Chapter 19
Integrated Approach for Privacy Preserving Itemset Mining

Barış Yıldız and Belgin Ergenç

Abstract In this work, we propose an integrated itemset hiding algorithm that eliminates the need of pre-mining and post-mining and uses a simple heuristic in selecting the itemset and the item in itemset for distortion. Base algorithm (matrix-apriori) works without candidate generation so efficiency is increased. Performance evaluation demonstrates (1) the side effect (lost itemsets) and time while increasing the number of sensitive itemsets and support of itemset and (2) speed up by integrating the post mining.

Keywords Matrix-apriori • Privacy preserving data mining • Sensitive itemset hiding

1 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 [1–6]. 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.

B. Yıldız (✉)
Department of Computer Engineering, Dokuz Eylul University, Buca-35160, Izmir, Turkey
e-mail: barisyildiz@computer.org

B. Ergenç
Department of Computer Engineering, Izmir Institute of Technology, Urla-35430, Izmir, Turkey
e-mail: belginergenc@iyte.edu.tr

S.I. Ao et al. (eds.), *Intelligent Control and Innovative Computing*, Lecture Notes in Electrical Engineering 110, DOI 10.1007/978-1-4614-1695-1_19,

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.

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 [7–10].

Most of the itemset or rule hiding approaches is 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 in finding the rules or itemsets in that data structure [15].

Our initial idea came with the idea of using Matrix-Apriori algorithm in itemset hiding in order to benefit from its advantages in terms of limited database scan and easily maintained compact matrix structure. In our previous work presented in [16], 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 (1) inputs sensitive itemsets, no matter whether they are frequent or not, which prevents privacy breech caused by pre-mining, (2) finds supports during hiding process and at the end returns sanitized database and frequent itemsets of this database as outputs eliminating the need of post-mining on sanitized database and (3) uses simple heuristics in itemset hiding avoiding heavy optimization cost. In this work we extended our study and analyzed the impact of integrating post-mining step in sanitization process.

In [16] a group of case study had been done to show the performance of four versions of our itemset hiding algorithm in terms of side effect, 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 (1) spmaxFI (select shortest pattern and maximum of frequent items) has better overall performance both as side effect and runtime, (2) side effect is related to given sensitive itemset, (3) support count or database size is not directly related to the number of lost itemsets and (4) time to hide sensitive itemset is a function of

distortion and database size and (5) distortion is related to support count. Now, we carried out another group of case study with the most effective version, spmaxFI, to see the impact of integrating post-mining in sanitization process on the performance and discovered that speed up can reach up to 10%.

The structure of the paper is as follows. Next section gives a short survey about association rule hiding and itemset hiding. In Sect. 3, in order to be self-contained itemset hiding process presented in [16] is revisited. Performance evaluation is given with discussion on results in Sect. 4. Then the paper is concluded with final remarks on the study and foreseen future work plan in Sect. 5.

2 Related Work

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 [17]. Input privacy is known as data hiding and output privacy is mostly known as rule or itemset 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. Wide range of data hiding techniques can be grouped as perturbation based techniques [18–20], cryptographic techniques [21, 22] and anonymization based techniques [23–25].

In rule or itemset hiding, sensitive knowledge which can be mined from the database is hidden while non-sensitive knowledge can still be mined [26]. 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 detailed survey about association rule hiding given in [27] where the association rule hiding methods are classified as heuristic, border based and exact approaches. Exact approaches provide completely sanitized database with no side effect but their computational cost is high. In [28, 29] exact techniques which formulate sanitization as constraint satisfaction problem and solve these by integer programming are given. Border based approaches uses border theory [30]. In [31–33] 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 [7, 9].

Another group of algorithms for hiding rules or itemsets can be labeled as fake transactions [34–36]. The idea behind this is to anonymize real transactions by insertion of fake transaction. The approach is practical since any rule or itemset hiding algorithm can be applied on the dataset including fake transactions while maintaining the high theoretical privacy [36]. The work presented in [35] tries to balance the high memory requirement and privacy. A combination of fake transaction randomization method and a new per-transaction randomization method is proposed in [34].

Most of the association rule hiding algorithms are Apriori [37] 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.

In the framework presented 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 replacing false values may increase side effects on non-sensitive rules so the algorithms use unknown values to hide given sensitive rules.

