2000

Hiding Association Rules by Using Confidence and Support

Elena Dasseni

Vassilios S. Verkios

Ahmed K. Elmagarmid
Purdue University, ake@cs.purdue.edu

Elisa Bertino
Purdue University, bertino@cs.purdue.edu

Report Number:
00-010

HIDING ASSOCIATION RULES BY USING CONFIDENCE AND SUPPORT

Elena Dasseni
Vassilios S. Verykios
Ahmed K. Elmagarmid
Elisa Bertino

CSD TR #00-010
June 2000

Hiding Association Rules by Using Confidence and Support

Elena Dasseni[1] Vassilios S. Verykios[2,*] Ahmed K. Elmagarmid[3] Elisa Bertino[1]

[1]Dipartimento di Scienze dell'Informazione, Universita' di Milano, Milano, Italy
[2]College of Information Science and Technology, Drexel University
[3]Department of Computer Sciences, Purdue University

June 13, 2000

Abstract

Large repositories of data contain sensitive information which 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 the government statistical agencies. Recent advances, in data mining and machine learning algorithms, have increased the disclosure risks 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, which are called association rules. If the disclosure risk of some of these rules is above a certain privacy threshold, those rules must be characterized as sensitive. Sometimes, sensitive rules should not be disclosed to the public since, among other things, they may be used for inferencing sensitive data, or they may provide business competitors with an advantage.

1 Introduction

Many government agencies, businesses and non-profit organizations in order to support their short and long term planning activities, are searching for a way to collect, 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.

The necessity to combine the confidentiality and the legitimate needs of data users is imperative. Every disclosure limitation method has an impact, which is not always a positive one, on true data values and relationships. Ideally, these effects can be quantified so that their anticipated impact on the completeness and validity of the data can guide the selection and use of the disclosure limitation method.

The increasing capacity of storing large amounts of data and the necessity to analyze them to support planning activities, have largely contributed to the diffusion of data mining techniques and related methodologies. 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 [1]. On the other hand, the impact of 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 [2], in a paper presented in the 1st International Conference in Knowledge Discovery and Databases. Piatetsky-Shapiro, in GTE Laboratories, was the chair of a mini-symposium 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 lim-

*Contact Author: Vassilios S. Verykios, 3141 Chestnut Street, Philadelphia, PA 19104-2875. tel: 215-895-1532, fax: 215-895-2494, email: verykios@cis.drexel.edu

itation 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 socio-economic 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, (that can also be performed by newly developed data mining techniques), 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 designed 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 describe the scenario below.

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 in reduced price, if we agree to give them access to our database of customer purchases. We accept the deal. Dedtrees now 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. We should recognize though, that the access of a company like the Dedtrees, is in general worthwhile since it improves the efficiency of distribution, lowers the costs and helps to predict inventory needs, even if that gives to Dedtrees the benefit of reducing competition.

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 and results obtained from the applications of the devised algorithms. Issues, concerning the implementation of the algorithms, are discussed in section 6. 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 the source database, augmenting the source database and limiting access to the source database by releasing only samples of the original data. Clifton in [7] adopts the last approach. In his paper, 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 analysis of the data mining techniques should be considered as the first step in the problem of security maintenance: if it is known which selection criteria are used to measure the significance of the induced patters it will be easier to identify what must be done to protect sensitive information from being disclosed. While the solution, which is 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 non-sensitive 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 use these criteria in the context of Decision-Region based algorithms, to inspect and also to modify, if necessary, the description of a sensitive object, so that they can be sure that it's not sensitive.

Disclosure limitation of sensitive knowledge by data mining algorithms, that are 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 induced by such algorithms. Towards this end, they apply a group of heuristic solutions for reducing the number of occurrences, (also referred to as support) of some frequent (large) groups of items, which are selected by the database security administrator, below a minimum user specified threshold. Because an association rule mining algorithm discovers association rules from large sets of items only, by decreasing the support of the selected sets of items has as a consequence that the selected rules escape from the mining. This approach focuses on the first step of the rules mining process, which is the discovery of large itemsets. The second step of the same process (e.g. the derivation of strong rules form frequent sets of items) is the starting point of the approach we will present in this paper.

3 Problem Formulation

Let $I = \{i_1, .., i_n\}$ be a set of literals, called items. Let D be a database of transactions, where each transaction T is an itemset such that $T \subseteq I$. A unique identifier, which we call it TID, is associated with each transaction. We say that a transaction T supports X, a set of items in I, if $X \subset T$. We assume that the items in a transaction or an itemset, are sorted in lexicographic order.

