Privacy Preserving Frequent Itemset Mining

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

Osmar R. Zaïane[2]
zaiane@cs.ualberta.ca

[1]Embrapa Information Technology
André Tosello, 209 – Barão Geraldo
13083-886 - Campinas, SP, Brasil

[2]Department of Computing Science
University of Alberta
Edmonton, AB, Canada, T6G 2E8

Abstract

One crucial aspect of privacy preserving frequent itemset mining is the fact that the mining process deals with a trade-off: privacy and accuracy, which are typically contradictory, and improving one usually incurs a cost in the other. One alternative to address this particular problem is to look for a balance between hiding restrictive patterns and disclosing non-restrictive ones. In this paper, we propose a new framework for enforcing privacy in mining frequent itemsets. We combine, in a single framework, 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 addition, we introduce performance measures for mining frequent itemsets that quantify the fraction of mining patterns which are preserved after sanitizing a database. We also report the results of a performance evaluation of our research prototype and an analysis of the results.

Keywords: Assosiation Rule Mining, Privacy Preserving Data Mining, Privacy Preservation in Association Rule Mining; Frequent Itemset Mining, Security.

1 Introduction

One of the most studied problems in data mining is the process of discovering frequent itemsets and, consequently, association rules. Discovering hidden patterns from large amounts of data plays an important role in marketing, business, medical analysis, and other applications where these patterns are paramount for strategic decision making.

Despite its benefits in various areas, data mining can also pose a threat to privacy and information security if not done or used properly. Recent advances in data mining and machine learning algorithms have introduced a new problem in database security (Johnsten & Raghavan 1999, Clifton 2000). The main problem is that from non-sensitive information or unclassified data, one is able to infer sensitive information, including personal information, facts, or even patterns that are not supposed to be disclosed.

Privacy issues in data mining cannot simply be addressed by restricting data collection or even by restricting the use of information technology. An appropriate balance between a need for privacy and knowledge discovery should be found (Brankovic & Estivill-Castro 1999). In (Clifton, Du, Atallah, Kantarcioglu, Lin & Vaidya 2001), Clifton et al. argued that there is no exact solution that resolves this privacy problem in data mining. An approximate solution could be sufficient depending on the application since the level of privacy can be interpreted in different contexts.

Let us consider a motivating example discussed in (Evfimievski, Srikant, Agrawal & Gehrke 2002). Suppose we have a server and many clients in which each client has a set of items (e.g. books or movies). The clients want the server to gather statistical information about associations among items in order to provide recommendations to the clients. However, the clients do not want the server to know some restrictive itemsets. Thus, when a client sends its dataset to the server, it hides some restrictive itemsets from the dataset according to some specific privacy policies, and the server then gathers statistical information from the modified dataset. In this case, the server is provided with the data and not the patterns and it is free to use its own tools so that the restriction for privacy has to be applied before the mining phase on the data itself.

The simplistic solution to address the motivating example is to implement a filter after the mining phase to weed out/hide the restricted discovered patterns, since discovery is a process that finds hidden patterns without a predetermined idea or hypothesis what the patterns may be. This solution is straightforward. However, in this particular case, the servers do not send patterns, but a dataset without the restrictive patterns. In addition, depending on the number of pattern discovered (e.g. millions or more), it is more reasonable and feasible to send a sanitized dataset than a huge set of patterns.

One simple and effective way to hide some restrictive patterns is to decrease their support in a given database. This procedure of altering the transactions is called the sanitization process and was introduced in (Atallah, Bertino, Elmagarmid, Ibrahim & Verykios 1999). To do so, a small number of transactions have to be modified by deleting one or more items from them or even changing items in transactions, i.e., adding noise to the data. This work relies on boolean association rules. The authors proved that the optimal sanitization problem is NP-hard. On one hand, this approach slightly modifies some data, but this is perfectly acceptable in some real applications (Clifton & Marks 1996, Dasseni, Verykios, Elmagarmid & Bertino 2001, Saygin, Verykios & Clifton 2001). On the other hand, such an approach must hold the following restrictions: (1) the impact on the data in D has to be minimal and (2) an appropriate balance between a need for privacy and knowledge discovery must be guaranteed.

