the item of transaction#J in S_{DB} and its support will be counted. The steps(9-17), fetch one td (transaction identifier) from X_{TIDS} and add this td to Y_{TIDS} and remove the td from X_{TIDS}. These operations iteratively work M times until X will be insensitive and Y become sensitive. Steps(19-24) remove X from SL and add Y to L_{newDB} after checking its support(Y_{sup} >=minsup) . Steps(4 - 24) continue iteratively until SL become empty. Step#25 removes all itemsets from L_{DB}. In step#26, L_{DB} stores all the items in S_{DB}. Step#27generates K-itemsets from the 1-itemsets in L_{DB}.

Example (2)

Table (8) shows a transaction of database, and one can find the support of each mined frequent k-itemset in Table (9), depending on minsup=2.

Table (8) Database With minsup=2.

Items	TIDS
A	1,2,3,4
B	2
C	1
D	1,2,3
E	1,2,4
F	3,5
G	1,2,3,5
H	3,5
I	4

Table (9) the new frequent Itemsets.

Support	Items	NO.

4	A,G	2
3	D,E,AD,AE,AG,ADG	6
2	F,H,DE,EG,GH,ADE	6

The first step select all 1-itemset from L_{DB} and store them in a list SL = {A,D,E,F,G,H}, the infrequent 1-itemsets of DB are stored in the list S_{DB}={B,C,I}. Select one item(A) from SL and one item(B) from S_{DB}, Count the supports of each, A_{sup}=4 and B_{sup}=1. Compute the number of transactions (M=3) in which we must hide A and switch it with B. Select one item(td {1}) from A_{tids}{1,2,3,4}, insert this td {1} to B_{tids} to be frequent and remove this td{1} from A_{tids}{2,3,4} to be infrequent , these steps continue iteratively M times until A be insensitive and B becomes sensitive. After that remove A from list SL = {D, E, F, G, H} and check if the B_{sup}=4 become frequent. After that add it to list L_{newDB} and remove it from list S_{DB}={C,I}, These steps continue iteratively until SL becomes empty. Table (10) depicts The new database after hiding all sensitive 1-itemset from the database.

Table (10) New Database

Items	TIDS
A	4
B	1,2,3
C	1,2
D	3
E	4
F	5
G	5
H	5
I	1,2,4

COMPARISON WITH THE FHSFI ALGORITHM

Measurement of Execution Time of Each Algorithm.

We have performed our experiments on a notebook with 1.6GHz processor and 2 GB memory, under Windows XP operating system. The IBM data generator [18] is used to synthesize the databases for the experiments.

Table (11) shows the CPU time spent to process Databases with sizes 10K, 20K, 30K, 40K, 50K used to comparison between the two proposed algorithms and FHSFI (Fast Hiding Sensitive Frequent Itemsets) algorithm [19].

Table (11) Execution time for each algorithm.

No. of transactions in DB	CPU time(ms) FHSFI	CPU time(ms) APSFI_HDB	CPU time(ms) APSFI_VDB
10000	578.8	320	140.6
20000	1155.6	654.6	335
30000	1899	1002.8	550
40000	2133.9	1354	730.2
50000	2700.2	1705.6	900.4

In Table (11) depicts that the two proposed algorithms outperformed the FHSFI from execution time aspects. The CPU time requirements is shown in Figure (5).