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 $\frac{|XUY|}{|X|} \geq c$ (where $|A|$ is the number of occurrences of the set of items A in the set of transactions D). We say that the rule $X \Rightarrow Y$ has *support* s if $\frac{|XUY|}{N} \geq s$, 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.

Association rule mining algorithms rely on support and confidence when they are searching for implications among sets of items. In this way, algorithms do not retrieve all the association rules that may be derivable from a database, but only a very small subset that satisfies the requirements set by the users. This is actually used as a form of searching bias, in order for the mining algorithm to be computationally more efficient.

An association rule-mining algorithm works as follows. It finds all the sets of items that appear frequently enough, so as to be considered relevant and then derive 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?

In [9] the authors demonstrated that solving this problem (also referred to as "sanitization" problem) is NP-hard; thus, we look 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: we can either prevent the rules in R_h from being generated, by hiding the frequent sets from which they are derived, or we can reduce their confidence by bringing it below a user-specified threshold (*min_conf*). In this paper we propose three strategies to hide rules using the both approach; work related to the former approach can also be found in [9].

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

Table 1: Database D

4 Proposed Solutions and Algorithms

For the simplicity of presentation and without loss of generality, we make the following assumptions in the development of the algorithms:

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

According to the first assumption we can choose to hide a rule by changing either its confidence or its support, but not both. By using this assumption, we can consistently evaluate each technique without any interactions from other techniques.
The second assumption means that, in order to decrease the confidence or the support of a rule, either we turn to 0 the value of a non-zero item in a specific transaction, or we turn to 1 all the zero items in a transaction that partially supports an itemset.
The third assumption states that hiding one rule must be considered as an atomic operation. This implies that the hiding of two different rules should take place in a sequential manner, by hiding one rule after the other. This assumption facilitates the analysis and the evaluation of the techniques. Note that in some cases, this approach may not give results as good as the ones one may get when using a dynamic scheme, in which the list of rules to be hidden can dynamically be reordered after each iteration.
The fourth assumption is based on the minimality of changes in the original database. By changing the confidence or the support of each rule, one step at a time, we act proactively in minimizing the side-effects of the hiding schemes.
The fifth assumption states that we hide only rules that involve disjoint sets of items. In a different situation, interactions among the rules (i.e., common subsets of items) should be considered beforehand.
The rest of this section is organized as follows: in Section 4.1 we introduce the required notation, in Section 4.2 we introduce three strategies that solve the problem of hiding association rules, by tuning the confidence and the support of these rules, while the building blocks of the algorithms that implement those strategies, are presented in Section 4.3.

4.1 Notation

Before presenting the solution strategies, we introduce some notation. Each database transaction is a triple:

t=<TID, list_of_elements, size>

where *TID* is the identifier of the transaction t and *list_of_elements* is a list with one element for each item in the database. Each element has value 1 if the corresponding item is supported by the transaction and 0 otherwise. *Size* is the number of elements in the list of elements having value 1 (e.g., the number of elements supported by the transaction). For example, if $I = \{A, B, C, D\}$, a transaction that contains the items $\{A, C\}$ would be represented as t=< T1, [1010], 2>.
According to this notation, a transaction t supports an itemset S if the elements of *t.list_of_elements* corresponding to items of S are all set to 1. A transaction t partially supports S if the elements of *t.list_of_elements* corresponding to items of S are not all set to 1. For example, if $S = \{A, B, C\} = [1110]$ and p=< T1, [1010], 2>, q=< T2, [1110], 3 > then we would say that q supports S while p partially supports S.

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

Table 2: Database D using the specified notation.

4.2 The Three Hiding Strategies

Decreasing the support of an itemset S means selecting a transaction t that supports S and setting to 0 at least one of the non-zero elements of *t.list_of_elements* that correspond to S. By choosing the minimum number of elements to modify (i.e., 1), we minimize the impact of hiding a sensitive rule in the database. In the example above, in order to decrease the support of S using q, we can turn to 0 the element corresponding to C, obtaining q=< T2, [1100], 2 >.
Increasing the support of an itemset S through a transaction t that partially supports it, means setting to 1 all the elements in *t.list_of_elements* corresponding to items of S. The increase of the support of an itemset that has the maximum number of elements in common with the itemset S, also contributes to minimizing the side-affects of hiding. Referring to the example above, in order to increase the support of S through p we must turn to 1 the element corresponding to the item B, obtaining p=< T1, [1110], 3 >.
Given a rule $X \Rightarrow Y$, we can write its confidence in terms of its support as follows:

$$Conf(X \Rightarrow Y) = \frac{Supp(X \cup Y)}{Supp(X)}$$

Starting from this relationship between the confidence and the support of a rule, we develop three strategies to hide a rule:

1. We decrease the confidence of the rule
   (a) by increasing the support of the rule antecedent X, through transactions that partially support it.
   (b) 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
   (a) by decreasing the support of either the rule antecedent X, or the rule consequent Y.