In the context of our work, we do not add noise to the data by turning some items from 0 to 1 in some transactions, but only strategically and selectively we remove individual items from sensitive transactions preventing the disclosure of some patterns while preserving as much of the original information as possible for other applications. To do so, we propose a new framework for enforcing privacy in mining frequent patterns that combines techniques for efficiently hiding restrictive rules. The framework is composed of a transaction retrieval engine relying on an inverted file and Boolean queries for retrieving transaction IDs from a database, and a set of sanitizing algorithms. One major novelty with our approach is that we take into account the impact of our sanitization not only on hiding the patterns that should be hidden but also on hiding legitimate patterns that should not be hidden. Thus, our framework tries to find a balance between privacy and disclosure of information by attempting to minimize the impact on the sanitized transactions. Other approaches presented in the literature focus on the hiding of restrictive patterns but do not study the effect of their sanitization on accidentally concealing legitimate patterns or even generating artifact patterns.

The main contributions of this paper are the following: (1) the design and implementation of the framework; (2) a taxonomy of algorithms for sanitizing a transactional database; and (3) performance measures for mining frequent patterns that quantify the fraction of mining patterns which are preserved after sanitizing a database.

This paper is organized as follows. In Section 2, we provide the basic concepts that are necessary to understand the scope and the issues addressed in this paper. In addition, the problem definition is given. We present our framework in detail in Section 3. In Section 4, we introduce our taxonomy of sanitizing algorithms. In Section 5, we present the experimental results and discussion. Related work is reviewed in Section 6. Finally, we summarize the conclusions of our study and outline future avenues to explore in Section 7. At the end of this paper, we address the reviewers' remarks.

2 Basic Concepts

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 (Han & Kamber 2001). 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.

2.1 The Basics of Mining Frequent Patterns and Association Rules

The discovery of the recurrent patterns in large transactional databases has become one of the main topics in data mining. In its simplest form, the task of finding frequent patterns can be viewed as the process of discovering all item sets, i.e., all combinations of items that are found in a sufficient number of examples, given a frequency threshold σ. If the frequency threshold is low, then there might be many frequent patterns in the answer set.

The items in a frequent pattern are Boolean, i.e., items are either present or absent. For this reason, a transactional database may be represented by a sparse matrix in which the rows correspond to transactions and the columns correspond to the items available in one store. If the element (i, j) is 1, this indicates that customer i purchased item j, while 0 indicates that the item j was not purchased.

When the frequent patterns are known, finding association rules is simple. Association rules provide a very simple but useful form of rule patterns for data mining.

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. Since the main challenge is the discovery of the frequent itemsets, we consider only this second phase in our analysis.

2.2 Privacy Preservation: Problem Definition

In this work, our goal is to hide a group of frequent patterns which contains highly sensitive knowledge. We refer to these frequent patterns as restrictive patterns, and we define them as follows:

Definition 1 *Let D be a transactional database, P be a set of all frequent patterns that can be mined from D, and $Rules_H$ be a set of decision support rules that need to be hidden according to some security policies. A set of patterns, denoted by R_P, is said to be restrictive if $R_P \subset P$ and if and only if R_P would derive the set $Rules_H$. $\tilde{}R_P$ is the set of non-restrictive patterns such that $\tilde{}R_P \cup R_P = P$.*

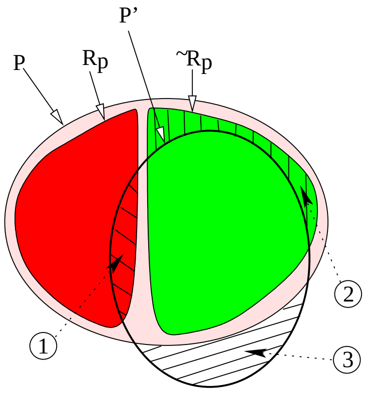


Figure 1: Visual representation of restrictive and non-restrictive patterns and the patterns effectively discovered after transaction sanitization

Figure 1 illustrates the relationship between the set P of all frequent patterns in the database D, the restrictive and non-restrictive patterns, as well as the set P' of frequent patterns discovered from the sanitized database D'. 1, 2, and 3 are potential problems that represent the restrictive patterns that were failed to be hidden, the legitimate patterns accidentally missed, and the artificial patterns created by the sanitization process. These are explained in Section 5 in the discussion of how to measure the effectiveness of our algorithms.

