Privacy Preserving Association Rule Mining

Yücel Saygın[1], Vassilios S. Verykios[2], Ahmed K. Elmagarmid[3]

[1]Faculty of Engineering and Natural Sciences, Sabancı University, Turkey
[2]College of Information Science and Technology, Drexel University, USA
[3]Computer Sciences Department, Purdue University, USA

Abstract

The current trend in the application space towards systems of loosely coupled and dynamically bound components that enables just-in-time integration jeopardizes the security of information that is shared between the broker, the requester, and the provider at runtime. In particular, new advances in data mining and knowledge discovery, that allow for the extraction of hidden knowledge in enormous amount of data, impose new threats on the seamless integration of information. In this paper, we consider the problem of building privacy preserving algorithms for one category of data mining techniques, the association rule mining. We introduce new metrics in order to demonstrate how security issues can be taken into consideration in the general framework of association rule mining, and we show that the complexity of the new heuristics is similar to this of the original algorithms.

1 Motivation

In a dynamic, unstable and ever changing business environment like that where enterprises conduct e-businesses, the old fashioned disclosure control and database inference protection techniques are inadequate to ensure complete data privacy. In a recent news article, fears were expressed for the online security of private information because a pharmaceutical company said that it had inadvertently released over the internet the e-mail addresses of more than 600 of its customers who were on some special type of medication. Although this is an extreme example of direct disclosure, it signifies the multiple risks that companies may run into, if they do not consider seriously the risks of not securing the sensitive information that they manipulate. For this reason, organizations should be able to evaluate the risk of disclosing information and proceed in adopting new more efficient approaches for information disclosure control, in order to maintain their competitive edge in the market.

The proliferation of new data mining techniques have increased the privacy risks because now it is possible to efficiently combine and interrogate enormous data stores, available on the web, in the search of previously unknown hidden patterns. In order to make a publicly available system secure, we must ensure not only that private sensitive data have been trimmed out, but also to make sure that certain inference channels have been blocked as well. In other words it is not only the data but the hidden knowledge in this data, that should be made secure. Moreover, the need for making our system as open as possible - to the degree that data sensitivity is not jeopardized - asks for various techniques that account for the disclosure control of sensitive data.

Government agencies make their data publicly available on the Web while some dot com companies sell their repository of customer records collected over time to another company. In both cases there may be some sensitive information that can be extracted by malicious users. The sensitive information can be extracted in the form of association rules with now popular association rule mining tools. For example, a rule which tells us that if a female customer buys a certain perfume then she also buys Prosac. Although this seemingly generic rule does not contain any personal information, it jeopardizes the privacy of the female customers since knowing this rule, someone can sniff a Prosac using female customer. Some other rule could be very critical for the company itself such as buying patterns of very rich customers.

Sensitivity of a rule is a semantic notion and it has a temporal dimension too. For example, an internet based company may give up selling hardware and may concentrate on selling books and videos. Therefore a rule relating the customer buying patterns of hardware may no longer be sensitive for that company.

For hiding sensitive rules, it is more desirable to replace a real value by an unknown value instead of placing a false value. Consider a medical institution which will make some of its data public, and the data is sanitized by replacing ac-

tual attribute values by false values. Misleading rules could be obtained from this sanitized data by using data mining tools. These misleading rules, when used for critical purposes (like diagnosis) may jeopardize patients' lives. Therefore, critical applications require that the sanitization process place unknown values instead of false values. Even for non-critical applications unknown values are preferable to false values when the data is going to be sold to another company. This is due to the fact that insertion of false values would degrade the quality of the released data. It also is not desirable to apply a trivial algorithm that hides data by deleting randomly since it will hide lots of rules as a side effect, degrading the quality of the released or sold data.

In this paper, we provide a framework for association rules when the data set contains unknown values and we propose an innovative technique for hiding rules (i.e., knowledge) from a data set using unknown values.

