Hiding Sensitive Predictive Frequent Itemsets

Barış Yıldız and Belgin Ergenç

***Abstract*—In this work, we propose an itemset hiding algorithm with four versions that use different heuristics in selecting the item in itemset and the transaction for distortion. The main strengths of itemset hiding algorithm can be stated as i) it works without pre-mining so privacy breech caused by revealing frequent itemsets in advance is prevented and efficiency is increased, ii) base algorithm (Matrix-Apriori) works without candidate generation so efficiency is increased, iii) sanitized database and frequent itemsets of this database are given as outputs so no post-mining is required and iv) simple heuristics like the length of the pattern and the frequency of the item in the pattern are used for selecting the item for distortion. We compare versions of our itemset hiding algorithm by their side effects, runtimes and distortion on original database.**

***Index Terms*—frequent itemset mining, privacy preserving data mining, sensitive itemset hiding**

I. Introduction

Data mining is simply defined as finding hidden information from large data sources. It became popular in last decades by the help of increase in abilities of computers and collection of large amount of data [1]. Although it is successfully applied in many fields such as marketing, forecasting, diagnosis and security, it is a challenge to extract knowledge without violating data owner's privacy [1, 2, 3, 4, 5 and 6]. 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.

PPDM can be divided into two 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. Frequent itemsets, association rules or classification rules are considered as outputs. Association rule hiding is studied mostly and considered as synonym to rule hiding and includes frequent itemset hiding.

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. The first study on rule hiding shows that sanitization of the database is NP-Hard and heuristic approaches are needed [7]. Heuristic approaches are based on support and confidence reduction. Following studies propose algorithms for itemset hiding and association rule hiding respectively [8 and 9]. These algorithms distort items in the database. However, there may be such conditions that writing false values may cause problems. The approach used in [10] use unknown values instead of writing false values on the database.

Many itemset or rule hiding approaches are based on Apriori algorithm which needs multiple database scans and pre-mining of association rules. On the other hand FP-Growth algorithm, which has a better performance compared to Apriori, makes two database scans for finding frequent itemsets [11]. The work presented in [12] uses hiding algorithm based on P-tree [13] similar to FP-tree of FP-Growth algorithm. They sanitize informative rules and eliminate need for pre-mining of association rules. Another, frequent itemset mining algorithm with two database scans is Matrix-Apriori [14]. It is simpler than FP-Growth in terms of maintenance of the compact data structure and performs better [15].

In this paper, four versions of the same itemset hiding algorithm are proposed where the Matrix-Apriori algorithm is modified to have itemset hiding capabilities. Each version uses different heuristics in selecting the transaction and the item in itemset to distort. Our algorithm i) inputs sensitive itemsets, no matter whether they are frequent or not, which prevents privacy breech caused by pre-mining, ii) supports are found during hiding process and at the end sanitized database and frequent itemsets from this database are given as outputs, this eliminates the need of frequent itemset mining to be done on new sanitized database and iv) uses simple heuristics in itemset hiding avoiding heavy optimization cost.

Case studies are done to show the performance of four versions of our itemset hiding algorithm to see the side effects, hiding time and distortion on initial database while changing the size of the original database, the number of sensitive itemsets and support of sensitive itemsets. Results showed that i) spmaxFI (select shortest pattern and maximum of frequent items) has better overall performance both as side effect and runtime, ii) selecting shortest pattern (spmaxFI and spminFI) is better in any case than selecting longest pattern (lpmaxFI and lpminFI), iii) side effect is related to given sensitive itemset., iv) support count or database size is not directly related to the number of lost itemsets and v) time to hide sensitive itemset is a function of distortion and database size and vi) distortion is related to support count.

The structure of the paper is as follows. Next section gives a short survey about association rule hiding and

Manuscript received December 28, 2010; revised January 23, 2011.

Barış Yıldız was with the Department of Computer Engineering, İzmir Institute of Technology, İzmir 35430 TURKEY. He is now with the Department of Computer Engineering, Dokuz Eylül University, İzmir 35160 TURKEY (e-mail: barisyildiz@computer.org).

Belgin Ergenç is with the Department of Computer Engineering, İzmir Institute of Technology, İzmir 35430 TURKEY (e-mail: belginergenc@iyte.edu.tr).

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

itemset hiding. In section 3 our itemset hiding algorithm is explained. Performance evaluation is given with discussion on results in section 4. Then the paper is concluded with our final remarks on the study and the future work in section 5.

II. Related Work

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 divided into two: input and output privacy [16]. Input privacy is known as data hiding and output privacy is known as rule hiding.

In data hiding, sensitive data is modified or trimmed out from the original database so that individual's private data will not be revealed by the data mining algorithm. Data hiding can be divided into three subgroups: perturbation based techniques [17, 18 and 19], cryptographic techniques [20 and 21] and anonymization based techniques [22, 23 and 24].