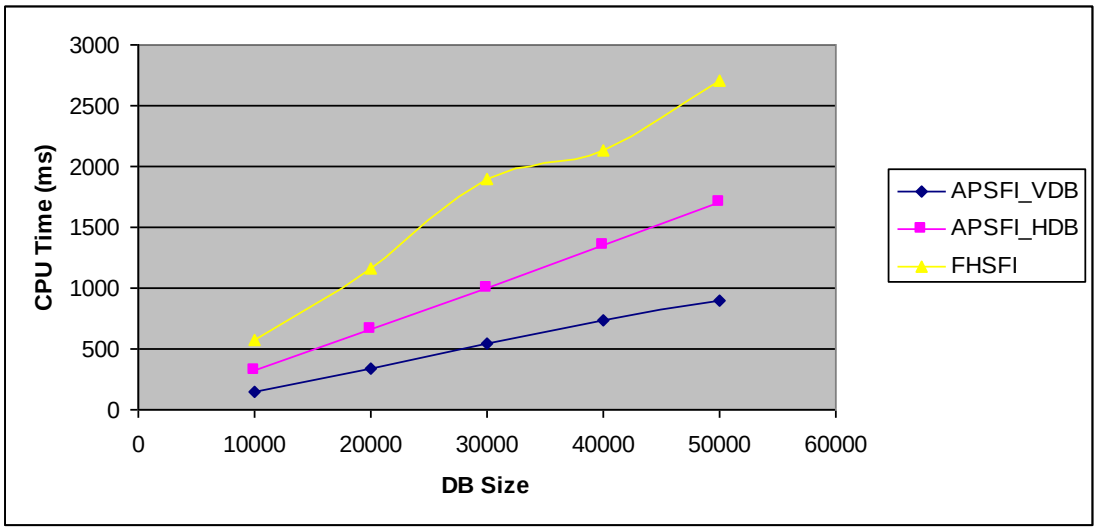

Figure (5) execution time of APSFI_VDB, APSFI_HDB, and FHSFI.

As shown in Figure (5), the execution time of the three algorithms is of linear growth with increasing size of databases, but APSFI_VDB overcomes APSFI_HDB and FHSFI in considerable time.

Counting of the number of DB Scan

Hiding some SFI from the database imperil this database to re-mined process to hide and uncover some itemsets. Most of the researchers ignore the increasing number of re-mining operation over the database.

This research presents two algorithms to manipulate this case by reducing re-mining operation to avoid a heavy computational cost during hiding SFI.

We can measure the number of re-mining of each algorithm by using sample of DB as shown in Table(12), and one can find the SFI in Table (13), depending on minsup=2.

Table (12) DB.

TIDS	ITEMSET
1	A,B,D,E,G
2	A,D,E,G
3	A,D,F,G,H
4	A,B,E,I
5	F,G,H

Table (13) the sensitive frequent itemset.

No.	Sensitive Frequent Itemsets	SUPPORT
1	A,G	4
2	D,E,AD,AE,AG, DG	3
3	B,F,H,AB,BE,DE, EG,FG, FH, ABE, DEG	2

This section illustrates the benefits of proposed algorithms by counting of the re-mining operation for each algorithm and compare the results as shown in Table (14).

Table (14) Comparison between the number of DB scan for each algorithm.

Name of algorithms	No. of hidden 1-itemset from DB	No. of hidden 2-itemset from DB	No. of hidden 3-itemset from DB	No. of re-mining in DB after hiding
FHSFI	5	4	2	11
APSFI_HDB	7	0	0	7
APSFI_VDB	7	0	0	1

Indeed, there is an extra scan operation in FHSFI not computed above; this operation counting weight of each transaction in DB, therefore its number of scanning over the DB is 12. It is well-known that database scan operation is the main cause of time consumption. According to the experimental results and the abstracted example in Table(14), APSFI_HDB excludes (12-7)/12=41% of scanning operations done by FHSFI, while APSFI_VDB eliminates (12-1)/12=91% of scanning operations of FHSFI, also it prohibits (7-1)/7=85% of the APSFI_HDB algorithm.

Another Comparison Factor

APSFI hides 1-itemset and inserts another in the same transaction to avoid the change in the weight of the transactions and the size of the DB, this operation produces two benefits:

- Doubt prevention.
- Itemsets losing prevention.
  The contrast of these two benefits is drawbacks of all PPDM algorithms.

DISCUSSION AND CONCLUSION

This research presents a new algorithm (APSFI) to hide extracted AR by hiding sensitive frequent itemsets. The main approach to hide sensitive frequent itemsets is minimizing the support to convert this sensitive itemset to insensitive, and emerging another insensitive to be sensitive by increasing its support in the same transaction to avoid the change in the weight of the transaction and DB size. The hide operation restricted to hide sensitive 1-itemset, to reduce the execution time consumption during candidates generating.

APSFI modifies only few parts of transactions in the DB, depending on the support reducing of the itemset to be insensitive. It adds this flexibility with a reasonable cost and blocking more inference channels.

Our approach has achieved the following goals:

- all sensitive frequent itemsets can be completely hidden without generating frequent k-itemsets.
- limited side effects by avoid the generating candidate(2-itemsets or more).
- APSFI_HDB reduces the scanning over the DB and the APSFI_VDB algorithm need only one scan over the DB.
- These algorithms are a solution to the forward and backward inference attacks problems.

