CERIAS Tech Report 2004-64

ASSOCIATION RULE HIDING

by V.Verykios, A.Elmagarmid, E.Dasseni, E. Bertino, Y.Saygin

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

Association Rule Hiding

Vassilios S. Verykios, *Member*, *IEEE*, Ahmed K. Elmagarmid, *Senior Member*, *IEEE*, Elisa Bertino, *Fellow*, *IEEE*, Yucel Saygin, *Member*, *IEEE*, and Elena Dasseni

Abstract—Large repositories of data contain sensitive information that must be protected against unauthorized access. The protection of the confidentiality of this information has been a long-term goal for the database security research community and for the government statistical agencies. Recent advances in data mining and machine learning algorithms have increased the disclosure risks that one may encounter when releasing data to outside parties. A key problem, and still not sufficiently investigated, is the need to balance the confidentiality of the disclosed data with the legitimate needs of the data users. Every disclosure limitation method affects, in some way, and modifies true data values and relationships. In this paper, we investigate confidentiality issues of a broad category of rules, the association rules. In particular, we present three strategies and five algorithms for hiding a group of association rules, which is characterized as sensitive. One rule is characterized as sensitive if its disclosure risk is above a certain privacy threshold. Sometimes, sensitive rules should not be disclosed to the public since, among other things, they may be used for inferring sensitive data, or they may provide business competitors with an advantage. We also perform an evaluation study of the hiding algorithms in order to analyze their time complexity and the impact that they have in the original database.

Index Terms—Privacy preserving data mining, association rule mining, sensitive rule hiding.

1 Introduction

Many government agencies, businesses, and nonprofit organizations in order to support their short and long-term planning activities are searching for a way to collect, store, analyze, and report data about individuals, households, or businesses. Information systems, therefore, contain confidential information, such as social security numbers, income, credit ratings, type of disease, customer purchases, etc., that must be properly protected.

Securing against unauthorized accesses has been a long-term goal of the database security research community and the government research statistical agencies. Solutions to such a problem require combining several techniques and mechanisms [1], [2], [3]. In an environment where data have different sensitivity levels, this data may be classified at different levels and made available only to those subjects with an appropriate clearance. It is, however, well known that simply restricting access to sensitive data does not ensure complete sensitive data protection. For example, sensitive or "high" data items may be inferred from nonsensitive, or "low" data through some inference process based on some knowledge of the semantics of the application the user has. Such a problem, known as the "inference problem," has been widely investigated and possible solutions have been identified. The proposed solutions address the problem of how to prevent disclosure of sensitive data through the combination of known inference rules with nonsensitive data. Examples of inference rules are deductive rules, functional dependencies, or material implications [3].

Recent advances in Data Mining (DM) techniques and related applications have, however, increased the security risks that one may incur when releasing data. The elicitation of knowledge that can be attained by such techniques has been the focus of the Knowledge Discovery in Databases (KDD) researchers' effort for years and by now, it is a well-understood problem [4]. On the other hand, the impact on the information confidentiality originating by these techniques has not been considered until very recently.

The process of uncovering hidden patterns from large databases was first indicated as a threat to database security by O' Leary [5]. Piatetsky-Shapiro, of GTE Laboratories, was the chair of a minisymposium on knowledge discovery in databases and privacy, organized around the issues raised in O' Leary's paper in 1991. The focal point discussed by the panel was the limitation of disclosure of personal information, which is not different in principle from the focal point of statisticians and database researchers since, in many fields like medical and socioeconomic research, the goal is not to discover patterns about specific individuals but patterns about groups.

The compromise in the confidentiality of sensitive information, that is not limited to patterns specific to individuals, is another form of threat which is analyzed in a recent paper by Clifton from Mitre Corporation and Marks from Department of Defense [6]. The authors provide a well-desined scenario of how different data mining techniques can be used in a business setting to provide business competitors with an advantage. For completeness purposes, we report such a scenario below.

- *V.S. Verykios is with the Data and Knowledge Engineering Group, Computer Technology Institute, 3 Kolokotroni Street, 26221 Patras, Greece. E-mail: verykios@cti.gr.*
- *A.K. Elmagarmid is with the Office of Strategy and Technology, Hewlett-Packard Company, USA. E-mail: ahmed_elmagarmid@hp.com.*
- *E. Bertino is with the Department of Computer Sciences, University of Milan, Via Comelico, 39/41, 20135 Milano, Italy. E-mail: bertino@dsi.unimi.it.*
- *Y. Saygin is with the Faculty of Engineering and Natural Sciences, Sabanci University, Orhanli, 34956, Tuzla, Istanbul, Turkey. E-mail: ysaygin@sabanciuniv.edu.*
- *E. Dasseni is with TXT e-solutions S.p.A., v. Frigia, 27 20126 Milano, Italy. E-mail: elena.dasseni@txtgroup.it.*

Manuscript received Aug. 2000; revised 4 June 2002; accepted 10 Feb. 2003. For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 112706.

1041-4347/04/$20.00 © 2004 IEEE Published by the IEEE Computer Society

Let us suppose that we are negotiating a deal with Dedtrees Paper Company, as purchasing directors of BigMart, a large supermarket chain. They offer their products with a reduced price if we agree to give them access to our database of customer purchases. We accept the deal and Dedtrees starts mining our data. By using an association rule mining tool, they find that people who purchase skim milk also purchase Green paper. Dedtrees now runs a coupon marketing campaign saying that "you can get 50 cents off skim milk with every purchase of a Dedtrees product." This campaign cuts heavily into the sales of Green paper, which increases the prices to us, based on the lower sales. During our next negotiation with Dedtrees, we find out that with reduced competition, they are unwilling to offer us a low price. Finally, we start to lose business to our competitors, who were able to negotiate a better deal with Green paper.

The scenario that has just been presented indicates the need to prevent disclosure not only of confidential personal information from summarized or aggregated data, but also to prevent data mining techniques from discovering sensitive knowledge which is not even known to the database owners.

In this paper, we propose new strategies and a suite of algorithms for hiding sensitive knowledge from data by minimally perturbing their values. The hiding strategies that we propose are based on reducing the support and confidence of rules that specify how significant they are (confidence and support will be formally defined in Section 3). In order to achieve this, transactions are modified by removing some items, or inserting new items depending on the hiding strategy. The constraint on the algorithms is that the changes in the database introduced by the hiding process should be limited, in such a way that the information loss incurred by the process is minimal. Selection of the items in a rule to be hidden and the selection of the transactions that will be modified is a crucial factor for achieving the minimal information loss constraint. We also perform a detailed performance evaluation in order to prove that the proposed algorithms are computationally efficient and provide certain provisions on the changes that they impose in the original database. According to this, we try to apply minimal changes in the database at every step of the hiding algorithms that we propose.

The rest of this paper is organized as follows: In Section 2, we present an overview of the current approaches to the problem of DM and security. Section 3 gives a formalization of the problem, while some solutions are presented in Section 4. Section 5 discusses performance results obtained from the applications of the devised algorithms. In Section 6, a discussion on the possible damage on the data in terms of possible queries is provided. Concluding remarks and future extensions are listed in Section 7.

2 Background and Related Work

The security impact of DM is analyzed in [6] and some possible approaches to the problem of inference and discovery of sensitive knowledge in a data mining context are suggested. The proposed strategies include fuzzyfying and augmenting the source database and also limiting the access to the source database by releasing only samples of the original data. Clifton [7] adopts the last approach as he studies the correlation between the amount of released data and the significance of the patterns which are discovered. He also shows how to determine the sample size in such a way that data mining tools cannot obtain reliable results.

Clifton and Marks in [6] also recognize the necessity of analyzing the various data mining algorithms in order to increase the efficiency of any adopted strategy that deals with disclosure limitation of sensitive data and knowledge. The solution proposed by Clifton in [7] is independent from any specific data mining technique; other researchers [8], [9] propose solutions that prevent disclosure of confidential information for specific data mining algorithms such as association rule mining and classification rule mining.

Classification mining algorithms may use sensitive data to rank objects; each group of objects has a description given by a combination of nonsensitive attributes. The sets of descriptions, obtained for a certain value of the sensitive attribute, are referred to as description space. For Decision-Region-based algorithms, the description space generated by each value of the sensitive attribute can be determined a priori. The authors in [8] first identify two major criteria which can be used to assess the output of a classification inference system and then they use these criteria, in the context of Decision-Region based algorithms, to inspect and to modify, if necessary, the description of a sensitive object so that they can be sure that it is not sensitive.

Agrawal and Srikant used data perturbation techniques to modify the confidential data values in such a way that the approximate data mining results could be obtained from the modified version of the database [10]. They considered applications where the individual data values are confidential rather than the data mining results and concentrated on a specific data mining model, namely, the classification by decision trees. Agrawal and Aggarwal, in their recent paper, enhance the data perturbation methods by using expectation maximization for reconstructing the original data distribution which is further used to construct the classification model [11].

Disclosure limitation of sensitive knowledge by data mining algorithms, based on the retrieval of association rules, has also been recently investigated [9]. The authors in [9] propose to prevent disclosure of sensitive knowledge by decreasing the significance of the rules using some heuristics which can be thought of as the precursors to the heuristics we proposed in this paper.

