One-Scan Sanitization of Collaborative Recommendation Association Rules
協同推薦關聯規則之一次掃瞄清除

Shyue-Liang Wang
Department of Information Management
National University of Kaohsiung, Taiwan
slwang@nuk.edu.tw

Tzung-Pei Hong
Department of Electrical Engineering
National University of Kaohsiung, Taiwan
tphong@nuk.edu.tw

Abstract

For a given recommended item, a collaborative recommendation association rule set is the smallest association rule set that makes the same recommendation as the entire association rule set by confidence priority. In this work, we propose an efficient one-scan sanitization algorithm to hide collaborative recommendation association rules. To hide association rules, previously proposed algorithms based on Apriori approach usually require multiple scanning of database to calculate the supports of the large itemsets. We propose here using a pattern-inversion tree to store related information so that only one scan of database is required. Numerical experiments show that the proposed algorithm out performs previous algorithms, with similar side effects.

摘要

給定一個推薦項目，一個協同推薦關聯規則集是一組根據信賴值排序之最小關聯規則集，且具備相同之推薦結果。在本文中，我們提出一個有效率之一次掃瞄清除演算法以隱藏協同推薦關聯規則集。一般隱藏關聯規則之演算法皆須做多次之資料庫掃瞄。我們則提出一個利用式樣反轉樹(pattern-inverse tree)儲存相關資料並且只須掃瞄資料庫一次之演算法。數值實驗顯示我們所提之演算法比其他方法更有效率且具有類似之副作用。

Keywords: privacy preserving, data mining, collaborative recommendation, association rule

關鍵字: 隱私保護、資料探勘、協同推薦、關聯規則

1. Introduction

Privacy-preserving data mining, is a novel research direction in data mining and statistical databases, where data mining algorithms are analyzed for the side effects they incur in data privacy [25]. For example, through data mining, one is able to infer sensitive information, including personal information or even patterns, from non-sensitive information or unclassified data. There have been two types of privacy concerning data mining. The first type of privacy, called output privacy, is that the data is minimally altered so that the mining result will not disclose certain privacy. Many techniques have been proposed for the output privacy [1,6,7,8,9,20,21,23,25]. For example, perturbation, blocking, aggregation or merging, swapping, and sampling are some alternation methods that have recently been proposed. The second type of privacy, called input privacy, is that the data is manipulated so that the mining result is not affected or minimally affected [10,11,12,15,24]. For example, the reconstruction-based technique tries to randomize the data so that the original distribution or patterns of the data can be reconstructed. The cryptography-based techniques like secure multiparty computation allow users access to only a subset of data while global data mining results can still be discovered.

In output privacy, given specific rules or patterns to be hidden, many data altering techniques for hiding association, classification and clustering rules have been proposed. For association rules hiding, two basic approaches have been proposed. The first approach [9, 23, 26] hides one rule at a time. It first selects transactions that contain the items in a give rule. It then tries to modify transaction by transaction until the confidence or support of the rule fall below minimum confidence or minimum support. The modification is done by either removing items from the transaction or inserting new items to the transactions. The second approach [19-21] deals with groups of restricted patterns or association rules at a time. It first selects the transactions that contain the intersecting patterns of a group of restricted patterns. Depending on the disclosure threshold given by users, it sanitizes a percentage of the selected transactions in order to hide the restricted patterns.

However, all these approaches require hidden rules or patterns been given in advance. This selection of rules would require data mining process to be executed first. Based on the discovered rules and privacy requirements, hidden rules or patterns are then selected manually. But, for some applications, we are only interested in hiding certain constrained classes of association rules such as collaborative recommendation association rule sets [27], which can be used for recommendation using less number of association rules. To hide such rule sets, the pre-process of finding these hidden rules can be integrated into the hiding process as long as the recommended items are given. In addition, these approaches, all based on Apriori algorithm, require multiple scanning of database to calculate the supports of the large itemsets. In this work, we propose using a pattern-inversion tree to store related information so that only one scan of database is required. Examples illustrating the proposed algorithm are given. Numerical experiments show that the proposed algorithm out performs previous multiple scanning algorithms, with similar side effects.