Our survey on rule or itemset hiding research motivated us to propose a hiding approach which eliminates the need of pre-mining, avoids multiple scans of database and heavy computational cost during hiding process. We used matrix-apriori algorithm [14] as the basis of our sanitization framework. It works without candidate generation and scans database only twice. It has a simpler data structure and performs better compared to FP-growth in [15]. The sanitization process is embedded into the itemset mining process and at the end frequent itemsets of sanitized database is available. In the following section our algorithm is introduced.

3 Integrated Sanitizatıon Framework for Itemset Hiding

As displayed in Fig. 19.1, 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 (1) it

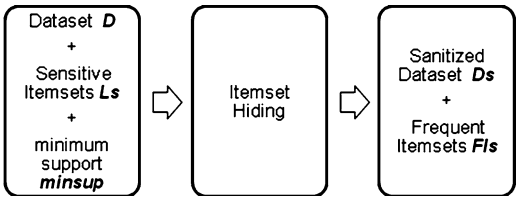

Fig. 19.1 Sanitization framework

```
INPUT: Original Database D, minimum support minsup, List of sensitive itemsets Ls

OUTPUT: Sanitized Database Ds, Frequent itemsets Fls 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

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

Fig. 19.2 Itemset hiding algorithm

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 (2) 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.

The overall process of itemset hiding algorithm is shown in Fig. 19.2. 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 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 (lines 1 to 15).

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). In [16] we used different heuristics in selecting the itemset and the item in itemset to distort and we discovered that selecting the shortest itemset and the most frequent item in that itemset causes minimum side effect. So in this work, we based our performance evaluation on spmaxMFI version. 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 (spmaxFI) stragety is applied and itemset of BA is assumed to be sensitive (Is). In Fig. 19.3 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.

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 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

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

MFI		
A	B	C
2	2	2
3	3	5
4	1	0
1	0	0
0	0	1

STE
3
1
1
1

TIDs
T1,T2,T3
T4
T5
T6

Fig. 19.3 Database D, MFI, STE and TList before hiding process

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

MFI		
A	B	C
2	2	2
3	3	5
4	6	0
1	0	0
0	0	7
0	7	0
0	1	1

STE
2
0
1
1
1
1

TIDs
T1,T2
T5
T6
T4
T3

Fig. 19.4 Sanitized database Ds, MFI, STE and TList after hiding process

selected transaction (Fig. 19.4). 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 Fig. 19.4 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.

4 Performance Evaluation

In this section, performance evaluation of our itemset hiding algorithm is given. Firstly, the performance of algorithm in two different cases is given. Secondly, impact of integrating post-mining on runtime is observed. Two synthetic databases are used to see effect of different database size. The algorithm is executed on databases (1) to see effect of increasing number of sensitive itemsets, (2) to see effect of increasing support of sensitive itemset. The effects observed are number of lost itemsets as side effect, runtime for hiding process and number of items distorted for hiding itemsets.

Table 19.1 Sensitive itemsets for case 1

Itemset No	Itemsets for 5 k database	Support (%)	Itemsets for 10 k 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

4.1 Simulation Environment

Test runs are performed on a computer with 2.4 GHz dual core processor and 3 GB memory. During evaluations, it is ensured that the system state is similar for all test runs and results are checked for consistency.

When hiding is applied inputs are original database and sensitive itemsets where the outputs are sanitized database and frequent itemsets which can be mined from this sanitized database. Two synthetic databases generated by ARtool [38] are used in evaluations. One database has 5,000 transactions while number of items is 50 and average length of transactions is 5. Other database has 10,000 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 2,714 frequent itemsets from 5 k database and 5,527 frequent itemsets from 10 k database can be found.