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

Table 3: The large itemsets from the database of Table 1 along with their support.

Example
Supposing that we have the database D shown in Table 1. According to the notation introduced above, the representation of the database is given in Table 2). Given that *min_supp*=2/6=33% and *min_conf*= 70% we are interested in hiding the rule $AC \Rightarrow B$, with support = 50%, and confidence = 75%.
Strategy 1.a We select the transaction t=< T5, [100], 1 > and turn to 1 the element of the list of item that corresponds to C. We obtain t=< T5, [101], 2 >. Now, the rule $AC \Rightarrow B$ has support=50% and confidence=60%, which means that the rule has been hidden since its confidence is below the *min_conf* threshold.
Strategy 1.b 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% and confidence=50%, which means that the rule is hidden.
Strategy 2.a We select the transaction t=< T1, [111], 3 > and turn to 0 one of the elements in the list of items that corresponds to A or to B or C. 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%, confidence=66%).

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

Table 4: The rules derived from the large itemsets of Table 3

4.3 Algorithms and Data Structures

We now present the algorithms for the previously introduced strategies. For each algorithm we specify the input and output requirements and we give a brief description of the data structures needed.

4.3.1 Algorithm 1.a

This algorithm hides 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. Figure 1 shows the sketch of this algorithm; a refinement of the algorithm is depicted in Figure 2. In both Figure 1 and 2 we used the compact notation $lhs(U)$ to represent the itemset on the left side of a rule U (also referred to as "rule antecedent").

4.3.2 Algorithm 1.b

This algorithm hides sensitive rules in accordance to the second of the proposed strategies. It reduces the support of each selected rule by decreasing the frequency of the consequent through transactions that support the rule. This process goes on until the rule confidence is below the minimum threshold. Figure 3 and 4 show the building blocks of algorithm 1.b. The compact notation $rhs(U)$ denotes the large itemset on the right side of a rule (also referred as "rule consequent").

INPUT: a set R_h of rules to hide, the source 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 U in R_h **do**
  {
    repeat until ($conf(U) <$ *min_conf*)
    {
      1. $T = \{ t$ in D / t partially supports $lhs(U) \}$
      2. count the number of items in each transaction of T
      3. sort the transactions in T in descending order of the number of supported items
      4. choose the transaction $t \in T$ with the highest number of items (the first transaction in T)
      5. modify t to support $lhs(U)$
      6. increase the support of $lhs(U)$ by 1
      7. recompute the confidence of U
    }
    8. remove U from R_h
  }
End

Figure 1: Sketch of Algorithm 1.a

4.3.3 Algorithm 2.a

This algorithm decreases the frequency of the sensitive rules until either their confidence is below the *min_conf* threshold or their support is below the *min_supp* threshold. Figure 5 shows the sketch of algorithm 2.a; more details about the steps it requires are given in Figure 6.

5 Performance Evaluation and Analysis Results

We performed our experiments on a Dell workstation with **P3 500** MHz processor and with **128 MB** of main memory, under Solaris **2.6** operating system.

```
Begin
   Foreach rule U in R_h do
   {
       While (conf(U) ≥ min_conf)
       {
            T = {t ∈ D/ t partially supports lhs(U)}
            // count how many items of lhs(U) are
            // in each trans. of T
            foreach transaction t in T do
            {
               t.num_itmes= |I|-
               Hamming_dist(lhs(U),t.list_of_items)
            }
            // sort transactions of T in descending
            // order of number of items of lhs(U)
            // contained
            sort(T)
            // pick the transaction of T with the
            // highest number of items
            t = T[1]
            // set to one all the bits of t that
            // represent items in lhs(U)
            set_all_ones(t.list_of_items, lhs(U))
            supp(lhs(U))=supp(lhs(U))+1
            conf(U)=supp(U)/supp(lhs(U))
       }
       R_h = R_h − U
   }
End
```

Figure 2: Refinement of Algorithm 1.a

INPUT: a set R_h of rules to hide, the source 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 U in R_h **do**
{
repeat until ($conf(U)$ <*min_conf*)
{
1. T = { t in D / t supports U }
2. choose the transaction t in T with the lowest number of items
3. choose the item j in $rhs(U)$ with the minimum impact on the $(|rhs(U)| - 1)$-itemsets
4. delete j from t
5. decrease the support of U by 1
6. recompute the confidence of U

}
7. remove U from R_h
}
End

Figure 3: Sketch of Algorithm 1.b

In order to generate the source databases, we made use of the IBM synthetic data generator. The generator creates output files in text format, which can be understood by the programs which implement our heuristics. The input to the synthetic data generator, among other parameters, is the database size ($|D|$), the number of literals appearing in the database ($|I|$) and the average number of items per transaction (ATL).