In rule hiding, sensitive knowledge which can be mined from the database is hidden while non-sensitive knowledge can still be mined [25]. Rule hiding research focuses on association rule hiding and frequent itemset hiding. It refers to the process of modifying the original database in such a way that certain sensitive association rules or frequent itemsets disappear without seriously affecting the data and non-sensitive rules or itemsets. The most wide ranging survey about association rule hiding in [26] divides association rule hiding methods as heuristic, border based and exact approaches.

Exact approaches give optimal solution and have no side effect, on the other hand have much computational cost. In [27 and 28] exact techniques are given which formulate sanitization as constraint satisfaction problem and solve these by integer programming. Border based approaches uses border theory [29]. In [30, 31 and 32] border based techniques for association rule hiding are proposed. The idea behind these approaches is that the elements on the border are boundary to the infrequent itemsets. During hiding process, instead of considering non-sensitive frequent itemsets, they are focused on preserving the quality of the border. Heuristic approaches uses heuristics for modifications in the database. These techniques are efficient, scalable and fast algorithms however they do not give optimal solution and may have side effects. These techniques based on support and confidence decreasing. There are two types of techniques: distortion and blocking. Distortion techniques select transactions which hold sensitive itemsets and then selected items are deleted from transaction and database is modified. Blocking techniques replaces items with unknown values instead of deletion of items to modify database. The first algorithm is based on support reduction [7]. In [9] five algorithms are proposed based on hiding strategies. Not only itemsets but also rules are considered through hiding in the algorithms.

A framework for frequent itemset hiding is proposed in [8]. Algorithms require two database scans. At first scan the inverted file index is created and at second scan items are deleted from selected transactions. In [10] blocking is used instead of distortion of items in the database. The idea behind this approach is that sometimes replacing false values may have bad consequences. The aim in the algorithms is hide given sensitive rules by replacing unknown values and minimize side effects on non-sensitive rules.

Many association rule hiding algorithms are Apriori [33] 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 [12] a tree structure which is similar to FP tree [11] is used to store information about database. This algorithm gets predictive item and sanitize informative rule set which is the smallest set of association rules that makes the same prediction as the entire rule set. The algorithm does not need data mining to be done before hiding process and does not scan database many times.

The idea of eliminating data mining to be done before hiding process and using a base algorithm which does not require multiple scans of the original database in [12], led us to propose our frequent itemset hiding algorithm. We used Matrix-Apriori algorithm [14] as the basis of hiding itemset process. It works without candidate generation and scans database only twice. It has a simpler data structure and showed to be faster compared to FP-growth in [15]. The sanitization process is embedded into the data mining process. In the following section our algorithm is introduced.

III. Itemset Hiding with Matrix Apriori

As displayed in Figure 1, our privacy preserving frequent itemset mining approach gets database *D*, sensitive itemsets *Ls* and minimum support *minsup* as input and returns sanitized database *Ds* with frequent itemsets which can be found from *Ds* as *FIs*. Sensitive itemsets are given without any knowledge about their frequency. If any itemset given as sensitive is frequent in original database then it is hidden through itemset hiding process. Most hiding approaches first do mining and calculate support of all frequent itemsets then start hiding process. This has two disadvantages i) it might cause a privacy breech if the one performing hiding process is not trusted because all frequent itemsets are required to be known before the hiding process and ii) it requires pre-mining causing decrease in efficiency. Our approach ensures that user does not know whether given sensitive itemset was frequent in original database because supports of sensitive itemsets are found during hiding process and eliminates the need for pre-mining process.

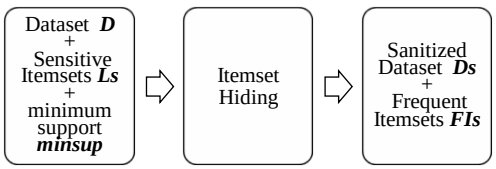

Figure 1. Sanitization Framework

The overall process of itemset hiding algorithm is shown in Figure 2. At first scan (P1), for the specified minimum support, frequent items are found. At second scan (P2), matrix MFI holding the frequent itemsets, vector STE holding the corresponding support counts of the itemsets in MFI and the TList holding the transaction ids of the rows of

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

database D containing the itemsets in MFI is build. Columns of the matrix MFI show the frequent items; each row shows a different itemset. If the corresponding item is present in the itemset corresponding cell value is set to "1", "0" otherwise. After that step MFI is modified to speed up frequent pattern search (P3). For each column of MFI, beginning from the first row, the value of a cell is set to the row number in which the item is "1". If there is not any "1" in remaining rows then the value of the cell is left as "1" which means down to the bottom of the matrix, there is no row that contains this item. After constructing the MFI matrix, finding patterns is simple. Beginning from the least frequent item, create candidate itemsets and count its support value. The support value of an itemset is the sum of the items at STE of which index are rows where all the items of the candidate itemset are included in MFI's related row. Frequent itemset mining is done on this compact data structure which eliminates the need for database scan for itemset support counting. This part of Matrix Apriori algorithm is modified to have itemset hiding capabilities (Line 1 to 15).

```
INPUT: Original Database D, minimum support minsup, List of sensitive itemsets Ls
OUTPUT: Sanitized Database Ds, Frequent itemsets FIs of Ds

BEGIN
P1   Read D and find frequent items           // first scan of database
P2   Read D and build MFI, STE and TList      // second scan of database
P3   Modify MFI                               //speeding-up the frequent pattern search

1  FOR every itemset in Ls
2     Calculate support of the sensitive itemset Is
3     Number of iterations:=(Support of Is -minsup) * number of transactions in TList +1
4     FOR 1 TO Number of iterations
5        Select pattern from MFI (shortest or longest one)
6        Select transaction from TList
7        Select item to distort (most frequent MaxFI or least frequent MinFI in Is)
8        Distort item in D
9        Update MFI
10       Update STE
11       Update TList
12    END
13 END
14 Find frequent itemsets FIs using up to date MFI
15 Return Ds, FIs
END
```