The rest of the paper is organized as follows. Section 2 presents the statement of the problem and the notation used in the paper. Section 3 presents the proposed algorithm for sanitizing collaborative recommendation association rule sets that contain the specified recommended items. Section 4 shows an example of the proposed algorithm. Section 5 shows the experimental results of the performance and various side effects of the proposed algorithm compared with multiple-scanning algorithm. Concluding remarks and future works are described in section 6.

2. Problem statement

2.1. Collaborative recommendation association rules

The problem of mining association rules was introduced in [2]. Let $I = \{ i_1,\ i_2, \cdots,\ i_m \}$ be a set of literals, called items. Given a set of transactions *D*, where each transaction T in *D* is a set of items such that $T \subseteq I$, an association rule is an expression $X \Rightarrow Y$ where $X \subseteq I$, $Y \subseteq I$, and $X \cap Y = \phi$. The X and Y are called respectively the body (left hand side) and head (right hand side) of the rule. An example of such a rule is that 90% of customers buy hamburgers also buy Coke. The 90% here is called the confidence of the rule, which means that 90% of transaction that contains *X* (hamburgers) also contains *Y* (Coke). The confidence is calculated as $| X \cup Y | / | X |$, where $|X|$ is the number of transactions containing *X* and $|X \cup Y|$ is the number of transactions containing both *X* and *Y*. The support of the rule is the percentage of transactions that contain both *X* and *Y*, which is calculated as $| X \cup Y | / N$, where *N* is the number of transactions in *D*. The problem of mining association rules is to find all rules that are greater than the user-specified minimum support and minimum confidence.

As an example, for a given database in Table 1, a minimum support of 33% and a minimum confidence of 70%, nine association rules can be found as follows: *B*=>*A* (66%, 100%), *C*=>*A* (66%, 100%), *B*=>*C* (50%, 75%), *C*=>*B* (50%, 75%), *AB*=>*C* (50%, 75%), *AC*=>*B* (50%, 75%), *BC*=>*A*(50%, 100%), *C*=>*AB*(50%, 75%), *B*=>*AC*(50%, 75%), where the percentages inside the parentheses are supports and confidences respectively.

However, mining association rules usually generates a large number of rules, most of which are unnecessary for the purpose of collaborative recommendation. For example, to recommend a target item *{C}* to a customer, the collaborative recommendation association rule set that contains only two rules, *B*=>*C* (50%, 60%) and *AB*=>*C* (50%, 60%), will generate the same recommendations as the entire nine association rules found from Table 1. This means that if the new customer has shown interests in purchasing item *{B}* or items *{AB}*, then the collaborative recommender will recommend the new customer to purchase target item *{C}*. Therefore, a collaborative recommendation association rule set can be informally defined as the smallest association rule set that makes the same recommendation as the entire association rule set by confidence priority.

Table 1: Database *D*

TID	Items
T_1	ABC
T_2	ABC
T_3	ABC
T_4	AB
T_5	A
T_6	AC

Formally, given an association rule set *R* and a target itemset *P*, we say that the collaborative recommendation for *P* from *R* is a sequence of items *Q*. Using the rules in *R* in descending order of confidence generates the sequence of *Q*. For each rule *r* that matches *P* (i.e., for each rule whose conclusion is a subset of *P*), each antecedent of *r* is added to *Q*. After adding a consequence to *Q*, all rules whose antecedents are in *Q* are removed from *R*.

The following is the definition of collaborative recommendation association rule set:

Definition Let R_A be an association rule set and R_A^1 the set of single-target rules in R_A. A set R_c is a collaborative recommender over R_A if (1) $R_c \subset R_A^1$, (2) $\forall\ r \in R_c$, there does not exist $r' \in R_c$ such that $r' \subset r$ and $conf(r') > conf(r)$, and (3) $\forall\ r'' \in R_A^1 - R_c$, $\exists\ r \in R_c$ such that $r'' \supset r$ and $conf(r'') \leq conf(r)$.