We performed two trials for each of the three algorithms: the goal of the first trial was to analyze the behavior of the developed algorithms when the number of literals appearing in the database increases; the second trial aimed at studying the behavior of the algorithms when the number of rules selected for hiding ($|R_h|$) increases.

For the first trial we generated 10 datasets of size $1K$, $1.5K, 2.5K, 4.2K, 5K, 6.7K, 7.5K, 8K, 9K$ and $10K$ respectively, for each tested value of $|I|$. Each one of the generated databases has an average transaction length of 5 items. Since we tested our algorithms for 4 different values of $|I|$ (20, 30, 40 and 50 items), we generated 4 groups (also referred to as "series") of 10 databases (one for each value of $|I|$). Details about the characteristics of the datasets, used in our first trial, are given in Table 5.

For the representation of the datasets in each series we used the compact notation 10:[1k..10k] to indicate that each series is made up of 10 databases and that the size of each database lies in the range 1k-10k. The parameter $|R_h|$ in Table 5 indicates that in our first trial we ran the proposed algorithms in order to hide a set of 2 disjoint rules (randomly selected among those having minimum confidence).

For the second trial, we used the first series of 10 datasets of Table 5 to hide sets of 2, 3 and 4 rules; Table 6 shows the values of the parameters $|D|$, $|I|$ and $|R_h|$ for our second trial. In the rest of this sec-

```
Begin
    Foreach rule U in R_h do
    {
        While (conf(U) ≥ min_conf)
        {
            T = {t ∈ D/ t supports U}
            // sort T in ascending order of
            // size of the transactions and choose
            // the one with the lowest size
            t = choose_transaction(T)
            // choose the item of rhs(U)
            // with the minimum impact on the
            // (|rhs(U)| − 1)-itemsets
            j = choose_item(rhs(U))
            // set to zero the bit of t.list_of_items
            // that represents item j
            set_to_zero(j, t.list_of_items)
            supp(U)=supp(U) - 1
            conf(U)=supp(U)/supp(lhs(U))
        }
        R_h = R_h − U
    }
End
```

Figure 4: Refinement of Algorithm 1.b

Series	$\|D\|$	$\|I\|$	$\|R_h\|$	ATL
1	10:[1k..10k]	20	2	5
2	10:[1k..10k]	30	2	5
3	10:[1k..10k]	40	2	5
4	10:[1k..10k]	50	2	5

Table 5: The datasets used in the first trial.

tion we will present a brief analysis for each one of the algorithms, followed by a discussion of the results obtained from the above experiments.

5.1 Analysis and Performance Evaluation of Algorithm 1.a

The time required by algorithm 1.a to hide a set of rules R_h is $O(\sum N(R_j) * \{|D| * |I| + |T_{j1}| * \log|T_{j1}|\})$, where j ranges from 1 to $|R_h|$, $N(R_j)$ is the number of executions of the inner loop performed to hide the rule R_j, $|D|$ is the number of transactions in the database, $|I|$ is the number of literals in D and $|T_{j1}|$ is the number of transactions partially supporting the

```
INPUT: a set R_h of rules to hide, the source
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 U in R_h do
    {
        repeat until (conf(U) <min_conf
                      or supp(U) <min_supp)
        {
            1. T = { t in D / t supports U }
            2. choose the transaction t in T
               with the lowest number of items
            3. choose the item j in U
               with the minimum impact on the
               (|U| − 1)-itemsets
            4. delete j from t
            5. decrease the support of U by 1
            6. recompute the confidence of U
        }
        7. remove U from R_h
    }
End
```

Figure 5: Sketch of Algorithm 2.a

left side of R_j during the first execution of the inner loop. We ran the experiments discussed at the beginning of this section using algorithm 1.a to hide 2 disjoint rules, under the constraint $N(R_j) = 3$. The significance of fixing the value of $N(R_j)$ is to simulate the hypothesis that rules with similar support and confidence are chosen for hiding in each dataset. From Figure 9 and 10 we can easily observe that - under the above constraint - our algorithm 1.a is linear in the size of the database and directly proportional to $|I|$ (Figure 9) and to $|R_h|$ (Figure 10).