The rest of the paper is organized as follows. In Section 2 we present some background information and the notation used in the rest of the paper. In Section 3 we introduce new metrics required for dealing with sensitive association rules. Section 4 provides an outline of the rule hiding process and demonstrates it by using an example. In Section 5, we present three algorithms that we developed for rule hiding and we comment on their performance. Section 6 presents some initial results from experiments that we have performed by using real data sets. Section 7 summarizes the related work in the area of privacy preserving data mining rules. Finally, we conclude our discussion in Section 8.

2 Background

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 $X \subseteq T$. The support of X on the other hand is the percentage of transactions that contain X. 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 $\frac{|XUY| \times 100}{|X|} = c$ (where $|A|$ is the number of occurrences of the set of items A in the set of transactions D). We also say that the rule $X \Rightarrow Y$ has *support* $s\%$ if $\frac{|XUY| \times 100}{N} = s$, 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. Because the number of itemsets and association rules increases exponentially with the number of items in the database, we only consider association rules which have support and confidence higher than two user specified thresholds: the Minimum Support Threshold *MST* and Minimum Confidence Threshold *MCT*.

In the context of the current work, we assume that an association rule (and its corresponding large itemset thereof) is also characterized by yet another metric, which we call it, the *sensitivity level*. The sensitivity level of a rule denotes whether the rule is sensitive or not. For the sake of this presentation, we assume that a rule whose support and confidence is below the *MST* and *MCT* is not sensitive. In other words, the sensitivity depends entirely on these two other metrics. In a general framework of sensitivity analysis, we consider that other factors affect the sensitivity of the rule (i.e., the rule refers to products of third parties). In our previous work [3, 9, 7] we have demonstrated how to hide a certain set of association rules which is considered sensitive from the database by using the support and the confidence of these rules. It is straightforward that if we turn to 0 the 1-values that provide support to a large itemset, then the support of the corresponding rule decreases, and consequently the rule is not sensitive any more.

3 Privacy Preserving Association Rules

In order to extend the idea of association rule discovery to privacy preserving association rule mining, we need to make some modifications to the original setting. First, we need to introduce a new symbol in the alphabet of an item. The possible set of values of an item in the new setting becomes $\{0, 1, ?\}$. For example, the value in the i^{th} position of a transaction is 1 if the transaction contains (or supports) the i^{th} item and, the value is 0 otherwise. A "?" mark in the i^{th} position of a transaction means that we do not have any information regarding whether the transaction contains the i^{th} item or not. With the new approach that involves "?" marks, the definition of support should be modified. Instead of a single value for the support of an itemset A, we have a *support interval*, $[minsup(A), maxsup(A)]$ where the actual support of itemset A can be any value between $minsup(A)$ and $maxsup(A)$. The $minsup(A)$ is the percentage of the transactions that contain 1's for all the items in A and $maxsup(A)$ is the percentage of the transactions that contain either 1 or "?" mark for all the items in A.

The confidence formula should also be modified since it will also have a degree of uncertainty. Instead of a single value for the confidence of a rule $A \Rightarrow B$, we have a *confidence interval* $[minconf(A \Rightarrow B), maxconf(A \Rightarrow B)]$, where the actual confidence of a rule $A \Rightarrow B$ can be any value between $minconf(A \Rightarrow B)$ and $maxconf(A \Rightarrow B)$. Given the minimum and maximum support values of itemsets AB and A, the minimum confidence value for a rule $A \Rightarrow B$ is, $minconf(A \Rightarrow B) = minsup(AB) \times 100/maxsup(A)$, and the maximum confidence value is $maxconf(A \Rightarrow B) = maxsup(AB) \times 100/minsup(A)$.

Note that $minconf(A \Rightarrow B) = maxconf(A \Rightarrow B)$, and $minsup(AB) = maxsup(AB)$ when there are no unknown values (i.e., "?" marks) in the database. During the sanitization process, when we start placing "?" marks, the minimum and maximum values will start to set apart, and in this way, the degree of uncertainty for the rule, will increase.