REFRENCE

[1]. Barış Yıldız, Belgin Ergenç "Integrated Approach for Privacy Preserving Itemset Mining", Intelligent Control and Innovative Computing and Electrical Engineering, Volume110, pp247-260,2012,availableat http://www.springerlink.com/index/GT876834J8575958.

[2] Verykios V, Elmagarmid A, Bertino E, Saygin Y, Dasseni E *"*Association rule hiding", IEEE Trans Knowledge Data Eng 16(4): 434–447, 2004.

[3] Dictionary.com. http://dictionary.reference.com/,accessed on the 9th of December 2005.

[4]. Farkas and S. Jajodia, C. "The inference problem: a survey", In Proceedings of the ACM SIGKDD Explorations Newsletter, volume 4(2), pages 6–11, New York, NY, USA, ACM Press, December 2002.

[5]. Oliveira, S. R. M. O. R. Zaiane, and Y. Saygin, "Secure association rule sharing", In H. Dai, R. Srikant, and C. Zhang, editors, Proceedings of the 8th PAKDD Conference, volume 3056, pages 74– 85, Sydney, Australia, May 2004.

[6]. Atallah, M. E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios, *"Disclosure limitation of sensitive rules*", In Proceedings of 1999 IEEE Knowledge and Data Engineering Exchange Workshop(KDEX'99), pages 45–52, Chicago, Illinois, USA, November 1999. IEEE Computer Society.

[7]. Agrawal R, Srikant R, "*Fast* algorithms for mining association

rules in large databases", Proceedings of 20th International Conference on very large databases, VLDB'94, Santiago de Chile, 12–15September 1994, pp 487–499, 1994.

[8]. Mitchell, M. "An Introduction to Genetic Algorithms," MIT press, 1996.

[9]. Ahluwalia M, Gangopadhyay, "APrivacy preserving data mining: taxonomy of existing techniques", In: Subramanian R (ed) Computer security, privacy and politics: current issues, challenges and solutions. IRM, New York, pp 70–93, 2008.

[10]. Dasseni, E. V. S. Verykios, A. K. Elmagarmid, and E. Bertino, "Hiding association rules by using confidence and support", The International Workshop on Information Hiding, pp. 369-383, 2001.

[11]. Oliveira and O. R. Za¨ıane, S. R. M. "Privacy preserving frequent itemset mining", IEEE International Conference on Privacy, Security and Data Mining, pp. 43-54, 2002.

[12]. Amiri, A. "Dare to share: Protecting sensitive knowledge with data Sanitization", Decision Support Systems, pp. 181–191, 2007.

[13]. Pontikakis, E. D. A. A. Tsitsonis, and V. S. Verykios, "An Experiential study of distortion-based techniques for association rule hiding", IFIP Workshop on Database Security, pp. 325-339, 2004.

[14]. Hong,T. P. C. W. Lin, K. T. Yang, and S. L. Wang, "A heuristic data-sanitization approach based on tf-idf", The International Conference on Industrial, Engineering andOther Applications of Applied Intelligent Systems, pp. 156-164, 2011.

[15]. Tzung-Pei Hong1,3, Chun-Wei Lin1, Chia-Ching Chang1 and Shyue-Liang Wang2, "Hiding Sensitive Itemsets by Inserting Dummy Transactions", 2011.

[16].Ramez Elmasri, Shamkant B. Navathe, "Fundamentals of Database Systems",(5th Edition), Addison-Wesley, 2007.

[17]. Manegold P. A. Bonz, M. Kelten, "Database Architecture Optimized for the New Bottleneck" In 26th Intl. Conf. VLDs, 1999.

[18].IBM Research almaden, available at http://www.almaden.ibm.com /cs/projects/iis/hdb /Projects/ data_ mining/mining.shtml, 2007.

[19]. Chih-Chia Weng, Shan-Tai Chen, Yuan-Chung Chang, " A Novel Algorithm for Hiding Sensitive Frequent Itemsets", Dept. of Information Science, Chung Cheng Institute of Technology, National Defense University, available at http://isis2007.fuzzy.or.kr/submission/upload/A1302.pdf