4.2 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 19.1. 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.

The side effect and time to hide is given in Fig. 19.5. In both databases number of lost itemsets is increased while number of sensitive itemsets is increased. What more can be inferred from these figure is that side effect is related to the characteristics of sensitive itemsets, not to the database size. For instance, we come across higher number of lost itemsets for 5 k database(29 itemsets) compared to 10 k database(22 itemsets) at 5 itemset hiding point.

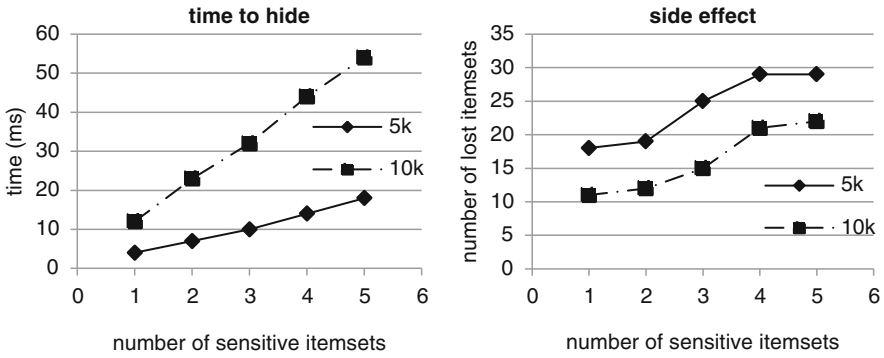

Fig. 19.5 Side effect and time to hide while increasing number of sensitive itemsets

Table 19.2 Sensitive itemsets for case 2

Itemset No	Itemsets for 5 k database	Support (%)	Itemsets for 10 k 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%

When we came to time cost of hiding we give pure hiding cost excluding the reading database and building up matrix to make observation simple. These are identical for the same databases. It is clear from the figure that database size effects time to hide itemsets for same cases. While the database size increases, time needed for hiding itemsets increases. The reason behind this is the cost of travelling on matrix to select pattern. It is clear that matrix size is bigger for 10 k database compared to 5 k database.

4.3 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 19.2. 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.

The side effect and time cost of increasing support value for sensitive itemset is given in Fig. 19.6. 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

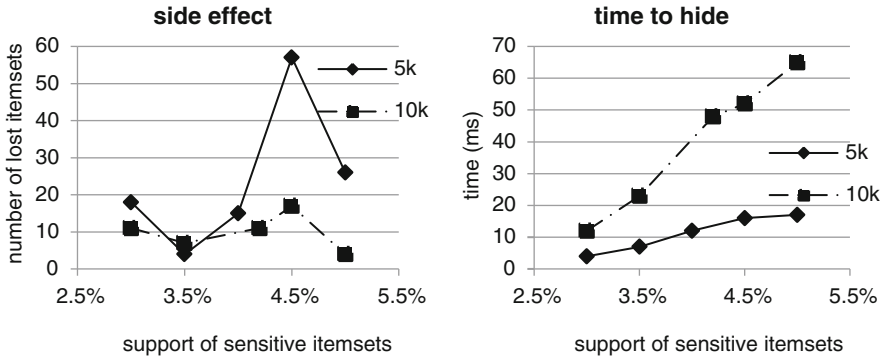

Fig. 19.6 Side effect and time to hide while increasing support of sensitive itemset

example, in the 5 k database for itemset no 1 the number of lost itemsets is 18 however, for itemset no 1 the number of lost itemsets is 4. Although the support is increased number of lost itemsets is decreased.

It is clear from the figure that database size effects time to hide itemsets for same cases. Like it was in first case study only pure hiding process is considered.

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.

4.4 Integrating Post-mining into Itemset Hiding

The time efficiency of integrated post-mining is observed for both two case studies above. Firstly, the program is executed without post-mining. This includes reading database, building matrix and pure hiding process. Secondly, the sanitized database is given to the pure matrix-apriori for frequent itemset mining. The running time of these two steps are added and time cost without integration is calculated. Afterwards, the program is executed with integrated post-mining. The difference between first calculation and integrated approach gives the gain. In Fig. 19.7 the speed up by integrating post-mining into itemset hiding is given for two cases mentioned in previous parts.

The evaluations reveal that integrating frequent itemset mining into hiding is always faster choice compared to do hiding and mining separated. For both cases on 5 k database the speed up of integrated approach is at least 2.1%, at most 6.9% and in average 5.4%. The gain increases when database size is increased. For 10 k database integrated approach is 9.2% faster than separated approach in average. The speed up is at least 7.8% and at most 10.3%.

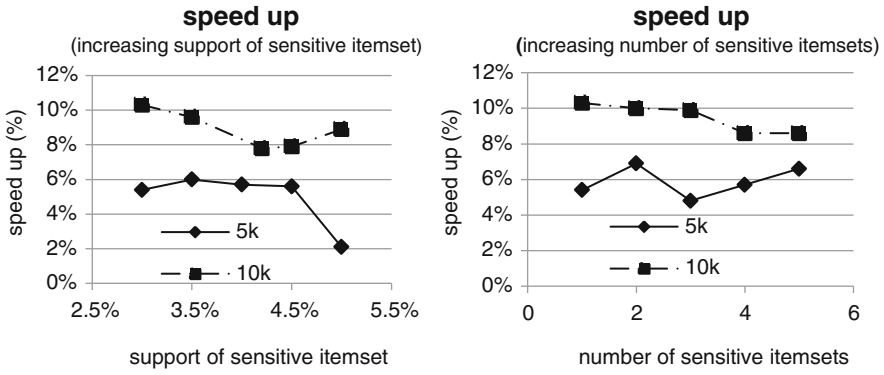

Fig. 19.7 The speed up of integrated post-mining

4.5 Discussion on Results