4 Sensitive Association Rule Hiding

Given a set of rules R extracted from the database with a certain minimum confidence and support threshold, we assume that sensitive rules in R are determined by the experts. The purpose of the rule hiding algorithms is to make the sensitive rules invisible to the association rule mining algorithms while giving as little harm as possible to the remaining non-sensitive rules to keep the data quality as high as possible. In order to hide a rule $A \Rightarrow B$, we can either decrease the support of the itemset AB below the minimum support threshold, or we can decrease the confidence below the minimum confidence threshold. This can be accomplished by placing "?" marks in place of the actual values to increase the uncertainty of the support and confidence of the rules (i.e., length of the support and confidence intervals). Considering the support interval and the minimum support threshold (MST) which is a point, we may have the following cases for an itemset A:

- A is hidden when the $minsup(A)$ is greater than or equal to *MST*,
- A is still visible when $maxsup(A)$ is smaller than *MST*,
- A is visible with a degree of uncertainty when $minsup(A) \leq$ *MST* $\leq maxsup(A)$

The same reasoning applies to the confidence interval and the minimum confidence threshold (*MCT*). Note that it is possible for the support of a rule to be above the *MST*, and for the confidence to have a degree of uncertainty and vice versa. Also, both the confidence and the support may be above the threshold.

From a rule hiding point of view, in order to hide a rule $A \Rightarrow B$ by decreasing its support, the only way is to replace 1's by "?" marks for the items in AB. In this way, we will only change the minimum support value while the maximum support value will be the same. As we replace 1's by "?" marks for the items in AB, the minimum support value of $A \Rightarrow B$ will decrease and after some point it will go below the minimum support threshold.

In order to hide a rule, $A \Rightarrow B$, by decreasing its confidence, we can replace both 1's and 0's by the "?" mark. The confidence interval of $A \Rightarrow B$ is $[minconf(A \Rightarrow B), maxconf(A \Rightarrow B)]$ and our aim is to decrease the $minconf(A \Rightarrow B)$ below the *MCT*. Recall that $minconf(A \Rightarrow B) = minsup(AB) \times 100/maxsup(A)$. So we should decrease $minsup(AB)$ and/or increase $maxsup(A)$. The $minsup(AB)$ can be decreased by either placing a "?" mark in place of either A or B. If we place a "?" mark in place of A then $minsup(A)$ will also decrease, which will cause an increase in the maximum confidence value, since $maxconf(A \Rightarrow B) = maxsup(AB) \times 100/minsup(A)$. For rule hiding, it would be desirable to keep the maximum confidence as low as possible, and for this reason, it is better to place a "?" mark for B. In order to increase $maxsup(A)$, we should replace the 0 values for the items in A with a "?" mark. Note that this process may cause an increase in the maximum support values of other rules as a side effect.

TID	A	B	C	D
T_1	1	1	0	1
T_2	0	1	0	0
T_3	1	0	1	1
T_4	1	1	0	0
T_5	1	1	0	1

Table 1. Sample Database of Transactions

TID	A	B	C	D
T_1	?	1	0	1
T_2	0	1	0	0
T_3	1	0	1	?
T_4	1	?	0	0
T_5	1	?	0	1

Table 2. Sample Database of Transactions with Unknown Attribute Values

A sample database of transactions is shown in Table 1. The database consists of 5 transactions whose items are drawn from the set $\{A, B, C, D\}$. For this database, when we set the minimum support threshold to 50% and the minimum confidence threshold to 70%, the frequent (large) items are A, B, and D with supports 80%, 80%, and 60%, respectively. Frequent itemsets of size 2 are the AB, and AD with support 60%. The rules obtained from these large itemsets are $A \Rightarrow B$, and $A \Rightarrow D$ both having 75% confidence. Table 2 shows a database with unknown attribute values. In case of unknown attribute values, we previously defined the concepts of minimum support and maximum support, as well as the minimum confidence and maximum confidence. For example, $minsup(A) = 60\%$, and $maxsup(A) = 80\%$. When we set the minimum support threshold to 50%, we see that