2.2. Problem description

The objective of data mining is to extract hidden or potentially unknown but interesting rules or patterns from databases. However, the objective of privacy preserving data mining is to hide certain sensitive information so that they cannot be discovered through data mining techniques [1,4-12,16]. In this work, we assume that only recommended (also called target or hidden) items are given and propose an algorithm to modify data in database so that collaborative recommendation association rule sets cannot be inferred through association rule mining. More specifically, given a transaction database *D*, a minimum support, a minimum confidence and a set of recommended items *Y*, the objective is to minimally modify the database *D* such that no collaborative recommendation association rules containing *Y* on the right hand side of the rule will be discovered.

As an example, for a given database in Table 1, a minimum support of 33%, a minimum confidence of 70%, and a hidden item *Y* = {*C*}, if transaction T_1 is modified from *ABC* to *AB*, then the following rules that contain item *C* on the right hand side will be hidden: *B*=>*C* (33%, 50%), *AB*=>*C* (33%, 50%). However, four rules are lost, *C*=>*B* (33%, 60%), *AC*=>*B* (33%, 66%), *C*=>*AB* (33%, 66%), *B*=>*AC* (33%, 50%), and no new rule is generated as side effects.

3. Proposed algorithm

In order to hide an association rule, *X*=>*Y*, we can either decrease its supports, ($|X|/N$ or $|X \cup Y|/N$), to be smaller than pre-specified minimum support or its confidence ($|X \cup Y|/|X|$) to be smaller than pre-specified minimum confidence. To decrease the confidence of a rule, two strategies can be considered. The first strategy is to increase the support count of *X*, i.e., the left hand side of the rule, but not support count of $X \cup Y$. The second strategy is to decrease the support count of the itemset $X \cup Y$. For the second strategy, there are in fact two options. One option is to lower he support count of the itemset $X \cup Y$ so that it is smaller than pre-defined minimum support count. The other option is to lower the support count of the itemset $X \cup Y$ so that $|X \cup Y|/|X|$ is smaller than pre-defined minimum confidence. In addition, in the transactions containing both *X* and *Y*, if we decrease the support of *Y* only, the right hand side of the rule, it would reduce the confidence faster than reducing the support of *X*. In fact, we can pre-calculate the number of transactions required to hide the rule. If there are not enough transactions exist, then the rule cannot be hidden. To decrease support count of an item, we will remove one item at a time in a selected transaction by changing from 1 to 0. To increase the support count of an item, we will add one item by changing from 0 to 1.

In order to hide collaborative recommendation association rules, we will consider hiding association rules with 2 items, *x=>z*, where *z* is a recommended item and *x* is a single large one item. In theory, collaborative recommendation association rules may have more specific rules that contain more items, e.g., *xY => z*, where *Y* is a large itemset. However, for such rule to exist, its confidence must be greater than the confidence of *x=>z*, i.e., *conf(xY=>z) > conf(x=>z)* or |*xYz*| > *conf(x=>z)* * |*xY*|. For higher confidence rules, such as *conf(x=>z) = 1*, there will be no more specific rules. In addition, once the more general rule is hidden, the more specific rule might be hidden as well.

To reduce the number of database scanning in generating large or frequent itemsets in association rule mining, a one scan of database method was proposed in [14]. The basic idea is to construct a tree structure, called Pattern tree (P-tree) and a frequency list which contains the frequency counts of each item in one database scan. The pattern tree is then restructured to a compact Frequent-Pattern tree (FP-tree) proposed by [13], which can store more information in less space. A FP-tree based pattern growth method (FP-growth) is used to find all the large tiemsets from the FP-tree.

In this work, we propose a Pattern-Inversion tree (PI-tree) based on the pattern tree. A PI-tree is similar to a P-tree except the following. Each node in a PI-tree contains three fields: item name (or item number), number of transactions containing the items on the path from the root to current node, and list of transaction ID that contains all the items on the path from the root to current node. For example, the PI-tree for the six transactions in Table 1 is shown in Figure 1. The frequency list is L = <(A:6), (B:4), (C:4)>.