In this section, we analyzed effects of itemset hiding algorithm on number of lost itemsets, time for hiding process and number of distortions needed for hiding itemsets and time efficiency on post-mining. 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 and support of sensitive itemsets. One of the results 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. Other important result is that integrating post-mining approach always performs faster than separated approach. The difference between 5 k and 10 k databases (5.4% and 9.2%) show that with bigger databases efficiency increases. Larger databases are likely to have more variety of patterns so larger matrix structures. Time for reading database and building up matrix are increased when database size is increased. Building this structure takes more time. The main aim of integrated approach is to eliminate time loss of reading database and rebuilding matrix structure. Our evaluations proved that the integrated approach is always faster than separated approach.

5 Conclusion and Future Work

In this paper we introduced an integrated approach for frequent itemset hiding. Main strengths of the algorithm are (1) it works without pre-mining so privacy breech caused by the knowledge obtained by finding frequent itemsets in advance is prevented, (2) efficiency is increased since no pre-mining is required, (3) 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, (4)

simple heuristic is 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 algorithm in terms of side effect (lost itemsets) and runtime while the size of the original database, the number of sensitive itemsets and the itemset supports change. Our findings are as follows: (1) side effect is related to given sensitive itemset, (2) neither support count nor database size is directly related to the number of lost itemsets, (3) time to hide sensitive itemset is a function of distortion and database size and (4) integration of post-mining brings speed up of 10%.

Our next aim is to compare our promising approach with different hiding algorithms and 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. Finally, we want to adapt this practical itemset hiding algorithm in dynamic environments allowing incremental itemset hiding.

References

1. Dunham M (2002) Data mining: introductory and advanced topics. Prentice Hall PTR Upper Saddle River, NJ, USA
2. Zhang N, Zhao W (2007) Privacy-preserving data mining systems. Computer 40(2):52–58
3. Grossman R, Kasif S, Moore R, Rocke D, Ullman J Data mining research: opportunities and challenges [Online]. http://pubs.rgrossman.com/dl/misc-001.pdf. Accessed on May 31, 2010
4. Yang Q, Wu X (2006) 10 challenging problems in data mining research. Int J Inform Technol Decision Making 5(4):597–604
5. Kantardzic M (2002) Data mining: concepts, models, methods, and algorithms. John Wiley & Sons, Inc. New York, NY, USA
6. Han J, Kamber M (2005) Data mining: concepts and techniques. Morgan Kaufman, Publishers Inc. San Francisco, CA, USA
7. Atallah M, Bertino E, Elmagarmid A, Ibrahim M, Verykios V (1999) Disclosure limitation of sensitive rules. In: Proceedings of 1999 workshop on knowledge and data engineering exchange, KDEX '99, Chicago, 7 November 1999
8. Oliveira S, Zaiane O (2002) Privacy preserving frequent itemset mining, Proceedings of 2nd IEEE international conference on data mining, ICDM'02, Maebashi City, 9–12 December 2002
9. Verykios V, Elmagarmid A, Bertino E, Saygin Y, Dasseni E (2004) Association rule hiding. IEEE Trans Knowledge Data Eng 16(4):434–447
10. Saygin Y, Verykios V, Clifton C (2001) Using unknowns to prevent discovery of association rules. ACM SIGMOD Records 30(4):45–54
11. Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. ACM SIGMOD Records 29(2):1–12
12. Wang S, Maskey R, Jafari A, Hong T (2008) Efficient sanitization of informative association rules. Exp Syst Appl 35(1–2):442–450
13. Huang H, Wu X, Relue R (2002) Association analysis with one scan of databases. Proceedings second IEEE international conference on data mining, ICDM'02, Maebashi City, 9–12 December 2002, pp 629–632
14. Pavon J, Viana S, Gomez S (2006) Matrix apriori: speeding up the search for frequent patterns. Proceedings 24th IASTED international conference on databases and applications, DBA 2006, Innsbruck, 14–16 February 2006, pp 75–82