Figure 2. Itemset Hiding Algorithm

As explained above while building MFI and STE, we also construct a transaction list as TList which keeps the transaction ids of transactions containing the itemset in each row of MFI. In proposed approach, transaction selection for modifying is done on MFI and database scan in order to find transaction is eliminated.

Between lines 1 and 15 for every itemset in sensitive itemsets list Ls, hiding process is run. Support value for sensitive itemset is calculated using MFI and STE. If the support of the itemset is above minsup then the number of iterations to hide itemset is calculated (line 3). This number indicates number of distortions to be done on the dataset to reduce the support of the sensitive itemset Is below minsup. Following this, at each iteration transaction to modify is selected (lines 5 and 6). There are two strategies for transaction selection. First one is to find shortest pattern in MFI that includes the sensitive itemset and select the last transaction from TList. Second strategy is to find longest pattern and select the transaction from TList. Most heuristic approaches use transaction length to decide transaction to modify. However, compact matrix structure of proposed approach has more valuable information as patterns of frequent items so length of the pattern is used instead of length of the transaction. This approach also eliminates the need for database access in choosing decision.

When transaction is selected we need to select an item of the transaction for distortion (line 7). There are two strategies for selection of item to distort: maxFI and minFI. Using maxFI, most frequent item of sensitive itemset is distorted on transaction. If minFI is used then least frequent item of sensitive itemset is distorted on transaction. Selected item is distorted in transaction (line 8), the distortion technique is replacing "1" with "0" in related cell. Matrix structure MFI is updated after distortion (line 9). We decrease the value of related row in STE (line 10) and delete transaction modified in that row of TList (line 11). By this way it is ensured that we have compact mirror of semi-sanitized dataset in MFI, STE and TList throughout the hiding process.

The selection and distortion process is repeated until the support of sensitive itemset Is is below minsupport. After sanitization of a Is the next itemset is read from Ls and sanitized. At final step (line 15) frequent itemsets FIs of sanitized dataset Ds are found using up-to-date MFI and STE.

Now, let us explain an itemset hiding process using an example. Shortest pattern and most frequent item maxFI stragety is applied and itemset of BA is assumed to be sensitive (Is). In Figure 3 sample database, MFI, STE and TList before hiding process is given. For minsupport value 3 (50%) 4 frequent itemsets (length 1 itemsets are not included) can be found. These are CB, CA, CBA and BA. But remember that our approach does not need frequent itemset mining to be performed before hiding process.

TID	Items
T1	ABC
T2	ABC
T3	ABC
T4	AB
T5	AD
T6	CD

MFI			STE	TIDs
A	B	C		
2	2	2		
3	3	5	3	T1,T2,T3
4	1	0	1	T4
1	0	0	1	T5
0	0	1	1	T6

Figure 3. Database D, MFI, STE and TList before hiding process

As in line 2 of the hiding algorithm, using MFI and STE support of BA is calculated to be 4 (66%). Since the minsupport value is 3 (50%), number of iterations to sanitize

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

BA can be calculated as 2 (line 2). At first iteration shortest pattern that holds BA is found as third row of MFI and related transaction is T4 from TList. Most frequent item of sensitive itemset BA is A so it will be deleted from selected transaction (Figure 4). Meanwhile STE value of selected row is decreased and modified transaction id is deleted from the list. After deletion the new pattern B is added to matrix and T4 is added to transaction list which is now the sixth row of the matrix. At second iteration second row is selected as shortest and T3 is selected for modification. In Figure 4 sanitized database Ds, MFI, STE and TList after sanitization process are shown.

After sanitization process we are able to find frequent itemsets for sanitized database using up-to-date matrix structure. Support values of itemsets are calculated as CB(50%), CA(33%), CBA(33%) and BA(33%). Support of itemset BA is now under minsupport and it is hidden. CBA is also hidden because it is a superset of BA. However, CA is now under minimum support and cannot be find as frequent although it was not sensitive. This is the side effect and CA is called lost itemset.