3 Problem Formulation

Let $I = \{i_1, \ldots, i_n\}$ be a set of literals, called items. Let D be a set of transactions which is the database that is going to be disclosed. Each transaction $t \in D$ is an itemset such that $t \subseteq I$. A unique identifier, which we call TID, is associated with each transaction. We say that a transaction t supports X, a set of items in I, if $X \subseteq t$. We assume that the items in a transaction or an itemset are sorted in lexicographic order. A sample database of transactions is shown in Table 1a. Each row in the table represents a transaction. There are three items and six transactions. AB is an itemset and transaction T_1 supports that itemset. An itemset X has support s if s percent of the transactions support X. Support

TABLE 1
(a) Sample Database D and (b) Large Itemsets Obtained from D

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

(a)

Itemset	Support
A	100%
B	66%
C	66%
AB	66%
AC	66%
BC	50%
ABC	50%

(b)

TABLE 2
The Rules Derived from the Large Itemsets of Table 3

Rules	Confidence	Support
$B \Rightarrow A$	100%	66%
$B \Rightarrow C$	75%	50%
$C \Rightarrow A$	100%	66%
$C \Rightarrow B$	75%	50%
$B \Rightarrow AC$	75%	50%
$C \Rightarrow AB$	75%	50%
$AB \Rightarrow C$	75%	50%
$AC \Rightarrow B$	75%	50%
$BC \Rightarrow A$	100%	50%

of X is denoted as $Supp(X)$. All possible itemsets and their supports obtained from the database in Table 1a are listed in Table 1b. For example, Itemset BC is supported by three transactions out of six and, therefore, has 50 percent support.

An association rule is an implication of the form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$. We say that the rule $X \Rightarrow Y$ holds in the database D with *confidence* c if $\frac{|XUY| \times 100}{|X|} \geq c$ (where $|A|$ is the number of occurrences of the set of items A in the set of transactions D, and A occurs in a transaction t, if and only if $A \subseteq t$). We also say that the rule $X \Rightarrow Y$ has *support* s if $\frac{|XUY| \times 100}{N} \geq s$, where N is the number of transactions in D. Note that, while the support is a measure of the frequency of a rule, the confidence is a measure of the strength of the relation between sets of items. All possible rules obtained from the large itemsets in Table 1b are shown in Table 2. For example, Rule $BC \Rightarrow A$ has 50 percent support since it appears in three out of six transactions in D and it has 100 percent confidence since all the transactions that contain BC also contain A.

A typical association rule-mining algorithm first finds all the sets of items that appear frequently enough to be considered significant and then it derives from them the association rules that are strong enough to be considered interesting. We aim at preventing some of these rules, that we refer to as "sensitive rules," from being disclosed. The problem can be stated as follows:

> Given a database D, a set R of relevant rules that are mined from D and a subset R_H of R, how can we transform D into a database D' in such a way that the rules in R can still be mined, except for the rules in R_H?

TABLE 3
The Sample Database that Uses the Proposed Notation

TID	Items	Size
T1	111	3
T2	111	3
T3	111	3
T4	110	2
T5	100	1
T6	101	2

In [9], the authors demonstrate that solving this problem by reducing the support of the large itemsets via removing items from transactions (also referred to as "sanitization" problem) is an NP-hard problem. Thus, we are looking for a transformation of D (the source database) in D' (the released database) that *maximizes* the number of rules in $R - R_H$ that can still be mined.

There are two main approaches that can be adopted when we try to hide a set R_H of rules (i.e., prevent them from being discovered by association rule mining algorithms): 1) we can either prevent the rules in R_H from being generated by hiding the frequent itemsets from which they are derived, or 2) we can reduce the confidence of the sensitive rules by bringing it below a user-specified threshold (*min_conf*). In this paper, we propose five strategies to hide rules using both the approaches; work related to the first approach can also be found in [9].

4 Proposed Strategies and Algorithms

This section is organized as follows: In Section 4.1, we introduce the required notation and in Section 4.2, we introduce three strategies that solve the problem of hiding association rules by tuning their confidence and their support. Few assumptions that we make are presented in Section 4.3, while the building blocks of the algorithms that implement those strategies are presented in Section 4.4.

4.1 Notation and Preliminary Definitions

Before presenting the strategies and the algorithms, we introduce some notation. We used a bitmap notation with a few extensions to represent a database of transactions. Bitmap notation is commonly used in the association rule mining context. In this representation, each transaction t in the database D is a triple:

$$t = < TID,\ values_of_items,\ size >,$$

where TID is the identifier of the transaction t and $values_of_items$ is a list of values with one value for each item in the list of items I. An item is represented by one of the initial capital letters of the English alphabet. The items in an itemset are sorted in lexicographic order. An item is supported by a transaction t if its value in the $values_of_items$ is 1 and it is not supported by t if its value in $values_of_items$ is 0. *Size* is the number of 1 values which appear in the $values_of_items$ (e.g., the number of items supported by transaction t). For example, Table 3 shows a database of transactions where the set of items is $I = \{A, B, C\}$. If t is a transaction that

contains the items $\{A, C\}$, it would be represented as $t = < T6, [101], 2 >$ as shown in the last row of Table 3.

Given a set P, we adopt the conventional representation $|P|$ to indicate the number of elements of the set P (also referred to as *size* or *cardinality* of P). According to this notation, the number of transactions stored in a database D is indicated as $|D|$, while $|I|$ represents the number of the different items appearing in D. The set of rules that can be mined from the database will be indicated by R and the subset of these rules that we're interested in hiding will be referred to as R_H. For each rule r in R_H, we use the compact notation $\text{lhs}(r)$ or l_r to indicate the itemset which appears in the left side of a rule r (also referred to as rule antecedent) and $\text{rhs}(r)$ or r_r to indicate the itemset which appears in the right side of a rule (also referred to as rule consequent).

Before going into the details of the proposed strategies, it is necessary to give some definitions. Given a transaction t and an itemset S, we say that t *fully supports* S if the values of the items of S in $t.values_of_items$ are all 1; t is said to *partially support* S if the values of the items of S in $t.values_of_items$ are not all 1s. For example, if $S = \{A, B, C\} = [1110]$ and $p = < T1, [1010], 2 >$, $q = < T2, [1110], 3 >$, then we would say that q fully supports S, while p partially supports S.

A rule r corresponds to an itemset. This itemset is the union of the items in the left-hand side and the right-hand side of the rule. We denote the itemset that corresponds to rule r as I_r, and we refer to it as the *generating itemset* of r. Notice that two different rules may have the same generating itemset.

We use the notation T_r to denote the set of transactions that fully support the generating itemset of a rule r. We also denote by T_{l_r} the set of transactions that fully support the left-hand side or the antecedent of the rule r, while by T_{r_r} we denote the set of transactions that fully support the right-hand side of the rule r. We slightly change the notations to represent a set of transactions that partially supports an itemset. In the previous notations, we add the prime symbol in all occurrences of T to indicate partial support. So, the set of transactions that fully support the consequent of the rule but partially support the antecedent is denoted by T'_{l_r}.

Let L also be the set of large itemsets with a greater support than the min_supp and $L_H \subseteq L$ be the set of large itemsets that we want to hide from D. Note that L_H is the set of generating itemsets of the rules in R_H. We denote with T_Z the set of the transactions that support an itemset Z. Finally, we denote by $A_D = |D| * ATL$ the average number of items in the database where ATL is the average transaction length.

4.2 The Hiding Strategies

The hiding strategies that we propose heavily depend on finding transactions that fully or partially support the generating itemsets of a rule. The reason for this is that, if we want to hide a rule, we need to change the support of some part of the rule (i.e., decrease the support of the generating itemset). Another issue is that the changes in the database introduced by the hiding process should be limited in such a way that the information loss incurred by the process is minimal. According to this, we try to apply minimal changes in the database at every step of the hiding algorithms that we propose.

The decrease in the support of an itemset S can be done by selecting a transaction t that supports S and by setting to 0 at least one of the nonzero values of $t.values_of_items$ that represent items in S. The increase in the support of an itemset S can be accomplished by selecting a transaction t that partially supports it and setting to 1 the values of all the items of S in $t.values_of_items$. As an example, consider the database in Table 3 and the rules in Table 2 that are obtained from this database. Given that $min_supp = 33\%$ and $min_conf = 70\%$, we are interested in hiding the rule $AC \Rightarrow B$, with support = 50 percent and confidence = 75 percent. In order to decrease the support of the rule $AC \Rightarrow B$, we can select the transaction $t = < T1, [111], 3 >$ and turn to 0 one of the elements in the list of items that corresponds to A, B, or C. Say, we decide to set to 0 the element corresponding to C, obtaining $t = < T1, [110], 2 >$. The rule $AC \Rightarrow B$ has been hidden (support = 33 percent, confidence = 66 percent).

