University of Alberta

An Efficient One-Scan Sanitization For Improving The Balance Between Privacy And Knowledge Discovery

by

Stanley R. M. Oliveira
Osmar R. Zaïane

Technical Report TR 03-15
June 2003

DEPARTMENT OF COMPUTING SCIENCE
University of Alberta
Edmonton, Alberta, Canada

An Efficient One-Scan Sanitization For Improving The Balance Between Privacy And Knowledge Discovery

Stanley R. M. Oliveira[1,2]
oliveira@cs.ualberta.ca

Osmar R. Zaïane[1]
zaiane@cs.ualberta.ca

[1]Department of Computing Science, University of Alberta, Canada
[2]Embrapa Information Technology, Campinas, São Paulo, Brazil

Abstract

In this paper, we address the problem of protecting some sensitive knowledge in transactional databases. The challenge is on protecting actionable knowledge for strategic decisions, but at the same time not losing the great benefit of association rule mining. To accomplish that, we introduce a new, efficient one-scan algorithm that meets privacy protection and accuracy in association rule mining, without putting at risk the effectiveness of the data mining per se. Our experiments demonstrate that our algorithm is effective and achieves significant improvement over the other approaches presented in the literature. We report the main results of our performance evaluation and discuss some open research issues.

1 Introduction

In business world, data mining has been used extensively to find the optimal customer targets and maximize return on investment [8]. In particular, the discovery of association rules from large databases has proven beneficial for companies. Association rules provide a company with leverage to expand business, improve profitability, and market more effectively [3]. Such rules can be very effective in revealing actionable knowledge that leads to strategic decisions.

Despite its benefit in modern business, association rule mining can also pose a threat to privacy and information security if not done or used properly. The main problem is that from non-sensitive information or unclassified data, one is able to infer sensitive knowledge, including personal information, facts, or even patterns that are not supposed to be disclosed [7, 5]. There are a number of realistic scenarios in which privacy and security issues in association mining arise. We describe two challenging scenarios as follows:

1. Two or more companies have a very large dataset of records of their customers' buying activities. These companies decide to cooperatively conduct association rule mining on their datasets for

their mutual benefit since this collaboration brings them an advantage over other competitors. However, some of these companies may not want to share some strategic patterns hidden within their own data (also called restrictive association rules) with the other parties. They would like to transform their data in such a way that these restrictive associations rules cannot be discovered. Is it possible for these companies to benefit from such collaboration by sharing their data while still preserving some restrictive association rules?

2. Let us consider the case in which one supplier offers products at reduced prices to some consumers and, in turn, this supplier receives permission to access the database of the consumers' customer purchases. The threat becomes real whenever the supplier is allowed to derive sensitive association rules that are not even known to the database owners (consumers). In this case, the consumers benefit from reduced prices, whereas the supplier is provided with enough information to predict inventory needs and negotiate other products to obtain a better deal for his consumers. This implies that the competitors of this supplier start losing business. How can the consumers protect some restrictive association rules of customer purchases, while allowing the supplier to mine other useful association rules?

The solution for the above scenarios have been treated with an "all-or-nothing" approach. This approach is easy to implement, but it is not attractive due to its lack of flexibility. A more challenging and useful approach should find an appropriate balance between privacy and knowledge discovery. This status indicates a pressing need for rethinking mechanisms to protect sensitive knowledge in large databases without compromising the great benefit of association rule mining.

In this paper, we address the problem of transforming a database into a new one that conceals some strategic patterns (restrictive association rules) while preserving the general patterns and trends from the original database. The procedure of transforming an original database into a sanitized one is called the sanitization process and it was initially introduced in [2]. The sanitization process acts on the data to remove or hide a group of restrictive association rules that contain sensitive knowledge. To do so, a small number of transactions that contain the restrictive rules have to be modified by deleting one or more items from them or even adding noise to the data by turning some items from 0 to 1 in some transactions. This approach relies on boolean association rules. On the one hand, this approach slightly modifies some data, but this is perfectly acceptable in some real applications [6, 11, 10]. On the other hand, such an approach must hold the following restrictions: (1) the impact on the non-restricted data has to be minimal and (2) an appropriate balance between a need for privacy and knowledge discovery must be guaranteed.

Our contribution in this paper is two-fold. First, we introduce an efficient one-scan algorithm, called Sliding Window Algorithm (SWA), that improves the balance between privacy and knowledge discovery in association rule mining. This algorithm requires only one pass over a transactional database regardless of the database size and the number of restrictive association rules that must be protected. This represents a significant improvement over the previous algorithms presented in the literature [6, 11, 9], which require various scans depending on the number of association rules to be hidden. Second, we

compare our proposed algorithm with the similar counterparts in the literature. Our experiments demonstrate that our algorithm is effective, scalable, and achieves significant improvement over the other approaches presented in the literature.

This paper is organized as follows. Related work is reviewed in Section 2. In Section 3, we provide the basic concepts that are necessary to understand the issues addressed in this paper. In Section 4, we describe the heuristic approach that we propose to improve the balance between privacy and knowledge discovery. We introduce the SWA algorithm in Section 5. In Section 6, we present the experimental results, including a comparison of effectiveness and scalability of the best algorithms for data sanitization in the literature. Finally, Section 7 presents our conclusions and a discussion of future work.

2 Related Work

Some effort has been made to address the problem of privacy preservation in association rule mining. The class of solutions for this problem has been restricted basically to randomization, data partition, and data sanitization. In this work, we focus on the latter category.

The idea behind data sanitization was introduced in [2]. Atallah et al. considered the problem of limiting disclosure of sensitive rules, aiming at selectively hiding some frequent itemsets from large databases with as little impact on other non-sensitive frequent itemsets as possible. Specifically, the authors dealt with the problem of modifying a given database so that the support of a given set of sensitive rules, mined from the database, decreases below the minimum support value. The authors focused on the theoretical approach and showed that the optimal sanitization is an NP-hard problem.

In [6], the authors investigated confidentiality issues of a broad category of association rules and proposed some algorithms to preserve privacy of such rules above a given privacy threshold. Although these algorithms ensure privacy preservation, they are CPU-intensive since they require multiple scans over a transactional database. In addition, such algorithms, in some way, modify true data values and relationships by turning some items from 0 to 1 in some transactions.

In the same direction, Saygin et al. [11] introduced a method for selectively removing individual values from a database to prevent the discovery of a set of rules, while preserving the data for other applications. They proposed some algorithms to obscure a given set of sensitive rules by replacing known values with unknowns, while minimizing the side effects on non-sensitive rules. These algorithms are CPU-intensive and require various scans depending on the number of association rules to be hidden.