both $minsup(A)$ and $maxsup(A)$ are above the minimum support threshold. However, for item B, $minsup(B) = 40\%$, and $maxsup(B) = 80\%$, and $minsup(B)$ is below the threshold while $maxsup(B)$ is above the threshold. Among the itemsets of size 2, $minsup(AB) = 0\%$, and $maxsup(AB) = 60\%$. By observing the rules, we note that $minconf(A \Rightarrow B) = minsup(AB) \times 100/maxsup(A)$ which is 0%, and $maxconf(A \Rightarrow B) = maxsup(AB) \times 100/minsup(A)$ which is 100%[1].

5 Algorithms for Rule Hiding

We have built two algorithms for rule hiding. The first one focuses on hiding the rules by reducing the minimum support of the itemsets that generated these rules (i.e., generating itemsets). The second one focuses on reducing the minimum confidence of the rules. We considered both support and confidence reduction algorithms as two alternative approaches. Support hiding is adequate against an association rule mining algorithm that uses support pruning to reduce the search space of rules which is usually the case for the currently available commercial products. However, algorithms that can efficiently extract high confidence rules without support pruning have recently been developed [6]. Therefore, we have also proposed an algorithm that hides rules by reducing their confidence. Based on the concepts of interval support and interval confidence that we introduced, we would like to reduce the minimum support and minimum confidence values below the *MST*, and *MCT* correspondingly by a certain *safety margin SM*. So, for a rule $A \Rightarrow B$, after the hiding process the following inequalities should hold; $minsup(A \Rightarrow B) \leq MST - SM$, and $minconf(A \Rightarrow B) \leq MCT - SM$.

5.1 Rule Hiding by Reducing the Support

This algorithm hides sensitive rules by decreasing the minimum support of their generating itemsets until the minimum support is below the *MST* by *SM*. The item with the largest minimum support is hidden from the minimum length transaction. The generating itemsets of the rules in R_h (set of sensitive rules) are considered for hiding. The generating itemsets of the rules in R_h are stored in L_h (set of large itemsets) and they are hidden one by one by decreasing their minimum support. The itemsets in L_h are first sorted in descending order of their size and minimum support. Then, they are hidden starting from the largest itemset. If there are more than one itemsets of maximum size, then the one with the highest minimum support is selected for hiding. The algorithm works like follows: Let Z be the next itemset to be hidden. Algorithm hides Z by decreasing its support. The algorithm first sorts the items in Z in descending order of their minimum support, and sorts the transactions in T_Z (transactions that support 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 highest minimum support is selected and a "?" mark is placed for that item in the transaction with minimum size. The execution stops after the support of the current rule to be hidden goes below the MST by SM. An overview of this algorithm is shown in Figure 1 where the generating itemsets of all the rules specified to be hidden is stored in L_h. After hiding an item from a transaction, the algorithm updates the minimum support of the remaining itemsets in L_h together with the list of transactions that support them. The algorithm chooses the item with highest minimum support for placing a "?" mark with the intention that an item of high minimum support will have less side effects since it has many more transactions that support it compared to an item of low minimum support. The idea behind choosing the shortest transaction for removal is that, a short transaction will possibly have less side effects on the other itemsets than a long transaction. Consider the transactions in Table 1. If rule $A \Rightarrow B$ needs to be hidden, then we need to choose one of the transactions in $\{T_1, T_4, T_5\}$. When the shortest transaction, T_4 among T_1, T_4, and T_5 is chosen, then placing a question mark for either item A, or item B in T_4 will only effect the rule $A \Rightarrow B$. However, if we had chosen T_1 or T_5, then rules $A \Rightarrow D$ and $B \Rightarrow D$ would also be affected by a modification in the transaction.