In order to decrease the confidence of a rule $X \Rightarrow Y$, we can decrease the support of its generating itemset, making sure that we hide items from the consequent or the right-hand side of the rule. This will decrease the support of the rule, while leaving the support of the left-hand side unchanged (i.e., the denominator in the confidence formula). Again, let's consider the rule $AC \Rightarrow B$ in Table 2. In order to decrease its confidence, we select the transaction $t = < T1, [111], 3 >$ and turn to 0 the element of the list of items that corresponds to B. The transaction becomes $t = < T1, [101], 2 >$ and we obtain: $AC \Rightarrow B$ with support = 33 percent and confidence = 50 percent, which means that the rule is hidden. We can also increase the denominator in the confidence formula (which is the support of the itemset in the antecedent) in order to decrease the confidence of the rule, while the support of the generating itemset of the rule remains fixed. We can achieve this by modifying the transactions that partially support the itemset in the antecedent of the rule but do not fully support the itemset in the consequent. Let's consider the rule $AC \Rightarrow B$ in Table 2 one more time. In order to decrease the confidence, we select the transaction $t = < T5, [100], 1 >$ and turn to 1 the element of the list that corresponds to C. We obtain $t = < T5, [101], 2 >$. Now, the rule $AC \Rightarrow B$ has support = 50 percent and confidence = 60 percent, which means that the rule has been hidden since its confidence is below the min_conf threshold.

Given a rule $X \Rightarrow Y$ on a database D, the strategies that we developed, based on the observations above, can be summarized as follows:

1. We decrease the confidence of the rule: 1) by increasing the support of the rule antecedent X through transactions that partially support it and 2) by decreasing the support of the rule consequent Y in transactions that support both X and Y.
2. We decrease the support of the rule by decreasing the support of either the rule antecedent X, or the rule consequent Y, through transactions that fully support the rule.

4.3 Assumptions

We make the following assumptions in the development of the algorithms:

1. We hide only rules that are supported by disjoint large itemsets.
2. We hide association rules by decreasing either their support or their confidence.
3. We select to decrease either the support or the confidence based on the side effects on the information that is not sensitive.
4. We hide one rule at a time.
5. We decrease either the support or the confidence, one unit at a time.

Without the first assumption, i.e., if we try to hide overlapping rules, then hiding a rule may have side effects on the other rules to be hidden. This may increase the time complexity of the algorithms since hiding a rule may cause an already hidden rule to haunt back. Therefore, the algorithms should reconsider previously hidden rules and hide them back if they are no longer hidden. We should also modify the transaction selection mechanism for hiding overlapping rules so that transactions do not choose the transactions that support only the antecedent of the overlapping rules.

The widely used data mining algorithms first find the large itemsets whose supports are greater than a threshold value. Rules are then generated and their confidence is calculated. When we are hiding a rule by decreasing its support below the threshold, we do not need to further reduce its confidence and vice versa. So, reducing the support or the confidence is enough for hiding a rule which is stated by the second assumption. Note that, while decreasing the support of a rule, its confidence will also decrease. Also, when using the first confidence reduction method to decrease the confidence, the support of the rule will also decrease.

The third assumption identifies the main constraint of the algorithms that aim to maximize the data quality in terms of nonsensitive information. If we relax this assumption, we can just randomly choose a transaction and an item to hide from the database without the need for any kind of heuristic approach.

The fourth assumption is related to the first assumption. Since the rules to be hidden are assumed to be disjoint, the items chosen for hiding a rule will also be different for different rules. Therefore, hiding a rule will not have a side effect on the rest of the rules. So, considering the rules one at a time or all together will not make any difference. If we relax the first assumption, we should reconsider the fourth assumption as well since items for the hiding process should be selected carefully taking into account the overlapping rules as explained in the discussion of the first assumption.

The proposed heuristics for rule hiding work step by step, considering an item and a transaction at each step which is stated by the fifth assumption. If we relax this assumption, we can assume that we remove clusters of transactions and remove items from the whole cluster which is an entirely different approach.

INPUT: a set R_H of rules to hide, the source database D, the number $|D|$ of transactions in D, the *min_conf* threshold, the *min_supp* threshold

OUTPUT: the database D transformed so that the rules in R_H cannot be mined

Begin
Foreach rule r in R_H **do**
{
1. $T'_{l_r} = \{t \in D /$ t fully supports r_r and partially supports $l_r\}$
// *count how many items of* l_r *are*
// *in each trans. of* T'_{l_r}
2. **foreach** transaction t in T'_{l_r} **do**
{
3. t.num_items= $|I|-$ $|l_r \cap$ t.values_of_items$|$
}
// *sort transactions of* T'_{l_r} *in descending*
// *order of number of items of* l_r
// *contained*
4. sort(T'_{l_r})
5. $N_iterations = \lceil |D| * (\frac{supp(r)}{min_conf} - supp(l_r)) \rceil$
6. **For** $i = 1$ to $N_iterations$ **do**
{
// *pick the transaction of* T'_{l_r} *with the*
// *highest number of items*
7. t = $T'_{l_r}[1]$
// *set to one all the bits of t that*
// *represent items in* l_r
8. set_all_ones(t.values_of_items, l_r)
9. supp(l_r)=supp(l_r)+1
10. conf(r)=supp(r)/supp(l_r)
11. $T'_{l_r} = T'_{l_r}$ - t
}
12. $R_H = R_H - r$
}
End

Fig. 1. Pseudocode of Algorithm 1.a.

4.4 Algorithms

We now present in detail five algorithms for the strategies that we developed and that we have already described in Section 4.2 and we analyze the time complexity of those algorithms.

4.4.1 Algorithm 1.a

The first algorithm hides the sensitive rules according to the first strategy: For each selected rule, it increases the support of the rule's antecedent until the rule confidence decreases below the min_conf threshold. An overview of this algorithm is depicted in Fig. 1.

Given a rule r, Algorithm 1.a starts by generating the set T'_{l_r} of transactions fully supporting the consequent but partially supporting the antecedent of r (i.e., l_r) and counting, for each transaction, the number of items of l_r that it contains. From T'_{l_r}, the transaction that supports the

highest number of items of l_r is then selected as the one through which the increase of the support of the rule's antecedent is accomplished. This step is performed by sorting the transactions in T'_{l_r} in decreasing order of number of items of the l_r contained and then selecting the first transaction. Note that the transaction length is bound, therefore, we can utilize a sorting algorithm that is of order $O(|T_{l_r}|)$. In order to increase the support of the rule's antecedent through t, Algorithm 1.a has to set to 1 the elements in t corresponding to items of l_r, by choosing the transaction containing the largest subset of items in l_r, the number of changes made to the database is tried to be minimized. Once the support of the rule's antecedent has been updated, the confidence of the rule is recomputed. A check is then applied in order to find out if the rule is still significant. If no further steps are required for the current rule, another rule is selected from the set R_H. If, on the other hand, the confidence of r is still greater than (or equal to) min_conf, Algorithm 1.a keeps executing the operations inside the inner loop before selecting a new rule from R_H.

Lemma 4.1. *Given a rule r, Algorithm 1.a performs $\lceil |D| * (\frac{supp(r)}{min_conf} - supp(l_r)) \rceil$ executions of the inner loop.*

Proof. During each execution of the inner loop, the number of transactions supporting the antecedent of the rule (l_r) is increased by one. After k iterations, the confidence of r will be $Conf(r)^{(k)} = \frac{N_r}{N_{l_r}+k}$, where N_r is the number of transactions supporting r and N_{l_r} is the number of transactions supporting l_r. The inner loop is executed until $Conf(r)^{(k)} < min_conf$ or, more specifically, $\frac{N_r}{N_{l_r}+k} < min_conf$. This inequality can be rewritten as $\frac{N_r}{min_conf} - N_{l_r} < k$. From the last inequality, we can derive that $k = \lceil \frac{N_r}{min_conf} - N_{l_r} \rceil$. The formula about k can be easily derived by observing that the iterations in the inner loop terminate as soon as the first integer value greater than (or equal to) $\frac{N_r}{min_conf} - N_{l_r}$ is reached. By incorporating the size of database in the formula for k, we can prove that $k = \lceil |D| * (\frac{N_r}{|D| * min_conf} - \frac{N_{l_r}}{|D|}) \rceil$ or that $k = \lceil |D| * (\frac{supp(r)}{min_conf} - supp(l_r)) \rceil$. □

4.4.2 Algorithm 1.b

This algorithm hides sensitive rules by decreasing the frequency of the consequent until either the confidence or the support of the rule is below the threshold. The algorithm is depicted in Fig. 2. It starts by generating the set T_r of transactions supporting the rule r and counting the number of items supported by each one of them. Algorithm 1.b sorts T_r in ascending order of transaction size. Then, it chooses the smallest transaction to minimize the impact on the database. The item that has the minimum impact on the database is selected as in the case of Algorithm 1.a. Once the selected item has been deleted form t and t has been removed from T_r, Algorithm 1.b updates the support and the confidence of the rule and checks if it's still significant. If no further steps are required, another rule is selected from the set R_H. A new rule is selected from R_H when either the support or the confidence of the current rule decreases below the minimum threshold. Therefore, the number of