Oliveira and Zaïane [9] introduced a unified framework that combines techniques for efficiently hiding restrictive patterns: a transaction retrieval engine relying on an inverted file and Boolean queries; and a set of algorithms to "sanitize" a database. In this framework, the sanitizing algorithms require two scans regardless of the database size and the number of restrictive patterns that must be protected. The first scan is required to build an index (inverted file) for speeding up the sanitization process, while the second scan is used to sanitize the original database.

The work presented here differs from the related work in some aspects, as follows: First, the hiding strategies behind our algorithms deal with the problem 1 and 2 in Figure 1, and most importantly,

they do not introduce the problem 3 since we do not add noise to the original data. Second, we study the impact of our hiding strategies in the original database by quantifying how much information is preserved after sanitizing a database. So, our focus is not only on hiding restrictive association rules but also on maximizing the discovery of rules after sanitizing a database. Third, in terms of balancing between privacy and disclosure, our approach is very flexible since one can adjust a disclosure threshold for every single association rule to be restricted. Fourth, our algorithm SWA can achieve a reasonable performance since it requires only one scan over the original database, regardless of the database size and the number of restrictive rules to be hidden. Another advantage is that SWA is not a memory-based algorithm and therefore can deal with very large databases. Finally, our new algorithm is more effective in terms of hiding patterns while minimizing the impact on legitimate patterns. It has the lowest misses cost among the known algorithms. This represents a significant improvement over the previous algorithms presented in the literature [6, 11, 9].

3 Basic Concepts

In this section, we briefly review the basics of transactional databases and association rules.

3.1 Transactional Databases

A transactional database is a relation consisting of transactions in which each transaction t is characterized by an ordered pair, defined as $t = \langle TID,\ list\_of\_elements \rangle$, where *TID* is a unique transaction identifier number and *list_of_items* represents a list of items making up the transactions. For instance, in market basket data, a transactional database is composed of business transactions in which the list of elements represents items purchased in a store. An example of a sample transactional database is depicted in Figure 2A.

3.2 The Basics of Association Rules

Association rules provide a very simple but useful form of rule patterns for data mining. A rule consists of a left-hand side proposition (the antecedent or condition) and a right-hand side (the consequent). Both the left and right-hand side consist of Boolean statements (or propositions). The rules state that if the left-hand side is true, then the right-hand side is also true.

Formally, association rules are defined as follows: 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, called *TID*, is associated with each transaction. A transaction t supports X, a set of items in I, if $X \subset t$. An association rule is an implication of the form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$ and $X \cap Y = \emptyset$. Thus, we say that a rule $X \Rightarrow Y$ holds in the database D with *confidence* φ if $\frac{|X \cup Y|}{|X|} \geq \varphi$, where $|A|$ is the number of occurrences of the set of items A in the set of transactions D. Similarly, we say that a rule $X \Rightarrow Y$ holds in the database D with *support* σ if $\frac{|X \cup Y|}{|N|} \geq \sigma$, where N is the number of transactions in D. Association rule mining algorithms rely on support and confidence and mainly have

two major phases: (1) based on a support σ set by the user, frequent itemsets are determined through consecutive scans of the database; (2) strong association rules are derived from the frequent item sets and constrained by a minimum confidence φ also set by the user.

3.3 Balancing Privacy and Knowledge Discovery

In our framework for database sanitization [9, 10], the process of modifying transactions to hide some patterns satisfies a disclosure threshold ψ controlled by the database owner. This threshold basically expresses how relaxed the privacy preserving mechanisms should be. When $\psi = 0\%$, no restrictive association rules are allowed to be discovered. When $\psi = 100\%$, there are no restrictions on the restrictive association rules. The advantage of having this threshold is that it enables a compromise to be found between hiding association rules while missing legitimate ones and finding all legitimate association rules but uncovering restrictive ones. This is indeed the balance between privacy and the disclosure of information.

In this paper we introduce the notion of disclosure threshold for every single pattern to restrict. In other words, rather than having one unique threshold ψ for the whole sanitization process, we can have a different threshold ψ_i for each pattern i to restrict. This provides a greater flexibility allowing an administrator to put different weights for different rules to hide. The algorithm SWA we present permits the setting of these different thresholds. If only one disclosure threshold is needed for the process, it suffices to set all thresholds to the same value.

4 Heuristic Approach

Our heuristic approach draws the following assumptions: (1) The database owners have to know in advance some knowledge (rules) that they want to protect. Such rules are fundamental in decision making, so they must not be discoverable; (2) The individual data values (e.g. a specific item or an attribute) are not restricted. Rather, some aggregates and relationships must be protected. Note that our approach works in the opposite way to the idea behind the statistical database techniques [4], which prevent against discovering individual tuples.

Our goal is to provide information for mining while protecting a group of association rules which contains highly sensitive knowledge. We refer to these rules as *restrictive association rules*. It should be noted that what constitute restrictive association rules depends on the application and the importance of these rules in a decision process. We define them as follows:

Definition 1 (Restrictive Association Rules) *Let D be a transactional database, σ the minimum support threshold, R be a set of all association rules that can be mined from D based on a minimum support σ, and $Rules_H$ be a set of decision support rules that need to be hidden according to some security policies. A set of association rules, denoted by R_R, is said to be restrictive if $R_R \subset R$, and $\forall\ r \in R_R, \exists\ r' \in Rules_H$ such that $r' \subset r$ (i.e. R_R would derive the set $Rules_H$). $\tilde{}R_R$ is the set of non-restrictive association rules such that $\tilde{}R_R \cup R_R = R$.*

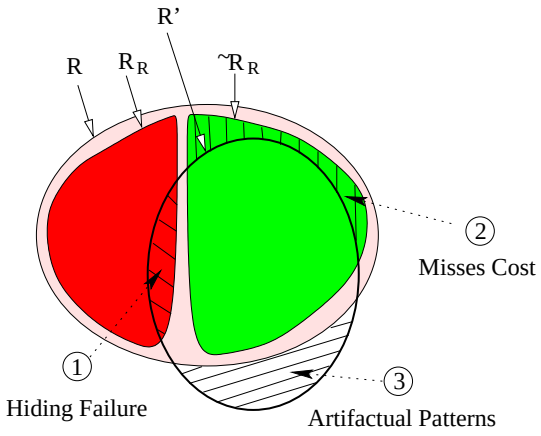


Figure 1: Visual representation of restrictive and non-restrictive association rules and the rules effectively discovered after transaction sanitization.

Figure 1 illustrates the relationship between the set R of all association rules in the database D, the restrictive and non-restrictive association rules, as well as the set R' of rules discovered from the sanitized database D'. 1, 2, and 3 are potential problems that respectively represent the restrictive association rules that were failed to be hidden, the legitimate rules accidentally missed, and the artificial association rules created by the sanitization process.