A group of restrictive patterns 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 *Let T be a set of all transactions in a transactional database D and R_P be a set of restrictive patterns 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 patterns can be mined from S_T and only from S_T.*

The specific problem addressed in this paper can be stated as follows: If D is the source database of transactions and P is a set of relevant patterns that could be mined from D, the goal is to transform D into a database D' so that the most frequent patterns in P can still be mined from D' while others will be hidden. In this case, D' becomes the released database.

2.3 The Sanitization Process

The goal of the sanitization process is to hide some restrictive patterns that contain highly sensitive knowledge. This process is composed of four steps as follows. In the first step, the set P of all patterns from D is identified. The second step distinguishes restricted patterns R_p from the non-restrictive patterns ˜R_P by applying some security policies. It should be noted that what constitute as restrictive patterns depends on the application and the importance of these patterns in a decision process. In Step 3, sensitive transactions are identified within D. In our approach, we use a very efficient retrieval mechanism called the transaction retrieval engine (discussed in Section 3.2) to speed up the process of finding the sensitive transactions. Finally, Step 4 is dedicated to the alteration of these sensitive transactions to produce the sanitized database D'. In our framework, the process of modifying such transactions satisfies a risk of disclosure threshold ψ controlled by the user. For short, we will refer to this threshold as disclosure threshold. This threshold basically expresses how relaxed the privacy preserving mechanisms should be. When $\psi = 0\%$, no restrictive patterns are allowed to be discovered. When $\psi = 100\%$, there are no restrictions on the restrictive patterns.

3 The Framework for Privacy Preservation

As depicted in Figure 2, our framework encompasses a transactional database (modeled into a document database), an inverted file, a set of sanitizing algorithms used for hiding restrictive patterns from the database, and a transaction retrieval engine for fast retrieval of transactions. We describe the inverted file and the transaction retrieval engine in this section, and the set of algorithms in Section 4.

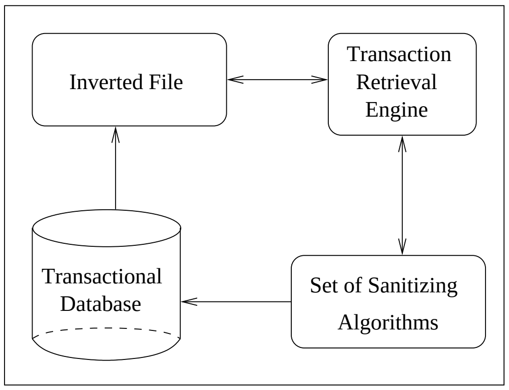


Figure 2: Privacy Preservation Framework

Sanitizing a transactional database consists of identifying the sensitive transactions and adjusting them. To speed up this process, we model transactions into documents in which the items simply become terms. This model preserves all the information and provides the basis for our indexing, borrowing from the information retrieval domain.

3.1 The Inverted File Index

One very efficient strategy for indexing a text database is an inverted file (Baeza-Yates & Ribeiro-Neto 1999). An inverted file, a structure comprising the *vocabulary* and the *occurrences*, is a word-oriented mechanism for indexing a text collection with the purpose of speeding up the searching task.

In our framework, the inverted file's vocabulary is composed of all different items in the transactional database, and for each item there is a corresponding list of transaction IDs in which the item is present. Figure 3 shows an example of an inverted file corresponding to the sample transactional database shown in the figure.