5.2 Rule Hiding by Reducing the Confidence

We propose two approaches for rule hiding using confidence reduction. The first approach is based on replacing 1s by "?" marks, while the second approach replaces 0s with "?" marks. It is important to have these two different approaches for the safety of the rule hiding. If we only used the first approach, then it would be obvious that all the "?" marks are actually 1's. Therefore, the two approaches should be used in an interleaved fashion for rule hiding via confidence reduction. The simplest way of Interleaving could be to hide the first half of sensitive rules by the first approach and the second half using the second approach.

The first algorithm shown in Figure 2 hides a sensitive rule r by decreasing the support of the generating itemset of r. The difference between this and the approach presented in Section 5.1 is that items in the consequent of r only, are chosen for hiding. This is due to the fact that by placing a "?" mark for the items in the antecedent of a rule r will cause the $minsup(l_r)$ (l_r is the left hand side of the rule r) to decrease, leading to an increase in the $maxconf(r)$, and this works against the rule hiding process which tries

[1] Note that we may have division by 0

INPUT: a set L of large itemsets, the set L_h
of large itemsets to hide, the database D, MST, and SM

OUTPUT: the database D modified by the
deletion of the large itemsets in L_h

Begin
1. Sort L_h in descending order
of size and minimum support of the large itemsets
Foreach Z **in** L_h
{
2. Sort the transactions in T_Z
in ascending order of transaction size
3. $N\_iterations = |T_Z| - (MST - SM) \times |D|$
For $k = 1$ **to** $N\_iterations$ **do**
{
4. Place a ? mark for the item with the largest minimum
support of Z in the next transaction in T_Z
5. Update the supports of the affected itemsets
6. Update the database, D
}
}
End

Figure 1. Rule Hiding by Support Reduction

INPUT: a set R_h of rules to hide,
the source database D, MCT, MST, and SM

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 $r\}$
2. for each t in T_r count the number of items in t
3. sort the transactions in T_r in ascending order of
the number of items supported
Repeat until $(minconf(r) < MCT - SM)$
{
4. Choose the first transaction $t \in T_r$
5. Choose the item j in r_r with the highest minsup
6. Place a ? mark for the place of j in t
7. Recompute the $minsup(r)$
8. Recompute the $minconf(r)$
9. Recompute the minconf of other affected rules
10. remove t from T_r
}
11. Remove r from R_h
}
End

Figure 2. Rule Hiding by Confidence Reduction (Algorithm 1)

to decrease confidence values of sensitive rules. The hiding process goes on until the $minsup(r)$ or the $minconf(r)$ goes below the *MST* and *MCT* thresholds by SM. The algorithm first generates the set T_r of transactions that support r, and then counts the number of items supported by each transaction. T_r is then sorted in ascending order of transaction size. In order to select the item in which we are going to place a "?" mark, we consider the impact on the rest of the rules. As a heuristic, the algorithm places a "?" mark for the item with the highest support in the minimum size transaction because of the same reason as we described in the case of rule hiding when the support of the generating itemsets was reduced.

The second algorithm shown in Figure 3 hides a rule r by increasing the $maxsup(l_r)$ via placing "?" marks in the place of the 0 values of items in l_r. By increasing the $maxsup(l_r)$ will cause the $minconf(r)$ to decrease. Given a rule r, the algorithm first generates the set T'_{l_r} of transactions that partially support l_r but that do not support r_r (the right hand side of the rule r). Then the number of items in l_r contained in each transaction is counted. The transaction t that contains the highest number of items in l_r is selected for processing, in order to make the minimum impact on the database. The 0 values for the items of l_r that are not supported by t is replaced by "?" marks to increase the $maxsup(l_r)$. The confidence of the rule is recomputed and the algorithm stops when the $minconf(r)$ goes below MCT by SM. In this method of rule hiding, we only consider the transactions that do not fully support r_r. Otherwise, by replacing 0 values for the items in l_r in the transactions that partially support l_r and fully support r_r, we will increase the $maxsup(r)$ leading to an increase in the $maxconf(r)$ which is not desirable. We choose the transaction that partially supports l_r while supporting the maximum number of items in l_r. In the best case, such a transaction will support $|l_r| - 1$ of the items in l_r and in this situation only one of the 0 values will be replaced by a "?" mark, achieving in this way the desired increase in the confidence while making the minimum change on the rest of the rules.