A group of restrictive association rules is mined from a database D based on a special group of transactions. We refer to these transactions as sensitive transactions and define them as follows.

Definition 2 (Sensitive Transactions) *Let T be a set of all transactions in a transactional database D and R_R be a set of restrictive association rules mined from D. A set of transactions is said to be sensitive, as denoted by S_T, if $S_T \subset T$ and if and only if all restrictive association rules can be mined from S_T, and only transactions in S_T contain all items involved in the restrictive association rules.*

For each restrictive association rule, we have to identify a candidate item that should be removed from its sensitive transactions. We refer to this item as the *victim item*. In many cases, a group of restrictive rules share one or more items. In this case, the selected victim item is one sharing item. The rationale behind this selection is that by removing the victim item from a sensitive transactions that contains a group of rules, all sensitive rules in the group would be hidden in one step.

Our heuristic approach has essentially five steps as follows. These steps are applied to every group of K transactions (window size) read from the original database D.

Step1: Distinguishing the sensitive transactions from the non-sensitives ones. For each transaction read from a database D, we identify if this transaction contains any restrictive association rules. If not, the transaction is copied directly to the sanitized database D'. Otherwise, this transaction is sensitive and must be sanitized.

Step2: Selecting the victim item. In this step, we first compute the frequencies of all items in the restrictive association rules presented in the current sensitive transaction. The item with the

highest frequency is the victim item since it is shared by a group of restrictive rules. If a restrictive rule shares no item with the other restrictive ones, the frequencies of its items are the same (freq = 1). In this case, the victim item of this particular restrictive rule is selected randomly. The rationale behind this selection is that by removing different items from the sensitive transactions would slightly minimize the support of the legitimate association rules that would be available for being mined in the sanitized database D'.

Step3: Computing the number of sensitive transactions to be sanitized. Given the disclosure threshold, ψ, set by the database owner, we compute the number of transactions to be sanitized. Every restrictive rule will have a list of sensitive transaction IDs associated with. The size of the list depends on the threshold ψ. This means that the disclosure threshold ψ is actually a measure of the impact of the sanitization rather than a direct measure of the restricted rules to hide or disclose. Indirectly, ψ does have an influence on the hiding or disclosure of restricted rules.

Step4: Sorting the sensitive transactions by size. In this step, we sort the number of sensitive transactions computed in the previous step, for each restrictive rule. The sensitive transactions are sorted in ascending order of size. Thus, we start marking the shortest transactions to be sanitized. By removing items of shortest transactions we will be minimizing the impact on the sanitized database since shortest transactions have less combinations of association rules.

Step5: Sanitizing a sensitive transaction. Given that the victim items for all restrictive association rules were selected in step 2, now they can be removed from the sensitive transactions. Every restrictive rule has now a list of sensitive transaction IDs with their respective selected victim item. Every time we remove a victim item from a sensitive transaction, we perform a look ahead procedure to verify if that transaction has been selected as a sensitive transaction for other restrictive rules. If so, and the victim item we just removed from the current transaction is also part of this other restrictive rule, we remove that transaction from the list of transaction IDs marked in the other rules. In doing so, the transaction will be sanitized, and then copied to the sanitized database D'. This look ahead procedure is done only when the disclosure threshold is 0%. This is because the look ahead improves the misses cost but could significantly degrade the hiding failure. When $\psi = 0$, there is no hiding failure (i.e. all restrictive rules are hidden) and thus there is no degradation possible but an improvement in the misses cost.

To illustrate how our heuristic works, let us consider the sample transactional database in Figure 2A. Suppose that we have a set of restrictive association rules R_R = {A,B→D; A,C→D} and we set the disclosure threshold $\psi = 50\%$. This example yields the following results: the sensitive transactions of A,B→D and A,C→D are {T3, T1} and {T4, T1} respectively. We purposely sorted the transactions in ascending order of size. After identifying the sensitive transactions, we select the victim items. For example, the victim item in the transaction T1 could be either A or D since these items are shared by the restrictive rules. However, the victim item for the restrictive rule A,B→D in T3 is selected randomly because there is no item shared with the other rule. Let us assume that the victim item selected is B.

Similarly, the victim item for the restrictive rule A,C→D, in transaction T4, is selected randomly, say, the item A. Then, we sanitize the transactions in each restrictive rule. Half of the transactions for each restrictive rule will be intact since the disclosure threshold $\psi = 50\%$. We start by sanitizing the shortest transactions. Thus, transactions T3 and T4 are sanitized and the released database is depicted in Figure 2B. Note that the restrictive rules are present in the sanitized database, but with lower support. This is an example of partial sanitization. The database owner could also set the disclosure threshold $\psi = 0\%$. In this case, we have a full sanitization since the restrictive rules will not be discoverable anymore. In this example we assume that the victim item in transaction T1 is D since this item is shared by both restrictive rules. Figure 2C shows the database after a full sanitization. As we can see, the database owner can tune the disclosure threshold to find a balance between protecting sensitive association rules by data sanitization and providing information for mining.

TID	Items
T1	A B C D
T2	A B C
T3	A B D
T4	A C D
T5	A B C
T6	B D

(A)

TID	Items
T1	A B C D
T2	A B C
T3	A D
T4	C D
T5	A B C
T6	B D

(B)

TID	Items
T1	A B C
T2	A B C
T3	A D
T4	C D
T5	A B C
T6	B D

(C)

Figure 2: (A): A sample transactional database; (B): An example of partial sanitization; (C): An example of full sanitization.

An important observation here is that any association rule that contains a restrictive association rule is also restrictive. Hence, if A,B→D is the only restrictive association rule but not A,C→D as it was the case above, any association rule derived from the itemset ABCD will also be restrictive since it contains ABD. This is because if ABCD is discovered to be a frequent itemset, it is straight forward to conclude that ABD is also frequent, which should not be disclosed. In other words, any superset containing ABD should not be allowed to be frequent.

5 The Sliding Window Algorithm

In this section, we introduce the Sliding Window Algorithm (SWA) and analyze its complexity in main memory. The intuition behind this algorithm is that SWA scans a group of K transactions, at a time. Then, SWA sanitizes the set of sensitive transactions, denoted by R_R, considering a disclosure threshold ψ defined by a database owner.

For each restrictive association rule there is a disclosure threshold assigned to it. We refer to the set of mappings of a restrictive association rule into its corresponding disclosure threshold as the set of mining permissions, denoted by M_P, in which each mining permission mp is characterized by an ordered pair, defined as $mp = < rr_i, \psi_i >$, where $\forall i\ rr_i \in R_R$ and $\psi_i \in [0 \dots 1]$.