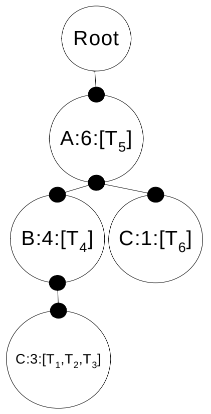

Figure 1 Pattern Inversion Tree for data in Table 1

The construction of a PI-tree contains two steps. In step one, we read transactions one by one from database. Each transaction is sorted according to item name (or item number) and inserted into the PI-tree. The frequency list is updated accordingly. In the second step, we sort the frequency list according to item support counts first. Then the PI-tree is restructured similar to the first step. Each branch of the original PI-tree is sorted according to the new frequency list and inserted to the restructured PI-tree.

Based on the strategies mentioned above, we propose a data-mining algorithm for sanitizing collaborative recommendation association rules, namely **D**ecrease **C**onfidence **B**y **S**upport (**DCBS**). The detail of the algorithm is described as follow.

Algorithm DCBS

```
Input:    (1) a source database D,
          (2) a min_support,
          (3) a min_confidence,
          (4) a set of hidden items Y
Output: a sanitized database D', where rules containing
          items of Y on RHS will be hidden

Step 1: Build Pattern Inversion Tree (PI-tree) and
          frequency list L
1.1     Construct an initial PI-tree and item frequency list L
          For each transaction t in D
              Sort t according to item name (or item #)
              Insert t into PI-tree
              Update L with items in t
1.2     Restructure the initial PI-tree
          For each path p' from root to leaf
              Set support count of each node of p' to common support count;
              Sort p' according to L;
              Insert p' to a new PI-tree
Step 2: Sanitize all rules of the form: $U: x \rightarrow y$
2.1     For each $y \in Y$ and y is a frequent item {
          For each large 2-itemset containing x { // e.g. {xy} and rule $U: x \rightarrow y$
              If confidence(U) >= min_conf, then {
                 //# of transactions required to lower the confidence
                 iterNumConf = $\lceil |D|*(supp(xy) - min\_conf * supp(x)) \rceil$;
                 //# of transactions required to lower the support
                 iterNumSupp = $\lceil |D|*(supp(xy) - min\_supp) \rceil$;
                 k = minimum (iterNumConf, iterNumSupp);
                 Sanitize the shortest k transactions containing xy by updating PI-tree, frequency list and D;
              }; //end if
          }; // end for each large 2-itemset
          remove y from Y;
        }; // end for each $y \in Y$
Step 3: Output updated database D';
```

4. Example

This section shows an example to demonstrate the proposed algorithm in sanitizing collaborative recommendation association rule sets. For a given database *D* in Table 2, a minimum support of 33% and a minimum confidence of 70%, the result of running the **DCBS** algorithm is given as follow.

In step one, a Pattern Inversion Tree and a frequency list *L* is built. Transactions are read one by one and the corresponding PI-trees and frequency lists are as follows. After reading T_1, root – A:1 – B:1 – C:1:[T_1], L = <(A:1), (B:1), (C:1)>. After reading T_2, root – A:2 – B:2 – C:2:[T_1, T_2], L = <(A:2), (B:2), (C:2)>. After reading T_3, root– A:3 – B:3 – C:3:[T_1, T_2, T_3], L = <(A:3), (B:3), (C:3)>. After reading T_4, root – A:4 – B:4:[T_4] – C:3:[T_1, T_2, T_3], L = <(A:4), (B:4), (C:3)>. After reading T_5, root – A:5:[T_5] – B:4:[T_4] – C:3:[T_1, T_2, T_3], L= <(A:5), (B:4), (C:3)>. After reading T_6, the PI-tree is shown in Figure 1and L = <(A:6), (B:4), (C:4)>. The restructured PI-tree is the same as the original PI-tree.