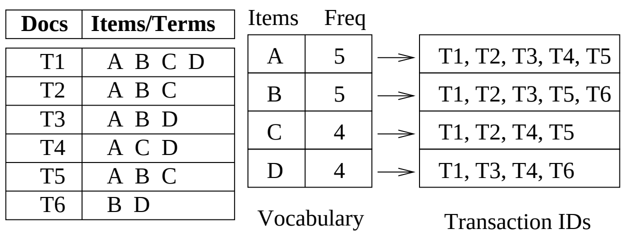


Figure 3: An example of transactions modeled by documents and the corresponding inverted file

We implemented the vocabulary based on a perfect hash table (Dietzfelbinger, Karlin, Mehlhorn, Heide, Rohnert & Tarjan 1994), with no collision, insertion, or deletion. For a given item, one access suffices to find the list of all transaction IDs that contain the item. The occurrences with transaction IDs are created and simultaneously sorted in ascending order of transaction IDs. Thus, to search for the transaction ID of a particular item, we use a binary search in which, in the worst case, the access time is $O(logN)$, where N is the number of transaction IDs in the occurrences.

3.2 The Transaction Retrieval Engine

To search for sensitive transactions in the transactional database, it is necessary to access, manipulate, and query transaction IDs. The transaction retrieval engine performs these tasks. It accepts requests for transactions from a sanitizing algorithm, determines how these requests can be filled (consulting the inverted file), processes the queries using a query language based on Boolean model, and returns the results to the sanitizing algorithm. The process of searching for sensitive transactions through the transactional database works on the inverted file. In general, this process follows three steps: (1) *Vocabulary search*: each restrictive pattern is split into single items. Isolated items are transformed into basic queries to the inverted index; (2) *Retrieval of transactions*: The lists of all transaction IDs of transactions containing each individual item respectively are retrieved; and (3) *Intersections of transaction lists*: The lists of transactions of all individual items in each restrictive pattern are intersected using a conjunctive Boolean operator on the query tree to find the sensitive transactions containing a given restrictive pattern.

4 The Sanitization Algorithms

Sanitizing algorithms for transactional databases can be classified into two classes: the algorithms that solely remove information from the transactional database and those that modify existing information. The first algorithms only reduce the support of some items, while the second may increase the support of some items. Our taxonomy of sanitizing algorithms, depicted in Figure 4, relies on the first category.

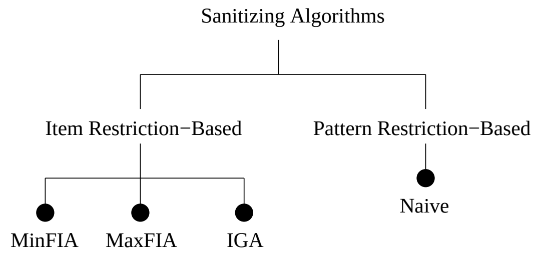


Figure 4: A Taxonomy of Sanitizing Algorithms

Algorithms that solely remove information create a smaller impact on the database since they do not generate artifacts such as illegal association rules that would not exist had the sanitizing not happened. Among the approaches that remove information only, we distinguish the pattern restriction-based approaches that remove complete restrictive patterns from the sensitive transactions and the item restriction-based approaches that selectively remove some items from sensitive transactions. The pattern restrictive-based approaches have a bigger impact on the database as more legal patterns may end-up hidden along with the restricted patterns.

To sanitize a database, each sanitizing algorithm requires an additional scan over the original database D in order to alter some sensitive transactions while keeping the other transactions intact. An initial scan is necessary to build the inverted index.

In most cases, a sensitive transaction contains more than one restrictive pattern. We refer to these transactions as *conflicting transactions* since modifying one of them causes an impact on other restrictive patterns or even on non-restrictive ones. The *degree of conflict* of a sensitive transaction is defined as the number of restrictive patterns that can be mined from the sensitive transaction.