The inputs for the Sliding Window algorithm are a transactional database D, a set of mining

permissions M_P, and the window size K. The output is the sanitized database D'. The sketch of the Sliding Window algorithm is given as follows:

Sliding_Window_Algorithm

Input: D, M_P, K

Output: D'

For each K transactions in D do {

Step 1. For each transaction $t \in K$ do {

1. Sort the items in t in ascending order
2. For each association rule $rr_i \in M_P$ do {

If $\exists$ rr_i such that $\forall j$ $item_j \in rr_i$ and $item_j \in t$ then

2.1. $T[rr_i] \leftarrow T[rr_i] \cup t$ //t is sensitive

2.2. $freq(item_j) \leftarrow freq(item_j) + 1$

}

}

Step 2. If t is sensitive then

1. Sort_Vector($freq$) //in descending order
2. For each association rule $rr_i \in M_P$ do

2.1. Select $item_v$ such that $item_v \in rr_i$ and $\forall$ $item_k \in rr_i$, $freq[item_v] \geq freq[item_k]$

2.2. if $freq[item_v] > 1$ then $Victim_{rr_i} \leftarrow item_v$

else $Victim_{rr_i} \leftarrow$ randomly selected $item_k$ such that $item_k \in rr_i$

}

Step 3. For each association rule $rr_i \in M_P$ do

1. $NumTrans_{rr_i} \leftarrow |T[rr_i]| \times (1 - \psi_i)$ //$|T[rr_i]|$ - number of sensitive transactions for rr_i in K

Step 4. For each association rule $rr_i \in M_P$ do

1. Sort_Transactions($T[rr_i]$) //in ascending order of size

Step 5. $D' \leftarrow D$

For each association rule $rr_i \in M_P$ do {

1. $TransToSanitize \leftarrow$ Select first $NumTrans_{rr_i}$ transactions from $T[rr_i]$
2. in D' foreach transaction $t \in TransToSanitize$ do

2.1. $t \leftarrow (t - Victim_{rr_i})$ //transaction is sanitized

2.2. if $\psi_i = 0$ then do $look\_ahead(rr_i, Victim_{rr_i}, t, M_P)$

}

End

SWA has essentially five steps. In the first, the algorithm sorts the items of each transaction t, in ascending order, to identify if the transaction is sensitive or not by using a binary search fashion. A transaction t is sensitive if it contains all items of at least one restrictive rule. In this case, the transaction ID is added to the list of transition IDs of the corresponding restrictive rule. Then, the algorithm computes the frequencies of the items of the restrictive rules that are present in such a transaction. This will support the selection of the victim items in the next step. In step 2, the vector

with the frequencies, computed in the previous step, is sorted in descending order. After that, the victim item $Victim_{rr_i}$ is selected. The item with the highest frequency is the victim item and must be removed from the transaction. If the frequencies of the items is equal to 1, any item from a restrictive association rule can be the victim item. In this case, we select the victim item randomly. Line 1 in step 3 shows that ψ_i is used to compute the number $NumTrans_{rr_i}$ of transactions to sanitize. In step 4, the sensitive transaction IDs in each restrictive rule are sorted in ascending order of size. Sanitizing the shortest transactions in each restrictive rule minimizes the impact on the sanitized database because shortest transactions contain less combinations of association rules. In step 5, the sensitive transactions to be cleansed are first marked and then sanitized, as can be seen in lines 1 and 2.1 respectively. If the disclosure threshold is 0 (i.e. all restrictive rules need to be hidden), we do a look ahead in the mining permissions (M_P) to check whether a sensitive transaction need not be sanitized more than once. This is to improve the misses cost. The function *look_ahead*() looks in M_P from rr_i onward whether a given transaction t is selected as sensitive transaction for another restrictive rule r. If it is the case, and $Victim_{rr_i}$ is part of the restrictive rule r, the transaction t is removed from the list since it has just been sanitized already.

Theorem 1 *The running time of the Sliding Window Algorithm is* $O(n_1 \times N \times log\ K)$ *when* $\psi \neq 0$ *and* $O((n_1 - 1) \times ((n_1 - 1) + 1))/2) \times N \times K)$ *when* $\psi = 0$*, where* n_1 *is the initial number of restrictive rules in the database* D*,* N *is the number of transactions in* D*, and* K *is the window size chosen.*

Proof. Let D be the source database, N the number of transactions in D, n_1 the initial number of restrictive association rules selected in D, n_2 the maximum number of items in a transaction $t \in D$, n_3 the maximum number of items in a restrictive association rule, K the number of transactions in a sliding window, and M_P a set of mining permissions in which each permission is defined as $mp = < rr_i, \psi_i >$, where $\forall i\ rr_i$ is a restrictive rule and ψ_i is the corresponding disclosure threshold.

In Step 1, first the items in each transaction are sorted in ascending order. So line 1 takes $n_2\ log\ n_2$. Then, for each association rule $rr_i \in R_R$, the algorithm verifies if such rule is present in the current transaction t. To do so, the algorithm binary searches, in the worst case, all items of the rule rr_i in the transaction t to make sure that this rule is present in t. If the transaction t is sensitive, its ID is added to the list of transaction IDs of the corresponding rr_i. This entire process encompasses n_3 binary searches over a transaction with at most n_2 items, so that the running time of each rule rr_i, in line 2.1 of Step 1, takes $O(n_3 \times log\ n_2)$. Line 2.2 in step 1 contains straightforward computations and takes $O(n_3)$, when all items in rr_i are present in the current transaction t. Because line 2 is executed n_1 times, the complexity of line 2 takes $O(n_1 \times n_3 \times log\ n_2 + n_1 \times n_3)$. The running time for Step 1 takes $O(n_2 \log\ n_2 + n_1 \times n_3 \times log\ n_2 + n_1 \times n_3)$. When n_1 is large, $n_1 \times n_3 \times log\ n_2$ grows faster than $n_2 \log\ n_2$ and $n_1 \times n_3$. Thus, Step 1 can be simplified to $O(n_1 \times n_3 \times log\ n_2)$.

In line 1 in Step 2, the vector of frequencies computed in the previous step contains at most $n_1 \times n_3$ items, when all items in the restrictive rules are present in the transaction t. The running time to sort this vector takes $O(n_1 \times n_3 \times log\ (n_1 \times n_3))$. The selection of the victim item in line 2.1 and line 2.2 is a straightforward computation that takes $O(1)$ per restrictive rule since the vector of frequencies is

already sorted. Thus, line 2.1 and line 2.2 take $O(n_1)$ since it is executed n_1 times. The running time in step 2 is the sum of the running times in lines 1, 2.1, and 2.2, i.e., $O(n_1 \times n_3 \times log\ (n_1 \times n_3) + n_1)$, which can be simplified to $O(n_1 \times n_3 \times log\ (n_1 \times n_3))$.