TID	Items
T1	ABC
T2	ABC
T3	BC
T4	B
T5	AD
T6	CD

MFI			STE	TIDs
A	B	C		
2	2	2		
3	3	5	2	T1,T2
4	6	0	0	
1	0	0	1	T5
0	0	7	1	T6
0	7	0	1	T4
0	1	1	1	T3

Figure 4. Sanitized database Ds, MFI, STE and TList after hiding process

IV. Performance Evaluation

In this section, performance evaluation of four versions of our itemset hiding algorithms is given. These are spmaxFI (select shortest pattern and maximum of frequent items in the itemset), spminFI (select shortest pattern and minimum of frequent items in the itemset), lpmaxFI (select longest pattern and maximum of frequent items in the itemset) and lpminFI (select longest pattern and minimum of frequent items in the itemset). Two synthetic databases are used to see effect of different database size. The algorithms are executed on databases i) to see effect of increasing number of sensitive itemsets, ii) to see effect of increasing support of sensitive itemset. The effects observed are number of lost itemsets as side effect, time cost for hiding process and number of items distorted for hiding itemsets. During evaluations, it is ensured that the system state is similar for all test runs and results are checked for consistency.

A. Simulation EnvironmentSimulation Environment

Test runs are performed on a computer with 2.7 GHz dual core processor and 1 GB memory. At each run inputs are original database and sensitive itemsets where the outputs are sanitized database and frequent itemsets which can be mined from this sanitized database. Synthetic databases used in the evaluations are generated using ARtool software [34]. Two databases are used for evaluations are different in the number of transactions since we want to compare the effects of the size of database on hiding process. One database has 5000 transactions while number of items is 50 and average length of transactions is 5. Other database has 10000 transactions while number of items is 50 and average length of transactions is 5. Minimum support is defined as 2.5% for all evaluations and if no hiding is applied then 2714 frequent itemsets from 5k database and 5527 frequent itemsets from 10k database can be found.

B. Increasing Number of Sensitive Itemsets

For both databases five of length three itemsets which are closest to 3.0% support are selected as sensitive itemsets. These itemsets are given in Table 1 below. Selected itemsets are mutual exclusive to ensure that one is not affected by hiding process of previous itemsets. The aim of this study is to understand the effect of increasing the number of sensitive itemsets on itemset hiding. For each run next itemset in the table is added to the sensitive itemsets given to program. At first run itemset no 1 is given as sensitive, at second run itemset no 1 and itemset no 2 are given as sensitive and so on.

Table 1. Sensitive Itemsets

Itemset no	Itemsets for 5k database	Support (%)	Itemsets for 10k database	Support (%)
1	37 31 32	2.96%	36 20 6	3.00%
2	7 47 41	3.06%	50 13 10	3.01%
3	5 6 4	2.92%	33 49 42	2.93%
4	24 13 46	3.08%	29 14 11	3.07%
5	45 34 20	2.94%	39 41 18	2.95%

The side effect which is the number of lost itemsets for increasing number of sensitive itemsets is given in Figure 5 for 5k database and in Figure 6 for 10k database. In both databases number of lost itemsets is increased for all hiding algorithms while number of sensitive itemsets is increased. It is clear that spmaxFI (select shortest pattern and maximum frequent item of itemset) algorithm has least side effects. The difference reaches up to 100% at five sensitive itemsets case. What more can be inferred from these figures is that side effect is related to the characteristics of sensitive itemsets, not to the database size. Even for the best algorithm we come across higher number of lost itemsets for 5k database such that for 5 itemset hiding point 29 itemsets are lost in 5k database while 22 itemsets are lost in 10k database.

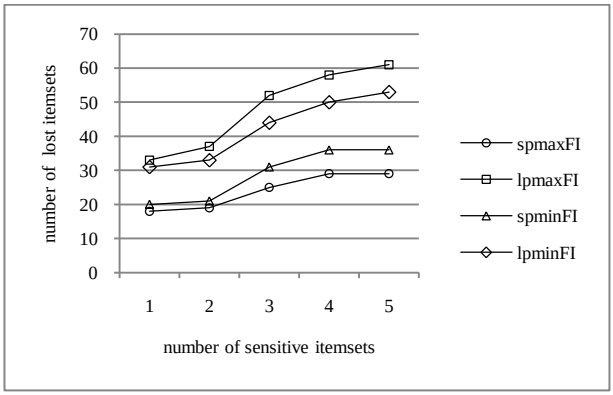

Figure 5. Side effect while increasing number of sensitive itemsets for 5k database

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

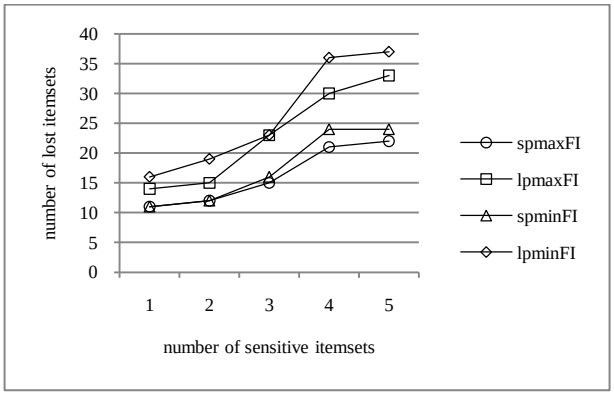

Figure 6. Side effect while increasing number of sensitive itemsets for 10k database

Time cost for itemset hiding is given in Figure 7 and Figure 8 for 5k database and 10k database respectively. Selecting shortest pattern seems as a better method no matter maximum or minimum frequent item is selected for distortion. Selecting longest pattern needs 20% to 100% more time compared to selecting shortest pattern method. It is clear from the figures that database size effects time to hide itemsets for same cases. The database size is doubled and time needed for hiding itemsets is increased more than 100%. The reason behind this is the cost of travelling on matrix to select pattern. It is clear that matrix size is bigger for 10k database compared to 5k database.