INPUT: a set R_H of rules to hide, the source database D, the size of the database $|D|$, the *min_conf* threshold, the *min_supp* threshold

OUTPUT: the database D transformed so that the rules in R_H cannot be mined

Begin
Foreach rule r in R_H **do**
{
1. $T_r = \{t \in D /$ t fully supports $r\}$
// count how many items are in each
// transaction of T_r
2. **foreach** transaction t in T_r **do**
{
3. t.num_items=count(t)
}
// sort T_r in ascending order of
// size of the transactions
4. sort(T_r)
5. $N_iter_conf = \lceil |D| * (\frac{supp(r)}{min_conf} - supp(l_r)) \rceil$
6. $N_iter_supp = \lceil |D| * \frac{supp(r)}{min_supp} \rceil$
7. $N_iterations = \min(N_iter_conf, N_iter_supp)$
8. **For** i=1 **to** $N_iterations$ **do**
{
// choose the transaction in T_r
// with the lowest size
9. t = T_r[1]
// choose the item of r_r
// with the minimum impact on the
// ($|r_r| - 1$)-itemsets
10. j = choose_item(r_r)
// set to zero the bit of t.values_of_items
// that represents item j
11. set_to_zero(j, t.values_of_items)
12. supp(r)=supp(r) - 1
13. Conf(r)=supp(r)/supp(l_r)
14. $T_r = T_r$ - t
}
15. $R_H = R_H - r$
}
End

Fig. 2. Pseudocode of Algorithm 1.b.

executions of the inner loop that are performed by Algorithm 1.b in order to hide a rule r, is given by the minimum of the number of iterations required to bring $conf(r) < min_conf$ and the number of iterations required to bring $supp(r) < min_supp$. The number of iterations required to reduce $conf(r)$ below the threshold is given by Lemma 4.1. Observe that we can use the same results obtained for Algorithm 1.a since they are independent from the specific algorithm. The number of iterations required to reduce the $supp(r)$ below the minimum threshold is given by the following lemma.

Lemma 4.2. *Given a rule r, Algorithm 1.b takes $\lceil |D| * \frac{supp(r)}{min_supp} \rceil$ steps to reduce $supp(r)$ below the min_supp threshold.*

INPUT: a set R_H of rules to hide, the source database D, the size of the database $|D|$, the *min_conf* threshold, the *min_supp* threshold

OUTPUT: the database D transformed so that the rules in R_H cannot be mined

Begin
Foreach rule r in R_H **do**
{
1. $T_r = \{t \in D/$ t fully supports $r\}$
// count how many items are supported
// by each trans. of T_r
2. **foreach** transaction t in T_r **do**
{
3. t.num_items=count(t)
}
// sort T_r in ascending order of
// size of the transactions
4. sort(T_r)
5. $N_iter_conf = \lceil |D| * (\frac{supp(r)}{min_conf} - supp(l_r)) \rceil$
6. $N_iter_supp = \lceil |D| * \frac{supp(r)}{min_supp} \rceil$
7. $N_iterations = \min(N_iter_conf, N_iter_supp)$
8. **For** i=1 **to** $N_iterations$ **do**
{
// choose the transaction in T_r
// with the lowest size
9. t = T_r[1]
// choose the item of r
// with the minimum impact on the
// ($|r| - 1$)-itemsets
10. j = choose_item(r)
// set to zero the bit of t.list_of_items
// that represents item j
11. set_to_zero(j, t.values_of_items)
12. supp(r)=supp(r) - 1
13. Conf(r)=supp(r)/supp(l_r)
14. $T_r = T_r$ - t
}
15. $R_H = R_H - r$
}
End

Fig. 3. Pseudocode of Algorithm 2.a.

Proof. During each execution of the inner loop, the number of transactions supporting the rule r is decreased by one. Therefore, after the k iteration, the support will be $Supp(r)^k = \frac{N_r - k}{|D|}$, where N_r is the number of transactions supporting r. The number of steps required to reduce the support below the min_supp threshold is given by the following relation $k = \lceil |D| * (\frac{supp(r)}{min_conf} - supp(l_r)) \rceil$ or, equivalently, by $\frac{N_r - k}{|D|} < min_supp$. The previous inequality can be rewritten as $\frac{N_r}{|D| * min_supp} < \frac{k}{|D|}$ or $|D| * \frac{N_r}{|D| * min_supp} < k$. We can then derive the value for k as follows $k = \lceil |D| * \frac{supp(r)}{min_supp} \rceil$ since k is an integer. □

4.4.3 Algorithm 2.a

This algorithm decreases the support of the sensitive rules until either their confidence is below the min_conf threshold or their support is below the min_supp threshold. Fig. 3 shows the pseudocode.

Algorithm 2.a first generates the set T_r of transactions supporting the rule r and counts the number of items supported by each of them. Then, it sorts T_r in increasing order of transaction size and chooses the first transaction in the so ordered T_r trying to minimize the impact on the database. The item to be removed is selected according to the "minimum impact" criterion as in the previous cases. Once the chosen item has been deleted form t and t has been removed from T_r, support and confidence of the rule is updated to check if it is still significant. Another rule is selected from the set R_H when support or confidence of the current rule is below the threshold. The number of executions of the inner loop performed by Algorithm 2.a is therefore given by the smallest between the number of iterations required to bring $conf(r) < min_conf$ and the number of iterations required to bring $supp(r) < min_supp$. According to the results from Lemma 4.1 and Lemma 4.2, we can state that this number is the minimum between $\lceil |D| * (\frac{supp(r)}{min_conf} - supp(l_r)) \rceil$ and $\lceil |D| * \frac{supp(r)}{min_supp} \rceil$ denoted by $N_iterations$ in Fig. 3.

4.4.4 Algorithm 2.b

This algorithm hides sensitive rules by decreasing the support of their generating itemsets until their support is below the minimum support threshold. An overview of this algorithm is depicted in Fig. 4. The item with maximum support is hidden from the minimum length transaction. The generating itemsets of the rules in R_H are considered for hiding. The generating itemsets of the rules in R_H are stored in L_H and they are hidden one by one by decreasing their support. The itemsets in L_H are first sorted in descending order of their size and support. Then, they are hidden starting from the largest itemset. If there are more than one itemset with maximum size, then the one with the highest support is selected for hiding. The algorithm works like follows: Let Z be the next itemset to be hidden. The algorithm hides Z by decreasing its support. The algorithm first sorts the items in Z in descending order of their support and sorts the transactions in T_Z in ascending order of their size. The size of a transaction is determined by the number of items it contains. At each step, the item $i \in Z$, with maximum support is selected and removed from the transaction with minimum size to minimize the impact on the database. The execution stops after the support of the current rule to be hidden goes below the minimum support threshold. Given a large itemset, Z is to be hidden.

4.4.5 Algorithm 2.c

This algorithm hides sensitive rules by decreasing the support of their generating itemsets until the support is below the minimum support threshold. If there are more than one large itemsets to hide, the algorithm first sorts the large itemsets with respect to their size and support as Algorithm 2.b does. Formally, let the next itemset to be hidden be Z. Algorithm 2.c hides Z from D by removing the

INPUT: a set L of large itemsets, the set L_H of large itemsets to hide, the database D and the *min_supp* threshold
OUTPUT: the database D modified by the deletion of the large itemsets in L_H
Begin
//sort L_H in descending order of size and support
1. sort(L_H)
//T_H is the data structure that keeps the
//transactions that support L_H
2. $T_H = \{T_Z | Z \in L_H$ and
$\forall t \in D : t \in T_Z \Rightarrow t$ supports $Z\}$
foreach Z **in** L_H
{
//sort T_Z in ascending order
//of transaction size
3. sort(T_Z)
4. $N_iterations = |T_Z| - min_supp * |D|$
For $k = 1$ **to** $N_iterations$ **do**
{
//get the top transaction of T_Z
//and delete it from T_Z
5. t = popfrom(T_Z)
6. a = maximal support item in Z
//a_S is the collection of itemsets
// that contain item a
7. $a_S = \{X \in L_H | a \in X\}$
//Propagate the effects of deleting a from t
//to other itemsets in L_H
// that are supported by t
foreach X **in** a_S
{
8. **if** (t **in** T_X)
9. delete(t,T_X)
}
//Delete item a from transaction t
//in database D
10. delete(a,t, D)
}
}
End

Fig. 4. Pseudocode of Algorithm 2.b.

items in Z from the transactions in T_Z, in round-robin fashion. The algorithm starts with a random order of items in Z and a random order of transactions in T_Z. Assume that the order of items in Z be $i_0, i_1, \ldots, i_{(n-1)}$ and let the order of transactions in T_Z be $t_0, t_1, \ldots, t_{(m-1)}$. In step 0 of the algorithm, the item i_0 is removed from t_0. At step 1, i_1 is removed from t_1 and, in general, at step k, item $i_{(k \bmod n)}$ is removed from transaction t_k. The execution stops after the support of the current itemset, to be hidden, goes below the minimum support threshold. The algorithm is fairly simple and due to the space limitations, we do not provide the pseudocode for it. Algorithm 2.c is the base algorithm which is going to be used for comparison with more sophisticated algorithms for reducing the support and confidence of the rules. The intuition behind the idea of hiding in round-robin fashion is fairness. This way, no item is overkilled and the chance of having a smaller number of side effects is higher than choosing an item at random and always trying to hide it.