Table 2: Databases before and after sanitization

TID	*D*	*D*'
T_1	111	11$\underline{0}$
T_2	111	111
T_3	111	111
T_4	110	110
T_5	100	100
T_6	101	101

In step 2, we sanitize all rules of the form, $U: x \rightarrow y$. For hidden item *C*, which is a frequent item, we find all large two itemsets that contain *C*, which includes {*AC*, *BC*}. For *AC*, the *A=>C* (66%, 66%) is not an association rule. For *BC*, the estimated number of transaction need to be sanitized is calculated as follows. The iterNumConf = $\lceil 3\text{-}70\%*4 \rceil = \lceil 3 - 2.8 \rceil = \lceil 0.2 \rceil = 1$, iterNumSupp = $\lceil 3\text{-}33\%*6 \rceil = \lceil 3 - 1.8 \rceil = \lceil 1.2 \rceil = 2$, and k = min [1, 2] = 1. Therefore, the shortest transaction containing *BC*, T_1 is sanitized. The resulting PI-tree has two paths from the root, root – A:6:[T_5] – B:4:[T_1, T_4] – C:2:[T_2, T_3], and root – A:6:[T_5] – C:1:[T_6], and L = <(A:4), (B:4), (C:3)>.

The resulting database is shown as *D*' in Table 2. Two rules are hidden, *B*=>*C* (33%, 50%), *AB*=>*C* (33%, 50%), four rules are lost, *C*=>*B* (33%, 60%), *AC*=>*B* (33%, 66%), *C=>AB* (33%, 66%), *B=>AC* (33%, 50%), and no new rule is generated as side effects.

5. Numerical experiments

In order to better understand the characteristics of the proposed algorithm numerically, we perform a series of experiments to measure various effects. The following effects are considered: time effects, database effects, side effects, and item ordering effects. For time effects, we measure the running time required to hide one and two recommended items, i.e., one and two collaborative recommendation association rule sets respectively. The database effects measure the percentage of altered transactions in the database. For side effects, we measure the hiding failures, new rules generated and lost rules. The hiding failure side effect measures the number of collaborative recommendation association rules that cannot be hidden. The new rule side effect measures the number of new rules appeared in the transformed database but is not in the original database. The lost rule side effect measures the number of rules that are in the original database but not in the transformed database. The item order effects examine the previous effects when the order of recommended item is changed.

The experiments are performed on a PC with Pentium four 498 MHz processor and 128 MB RAM running on Windows XP operating system. The data sets used are generated from IBM synthetic data generator. The sizes of the data sets range from 5K to 25K transactions with average transaction length, |ATL| = 5, and total number of items, |I| = 50

For each data set, various sets of association rules are generated under various minimum supports and minimum confidences. The minimum support range tested is from 0.5% to 10%. The minimum confidence range tested is from 20% to 40%. Total number of association rules is from 6 to 404. The number of hidden rules ranges from 0 to 73, which in percentage over total association rules is from 0% to 66%. The number of recommended items considered here are one and two items. For the following results, the minimum support and minimum confidence used are 3% of 20% respectively. The total numbers of association rules are 150, 177, 177, 175, 176 for 5k to 25k data sizes respectively. The percentage of hidden rule over association rules is about 8% and 17% for one and two recommended items respectively.

Figure 1 shows the time required for multiple-scan DCDS [27] and one-scan DCBS algorithms to hide collaborative recommendation association rule sets for one and two recommended items. The processing times increase in proportion to the sizes of the data sets and the number of recommended items for both algorithms. However, the one-scan DCBS requires less processing time than the multiple-scan DCDS algorithm. Noted that the results of multiple-scan DCDS algorithm are performed under faster platform of 3 GHz processor and 256 MB RAM. Figure 2 shows the percentages of transaction altered for multiple-scan DCDS and one-scan DCBS algorithms. The percentages of altered transactions remain constant for both algorithms as well as one and two items respectively. Even though DCBS algorithm modifies more transactions, it is still faster than DCDS algorithm. Figure 3 and 4 show the various side effects of multiple-scan DCDS and one-scan DCBS algorithms for hiding two recommended items. For multiple-scan DCDS, there is no hiding failure, meaning every rule can be hidden. There is about 1% or less of new rules generated and about 4% of association rules are lost on the average. For one-scan DCBS, there is at most 9% hiding failures and at most 11% new rules but no lost rules.

6. Conclusion

In this work, we have studied the database privacy problems caused by data mining technology and proposed an efficient one-scan algorithm for sanitizing collaborative recommendation association rule sets. The proposed algorithm can automatically sanitize collaborative recommendation rule sets without pre-mining and selection of a class of rules under one database scan. Example illustrating the proposed algorithm is given. Numerical experiments are performed to show the time effects, database effects, and side effects of the algorithm and compared with multiple-scan algorithm. It can be seen that the proposed one-scan algorithm out performs the multiple scan algorithm in processing time with similar side effects. In the future, we will consider the problem of efficient maintenance of privacy when databases are updated frequently.