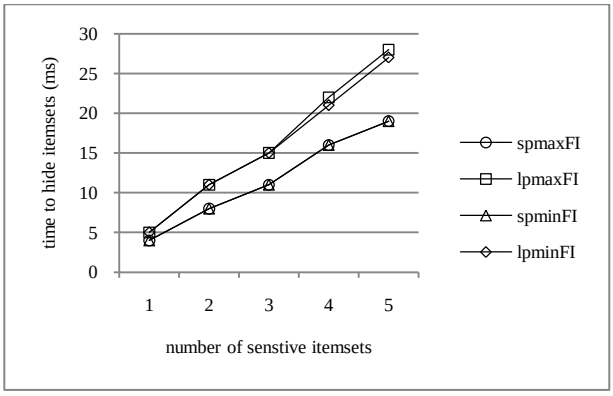

Figure 7. Time to hide itemsets while increasing number of sensitive itemsets for 5k database

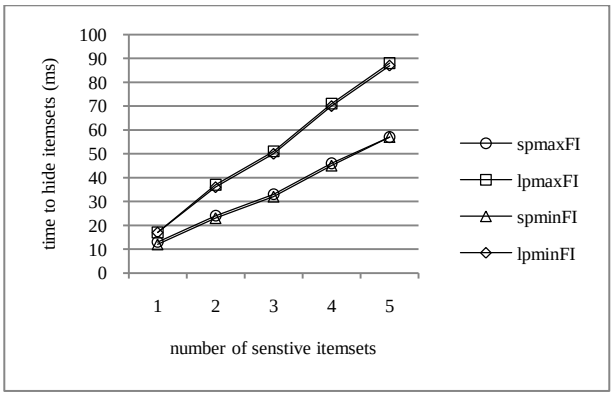

Figure 8. Time to hide itemsets while increasing number of sensitive itemsets for 10k database

The number of items to distort is similar for all algorithms. In Figure 9 and Figure 10 number of distorted items for increasing number of sensitive itemsets is given for 5k and 10k databases. Values are identical because calculation of number of items to distort is identical for all algorithms. For our case study this is also equal to number of transactions distorted since selected sensitive itemsets are mutual exclusive and there is no frequent itemset includes more than one sensitive itemset. Figures demonstrate that increasing the size of database will increase distorted items. This is a result of support count. For itemsets with same support value we have different support counts such that 3.0% support itemset in 5k database has 150 support count and itemset with same support value in 10k database has 300 support count.

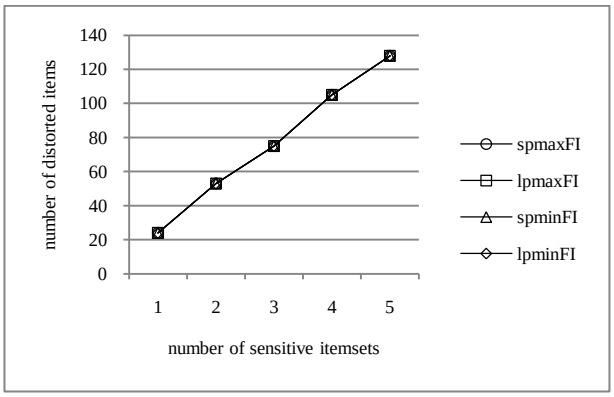

Figure 9. Number of items to distort while increasing number of sensitive itemsets for 5k database

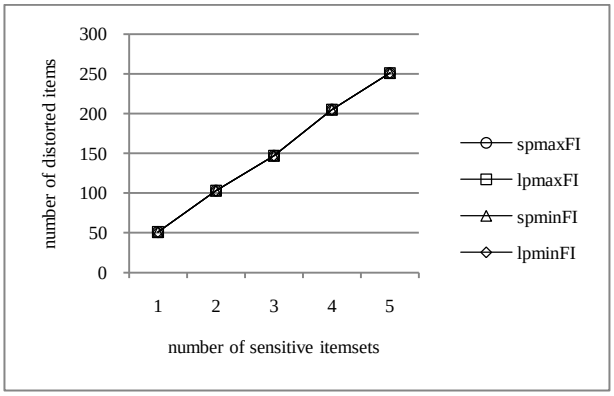

Figure 10. Number of items to distort while increasing number of sensitive itemsets for 10k database

C. Increasing Support of Sensitive Itemset

For both databases five of length three itemsets which have increasing support values between 3.0% and 5.0% are selected as sensitive itemsets. These itemsets are given in Table 2 below. The aim of this study is to understand the effect of increasing the support value of sensitive itemsets on itemset hiding. For each run next itemset in the table is selected as the sensitive itemsets given to program. At first run itemset no 1 is given as sensitive, at second run itemset no 2 is given as sensitive and so on.