To illustrate the presented concepts, let us consider the sample transactional database in Figure 3. Suppose that we have a set of restrictive patterns R_P = {ABD, ACD}. This example yields the following results. The sensitive transactions S_T containing the restrictive patterns are {T1, T3, T4}. The degrees of conflict for the transactions T1, T3 and T4 are 2, 1 and 1 respectively. Thus, the only conflicting transaction is T1, which covers both restrictive patterns at the same time. An important observation here is that any pattern that contains a restrictive pattern is also a restrictive pattern. Hence, if ABD is a restricted pattern but not ACD as above, the pattern ABCD will also be restrictive since it contains ABD. This is because if ABCD is discovered to be a frequent pattern, it is straight forward to conclude that ABD is also frequent, which should not be disclosed.

All our item restriction-based algorithms have essentially four major steps: (1) Identify sensitive transactions for each restrictive pattern; (2) For each restrictive pattern, identify a candidate item that should be eliminated from the sensitive transactions. This candidate item is called the *victim item*; (3) Based on the disclosure threshold ψ, calculate for each restrictive pattern the number of sensitive transactions that should be sanitized; and (4) Based on the number found in step 3, identify for each restrictive pattern the sensitive transactions that have to be sanitized and remove the victim item from them.

Our sanitizing algorithms mainly differ in step 2 in the way they identify a victim item to remove from the sensitive transactions for each restrictive pattern, and in step 4 where the sensitive transactions to be sanitized are selected. Steps 1 and 3 remain essentially the same for all approaches.

The complexity of our sanitization algorithms in main memory is $O(n_1 \times NlogN)$, where n_1 is the number of restrictive patterns and N the number of transactions in the database. This is considering the number of items per restrictive pattern relatively small compared to the size of the database. The proof of this is given in (Oliveira & Zaïane 2002).

4.1 The Naïve Algorithm

The main idea behind the Naïve Algorithm is to select all items in a given restrictive pattern as victims. The rationale behind this selection is that by removing from the sensitive transactions the items of a restrictive pattern, such a pattern will be hidden. If a sensitive transaction contains exactly the same items as a restrictive pattern, the Naïve Algorithm removes all items of this transaction except for the item with the highest frequency in the database. Because one item must be kept, the number of transactions is not modified.

Selecting the sensitive transactions to sanitize is based simply on their degree of conflict. Given the number of sensitive transactions to alter, based on ψ, this approach selects for each restrictive pattern the transactions with the smallest degree of conflict. The rationale is, as above, to minimize the impact of the sanitization on the discovery of the legitimate patterns.

The sketch of the Naïve Algorithm is given as follows:

```
**Naive_Algorithm**
**Input:** $D$, $R_P$, $\psi$
**Output:** $D'$
Step 1. For each restrictive pattern $rp_i \in R_P$ do
    1. $T[rp_i] \leftarrow$Find_Sensitive_Transactions($rp_i$, $D$);
Step 2. For each restrictive pattern $rp_i \in R_P$ do
    1. $Victims_{rp_i} \leftarrow \forall\ item_k$ such that $item_k \in rp_i$
Step 3. For each restrictive pattern $rp_i \in R_P$ do
    1. $NumTrans_{rp_i} \leftarrow |T[rp_i]| \times (1 - \psi)$   // $|T[rp_i]|$ is the number of sensitive transactions for $rp_i$
Step 4. $D' \leftarrow D$
    For each restrictive pattern $rp_i \in R_P$ do
    1. Sort_Transactions($T[rp_i]$);   //in ascending order of degree of conflict
    2. $TransToSanitize \leftarrow$ Select first $NumTrans_{rp_i}$ transactions from $T[rp_i]$
    3. in $D'$ foreach transaction $t \in TransToSanitize$ do
        3.1. $t \leftarrow (t - Victims_{rp_i})$
**End**
```