5.3 Complexity of the Rule Hiding Algorithms

All the algorithms first sort a subset of transactions in the database with respect to the items they have or with respect

INPUT: a set R_h of rules to hide,
the source database D, MCT, MST, and SM

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 partially supports l_r
and t does not fully support $r_r\}$
2. for each transaction of T'_{l_r} count the number
of items of l_r in it
3. sort the transactions in T'_{l_r} in descending
order of the calculated counts
Repeat until ($minconf(r) < MCT - SM$
or $minsup(r) < MST - SM$)
{
4. Choose the first transaction $t \in T'_{l_r}$
5. Place a "?" mark in t for the items in l_r
that are not supported by t
6. Recompute the $maxsup(l_r)$
7. Recompute $minconf(r)$
8. Recompute the minconf of other affected rules
9. remove t from T'_{l_r}
}
10. Remove r from R_h
}
End

Figure 3. Rule Hiding by Confidence Reduction (Algorithm 2)

to the particular items they support. Sorting N numbers is $O(NlogN)$ in the general case, however in our case the length of the transactions has an upper bound which is very small compared to the size of the database. In such a case we can sort N transactions in $O(N)$. The inner loop of the algorithm shown in Figure 1 executes $|T_Z| - (MST - SM) \times |D|$ times, and the operations in the inner loop can be done in constant time. The algorithm shown in Figure 2 executes its inner loop $|T_r| \times (minconf(r) - MCT + SM)$ times in order to reduce the minimum confidence of the sensitive rule r below the MCT by SM. The value of $(minconf(r) - MCT + SM)$ is the reduction needed in the minimum confidence represented as fraction. And this fraction multiplied by the number of the transactions supporting the rule to be hidden gives the actual number of iterations. For the algorithm shown in Figure 3, the inner loop is executed k times until the $minsup(r)$ goes below

rule	confidence
18 79 $\Rightarrow$ 31	76%
2 168 $\Rightarrow$ 4	79%
9 10 57 $\Rightarrow$ 33	83%
4 19 39 $\Rightarrow$ 27	77%
9 18 47 $\Rightarrow$ 19 35	53%

Table 3. Rules Selected for Hiding

MCT by SM. The $minconf(r)$ is initially $\frac{|T_r|}{|T'_{l_r}|}$, and after k iterations the fraction becomes $\frac{|T_r|}{|T'_{l_r}|+k}$ which should be smaller than $MCT - SM$ in order for the rule r to be hidden. When we isolate k from this fraction, we obtain $k < |T'_{l_r}| - \frac{|T_r|}{MCT-SM}$. The operations in the inner loops can be performed in constant time with proper hash structures.

6 Experiments

We used the anonymous Web usage data of the Microsoft web site created by Jack S. Breese, David Heckerman, and Carl M. Kadie from Microsoft. The data was created by sampling and processing the www.microsoft.com logs and donated to the Machine Learning Data Repository stored at University of California at Irvine Web site [8]. The Web log data keeps track of the use of Microsoft Web site by 38000 anonymous, randomly-selected users. For each user, the data records list all the areas of the Web sites that the user visited in a one week time frame. We used the training set only which has 32711 instances. Each instance represents an anonymous, randomly selected user of the Web site and corresponds to the transactions in market basket data. The number of attributes is 294 where each attribute is an area of the www.microsoft.com Web site and each attribute corresponds to an item in the store in the context of market basket data. We cleaned the data by removing the instances with less than or equal to non-zero attribute values and the resulting data set contained about 22k transactions.