Table 2. Sensitive Itemset

Itemset no	Itemsets for 5k database	Support (%)	Itemsets for 10k database	Support (%)
1	37 31 32	2.96%	36 20 6	3.00%
2	18 28 47	3.50%	4 49 42	3.54%
3	14 17 24	4.00%	9 8 3	4.23%
4	28 47 4	4.50%	7 33 18	4.47%
5	46 20 4	5.00%	24 39 13	5.03%

The side effects of increasing support value for sensitive itemset is given in Figure 11 and Figure 12 for 5k and 10k databases. Like it was in first case study selecting shortest

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

pattern has better performance. Selecting shortest pattern and maximum frequent item (spmaxFI) for distortion is the best algorithm to have less number of lost itemsets. The statement "side effect is related to characteristics of selected itemsets" which was written in the first case study is approved in this study. For example, in the 5k database using the strategy spmaxFI, for itemset no 2 the number of lost itemsets is 4 however, for itemset no 4 the number of lost itemsets is 57. One interesting point in the figure is the itemset no 3 for 10k database. This is a good example how pattern selection has effect on the results. Selecting shortest pattern results about 10 lost itemsets while selecting longest pattern results about 300 lost itemsets.

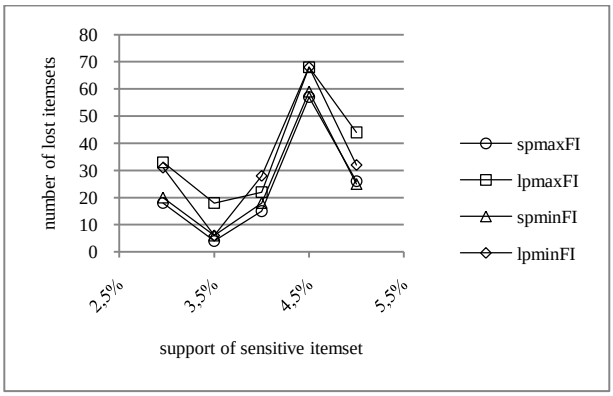

Figure 11. Side effect while increasing support of sensitive itemsets for 5k database

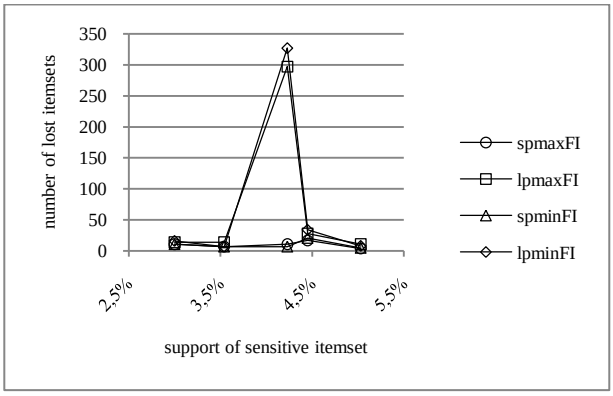

Figure 12. Side effect while increasing support of sensitive itemsets for 10k database

Time cost for itemset hiding is given in Figure 13 and Figure 14 for 5k and 10k databases. Selecting shortest pattern seems as a better method. In addition, spminFI algorithm is slightly faster than spmaxFI algorithm. It is clear from the figure that database size effects time to hide itemsets for same cases. The database size is doubled and like in the first case study time needed for hiding itemsets is increased more than 100%.

The number of distortions is related to support count of sensitive itemsets as it was stated in previous part and so we will have increasing number of distorted items with increasing support.

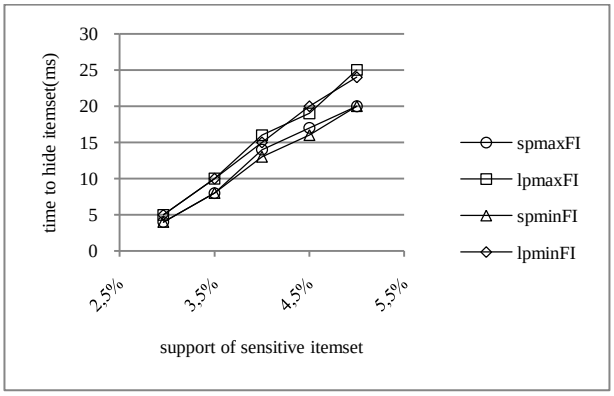

Figure 13. Time to hide itemsets while increasing support of sensitive itemsets for 5k database