TABLE 4
The Data Sets Used in the Evaluation Trials

$\|D\|$	$\|I\|$	ATL
10k	50	5
50k	50	5
100k	50	5

5 Performance Evaluation

We performed our experiments on a SPARC workstation with **400** MHz processor and with **1 GB** of main memory, under SunOS **5.6** operating system. In order to generate the source databases, we made use of the IBM synthetic data generator. The performance of the developed algorithms has been measured according to two criteria: time requirements and side effects produced. As time requirements, we considered the time needed by each algorithm to hide a specified set of rules; as side effects, we considered the number of "lost" rules and the number of "new" rules introduced by the hiding process. Hiding rules produces some changes in the original database, changes that affect the set of rules mined. In particular, not all the rules that can be mined from the source DB can still be retrieved in the released DB and some rules that couldn't be mined in the source DB can be retrieved after the hiding process. We call the former rules "lost rules" and the latter rules "new rules."

To assess the performance of the proposed algorithms, we used them to hide sets of 5 and 10 rules mined from the data sets of Table 4, each time measuring the time required by the hiding process. For each data set, we generated all the rules that have minimum support and minimum confidence and we stored them in an appropriate data structure. After the completion of the hiding process, we mined the released database and then we compared the rules generated by the two databases in the following way: We checked if the nonsensitive rules mined from the source database could still be mined in the released database. To do so, we compared each rule mined from the original database with each rule mined form the released DB. If the rule wasn't found, we considered it "lost." Note that this process of rule checking tends to consider the rules selected for hiding as "lost" since they cannot be retrieved from the released database. However, since they have been hidden on purpose, we excluded them from the set of "lost rules."

Once all the "lost rules" had been determined, we checked if any new rule had been introduced by the hiding process. To do so, we computed the difference between the number of rules mined from the released DB and the number of rules mined from the source database and subtracting from this difference the number of rules lost. If the result is greater than zero, it gives the number of the "new rules" introduced.

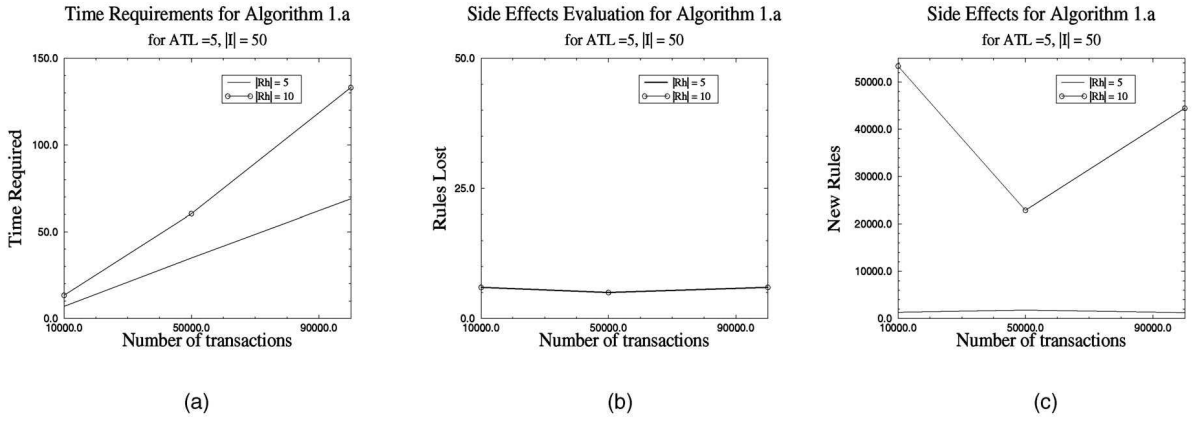


Fig. 5. Evaluation of Algorithm 1.a.

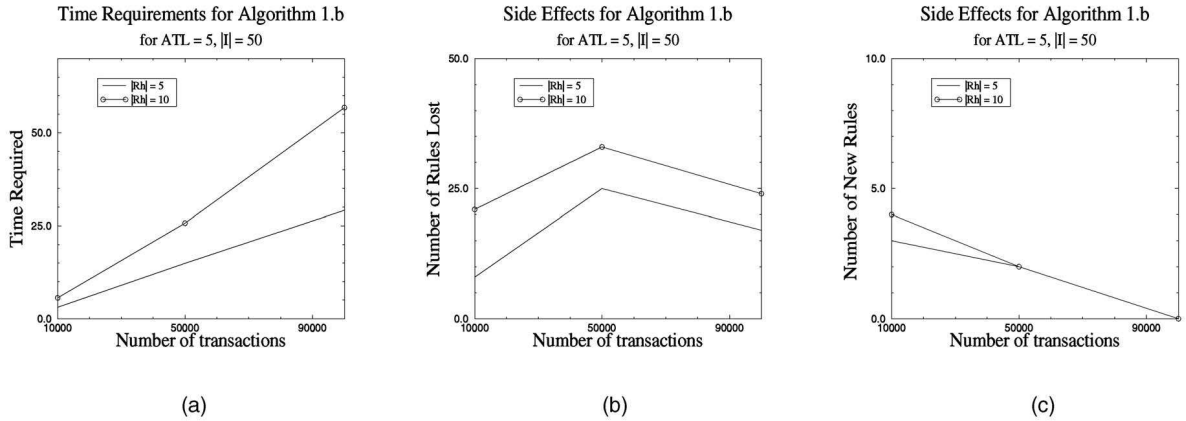


Fig. 6. Evaluation of Algorithm 1.b.

5.1 Performance Evaluation of Algorithm 1.a

The time requirements and side effects evaluation for Algorithm 1.a are shown in Figs. 5a, 5b, and 5c, respectively. As depicted from Fig. 5a, the time needed by Algorithm 1.a is linear in $|D|$ and $|R_H|$, according to the theoretical analysis and to the validation results. The increase of the support of the antecedent of a set of rules increases the number and the size of the frequent itemsets; the effect is two-fold: The existing nonsensitive rules cannot reach the minimum requirements in the released database (rules lost) and the rules, whose support or confidence—in the source database—did not reach the minimum threshold, now they do in the released database (new rules). As Fig. 5b shows, the number of nonsensitive rules lost does not depend neither on the number of transactions in the database nor on the number of rules selected for hiding. In the opposite, the number of new rules generated depends on $|R_H|$: As Fig. 5c shows, the number of new rules introduced by Algorithm 1.a increases when $|R_H|$ increases. We experienced that, if we hide larger sets of rules, a larger number of new frequent itemsets is introduced and, therefore, an increasing number of new rules is generated.

5.2 Performance Evaluation of Algorithm 1.b

The time needed by Algorithm 1.b increases proportionally to $|D|$ and $|R_H|$, as Fig. 6a shows. This is in accordance to the theoretical analysis and to the validation results.

Fig. 6b gives the number of rules mined from the source database that cannot be mined after the hiding process; as can be seen, the number of nonsensitive rules lost by Algorithm 1.b increases when $|R_H|$ increases. The peak in correspondence to the 50k data set can be explained by looking at the characteristics of the rules hidden. As Table 5 shows, the fourth rule selected in the 50k data set has a very high support and there are no such rules in the other two data sets. Moreover, the support of the antecedent of this rule was also very high. To hide this rule required the deletion of its consequent from numerous transactions supporting it. This results to the increase of the number of nonsensitive rules whose support or confidence decreased below the threshold. For the Algorithm 1.b, new rules can be generated starting from those that have support greater than (or equal to) the min_supp but confidence lower than min_conf: If the deletion of some literals decreases the support of the antecedent of these rules, so that their confidence overcomes the minimum threshold, these rules become nonsensitive. As depicted in Fig. 6c, the number of new rules is quite low and tends to decrease when the number of transactions in the database increases. We can therefore say that the changes, introduced by Algorithm 1.b to hide the rules of Tables 5 and 6, reduce the number of nonsensitive rules mined from the original database proportionally to $|R_H|$, but slightly affect the number of rules that become nonsensitive.