Considering that Steps 1 and 2 are executed K times, the running time for these steps are respectively $O(K \times n_1 \times n_3 \times log\ n_2)$ and $O(K \times n_1 \times n_3 \times log\ (n_1 \times n_3))$.

Step 3 contains straightforward computations and takes $O(1)$ per restrictive rule since the algorithm simply selects the number of sensitive transactions to be touched. Because this step is executed n_1 times, the running time for step 3 takes $O(n_1)$.

In Step 4, for each restrictive rule, we sort the sensitive transaction IDs marked to be sanitized. In the worst case in line 1, all the K transactions will be marked to be sanitized. This entire step takes $n_1 \times K\ log\ K$, since this step is executed n_1 times.

In line 1 of Step 5, all transactions in each sliding window are sensitive, in the worst case, and then they are stored in the data structure $TransToSanitize$ to be sanitized. So line 1 takes $O(n_1 \times K)$. In line 2.1, the algorithm cleanses at most n_2 items in the current transaction, given that all restrictive rules are present in such a transaction. Because this line is executed n_1 times, line 2.1 takes $O(n_1 \times n_2)$. In line 2.2, the *look_ahead*() procedure is performed for each victim item sanitized only when $\psi = 0$.

First, let us consider $\psi \neq 0$. In this case, Step 5 takes $O(n_1 \times K + n_1 \times n_2)$. When n_1 and K are large, Step 5 can be simplified to $O(n_1 \times K)$. Thus, the running time of the Sliding Window algorithm is the sum of running times for each step. Thus, the running time takes $O(K \times (n_1 \times n_3 \times log\ n_2) + K \times (n_1 \times n_3 \times log\ (n_1 \times n_3)) + n_1 + n_1 \times K\ log\ K + n_1 \times K)$. When K and n_1 are large, $n_1 \times K\ log\ K$ grows faster than $K \times (n_1 \times n_3 \times log\ n_2)$, $K \times (n_1 \times n_3 \times log\ (n_1 \times n_3))$, n_1, and $n_1 \times K$. Thus, the running time of SWA takes $O(n_1 \times K\ log\ K)$, when $\psi \neq 0$.

The running time above refers to a group of K transactions. Considering that the whole database contains N transactions, and there exists N/K windows, the running time of the Sliding Window Algorithm takes $O(N/K \times n_1 \times K\ log\ K)$, which can be simplified to $O(n_1 \times N \times log\ K)$.

When $\psi = 0$, SWA performs the look ahead procedure. In the worst case, all K transactions are marked to be sanitized in all restrictive rules. Then, SWA checks if the current victim item sanitized is present in the next rules. This procedure takes $(n_1 - 1) \times K \times K$ for the first restrictive rule, $(n_1 - 2) \times K \times K$ for the second rule, and it repeats until the last restrictive rule. So the number of times that the look ahead procedure is performed is the sum of the terms of an arithmetic progression multiplied by K transactions of each restrictive rule, i.e., $((n_1 - 1) \times ((n_1 - 1) + 1))/2) \times K \times K)$. In this case, Step 5 takes $((n_1 - 1) \times ((n_1 - 1) + 1))/2) \times K \times K)$ since it is the most expensive step of the algorithm. Thus, the running time of Step 5 takes approximately $O((n_1 - 1) \times ((n_1 - 1) + 1))/2) \times K^2)$.

Therefore, considering $\psi = 0$, and given that the Step 5 is the most expensive step of the algorithm, the running time of SWA takes $O((n_1-1)\times((n_1-1)+1))/2)\times N\times K)$. □

6 Experimental Results

In this section, we present the results of our performance evaluation. We start by describing the methodology used in this paper. Then, we study the efficiency and scalability of our algorithm SWA and the similar counterparts in the literature.

6.1 Methodology

We compare SWA with respect to the following benchmarks: (1) the result of Apriori algorithm without transformation; (2) the results of similar algorithms in the literature.

We compare the effectiveness and scalability of SWA with a similar one proposed in [6] to hide rules by reducing support, called Algo2a. The algorithm GIH designed by Saygin et al. [11] is similar to Algo2a. The basic difference is that in Algo2a some items are removed from sensitive transactions, while in GIH a mark "?" (unknowns) is placed instead of item deletions. We also compare SWA with the Item Grouping Algorithm (IGA), our best algorithm so far published and presented in [9]. The main idea behind the Item Grouping Algorithm, denoted by IGA, is to group restricted association rules in groups of rules sharing the same itemsets. If two restrictive rules intersect, by sanitizing the conflicting sensitive transactions containing both restrictive rules, one would take care of hiding these two restrictive rules in one step and consequently reduce the impact on the released database. However, clustering the restrictive rules based on the intersections between rules leads to groups that overlap since the intersection of itemsets is not transitive. By solving the overlap between clusters and thus isolating the groups, we can use a representative of the itemset linking the restrictive rules in the same group as a victim item for all rules in the group. By removing the victim item from the sensitive transactions related to the rules in the group, all sensitive rules in the group would be hidden in one step. This again minimizes the impact on the database and reduces the potential accidental hiding of legitimate rules.

We performed two series of experiments: the first to measure the effectiveness of SWA, IGA, and Algo2a, and the second to measure the efficiency and scalability of these algorithms. All the experiments were conducted on a PC, AMD Athlon 1900/1600 (SPEC CFP2000 588), with 1.2 GB of RAM running a Linux operating system. To measure the effectiveness of the algorithms, we used a dataset generated by the IBM synthetic data generator [1] to generate a dataset containing 500 different items, with 100K transactions in which the average size per transaction is 40 items. The effectiveness is measured in terms of the number of restrictive association rules effectively hidden, as well as the proportion of legitimate rules accidentally hidden due to the sanitization.

We selected for our experiments a set of ten restrictive association rules from the dataset ranging from two to five items in length, as can be seen is Table 1. To do so, we ran the Apriori algorithm to select such association rules. With our ten original restrictive association rules, 94701 rules became restricted in the database since any association rule that contains restrictive rules should also be restricted.

Restrictive Association Rules	Support (%)	Confidence (%)
$2 \rightarrow 7$	23.8	87.7
$33 \rightarrow 19$	41.9	80.8
$3, 16 \rightarrow 35$	22.5	90.6
$12, 16 \rightarrow 13$	28.8	99.2
$13, 37 \rightarrow 9$	30.7	87.8
$33, 34 \rightarrow 17$	41.3	80.4
$2, 3, 28 \rightarrow 13$	20.7	94.6
$13, 14, 19 \rightarrow 7$	30.3	80.8
$20, 27, 33, 39 \rightarrow 35$	30.5	96.6
$26, 28, 37, 38 \rightarrow 31$	34.0	93.8

Table 1: The set of restrictive rules

6.2 Measuring effectiveness

In this section, we measure the effectiveness of SWA, IGA, and Algo2a taking into account the performance measures introduced in [9]. We summarize such performance measures as follows:

Hiding Failure (HF): measures the amount of restrictive association rules that are disclosed after sanitization. The hiding failure is measured by $HF = \frac{\#R_R(D')}{\#R_R(D)}$ where $\#R_R(X)$ denotes the number of restrictive association rules discovered from database X.