The four steps in this algorithm correspond to the four steps described above for all pattern restriction-based algorithms. The first step builds an inverted index of the item in D in one scan of the database. As illustrated in the example in Figure 3, the support of each item in the database is also calculated during this scan and attached to the respective items in the inverted index. This support of the items is used in step 2 to identify the victim items $Victims_{rp_i}$ for each restrictive pattern. For the Naïve algorithm, all items in a given restrictive pattern are selected. Line 1 in step 3 shows that ψ is used to compute the number $NumTrans_{rp_i}$ of transactions to sanitize. This means that the threshold ψ is actually a measure of the impact of the sanitization rather than a direct measure of the restricted patterns to hide or disclose. Indirectly, ψ does have an influence on the hiding or disclosure of restricted patterns. There is actually only one scan of the database in the implementation of step 4. Transactions that do not need sanitization are directly copied from D to D', while the others are sanitized before being copied to D'. In our implementation, the sensitive transactions to be cleansed are first marked before the database scan for copying. The selection of the sensitive transactions to sanitize, $TransToSanitize$, is based on their degree of conflict, hence the sorting in line 1 of step 4. When a transaction is selected for sanitization, the victim items are removed from it (line 3.1 in step 4).

4.2 The Minimum Frequency Item Algorithm

The main idea behind the Minimum Frequency Item Algorithm, denoted by MinFIA, is to select as a victim item, for a given restrictive pattern, the restrictive pattern item with the smallest support in the pattern. The rationale behind this selection is that by removing the item from the sensitive transactions with the smallest support will have the smallest impact on the database and the legitimate patterns to be discovered. Selecting the sensitive transactions to sanitize is simply based on their degree of conflict. Given the number of sensitive transactions to alter, based on ψ, this approach selects for each restrictive pattern the transactions with the smallest degree of conflict. The rationale is, as above, to minimize the impact of the sanitization on the discovery of the legitimate patterns. The sketch of the Minimum Frequency Item Algorithm is given as follows:

```
**Minimum_Frequency_Item_Algorithm**
**Input:** $D$, $R_P$, $\psi$
**Output:** $D'$
Step 1. For each restrictive pattern $rp_i \in R_P$ do
    1. $T[rp_i] \leftarrow$Find_Sensitive_Transactions($rp_i$, $D$);
Step 2. For each restrictive pattern $rp_i \in R_P$ do
    1. $Victim_{rp_i} \leftarrow item_v$ such that $item_v \in rp_i$ and
       $\forall\ item_k \in rp_i$ sup($item_k$, $D$) $\geq$ sup($item_v$, $D$)
Step 3. For each restrictive pattern $rp_i \in R_P$ do
    1. $NumTrans_{rp_i} \leftarrow |T[rp_i]| \times (1 - \psi)$   // $|T[rp_i]|$ is the number of sensitive transactions for $rp_i$
Step 4. $D' \leftarrow D$
    For each restrictive pattern $rp_i \in R_P$ do
    1. Sort_Transactions($T[rp_i]$);   //in ascending order of degree of conflict
    2. $TransToSanitize \leftarrow$ Select first $NumTrans_{rp_i}$ transactions from $T[rp_i]$
    3. in $D'$ foreach transaction $t \in TransToSanitize$ do
        3.1. $t \leftarrow (t - Victim_{rp_i})$
**End**
```

The four steps of this algorithm correspond to those in the Naïve Algorithm. The only difference is that the Minimum Frequency Item Algorithm selects exactly one victim item, as aforementioned.

Unlike the Minimum Frequency Item Algorithm, the idea behind the Maximum Frequency Item Algorithm, denoted by MaxFIA, is to select as a victim item, for a given restrictive association rule, the item with the maximum support in the restrictive association rule. The algorithms MinFIA and MaxFIA are thus conceptually very similar.