5.2 Analysis and Performance Evaluation of Algorithm 1.b

The time required by algorithm 1.b is $O(\sum N(R_j) * \{|D| * |I| + |T_{j1}| * \log|T_{j1}| + |rhs(R_j)| * |I| * |R|)$, where $|rhs(R_j)|$ is the number of literals in the itemset on the right side of the rule R_j and j ranges from 1 to $|R_h|$. We tested algorithm

```
Begin
    Foreach rule U in R_h do
    {
        While (conf(U) ≥ min_conf
                and supp(U) ≥ min_supp)
        {
            T = {t ∈ D/ t supports U}
            // sort T in ascending order of
            // size of the transactions and choose
            // the one with the lowest size
            t = choose_transaction(T)
            // choose the item of U
            // with the minimum impact on the
            // (|U| − 1)-itemsets
            j = choose_item(U)
            // set to zero the bit of t.list_of_items
            // that represents item j
            set_to_zero(j, t.list_of_items)
            supp(U)=supp(U) - 1
            conf(U)=supp(U)/supp(lhs(U))
        }
        R_h = R_h − U
    }
End
```

Figure 6: Refinement of Algorithm 2.a

Series	$\|D\|$	$\|I\|$	$\|R_h\|$	ATL
1	10:[1k..10k]	20	2	5
2	10:[1k..10k]	20	3	5
3	10:[1k..10k]	20	4	5

Table 6: The datasets used in the second trial.

1.b on the same datasets and under the same constraints used for algorithm 1.a (no constraints were imposed neither on $|R|$ nor on $|rhs(R_j)|$). The results of these experiments are shown in Figure 11 and 12. ¿From these pictures we can see that the time of algorithm 1.b is linear in the size of the database, and in the number of literals that appear in the database increase (Figure 11) and in the number of rules selected for hiding (Figure 12). Notice also, that these graphs indicate that algorithm 1.a performs slightly better than algorithm 1.b.

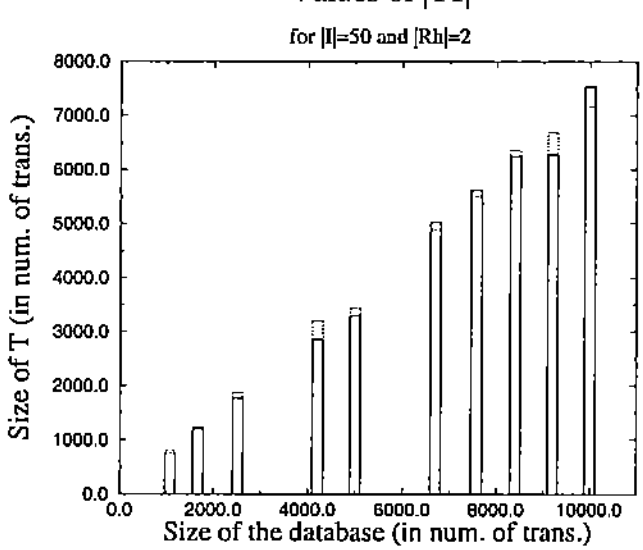

Figure 7: The values of $|T_{j1}|$ for the last series of Table 5

5.3 Analysis and Performance Evaluation of Algorithm 2.a

Algorithm 2.a performs in $O(\sum N(R_j) * \{|D| * |I| + |T_{j1}| * \log|T_{j1}| + |R_j| * |I| * |R|\})$ time, where $|R_j|$ is the number of literals that appear in the rule R_j, $|R|$ is the number of rules that can be mined from the database D and j ranges from 1 to $|R_h|$. Figure 13 and 14 show the results of the trials discussed at the beginning of this section, when using algorithm 2.a to hide the rules. Again, if the same rules where chosen for hiding in each dataset, algorithm 2.a would perform in a time directly proportional to the size of the database. Its time requirements would also increase linearly when increasing the number of literals appearing in the database or the number of rules selected for hiding.