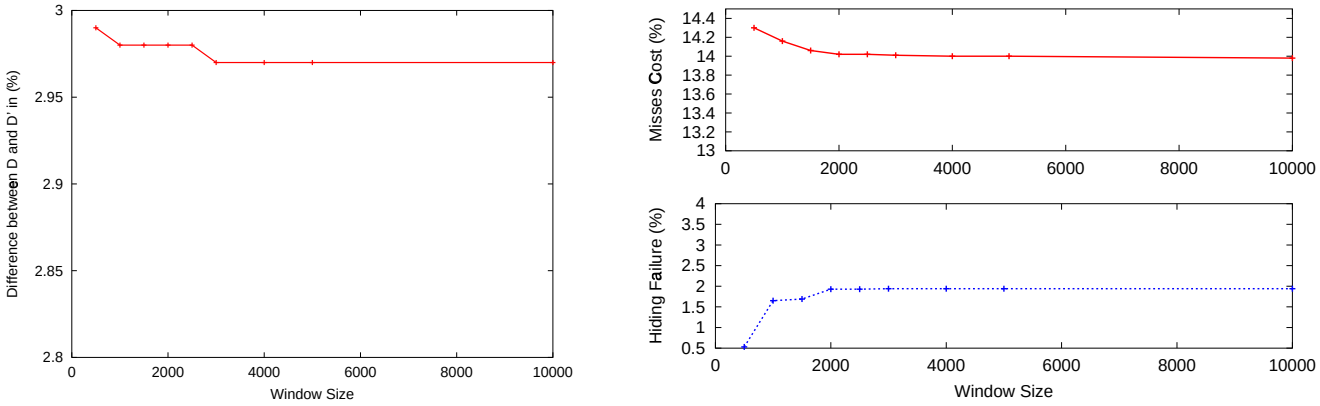
Misses Cost (MC): measures the amount of legitimate association rules that are hidden by accident after sanitization. The misses cost is calculated as follows: $MC = \frac{\#\sim R_R(D) - \#\sim R_R(D')}{\#\sim R_R(D)}$ where $\# \sim R_R(X)$ denotes the number of non-restrictive association rules discovered from database X.

Artifactual Patterns (AP): measure the artificial association rules created by the addition of noise in the data. Artifactual patterns are measured as: $AP = \frac{|R'| - |R \cap R'|}{|R'|}$, where $|X|$ denotes the cardinality of X.

Difference between the original and sanitized databases: the difference between the original (D) and the sanitized (D') databases, denoted by $dif(D, D')$, is given by:
$dif(D, D') = \frac{1}{\sum_{i=1}^{n} f_D(i)} \times \sum_{i=1}^{n} [f_D(i) - f_{D'}(i)]$, where $f_X(i)$ represents the frequency of the ith item in the dataset X, and n is the number of distinct items in the original dataset.

We evaluated the effect of window size with respect to the difference of the original database D and the sanitized one D'. To do so, we varied K from 500 to 10000 transactions with the disclosure threshold $\psi = 15\%$. Figure 3A shows that up to 3000 transactions the difference between the original and the sanitized database improves slightly. After 3000 transactions, the difference remains the same. Similarly, Figure 3B shows that after 3000 transactions the values of misses cost (MC) and hiding failure (HF) tend to be constant. This shows that on our example database, a window size representing 3% of the size of the database suffices to stabilize the misses cost and hiding failure.

The distribution of the data may affect these values. However, we have observed that the larger the window size the better the results. The reason is that when the heuristic is applied to a large number of transactions, the impact in the database is minimized. Consequently, the value of misses cost and the difference between D and D' improve slightly.



(A) (B)

Figure 3: (A): Effect of window size on the difference of D and D' (B): Effect of window size on misses cost and hiding failure

Figure 4A shows a special case in which the disclosure threshold ψ is set to 0%. As we can see, no restrictive rule is allowed to be mined from the sanitized database. In this situation, 18.30% of the legitimate association rules in the case of SWA, 20.08% in the case of IGA, and 24.76% in the case of Algo2a are accidentally hidden.

We intentionally selected restrictive association rules with high support in the reported experiments to accentuate the differential between the sizes of the original database and the sanitized database and thus to better illustrate the impact of the sanitization on the mining process.

Figure 4B shows the differential between the initial size of the database and the size of the sanitized database when the disclosure threshold $\psi = 0\%$. To have the smallest impact possible on the database, the sanitization algorithm should not reduce the size of the database significantly. SWA and IGA are the ones that impact the least on the database. In this particular case, 3.55% of the database is lost in the case of SWA and IGA, and 5.24% in the case of Algo2a.

While the algorithms proposed in [6, 11] hide rules reducing their absolute support below a privacy threshold controlled by the database owner, SWA and IGA hide rules based on a disclosure threshold ψ.

Figure 5 shows the effect of the disclosure threshold ψ on the hiding failure and the misses cost for SWA and IGA, considering the minimum support threshold $\sigma = 5\%$. Since Algo2a doesn't allow the input of a disclosure threshold, it is not compared in this figure with our algorithms. As can be observed, when ψ is 0%, no restrictive association rule is disclosed for both algorithms. However, 20.08% of the legitimate association rules in the case of IGA, and 18.30% in the case of SWA are accidentally hidden. When ψ is equal to 100%, all restrictive association rules are disclosed and no misses are recorded for