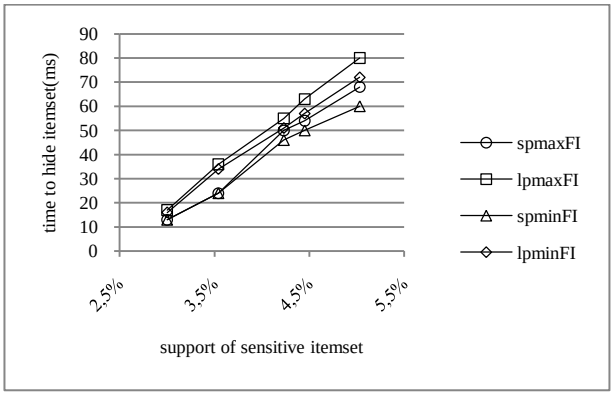

Figure 14. Time to hide itemsets while increasing support of sensitive itemsets for 10k database

D. Discussion on Results

In this section, we analyzed effects of spmaxFI, spminFI, lpmaxFI and lpminFI algorithms on number of lost itemsets, time for hiding process and number of distortions needed for hiding itemsets. We used two different databases to understand the effect of database size and two different set of sensitive itemsets to understand the effects of number of sensitive items and support of sensitive items. If we compare the algorithms, it is clear that spmaxFI algorithm has least side effects at any case. Another point is that selecting shortest pattern causes fewer side effects compared to selecting longest pattern and it needs less time for hiding. Number of distorted items is the same for all algorithms because items are distorted based on the difference between support count of sensitive itemsets and minimum support count no matter which algorithm is used. The most important result from these studies is that side effect is related to characteristics of selected sensitive itemsets because subsets or supersets of that itemset are affected too.

V. CONCLUSION

In this paper we introduced a new algorithm for frequent itemset hiding with four different versions. The algorithm is based on Matrix-Apriori which is an efficient algorithm since it eliminates multiple database scans by using a compact matrix structure as a summary of the original database. Each version uses different heuristic in selecting the transaction and the item in itemset for distortion; spmaxFI (select shortest pattern and maximum of frequent items in the itemset), spminFI (select shortest pattern and minimum of frequent items in the itemset), lpmaxFI (select longest pattern and maximum of frequent items in the

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

itemset) and lpminFI (select longest pattern and minimum of frequent items in the itemset).

Main strengths of the algorithm are i) all versions work without pre-mining so privacy breech caused by the knowledge obtained by finding frequent itemsets in advance is prevented, ii) efficiency is increased since no pre-mining is required, iii) supports are found during hiding process and at the end sanitized database and frequent itemsets of this database are given as outputs so no post-mining is required, iv) simple heuristics are used in transaction and item selection for distortion eliminating the need of extra computational cost.

Performance evaluation study is done on different databases to show the efficiency of the versions of the algorithms while the size of the original database, the number of itemsets and the itemset supports change. The efficiency of four versions are observed as side effects (lost itemsets), time to hide itemsets and amount of distortion caused on the original database. Our findings are as follows; i) among four versions, spmaxFI has better overall performance, ii) the algorithms spmaxFI and spminFI are better in any case than lpmaxFI and lpminFI algorithms, iii) results show that side effect is related to given sensitive itemset, iv) neither support count nor database size is directly related to the number of lost itemsets, v) time to hide sensitive itemset is a function of distortion and database size and vi) distortion is related to support count.

This was a preliminary study to establish our itemset hiding framework and test our itemset hiding algorithm with different heuristics. Our next aim is to compare our promising approach with different itemset hiding algorithms. We plan to carry out further evaluations on different databases, especially those having bigger average transaction lengths, to see the impact of having multiple sensitive itemsets in a single transaction on distortion. Secondly, the effect of the sensitive itemset sanitization order can be observed since in this work we only picked mutually exclusive sensitive itemsets. Finally, we want to adapt this practical itemset hiding algorithm in dynamic environments allowing incremental itemset hiding.