15. Yıldız B, Ergenç B (2010) Comparison of two association rule mining algorithms without candidate generation. In: Proceedings 10th IASTED international conference on artificial intelligence and applications, AIA 2010, Innsbruck, 15–17 February 2010, pp 450–457
16. Yıldız B, Ergenç B (2011) Hiding sensitive predictive frequent itemsets, lecture notes in engineering and computer science. In: Proceedings of the international multiconference of engineers and computer scientists 2011, IMECS 2011, Hong Kong, 16–18 March 2011, pp 339–345
17. Ahluwalia M, Gangopadhyay A (2008) Privacy 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
18. Agrawal D, Aggarwal C (2001) On the design and quantification of privacy preserving data mining algorithms. Proceedings 20th ACM SIGMOD SIGACT-SIGART symposium on principles of database systems, PODS'01, CA, 21–24 May 2001, pp 247–255
19. Agrawal R, Srikant R (2000) Privacy-preserving data mining. ACM SIGMOD Records 29(2):439–450
20. Liu L, Kantarcioglu M, Thuraisingham B (2008) The applicability of the perturbation based privacy preserving data mining for real-world data. Data Knowledge Eng 65(1):5–21
21. Lindell Y, Pinkas B (2002) Privacy preserving data mining. J Crytol 15(3):177–206
22. Pinkas B (2006) Cryptographic techniques for privacy-preserving data mining. ACM SIGKDD Explorations Newslett 4(2):12–19
23. Bayardo R, Agrawal R (2005) Data privacy through optimal k-anonymization. Proceedings of 21st international conference on data engineering, ICDE'05, Tokyo, 5–8 April 2005, pp 217–228
24. Sweeney L (2002) Achieving k-anonymity privacy protection using generalization and suppression. Int J Uncertain, Fuzziness Knowledge-Based Syst 10(5):571–588
25. Brickell J, Shmatikov V (2008) The cost of privacy: destriction of data-mining utility in anonymized data publishing. Proceedings of 14th ACM SIGKDD international conference on knowledge discovery and data mining, KDD'08, Las Vegas, 24–27 August 2008, pp 70–78
26. Verykios V, Bertino E, Fovino I, Provenza L, Saygin Y, Theodoridis Y (2004) State-of-the-art in privacy preserving data mining. ACM SIGMOD Records 33(1):50–57
27. Verykios V, Gkoulalas-Divanis A (2008) A survey of association rule hiding methods for privacy. In: Aggarwal C, Yu P (ed) Privacy-preserving data mining: models and algorithms. Springer, New York, pp 267–289
28. Gkoulalas-Divanis A, Verykios V (2006) An integer programming approach for frequent itemset hiding. Proceedings of 15th ACM international conference on Information and knowledge management, CIKM'06, Virginia, 5–11 November 2006, pp 748–757
29. Gkolalas-Divanis A, Verykios V (2008) Exact knowledge hiding through database extension. IEEE Trans Knowledge Data Eng 21(5):699–713
30. Mannila H, Toivonen H (1997) Levelwise search and borders of theories in knowledge discovery. Data Mining Knowledge Discov 1(3):241–258
31. Sun X, Yu P (2005) A border-based approach for hiding sensitive frequent itemsets. Proceedings of 5th IEEE international conference on data mining, ICDM'05, Houston, 27–30 November 2005, pp 426–433
32. Sun X, Yu P (2007) Hiding sensitive frequent itemsets by a border-based approach. J Comput Sci Eng 1(1):74–94
33. Mousakides G, Verykios V (2008) A max min approach for hiding frequent itemsets. Data Knowledge Eng 65(1):75–89
34. Boora RK, Shukla R, Misra AK (2009) An improved approach to high level privacy preserving itemset mining. Int J Comput Sci Inform Security 6(3):216–223
35. Mohaisen A, Jho N, Hong D, Nyang D (2010) Privacy preserving association rule mining revisited: privacy enchancement and resource efficiency. IEICE Trans Inform Syst E93(2): 315–325

36. Lin JL, Liu JYC (2007) Privacy preserving itemset mining through fake transactions. Proceedings of 22nd ACM symposium on applied computing, SAC 2007, Seoul, 11–15 March 2007, pp 375–379
37. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules in large databases. Proceedings of 20th international conference on very large data bases, VLDB'94, Santiago de Chile, 12–15 September 1994, pp 487–499
38. Cristofor L artool project [Online]. http://www.cs.umb.edu/~laur/ARtool/. Accessed on May 13, 2010