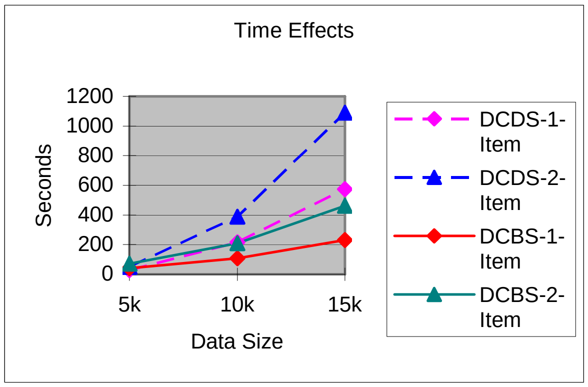

Figure 1 Time Effects

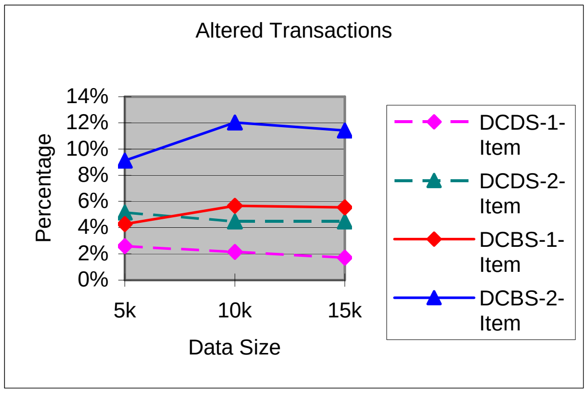

Figure 2 Database Effects

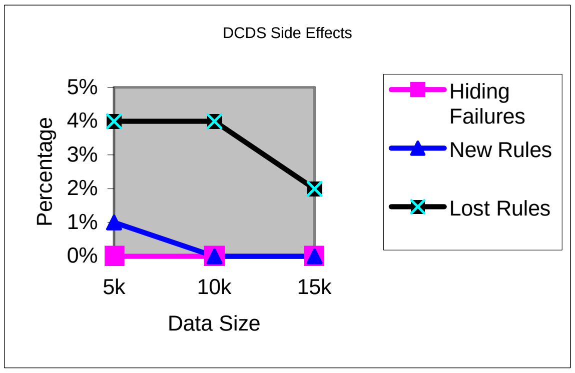

Figure 3 Side Effects of DCDS

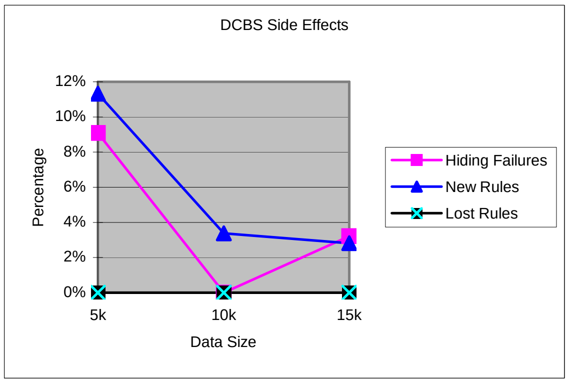

Figure 4 Side Effects of DCBS

7. References

[1] D. Agrawal and C. C. Aggarwal, "On the design and quantification of privacy preserving data mining algorithms", In Proceedings of the 20th Symposium on Principles of Database Systems, Santa Barbara, California, USA, May 2001.

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

[3] R. Agrawal and R. Srikant, "Privacy preserving data mining", In ACM SIGMOD Conference on Management of Data, pages 439–450, Dallas, Texas, May 2000.

[4] Ljiljana Brankovic and Vladimir Estivill-Castro, "Privacy Issues in Knowledge Discovery and Data Mining", Australian Institute of Computer Ethics Conference, July, 1999, Lilydale.