REFERENCES

[1] M. Dunham, Data Mining: Introductory and Advanced Topics. New Jersey: Prentice Hall, 2002.
[2] N. Zhang, W. Zhao, "Privacy-preserving data mining systems," Computer, vol. 40, 2007, pp. 52-58.
[3] R. Grossman, S. Kasif, R. Moore, D. Rocke, J. Ullman, "Data mining research: opportunities and challenges," 1998. [Online]. Avalible: http://pubs.rgrossman.com/dl/misc-001.pdf. [Accessed: May. 31, 2010].
[4] Q. Yang, X. Wu, "10 challenging problems in data mining research," International Journal of Information Technology and Decision Making, vol. 5, 2006, pp. 597-604.
[5] M. Kantardzic, Data Mining: Concepts, Models, Methods, and Algorithms. New Jersey: IEEE Press, 2002.
[6] J. Han, M. Kamber, Data Mining: Concepts and Techniques. San Francisco: Morgan Kaufman, 2006.
[7] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, V. Verykios, "Disclosure limitation of sensitive rules," Proc. 1999 Workshop on Knowledge and Data Engineering Exchange, 1999, pp. 45-52.
[8] S. Oliveira, O. Zaiane, "Privacy preserving frequent itemset mining," Proc2002 IEEE International Conference on Privacy, Security and Data Mining, 2002, pp. 43-54.
[9] V. Verykios, A. Elmagarmid, E. Bertino, Y. Saygin, E. Dasseni, "Association rule hiding," IEEE Transactions on Knowledge and Data Engineering, vol. 16, 2004, pp. 434-447
[10] Y. Saygin, V. Verykios, C. Clifton, "Using unknowns to prevent discovery of association rules," ACM SIGMOD Records, vol. 30, 2001, pp. 45-54
[11] Han, J., Pei, J., Yin, Y.: "Mining frequent patterns without candidate generation," Proc. 2000 ACM SIGMOD international conference on Management of data, 2000, pp. 1-12.
[12] S. Wang, R. Maskey, A. Jafari, T. Hong, "Efficient sanitization of informative association rules." Expert Systems with Applications, vol. 35, 2008, pp. 442-450
[13] H. Huang, X. Wu, R. Relue, "Association analysis with one scan of databases," In: 2002 IEEE International Conference on Data Mining, 2002, pp. 629-632. IEEE Computer Society, Washington
[14] J. Pavon, S. Viana, S. Gomez, "Matrix Apriori: speeding up the search for frequent patterns," Proc. 24th IASTED International Conference on Databases and Applications, 2006, pp. 75-82.
[15] B. Yıldız, B. Ergenç, "Comparison of two association rule mining algorithms without candidate generation," Proc. 10th IASTED International Conference on Artificial Intelligence and Applications, 2010, pp. 450-457.
[16] M. Ahluwalia, A. Gangopadhyay, "Privacy preserving data mining:taxonomy of existing techniques," in Computer Security, Privacy and Politics: Current Issues, Challenges and Solutions, R. Subramanian, Eds. New York: IRM Press, 2008, pp.70-93.
[17] D. Agrawal, C. Aggarwal, "On the design and quantification of privacy preserving data mining algorithms," Proc. 20th ACM SIGMOD SIGACT-SIGART Symposium on Principles of Database Systems, 2001, pp. 247-255.
[18] R. Agrawal, R. Srikant, "Privacy-preserving data mining," ACM SIGMOD Record, vol. 29, 2000, pp. 439-450.
[19] L. Liu, M. Kantarcioglu, B. Thuraisingham, "The applicability of the perturbation based privacy preserving data mining for real-world data," Data and Knowledge Engineering, vol. 65, 2008, pp. 5-21.
[20] Y. Lindell, B. Pinkas, "Privacy preserving data mining." Journal of Crytology, vol. 15, 2002, pp. 177-206.
[21] B. Pinkas, "Cryptographic techniques for privacy-preserving data mining," ACM SIGKDD Explorations, vol. 4, 2002, pp. 12-19.
[22] R. Bayardo, R. Agrawal, "Data privacy through optimal k-anonymization," Proc. 21st international Conference on Data Engineering, 2005, pp. 217-228.
[23] L. Sweeney, "Achieving k-anonymity privacy protection using generalization and suppression," International Journal on Uncertain, Fuzziness Knowledge-Based Systems, vol. 10, 1998, pp. 571-588.
[24] J. Brickell, V. Shmatikov, " The cost of privacy: destriction of data-mining utility in anonymized data publishing," Proc. 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008, pp. 70-78.
[25] V. Verykios, E. Bertino, I. Fovino, L. Provenza, Y. Saygin, Y. Theodoridis, "State-of-the-art in privacy preservng data mining," ACM SIGMOD Record, vol. 33, 2004, pp. 50-57
[26] V. Verykios, A. Gkoulalas-Divanis, "A survey of association rule hiding methods for privacy," in Privacy-Preserving Data Mining: Models and Algorithms, C. Aggarwal and P. Yu, Eds. New York: Springer, 2008, pp. 267-289.
[27] A. Gkoulalas-Divanis, V. Verykios, "An integer programming approach for frequent itemset hiding," Proc. 15th ACM international conference on Information and knowledge management, 2006, pp. 748-757.
[28] A. Gkolalas-Divanis, V. Verykios, "Exact knowledge hiding through database extension," IEEE Transactions on Knowledge and Data Engineering, vol. 21, 2008, pp. 699-713
[29] H. Mannila, H. Toivonen, "Levelwise search and borders of theories in knowledge discovery," Data Mining and Knowledge Discovery, vol. 1, 1997, pp. 241-258.
[30] X. Sun, P. Yu, "a border-based approach for hiding sensitive frequent itemsets," Proc. 5th IEEE International Conference on Data Mining, 2005, pp. 426-433.
[31] X. Sun, P. Yu, "Hiding sensitive frequent itemsets by a border-based approach," Journal of Computing Science and Engineering, vol. 1, 2007, pp. 74-94
[32] G. Mousakides, V. Verykios, "A max min approach for hiding frequent itemsets," Data and Knowledge Engineering 65, 2008, pp. 75-89
[33] R. Agrawal, R. Srikant, "Fast algorithms for mining association rules in large databases," Proc. 20th International Conference on Very Large Data Bases, 1994, pp. 487-499. Morgan Kaufmann, San Francisco
[34] L. Cristofor, "ARtool project," 2002. [Online]. Avalible: http://www.cs.umb.edu/~laur/ARtool/. [Accessed: May. 13, 2010]

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)