6 Implementation Issues

In Section 4.3 we introduced the following data structures which are manipulated by the proposed algorithms:

- sets of transactions (D and T)
- set of rules (R_h)
- set of large itemsets (R)

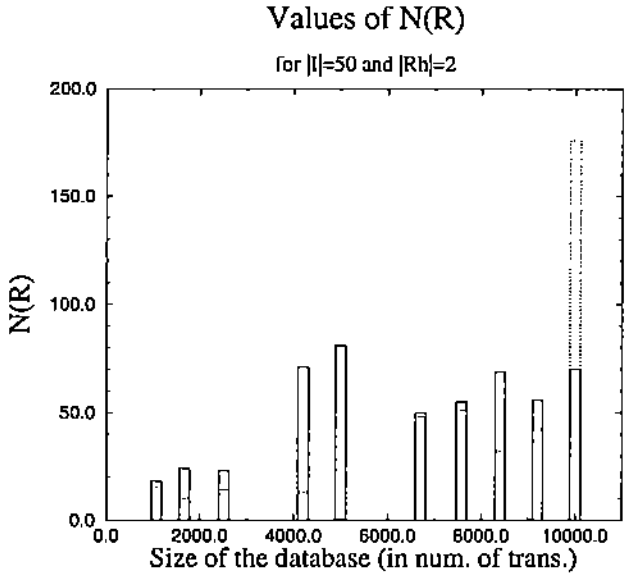

Figure 8: The values of $N(R_j)$ for the last series of Table 5

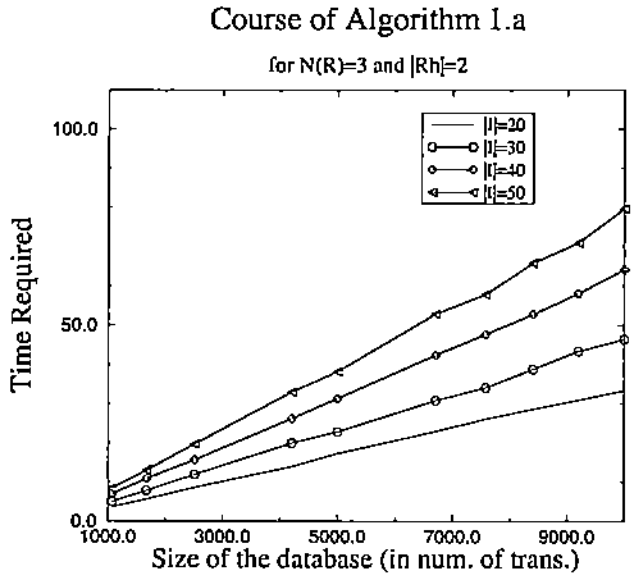

Figure 9: Results of the first trial

- list of items I, that appear in the database D.

We now give a brief description of how each data structure has been implemented.

For the SET_OF_ELEMENTS data structure we adopted the following implementation: each set has been represented as an array of references to the elements of this set. These elements can be: transactions for D and T, rules for R_h and large itemsets for R. The elements of a set have been represented as arrays. According to this convention, each transaction t is an array, with one field storing the transaction ID (TID), another one storing the list of items contained in a transaction and still another one storing the number of these items. The list of items has been implemented as a hash structure which has literals for keys and integers for values. The number of keys equals I and the value associated with each key is either 1 or 0, depending whether the literal is supported by the transaction or not. The same implementation has been adopted to represent the list of items contained in a large itemset, which turns out to be an array with one field for the list of items and one filed storing the support. Each rule U has been implemented as an array with 4 fields, storing the confidence of the rule, a reference to the large itemsets representing the rule antecedent and the rule consequent and a reference to the large itemset from which the rule has been derived. Finally, to represent the list I of literals appearing in the database, we used an array of strings.

Generation of the set T The set T of transactions (partially) supporting a large itemset l is generated by analyzing each transaction t in D and checking if the list of items in the transaction equals the list of items in large itemset (for algorithm 1.a we check if the items in t are a proper subset of l).

Computing the Hamming distance (Algorithm 1.a) This function computes the distance between two lists of items. Since we represented the LIST_OF_ITEMS data structure as a hash structure with $|I|$ keys, each list has the same number of elements. For each literal (key), we compute the difference between the corresponding values, and add 1 to the Hamming distance counter, if this difference is 1.

Choosing the best transaction (Algorithm 1.b) The decrease in the frequency of an itemset l, is made through a transaction that minimizes the side-effects in the remaining set of rules. This transaction is selected among the set of transactions that support l, and it is the one with a minimum number of non-zero items. To do this, we sort the set T in decreasing order of size of transactions and then pick the first transaction from T.