[5] C. Clifton and D. Marks, "Security and Privacy Implications of Data Mining", in SIGMOD Workshop on Research Issues on Data Mining and knowledge Discovery, 1996.

[6] C. Clifton, "Protecting Against Data Mining Through Samples", in Proceedings of the Thirteenth Annual IFIP WG 11.3 Working Conference on Database Security, 1999.

[7] C. Clifton, "Using Sample Size to Limit Exposure to Data Mining", Journal of Computer Security, 8(4), 2000.

[8] Chris Clifton, Murant Kantarcioglu, Xiaodong Lin and Michael Y. Zhu, " Tools for Privacy Preserving Distributed Data Mining", SIGKDD Explorations, 4(2), 1-7, Dec. 2002.

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

[10] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, "Privacy preserving mining of association rules", In Proc. Of the 8th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining, Edmonton, Canada, July 2002.

[11] Alexandre Evfimievski, "Randomization in Privacy Preserving Data Mining", SIGKDD Explorations, 4(2), Issue 2, 43-48, Dec. 2002.

[12] Alexandre Evfimievski, Johannes Gehrke and Ramakrishnan Srikant, "Limiting Privacy Breaches in Privacy Preserving Data Mining", PODS 2003, June 9-12, 2003, San Diego, CA.

[13] H. Huang, X. Wu, and R. Relue, "Association Analysis with One Scan of Databases", Proceedings of IEEE International Conference on Data Mining, Maebashi City, Japan, December, 2002, 629-632.

[14] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation", Proceedings of ACM International Conference on Management of Data (SIGMOD), 2002, 1-12.

[15] M. Kantarcioglu and C. Clifton, "Privacy-preserving distributed mining of association rules on horizontally partitioned data", In ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, June 2002.

[16] Jiuyong Li, Hong Shen and Rodney Topor, "Mining the Smallest Association Rule Set for Predictions", Proceedings of the 2001 IEEE International Conference on Data Mining, 361-368.

[17] Y. Lindell and B. Pinkas, "Privacy preserving data mining", In CRYPTO, pages 36–54, 2000.

[18] D. E. O' Leary, "Knowledge Discovery as a Threat to Database Security", In G. Piatetsky-Shapiro and W. J. Frawley, editors, Knowledge Discovery in Databases, 507-516, AAAI Press/ MIT Press, Menlo Park, CA, 1991.

[19] S. Oliveira, O. Zaiane, "Privacy Preserving Frequent Itemset Mining", Proceedings of IEEE International Conference on Data Mining, November 2002, 43-54.

[20] S. Oliveira, O. Zaiane, “Algorithms for Balancing Privacy and Knowledge Discovery in Association Rule Mining”, Proceedings of 7th International Database Engineering and Applications Symposium (IDEAS03), Hong Kong, July 2003.
[21] S. Oliveira, O. Zaiane, “Protecting Sensitive Knowledge by Data Sanitization”, Proceedings of IEEE International Conference on Data Mining, November 2003.
[22] S. J. Rizvi and J. R. Haritsa, “Privacy-preserving association rule mining”, In Proc. of the 28th Int’l Conference on Very Large Databases, August 2002.
[23] Y. Saygin, V. Verykios, and C. Clifton, “Using Unknowns to Prevent Discovery of Association Rules”, SIGMOND Record 30(4): 45-54, December 2001.
[24] J. Vaidya and C.W. Clifton. “Privacy preserving association rule mining in vertically partitioned data”, In Proc. of the 8th ACM SIGKDD Int’l Conference on Knowledge Discovery and Data Mining, Edmonton, Canada, July 2002.
[25] V. Verykios, E. Bertino, I.G. Fovino, L.P. Provenza, Y. Saygin, and Y. Theodoridis, “State-of-the-art in Privacy Preserving Data Mining”, SIGMOD Record, Vol. 33, No. 1, 50-57, March 2004.
[26] V. Verykios, A. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni, “Association Rules Hiding”, IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 4, 434-447, April 2004.
[27] S.L. Wang, D. Patel, A. Jafari, and T.P. Hong “Hiding Collaborative Recommendation Association Rules”, Applied Intelligence, Vol. 26, No. 1, 66-77, 2007.