We have implemented the support reduction and the first algorithm for confidence reduction, using the Perl programming language. We have also implemented a naive algorithm that hides a rule by replacing 1's by "?" marks in a round robin fashion by selecting the next item from the next transaction with no particular preference. The naive algorithm is used as a base for comparison with the rule and support reduction algorithms.

As a first step, we run an Apriori based mining algorithm on the data with support 0.1%. We then obtained the rules out of the resulting large itemsets with 50% minimum support. The minimum confidence and support values are

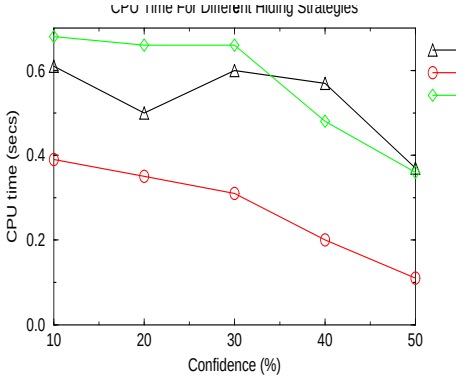

Figure 4. CPU Requirement

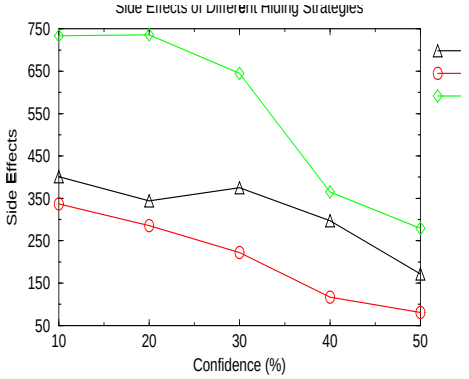

Figure 5. Side Effects

chosen with regard to typical minimum confidence and support thresholds from the literature. Sensitive rules should be determined by the domain experts. We did not have the necessary domain knowledge, therefore we randomly selected 5 different (not necessarily disjoint) rules and assumed that they are sensitive in order to test the hiding strategies. The selected rule set to be hidden is provided in Table 3. In order to assess the performance of the hiding strategies, we performed experiments on a PC, running Linux operating system, with 512 MB of main memory, and a Pentium III processor that has a CPU clock rate of 500 MHz.

In this exploratory study, we measured the CPU time requirement of the hiding strategies for different confidence values which is depicted in Figure 4. As can be seen from the figure, all the hiding strategies hide the given rule set successfully in less than a second that is considerably less than the time for mining which takes 57 seconds for 0.1% support. For various confidence values the GIH method (generating itemset support reduction algorithm Shown in Figure 1) and CH (Cyclic Hide) perform similarly while the CR (confidence reduction algorithm shown in Figure 2) hides the rules faster. However our main performance criterion of the different algorithms is the side effects they incur on the database. We measure the side effects by summing up the number of rules hidden unintentionally and the number of newly introduced rules. The performance of the hiding strategies in terms of the side effects are depicted in Figure 5. As can be observed from the figure, the CR causes the least number of side effects followed by GIR. CR and GIR outperform CH for all confidence values.

7 Related Work

The problem addressed in this paper is closely related to the inference problem in databases and the privacy preservation problem in data mining. Chang and Moskowitz [4] consider a solution of the database inference problem by using a new paradigm where decision tree analysis is combined with parsimonious downgrading. In their scheme, Chang and Moskowitz, propose that High decides what not to downgrade based upon the rules that it thinks Low can infer (i.e., by using decision tree analysis) and upon the importance of the information that Low should receive. Their objective then in developing this paradigm is to assign a penalty function to the parsimonious downgrading in order to minimize the amount of information that is not downgraded and to compare the penalty costs to the extra confidentiality that is obtained.