Choosing the best item (Algorithm 1.b) The best item (in a large itemset l) to delete from a trans-

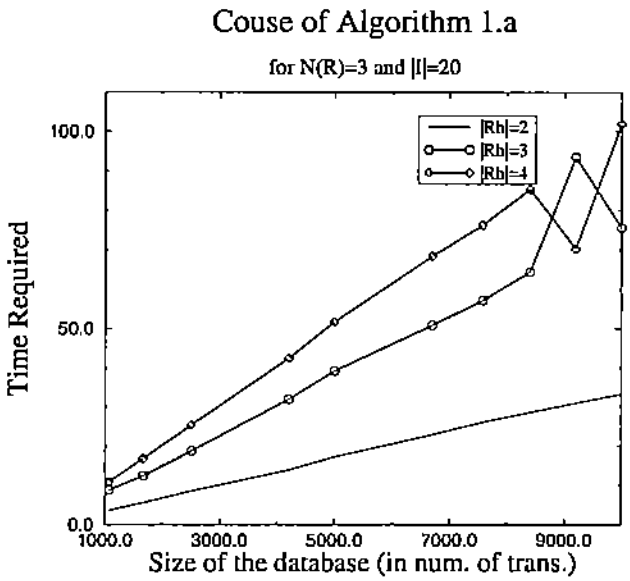

Figure 10: Results of the second trial

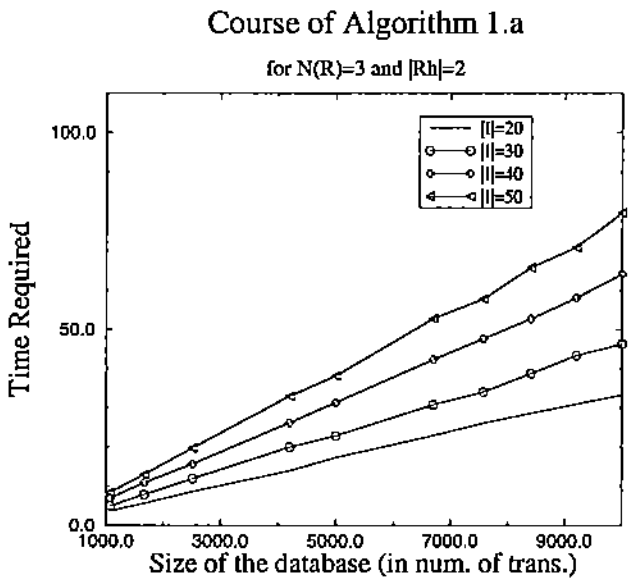

Figure 11: Results from the second trial

action t is chosen so to have the minimum impact on the set of large itemsets. To do this, we determine all the $(|l|-1)$-itemsets and pick one item from $(|l|-1)$-itemset with the lowest support.

7 Conclusions

To protect sensitive rules from disclosure, two main approaches can be adopted. We can either prevent rules from being generated, by hiding the frequent sets from which they are derived, or we can reduce their importance by setting their confidence below a user-specified threshold. We developed three strategies that hide sensitive association rules based on these two approaches. These strategies work either on the support or on the confidence of the rules, by decreasing either one of these until the rule is not important.

Some assumptions have been made when developing these strategies. We are currently considering extensions on these algorithms by dropping these assumptions. We also need to define some metrics to measure the impact that our algorithms have on the source database. Another interesting issue which we will be investigating, is the applicability of the ideas introduced in this paper to other data mining contexts, such as classification mining, clustering, etc.

References

[1] U. Fayyad, G. Piatetsky-Shapiro and P. Smyth. "From Data Mining to Knowledge Discovery: An Overview". In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 1-34. AAAI Press/MIT Press, 1996.

[2] 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*, pages 507-516, AAAI Press/MIT Press, Menlo Park, CA, 1991.

[3] R. Agrawal, T. Imielinski and A. Swami. "Mining Association Rules between Sets of Items in Large Databases". In *Proceedings of the ACM-SIGMOD Conference on Management of Data*, pages 207-216, Washington, DC, 1993.

[4] R. Agrawal, H. Mannila, R. Srinkant, H. Toivonen and A. I. Verkamo. "Fast Discovery of Association Rules". In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307-328. AAAI Press/MIT Press, Menlo Park, CA, 1996.