TABLE 5
Fives Rules Selected for Hiding
from Databases of Size 10k, 50k, and 100k

|DB|	Rules	Support	Confidence
10k	8 25 ⇒ 45	2.25%	40.34%
	12 27 ⇒ 21	2.15%	37.95%
	6 ⇒ 43	2.44%	30.56%
	37 ⇒ 38	2.04%	27.09%
	28 ⇒ 31	2.57%	23.87%
50k	8 25 ⇒ 45	2.18%	39.71%
	12 27 ⇒ 21	2.40%	36.83%
	6 ⇒ 43	2.30%	29.14%
	1 ⇒ 38	4.65%	25.86%
	10 ⇒ 27	2.24%	21.60%
100k	8 25 ⇒ 45	2.19%	39.91%
	12 27 ⇒ 21	2.36%	36.35%
	6 ⇒ 43	2.33%	29.87%
	37 ⇒ 38	2.02%	26.47%
	1 ⇒ 10	2.27%	21.59%

TABLE 6
Five Additional Rules for Hiding
from Databases of Size 10k, 50k, and 100k

|DB|	Rules	Support	Confidence
10k	33 25 ⇒ 46	2.02%	22.82%
	10 ⇒ 48	2.16%	21.21%
	0 ⇒ 1	2.77%	20.65%
	4 ⇒ 35	2.55%	18.76%
	3 ⇒ 32	2.22%	18.02%
50k	46 ⇒ 48	3.50%	20.58%
	4 ⇒ 35	2.63%	19.11%
	32 ⇒ 39	2.02%	18.19%
	0 ⇒ 2	2.19%	16.05%
	3 ⇒ 5	2.20%	15.89%
100k	3 ⇒ 27	2.59%	20.74%
	46 ⇒ 48	3.46%	20.24%
	4 ⇒ 35	2.64%	18.96%
	2 ⇒ 32	2.58%	16.27%
	0 ⇒ 5	2.11%	15.41%

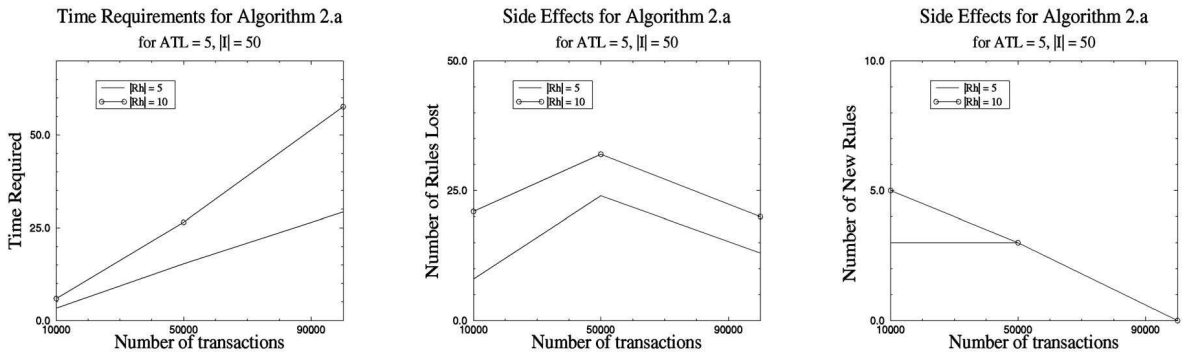
5.3 Performance Evaluation of Algorithm 2.a

The time needed by Algorithm 2.a increases proportionally to $|D|$ and $|R_H|$, as Fig. 7a shows. This is consistent with the theoretical analysis and the validation results. Fig. 7b gives the number of rules mined from the source database that cannot be mined after the hiding process. As it can be seen, the number of nonsensitive rules lost by Algorithm 2.a increases when $|R_H|$ increases. Regarding the peak performance of Algorithm 2.a for the 50k data set, the same considerations made for Algorithm 1.b, apply. Like Algorithm 1.b, new rules are generated by Algorithm 2.a when the deletion of some literals decreases the support of the antecedent of some (minimum support, but not minimum confidence) rules so that their confidence overcomes the minimum threshold. Fig. 7c gives the number of new rules mined from the database after hiding the rules of Tables 5 and 6. As shown, the number of new rules introduced tends to decrease when the number of transactions in the database increases. Furthermore, similar with the Algorithm 1.b, the changes introduced by Algorithm 2.a reduce the number of nonsensitive rules mined from the original database proportionally to $|R_H|$, but slightly affect the number of rules that become nonsensitive.

5.4 Performance Evaluation of Algorithm 2.b

The time requirements of Algorithm 2.b are shown in Fig. 8a for large-scale data. Fig. 8a shows the results when five and 10 rules are hidden for data sets of size 10k, 50k, and 100k. As can be seen from the figure, the time requirements for hiding a set of rules increase linearly with the database size for large data sets as well. In fact, the linear behavior is more obvious for larger scale data. Another observation is that the time requirements for hiding 10 rules are higher than hiding five rules which is also another expected result.

Evaluation of side effects in terms of the lost rules and the new rules as a result of the hiding process are shown in Fig. 8b and 8c. We can see from Fig. 8b that the number of rules lost after hiding 10 rules is higher than the number of rules lost after hiding five rules. This is an intuitive result since hiding more rules means deleting more items, therefore, more side effects in terms of the number of rules lost. One would expect that the size of the database would not change the characteristics of the rules in the database; therefore, the side effects would be more or less similar for



(a) (b) (c)

Fig. 7. Evaluation of Algorithm 2.a.

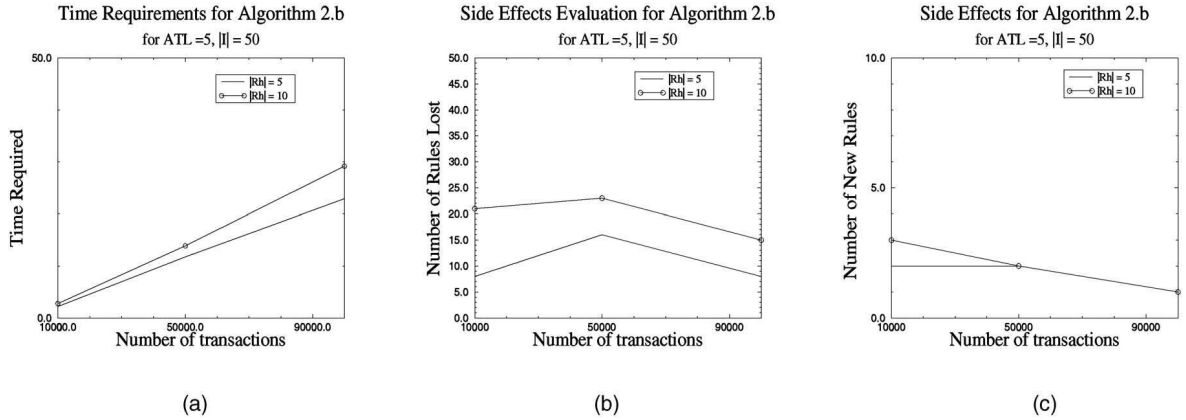


Fig. 8. Evaluation of Algorithm 2.b.

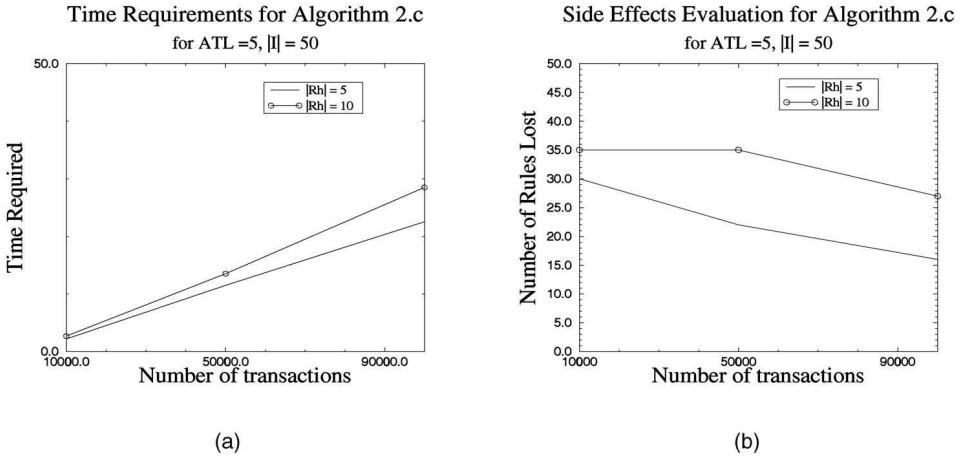


Fig. 9. Evaluation of Algorithm 2.c.

different database sizes. However, we observed an unexpected behavior in the number of rules lost which is shown in Fig. 8b. When we look closely at the rules selected to be hidden, we can see why this happens. Since we choose the rules to be hidden randomly, the selected rule set affects the performance of the algorithms in terms of the side effects. The sets of rules hidden for different database sizes are shown in Table 5 and Table 6. Table 5 shows the five rules selected to be hidden for different databases together with their support and confidence values. Table 6 shows the additional five rules selected to bring the number of rules to be hidden to 10. In the set of five rules selected for hiding for the database of 50k transactions, we can see that there is a rule, $1 \Rightarrow 38$ with support 4.65%, and in the additional five rules, there is a rule, $46 \Rightarrow 48$, with support 3.50%. When we look at the rule sets for 10k and 100k transactions, we can see that all the rules have their support less than 4%. To hide a rule with high support, more items need to be removed from transactions which will increase the number of rules lost. And, the rule with a high support selected for hiding in the 50k transactions case explains the abnormal behavior in the rules lost which is more obvious for the case of hiding five rules.

5.5 Performance Evaluation of Algorithm 2.c

The time requirements of Algorithm 2.c are shown in Fig. 9a for large-scale data. Fig. 9a shows the results when five and 10 rules are hidden for data sets of size 10k, 50k, and 100k. As it can be seen from the figure, the time requirement for hiding a set of rules increases linearly with the database size. Also, the time requirement for hiding 10 rules is higher than hiding five rules which is also an expected result.