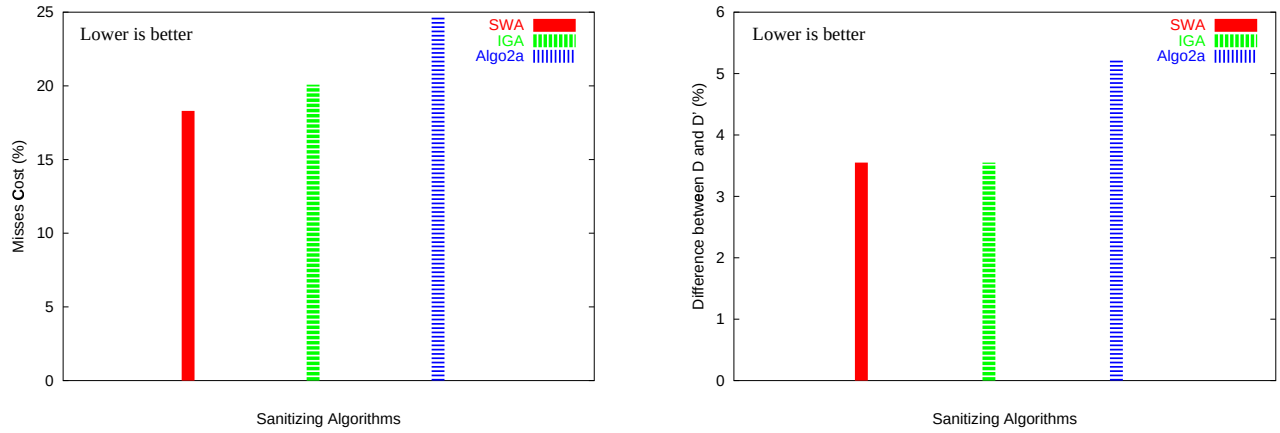


Figure 4: (A): Effect of $\psi = 0\%$ on misses cost (B): The difference in size between D and D'

legitimate rules. What can also be observed is that the impact of SWA on the database is smaller and the misses cost of SWA is slightly better than that of IGA. Moreover, the hiding failure for SWA is slightly better than that for IGA in all the cases, except when $\psi = 50\%$.

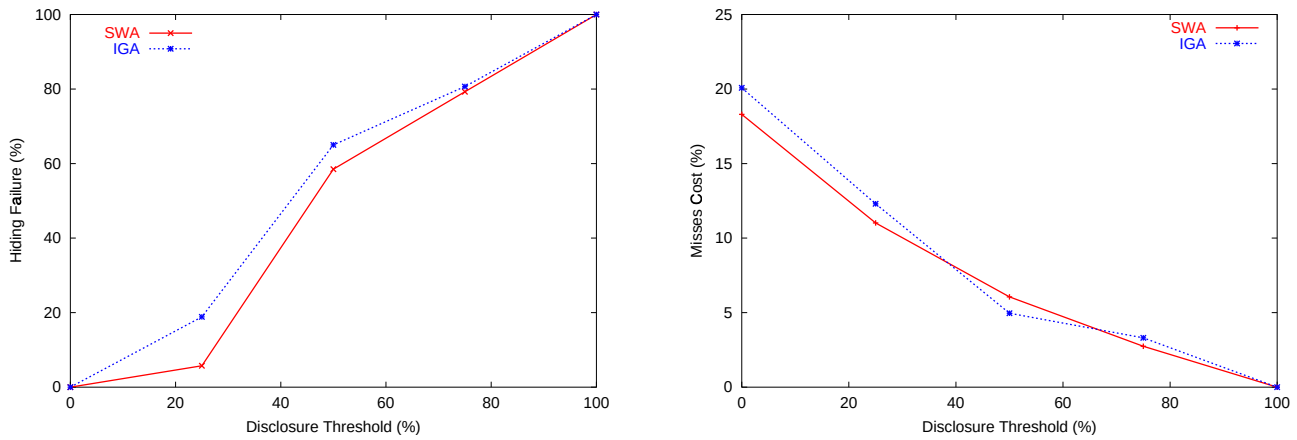


Figure 5: Effect of ψ on hiding failure and misses cost

Regarding the third performance measure, artifactual patterns, one may claim that when we decrease the frequencies of some items, the relative frequencies in the database may be modified by the sanitization process, and new rules may emerge. However, in our experiments, the problem artifactual pattern AP was always 0% with all algorithms regardless of the values of ψ. Our sanitization, indeed, does not remove any transaction. The same results can be observed for the algorithms presented in [6, 11].

Figure 6 shows the differential between the initial size of the database and the size of the sanitized database for SWA and IGA with respect to the disclosure threshold ψ. Again, SWA is the one that impacts the least on the database for all values of the disclosure threshold ψ. Thus, as can be seen, both algorithms slightly alter the data in the original database, while enabling flexibility for someone

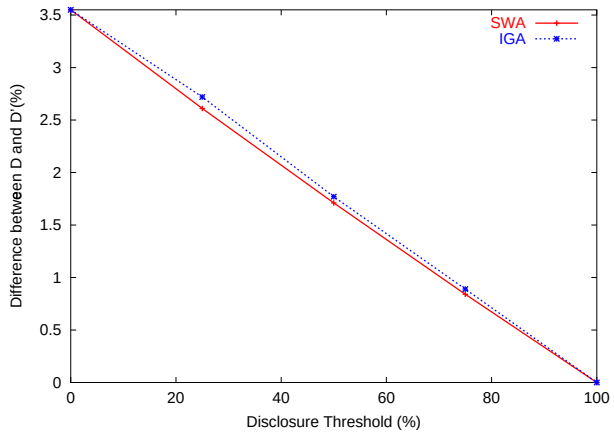


Figure 6: The difference in size between D and D'

to tune them.

SWA also has an advantage over the counterpart algorithms. The advantage is that SWA allows a database owner to set a specific disclosure threshold for each restrictive rule. This specific disclosure threshold works as a weight. In many cases, some rules are more important than others. So giving different disclosure thresholds to different rules is reasonable and may reflect the need in the real world. For instance, let us consider the set of restrictive rules depicted in Table 1. In our previous examples, we set the disclosure thresholds of such rules with a unique value. This was done due the fact that we compared SWA with the other algorithms in the literature. Now we give different disclosure thresholds, as follows: {[rule 1, 20%], [rule 2, 10%], [rule 3, 20%], [rule 4, 25%], [rule 5, 12%], [rule 6, 10%], [rule 7, 15%], [rule 8, 8%], [rule 9, 5%], [rule 10, 5%]}, where for each ordered pair [rule i, ψ_i], rule i represents a restrictive rule, and ψ_i the corresponding disclosure threshold. In this example, we set the sliding window $K = 40000$, and we yielded the following results: misses cost = 14.80% and hiding failure = 0.42%.