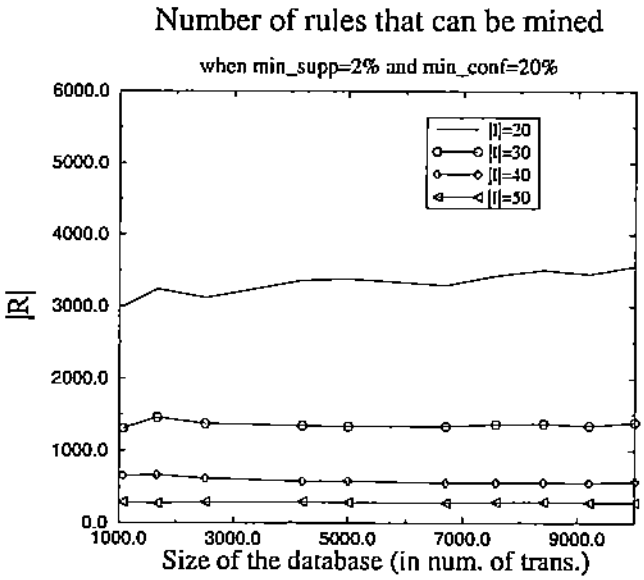

Figure 12: Experienced values of $|R|$

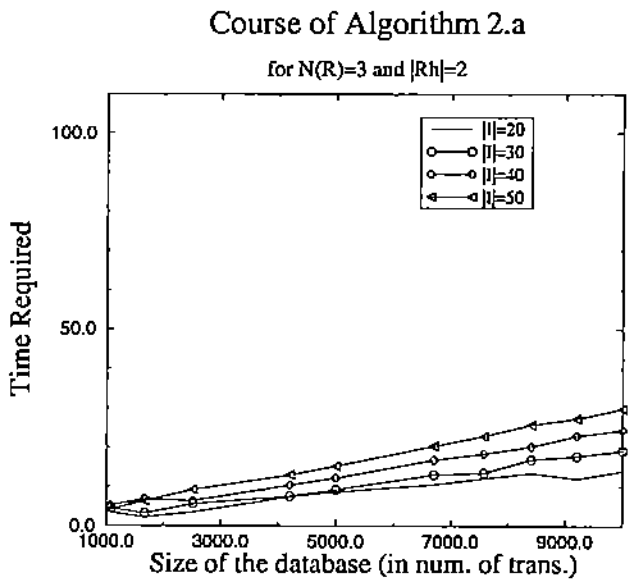

Figure 13: Results from the first trial

[5] R. Agrawal, R. Srikant. "Fast Algorithm for Mining Association Rules". In J. Bocca, M. Jarke and C. Zaniolo, editors, *Proceedings of the Twentieth International Conference on Very Large Data Bases (VLDB '94)*, pages 487-499, Santiago, Chile, 2994. Morgan Kaufmann.

[6] C. Clifton and D. Marks. "Security and Privacy Implication of Data Mining". In *Proceedings of the 1996 ACM Workshop on Data Mining and Knowledge Discovery*, 1996.

[7] C. Clifton. "Protecting against Data Mining through Samples". In *Proceedings of 13th IFIP WG11.3 Conference on Database Security*, Seattle, Washington, 1999.

[8] T. Johnsten, Vijay V. Raghavan. "Impact of Decision-Region Based Classification Mining Algorithms on Database Security". In *Proceedings of thirteenth IFIP WG11.3 Conference on Database Security*, Seattle, WA, Jul. 1999

[9] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, V. Verykios. "Disclosure Limitation Of Sensitive Rules". In *Proceedings of KDEX'99*, Chicago, IL, 1999.

[10] R. Agrawal. "Data Mining: Crossing The Chasm". In *Proceedings of 5th Int'l Conference on Knowledge Discovery in Databases and Data Mining*, San Diego, California, August 1999.

[11] R. Agrawal, R. Srikant. "Privacy-Preserving Data Mining". In *PODS'2000*. IBM Almaden Reserach Center, San Jose, California.

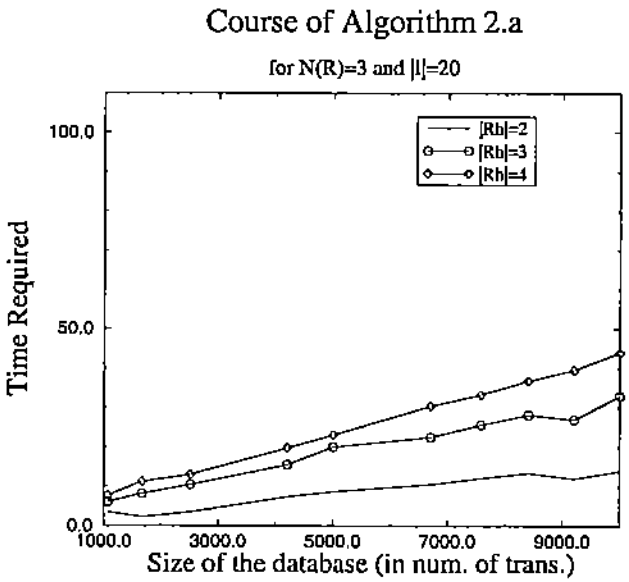

Figure 14: Results from the second trial