4.3 The Item Grouping Algorithm

The main idea behind the Item Grouping Algorithm, denoted by IGA, is to group restricted patterns in groups of patterns sharing the same itemsets. If two restrictive patterns intersect, by sanitizing the conflicting sensitive transactions containing both restrictive patterns, one would take care of hiding these two restrictive patterns at once and consequently reduce the impact on the released database. However, clustering the restrictive patterns based on the intersections between patterns 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 patterns in the same group as a victim item for all patterns in the group. By removing the victim item from the sensitive transactions related to the patterns in the group, all sensitive patterns in the group will be hidden in one step. This again will minimize the impact on the database and reduce the potential accidental hiding of legitimate patterns. The sketch of the Item Algorithm is given as follows:

```
**Item_Grouping_Algorithm**
**Input:** $D$, $R_P$, $\psi$
**Output:** $D'$
Step 1. For each restrictive pattern $rp_i \in R_P$ do
    1. $T[rp_i] \leftarrow$Find_Sensitive_Transactions($rp_i$, $D$);
Step 2.
    1.Group restrictive patterns in a set of groups $GP$ such that $\forall\ G \in GP, \forall\ rp_i, rp_j \in G$, $rp_i$ and $rp_j$ share the same itemset $I$. Give the class label $\alpha$ to $G$ such that $\alpha \in I$ and $\forall \beta \in I$, sup($\alpha$, $D$) $\leq$ sup($\beta$, $D$).
    2. Order the groups in $GP$ by size in terms of number of restrictive patterns in the group.
    3. Compare groups pairwise $G_i$ and $G_j$ starting with the largest. For all $rp_k \in G_i \cap G_j$ do
        3.1. if size($G_i$) $\neq$ size($G_j$) then remove $rp_k$ from smallest($G_i, G_j$)
        3.2. else remove $rp_k$ from group with class label $\alpha$ such that sup($\alpha$, $D$) $\leq$ sup($\beta$, $D$) and $\alpha, \beta$ are class labels of either $G_i$ or $G_j$
    4. For each restrictive pattern $rp_i \in R_P$ do
        4.1. $Victim_{rp_i} \leftarrow \alpha$ such that $\alpha$ is the class label of $G$ and $rp_i \in G$
Step 3. For each restrictive pattern $rp_i \in R_P$ do
    1. $NumTrans_{rp_i} \leftarrow |T[rp_i]| \times (1 - \psi)$   // $|T[rp_i]|$ is the number of sensitive transactions for $rp_i$
Step 4. $D' \leftarrow D$
    For each restrictive pattern $rp_i \in R_P$ do
    1. Sort_Transactions($T[rp_i]$);   //in descending order of degree of conflict
    2. $TransToSanitize \leftarrow$ Select first $NumTrans_{rp_i}$ transactions from $T[rp_i]$
    3. in $D'$ foreach transaction $t \in TransToSanitize$ do
        3.1. $t \leftarrow (t - Victim_{rp_i})$
**End**
```

In this algorithm, Steps 1 and 3 are identical to the respective steps in the previous algorithm MinFIA. Step 4 is slightly different from Step 4 in MinFIA since the sensitive transactions are now ordered in descending order of their degree of conflict so that more conflicting transactions are selected for sanitization instead of non conflicting ones. The reason is that since the victim item now represents a set of restrictive patterns (from the same group), sanitizing a conflicting transaction will allow many restrictive patterns to be taken care of at once per sanitized transaction. There are, however, other possible strategies for step 4 that we report only in (Oliveira & Zaïane 2002) for lack of space. The goal of step 2 is to identify a victim item per restrictive pattern. This is done by first clustering restrictive patterns in a set of overlapping groups GP (task 1), such that all restrictive patterns in the same group G share the some items that are the same. The shared items are the class label of the group. For example, the patterns "ABC" and "ABD" would be in the same group labeled either A or B (depending on support of A and B - task 1, line 3). However, "ABC" could also be in another group if there was one where restrictive patterns shared "C". Tasks 2 and 3 identify such overlap between groups and eliminate it by favoring larger groups or groups with a class label with higher support in the database.