6.3 CPU Time for the Sanitization Process

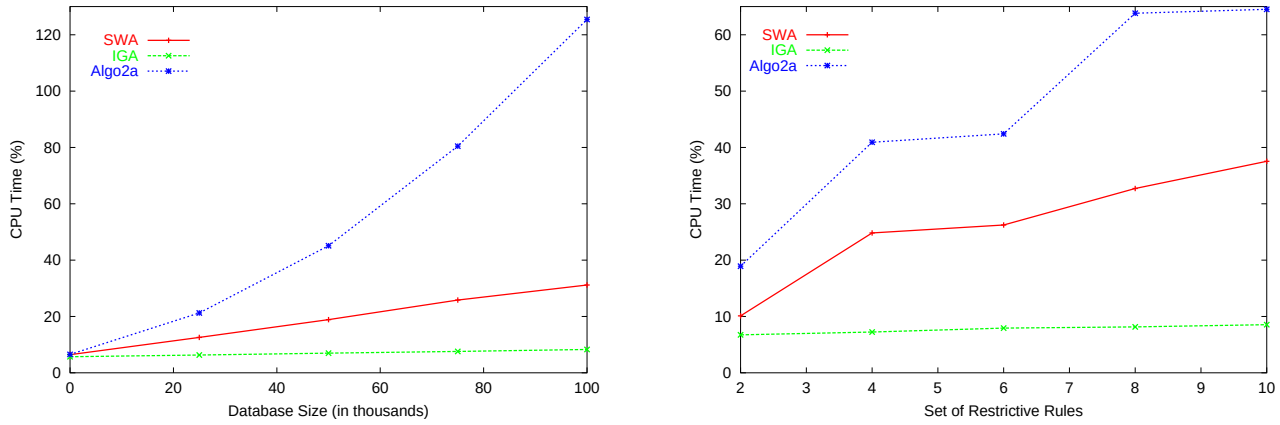
We tested the scalability of our sanitization algorithms vis-à-vis the size of the database as well as the number of rules to hide. Our comparison study also includes the algorithm Algo2a.

We varied the size of the original database D from 20K transactions to 100K transactions, while fixing the disclosure threshold $\psi = 0$ and the support threshold $sigma = 5\%$, and keeping the set of restrictive rules constant (10 original patterns). We set the window size for SWA with $K = 20000$. Figure 7A shows that IGA and SWA increase CPU time linearly with the size of the database, while the CPU time in Algo2a grows fast. This is due the fact that Algo2a requires various scans over the original database, while IGA requires two, and SWA requires only one.

Although IGA requires 2 scans, it is faster than SWA. The main reason is that IGA clusters restrictive association rules in groups of rules sharing the same itemsets. Then by removing the victim item from the sensitive transactions related to the rules in the group, all sensitive rules in the group would be

hidden in one step.

As can be observed, SWA increases CPU linearly, even though its complexity in main memory is not linear. If we increase the number of restrictive rules or even if we select a group of restrictive rules with very high support, SWA may not scale linearly. However, there is no compelling need for sanitization to be a fast operation since it can be done offline.



(A) (B)

Figure 7: Results of CPU time for the sanitization process

We also varied the number of restrictive rules to hide from approximately 6000 to 29500, while fixing the size of the database to 100K transactions and fixing the support and disclosure thresholds to $\psi = 0\%$. Figure 7B shows that our algorithms scale well with the number of rules to hide. The figure reports the size of the original set of restricted rules, which varied from 2 to 10. This makes the set of all restricted rules range from approximately 6097 to 29558. This scalability is mainly due to the inverted files we use in our approaches for indexing the sensitive transaction IDs per restrictive rules. There is no need to scan the database again whenever we want to access a transaction for sanitization purposes. The inverted file gives direct access with pointers to the relevant transactions. The CPU time for Algo2a is more expensive due the number of scans over the database.

7 Conclusions

In this paper, we have introduced and efficient algorithm that improves the balance between protection of sensitive knowledge and pattern discovery, called Sliding Window Algorithm (SWA). This algorithm is useful for sanitizing large transactional databases based on a disclosure threshold (or a set of thresholds) controlled by a database owner.

The experimental results revealed that SWA is effective and can achieve significant improvement over the other approaches presented in the literature. SWA slightly alters the data while enabling flexibility for someone to tune it. Another advantage of SWA is the fact that it does not introduce false drops to the data. However, an extra cost is incurred because some rules are removed inadvertently.

It is important to note that our sanitization method is robust in the sense that there is no desanitization possible. The alterations to the original database are not saved anywhere since the owner of the database still keeps an original copy of the database intact while distributing the sanitized database. Moreover, there is no encryption involved. There is no possible way to reproduce the original database from the sanitized one.

Currently, we are investigating new optimal sanitization algorithms that minimize the impact in the sanitized database without compromising the benefit of mining. We are also expanding our work with a probabilistic analysis to supplement the empirical results, which require further exploration.

8 Acknowledgments

Stanley Oliveira was partially supported by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) of Ministry for Science and Technology of Brazil, under Grant No. 200077/00-7. Osmar Zaïane was partially supported by a Research Grant from NSERC, Canada.

References

[1] IBM. Almaden. Quest synthetic data generation code. http://www.almaden.ibm.com/cs/quest/syndata.html.

[2] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios. Disclosure Limitation of Sensitive Rules. In *Proc. of IEEE Knowledge and Data Engineering Workshop*, pages 45–52, Chicago, Illinois, November 1999.

[3] M. Berry and G. Linoff. *Data Mining Techniques - for Marketing, Sales, and Customer Support.* John Wiley and Sons, New York, USA, 1997.

[4] S. Castano, M. Fugini, G. Martella, and P. Samarati. *Database Security.* Addison-Wesley Longman Limited, England, 1995.

[5] C. Clifton and D. Marks. Security and Privacy Implications of Data Mining. In *Workshop on Data Mining and Knowledge Discovery*, pages 15–19, Montreal, Canada, February 1996.

[6] E. Dasseni, V. S. Verykios, A. K. Elmagarmid, and E. Bertino. Hiding Association Rules by Using Confidence and Support. In *Proc. of the 4th Information Hiding Workshop*, pages 369–383, Pittsburg, PA, April 2001.

[7] T. Johnsten and V. V. Raghavan. Impact of Decision-Region Based Classification Mining Algorithms on Database Security. In *Proc. of 13th Annual IFIP WG 11.3 Working Conference on Database Security*, pages 177–191, Seattle, USA, July 1999.

[8] V. S. Y. Lo. The True Lift Model - A Novel Data Mining Approach to Response Modeling in Database Marketing. *SIGKDD Explorations*, 4(2):78–86, December 2002.

[9] S. R. M. Oliveira and O. R. Zaïane. Privacy Preserving Frequent Itemset Mining. In *Proc. of the IEEE ICDM Workshop on Privacy, Security, and Data Mining*, pages 43–54, Maebashi City, Japan, December 2002.

[10] S. R. M. Oliveira and O. R. Zaïane. Algorithms for Balancing Privacy and Knowledge Discovery in Association Rule Mining. In *Proc. of the 7th International Database Engineering and Applications Symposium (IDEAS'03)*, Hong Kong, China, July 2003.

[11] Y. Saygin, V. S. Verykios, and C. Clifton. Using Unknowns to Prevent Discovery of Association Rules. *SIGMOD Record*, 30(4):45–54, December 2001.