Clifton [5] investigates the techniques to address the basic problem of using non-sensitive data to infer sensitive data in the context of data mining. His goal is to accomplish privacy by ensuring that the data available to the adversary is only a sample of the data on which the adversary would like the rules to hold. In addition, Clifton shows that for classification purposes, the security officer is able to draw a relationship between the sample size and the likelihood that the rules are correct.

Agrawal and Srinkant [2] investigate the development of a data mining technique that incorporates privacy concerns. In particular, they consider the concrete case of building a decision tree classifier from training data in which the values of individual records have been perturbed. Their goal is to use the perturbed data (acquired either by a discretization or by a value distortion technique) in order to accurately estimate the original distribution of the data values. By doing this, they are able to build classifiers whose accuracy is comparable to the accuracy of classifiers built with the original data.

Agrawal and Aggarwal [1] improve on the distribution reconstruction technique presented in [2] by using the Expectation Maximization (EM) method. The authors claim that EM is more effective than the currently available technique in terms of the level of information loss. They also prove that EM converges to the maximum likelihood estimate of the original distribution based on the perturbed data

and that it provides robust estimates of the original distribution. Finally, they propose novel metrics for the quantification and measurement of privacy-preserving data mining algorithms.

A new class of privacy preserving techniques is introduced in [3, 9, 7]. In particular Atallah et. al [3], Dasseni et. al. [7] and Verykios et. al. [9] have considered the problem of privacy preserving mining of association rules. The authors have demonstrated how certain sensitive rules can be hidden by some data modification techniques and they have proposed efficient heuristics for solving this problem since Atallah et. al. [3] proved that the problem is NP-Hard. In the current work we are considering the same problem but instead of allowing random data modification, we have restricted ourselves to introducing "?" a special symbol that indicates that information is missing. Some changes to the original association rule discovery program are necessary for the introduction of heuristics based on this idea.

8 Conclusions

In this paper, we have presented a new set of concepts for making association rules sensitive, and we have extended the association rule mining process, in order to account for sensitive association rules. Association rules is one category of data mining techniques; other data mining techniques should also be considered for securing both data and knowledge in a virtual e-business environment that is open-ended and vulnerable to all kinds of different malevolent attacks. Our initial results indicate that deterministic algorithms for privacy preserving association rules are a promising framework for controlling disclosure of sensitive data and knowledge. In the near future, we will investigate how probabilistic and information theoretic techniques can also be applied to this problem.

References

[1] D. Agrawal and C. Aggarwal. On the Deisgn and Quantification of Privacy Preserving Data Mining Algorithms. *Proceedings of PODS*, pages 247–255, 2001.

[2] R. Agrawal and R. Srikant. Privacy Preserving Data Mining. *Proceedings of SIGMOD Conference*, pages 45–52, 2000.

[3] M. J. Atallah, E. Bertino, A. K. Elmagarmid, M. Ibrahim, and V. S. Verykios. Disclosure Limitation of Sensitive Rules. *Proceedings of IEEE Knolwedge and Data Engineering Workshop*, pages 45–52, November 1999.

[4] L. Chang and I. S. Moskowitz. Parsimonious Downgrading and Decision Trees Applied to the Inference Problem. *Proceedings of the Workshop of New Security Paradigms*, pages 82–89, 1999.

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

[6] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. D. Ullman, and C. Yang. Finding Interesting Associations Without Support Pruning. *IEEE Transactions on Knowledge and Data Engineering*, 13(1), 2001.

[7] D. Elena, V. S. Verykios, A. K. Elmagarmid, and E. Bertino. Hiding Association Rules by using Confidence and Support. *To appear in the Proceedings of Information Hiding Workshop*, 2001.

[8] U. of California at Irvine Machine Learning Repository. `http://www.ics.uci.edu/~mlearn/MLSummary.html`.

[9] V. S. Verykios, A. K. Elmagarmid, B. Elisa, D. Elena, and Y. Saygin. Association Rule Hiding. *IEEE Transactions on Knowledge and Data Engineering*, 2000. Under review.