Evaluation of side effects in terms of the rules lost as a result of the hiding process are shown in Fig. 9b. The number of rules that are lost when hiding 10 rules is higher that the number in the case of hiding five rules. This is an intuitive result since hiding more rules will affect more rules. We did not include the graphs depicting the number of new rules introduced by Algorithm 2.c since there are no new rules introduced. This may first seem counterintuitive since Algorithm 2.c is a naive algorithm that emphasizes fairness by removing items from transactions in a round-robin fashion. Algorithm 2.c chooses the transactions randomly as opposed to choosing the smallest transaction in size, therefore it will probably choose mostly average size transactions which will have small side effects to the confidence of the other rules. This will be clearer with an example. Suppose that we would like to decrease the support of a rule $A \Rightarrow B$. The smallest possible transaction that supports this rule is $\{A, B\}$ and suppose that such a transaction exists. Removing A from that transaction will cause the confidence of the rules (other than $A \Rightarrow B$) that contain A in their antecedent to increase which will cause the introduction of new rules observing the minimum

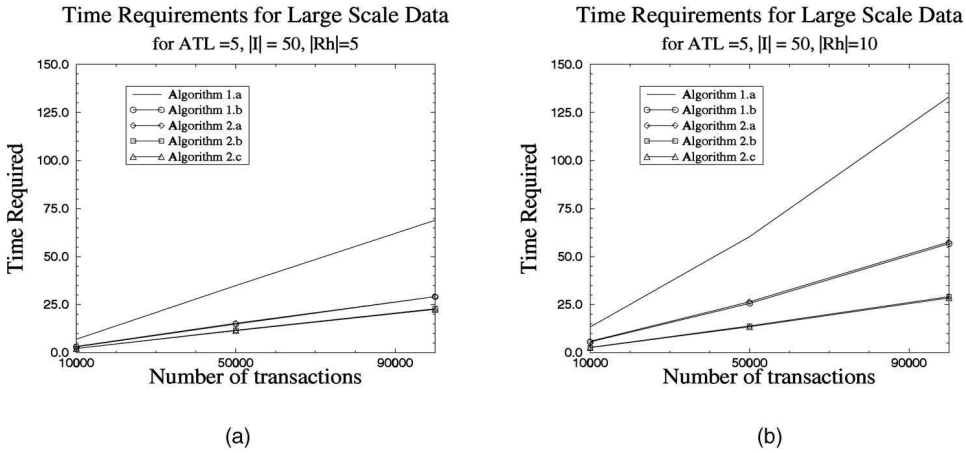


Fig. 10. Time requirements for hiding (a) 5 rules and (b) 10 rules.

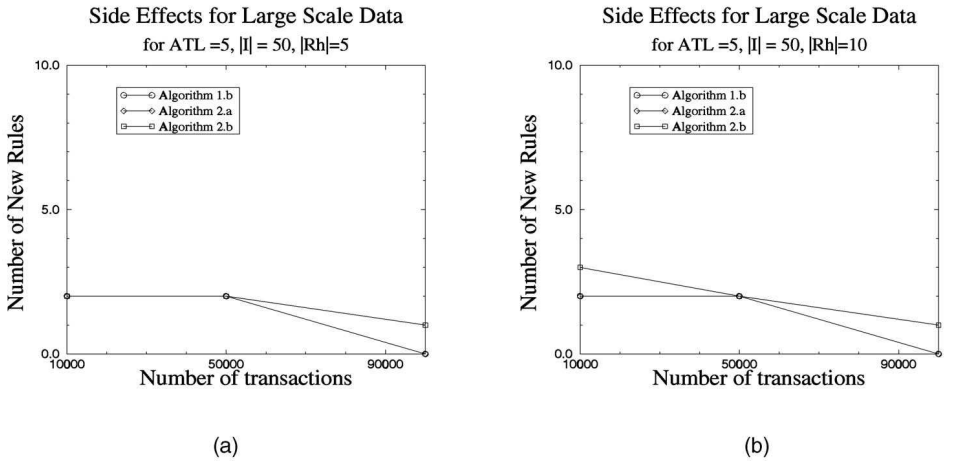


Fig. 11. New rules introduced after hiding (a) 5 rules and (b) 10 rules.

confidence requirement. However, for average size transactions, this will probably not be the case since they will contain both the antecedent and the consequent of the rules and the confidence of these rules will decrease upon the removal of A.

5.6 Comparative Evaluation of the Algorithms

The time requirements of the proposed algorithms are shown in Fig. 10a and Fig. 10b for hiding five and 10 rules, respectively. In both cases, Algorithm 1.b and Algorithm 2.c perform very similar to each other and they have the lowest time requirement. Algorithm 1.a has the highest time requirement.

The evaluation of side effects, in terms of the new rules introduced, is shown in Fig. 11a and 11b for hiding five and 10 rules, respectively. We did not include the number of rules introduced for Algorithm 1.a since it introduces new rules in the order of thousands. On the contrary, Algorithm 2.c does not introduce any new rule; therefore, it is not included in Figs. 11a and 11b either. In terms of the number of rules introduced, all the algorithms behave similarly, introducing only a few rules, except Algorithm 2.c, which does not introduce new rules at all and Algorithm 1.a, which introduces an unacceptable number of new rules.

Performance results in terms of the number of rules lost are shown in Figs. 12a and 12b for five and 10 rules to be hidden. In terms of the number of rules lost, Algorithm 1.a is the best performing algorithm in both Figs. 12a and 12b. Algorithm 2.b also hides fewer rules if compared to the other algorithms in both figures. Algorithm 2.c is the worst with respect to the number of rules lost (see Fig. 12b), but it is still close, in performance, to the other algorithms.

6 Discussion on the Possible Damage in Terms of Query Results

Possible queries on a database of transactions could be selection and projection which may also involve statistical operations, and maybe a temporal extension to those. In terms of data mining, users would like to know what are the maximal set of items purchased having a count greater than a threshold value. These types of queries cannot be answered by standard querying tools. But, queries such as what is the count of transactions where milk and bread are purchased together can be answered by the standard querying tools. Users may also ask queries that return a set of transactions in the database. Given the time-stamp of each transaction, users may want to write queries with a temporal dimension such as how many customer transactions for April, 2002 contain both milk and bread.

Among statistical operations, min, max, and average does not make sense in a database of binary values where

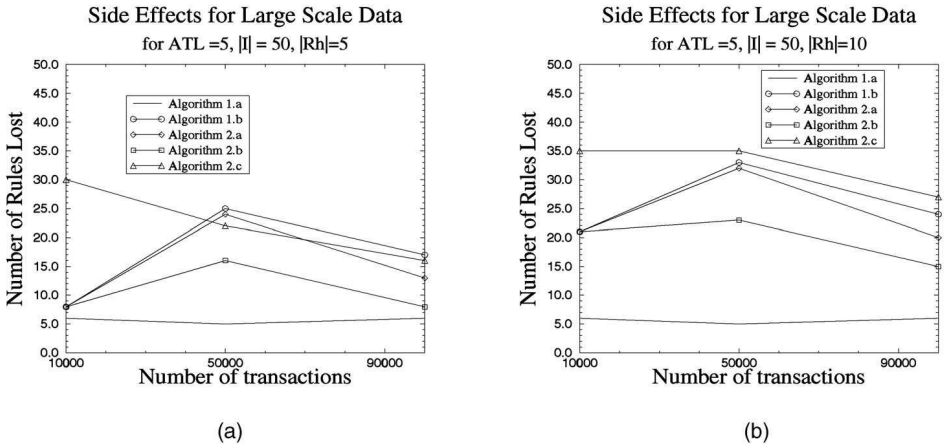


Fig. 12. Rules lost after hiding (a) 5 rules and (b) 10 rules.

quantities of items sold, or price information is not involved. Therefore, we will not consider these types of queries in our discussion.

Queries that return a set of transactions may be considered as microqueries and the queries that return only statistical information can be considered as macrolevel queries. Given a set of rules R_h that are hidden from the database D, we can construct a view of the database D_v, where D_v is a subset of D which consists of the transactions not modified by the hiding process. IDs of modified transactions could be released so that D_v could be easily constructed. Macrolevel queries whose results are a subset of D_v return correct results. The rest of the queries may return incorrect results. This is also true for queries that involve a temporal dimension. In order to improve the correctness of temporal queries, the hiding process may be biased over older transactions, this way ensuring the correctness of queries over more recent transactions.

Queries that contain count operation return correct results if they are issued over items that are not used by the hiding process. We can also give a maximum error range for the count queries which can be used by the user to have a rough idea of the error in the returned count values. This maximum error could be the highest support reduction percentage among the items used by the hiding strategies.

In terms of data mining, the user would like to obtain association rules from the database. In the previous section, we have already discussed the side effects of the hiding process in terms of the hidden or newly appearing association rules.

7 Conclusions

In this paper, we presented two fundamental approaches in order to protect sensitive rules from disclosure. The first approach prevents rules from being generated by hiding the frequent sets from which they are derived. The second approach reduces the importance of the rules by setting their confidence below a user-specified threshold. We developed five algorithms that hide sensitive association rules based on these two approaches. The first three algorithms are rule-oriented. In other words, they decrease either the confidence or the support of a set of sensitive rules, until the rules are hidden. This can happen either because the large itemsets that are associated with the rules are becoming small or because the rule confidence goes below the threshold. The last two algorithms are itemset oriented. They decrease the support of a set of large itemesets until it is below a user-specified threshold, so that no rules can be derived from the selected itemsets.

We also measured the performance of the proposed algorithms according to two criteria: 1) the time that is required by the hiding process and 2) the side effects that are produced. As side effects, we considered both the loss and the introduction of information in the database. We loose information whenever some rules, originally mined from the database, cannot be retrieved after the hiding process. We add information whenever some rules, that could not be retrieved before the hiding process, can be mined from the released database.

We compared the proposed algorithms on the base of the results of these experiments and we concluded that there is not a best solution for all the metrics. The choice of the algorithm to adopt depends on which criteria one considers as the most relevant: the time required, the information loss, or the information that is added.

Some assumptions have been made for the development of the proposed algorithms. We are currently considering extensions on these algorithms by dropping these assumptions. Another interesting issue, which we plan to investigate, is the application of the ideas introduced in this paper to other data mining contexts, such as classification mining, clustering, etc.

Acknowledgments

This work was partially funded by the Information Society Technologies programme of the European Commission, Future, and Emerging Technologies under the IST-2001-39151 CODMINE project.

References

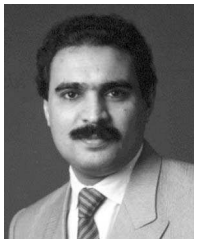
[1] N.R. Adam and J.C. Wortmann, "Security-Control Methods for Statistical Databases: A Comparison Study," *ACM Computing Surveys*, vol. 21, no. 4, pp. 515-556, 1989.

[2] B. Thuraisingham and W. Ford, "Security Constraint Processing in a Multilevel Secure Distributed Database Management System," *IEEE Trans. Knowledge and Data Eng.*, vol. 7, no. 2, pp. 274-293, Apr. 1995.

[3] D.G. Marks, "Inference in MLS Database," *IEEE Trans. Knowledge and Data Eng.*, vol. 8, no. 1, pp. 46-55 Feb. 1996.

[4] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining*. AAAI Press/The MIT Press, 1996.
[5] D.E. O'Leary, "Knowledge Discovery as a Threat to Database Security," *Proc. First Int'l Conf. Knowledge Discovery and Databases*, pp. 107-516, 1991.
[6] C. Clifton and D. Marks, "Security and Privacy Implications of Data Mining," *Proc. 1996 ACM Workshop Data Mining and Knowledge Discovery*, 1996.
[7] C. Clifton, "Protecting against Data Mining through Samples," *Proc. 13th IFIP WG11.3 Conf. Database Security*, 1999.
[8] T. Johnsten and V.V. Raghavan, "Impact of Decision-Region Based Classification Mining Algorithms on Database Security," *Proc. 13th IFIP WG11.3 Conf. Database Security*, 1999.
[9] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios, "Disclosure Limitation Of Sensitive Rules," *Proc. Knowledge and Data Exchange Workshop*, 1999.
[10] R. Agrawal and R. Srikant, "Privacy Preserving Data Mining," *Proc. ACM SIGMOD Conf.*, 2000.
[11] D. Agrawal and C.C. Aggarwal, "On the Design and Quantification of Privacy Preserving Data Mining Algorithms," *Proc. ACM PODS Conf.*, 2001.

Vassilios S. Verykios received the Diploma degree in computer engineering from the University of Patras, Greece, in 1992. From 1992 to 1994, he worked as a software engineer at FIRST Informatics S.A. In 1995, he started graduate studies in the Computer Sciences Department at Purdue University, Indiana, where he received the master's and PhD degrees in 1997 and 1999, respectively. In 1999, he joined the Division of Information Systems in the College of Information Science and Technology at Drexel University, Pennsylvania, as a tenure track assistant professor. He has published more than 30 research papers in various areas such as databases, data mining, database security, and data cleaning, and he has been on the program committees of several international conferences and workshops. He has consulted for Bellcore/Telcordia Technologies, ChoiceMaker Technologies, the US Naval Research Laboratory, and INTRACOM S.A. Since April 2002, he has worked in the Data and Knowledge Engineering Group at the Research and Academic Computer Technology Institute in Patras, Greece. He is a member of the IEEE.



Ahmed K. Elmagarmid received the BSc degree from the University of Dayton, the MS and PhD degrees from the Ohio State University in 1977, 1980, and 1985, respectively. He is responsible for software strategy coming out of the corporate CTO office. As chief scientist in the Office of Strategy and Technology, he contributes to cross company roadmap initiatives and serves on the technology council for Hewlett Packard (HP). He works closely with the business units to identify areas of leverage in the software directions for HP. Dr. Elmagarmid was director of the Indiana Center for Database Systems and the Indiana Telemedicine Incubator. He is on leave from Purdue University where he serves as a professor of computer science. He also served on the faculty at Pennsylvania State University and the University of Padua. He has worked on long-term consulting engagements with Harris Commercial systems, IBM, Bellcore, Telcordia, MDL, UniSql, MCC, CSC, DoD, the Padua Chamber of Commerce, and the Italian Government. He received a US National Science Foundation Presidential Young Investigator award from President Ronald Reagan, and distinguished alumni awards from Ohio State University and the University of Dayton in 1988, 1993, and 1995, respectively. Dr. Elmagarmid is the editor-in-chief of *Distributed and Parallel Databases: an International Journal*, editor of the *IEEE Transactions on Knowledge and Data Engineering*, *Information Sciences Journal*, *Journal of Communication Systems*, and editor of the book series on advances in database systems. He has written six books and more than 150 papers in database systems. He is a senior member of the IEEE.

Elisa Bertino is professor of database systems in the Department of Computer Science at the University of Milan where she is currently the chair of the department. She has also been on the faculty of the Department of Computer and Information Science at the University of Genova, Italy. Until 1990, she was a researcher for the Italian National Research Council in Pisa, Italy, where she headed the Object-Oriented Systems Group. She has been a visiting researcher at the IBM Research Laboratory (now Almaden) in San Jose, at the Microelectronics and Computer Technology Corporation in Austin, Texas, at George Mason University, at Rutgers University, at Purdue University, and at Telcordia Technologies. Her main research interests include database security, object-oriented databases, distributed databases, deductive databases, multimedia databases, interoperability of heterogeneous systems, integration of artificial intelligence, and database techniques. In those areas, Dr. Bertino has published more than 200 papers in all major refereed journals and in proceedings of international conferences and symposia. She is a coauthor of the books *Object-Oriented Database Systems—Concepts and Architectures* (Addison-Wesley International Publishers, 1993), *Indexing Techniques for Advanced Database Systems* (Kluwer Academic Publishers, 1997), and *Intelligent Database Systems* (Addison-Wesley International Publishers, 2001). She is member of the advisory board of the *IEEE Transactions on Knowledge and Data Engineering* and a member of the editorial boards of several scientific journals, incuding the *ACM Transactions on Information and System Security*, *IEEE Internet Computing*, the *Very Large Database Systems (VLDB) Journal*, the *Parallel and Distributed Database Journal*, the *Journal of Computer Security*, *Data and Knowledge Engineering*, the *International Journal of Information Technology*, the *International Journal of Cooperative Information Systems*, and *Science of Computer Programming*. She has been a consultant to several Italian companies on data management systems and applications and has given several courses to industries. She is involved in several projects sponsored by the EEC. She is a fellow of the IEEE and a member of ACM, and has been been named a Golden Core Member for her service to the IEEE Computer Society. She has served as a program committee member of several international conferences, such as ACM SIGMOD, VLDB, ACM OOPSLA, as program cochair of the 1998 IEEE International Conference on Data Engineering (ICDE), as program chair of the 2000 European Conference on Object-Oriented Programming (ECOOP 2000), and as program chair of the Seventh ACM Symposium of Access Control Models and Technologies (SACMAT 2002).

Yucel Saygin received the PhD degree in computer engineering from Bilkent University, Turkey, in 2001. He is currently an assistant professor in the Faculty of Engineering and Natural Sciences, Sabanci University, Turkey. His main research interests include mobile data management, data mining, and data confidentiality against data mining methods. He is a member of the IEEE.

Elena Dasseni graduated from the Computer Science Department at the University of Milan, Italy. She is currently a software engineer for a software house in Milan, Italy, where she's been developing distributed java applications for privacy protection. During the past year, she has been involved in the design and development of the W3C P3P standard for the European Joint Research Centre. She has been a research assistant at the Purdue University ICDS in Lafayette, Indiana, where she contributed in searching and developing strategies to protect sensitive information stored in databases. Her main research interests include security and privacy protection over the networks, criptography and PKI infrastructures, database security, object-oriented databases, distributed databases, and wireless networks.