5 Experimental Results

We performed two series of experiments: the first to measure the effectiveness of our sanitization algorithms and the second to measure the efficiency and scalability of the 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 to generate a dataset containing 500 different items, with 100K transactions in which the minimum size per transaction is 40 items. The effectiveness is measured in terms of the number of restrictive patterns effectively hidden, as well as the proportion of legitimate patterns accidentally hidden due to the sanitization. Some types of sanitization could also lead to the discovery of non-existing patterns in the original database D, but not in D' the sanitized database. As depicted in Figure 1 of Section 2.2, if P is the set of all mining patterns in the original database D and we have some restricted patterns R_P, the legitimate patterns are $\tilde{\ }R_P$ such that $P = \tilde{\ }R_P \cup R_P$. After sanitization, the patterns P' that should be discovered from the sanitized database D' should be equal to $\tilde{\ }R_P$ and only $\tilde{\ }R_P$. However, this is not the case, and we have three possible problems, as illustrated in Figure 1.

Problem 1 occurs when some restrictive patterns are discovered. We call this problem ***Hiding Failure***, and it is measured in terms of the percentage of restrictive patterns that are discovered from D'. Ideally, the hiding failure should be 0%. The hiding failure is measured by $HF = \frac{\#R_P(D')}{\#R_P(D)}$ where $\#R_P(X)$ denotes the number of restrictive patterns discovered from database X. In our framework, the proportion of restrictive patterns that are nevertheless discovered from the sanitized database can be controlled with the disclosure threshold ψ, and this proportion ranges from 0% to 100%. Note that ψ does not control the *hiding failure* directly, but indirectly by controlling the proportion of sensitive transactions to be sanitized for each restrictive pattern.

Problem 2 occurs when some legitimate patterns are hidden by accident. This happens when some non-restrictive patterns lose support in the database due to the sanitization process. We call this problem ***Misses Cost***, and it is measured in terms of the percentage of legitimate patterns that are not discovered from D'. In the best case, this should also be 0%. The misses cost is calculated as follows: $MC = \frac{\#{\sim}R_P(D) - \#{\sim}R_P(D')}{\#{\sim}R_P(D)}$ where $\# \sim R_P(X)$ denotes the number of non-restrictive patterns discovered from database X. Notice that there is a compromise between the misses cost and the hiding failure. The more restrictive patterns we hide, the more legitimate patterns we miss. This is basically the justification for our disclosure threshold ψ, which with tuning, allows us to find the balance between privacy and disclosure of information whenever the application permits it.

Problem 3 occurs when some artificial patterns are generated from D' as a product of the sanitization process. We call this problem **Artifactual Patterns**, and it is measured in terms of the percentage of the discovered patterns that are artifacts. This is measured as: $AP = \frac{|P'| - |P \cap P'|}{|P'|}$ where $|X|$ denotes the cardinality of X. 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 patterns may emerge. However, in our experiments, AP was always 0% with all algorithms regardless of the values of ψ.

5.1 Measuring effectiveness

We selected for our experiments a set of ten restrictive patterns from the dataset ranging from two to five items in length, with support ranging from 20% to 40% in the database.

Note that the higher the support for the restrictive patterns, the larger the number of sensitive transactions, and the greater the impact of the sanitization on the database.

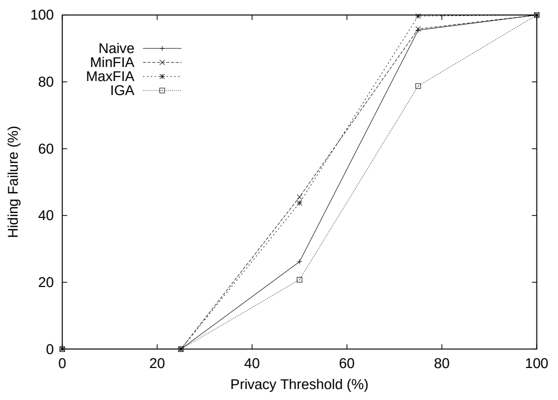
We ran the Apriori algorithm to select such patterns. The time required to build the inverted file in main memory was 4.05 seconds. Based on this inverted file, we retrieved all the sensitive transactions in 1.02 seconds. With our ten original restrictive patterns, 22479 patterns (out of 1866693) became restricted in the database since any pattern that contains restrictive patterns should also be restricted. Thus, in our experiments R_P contained the 22479 restrictive patterns.

Figure 5 shows the effect of the disclosure threshold ψ on the hiding failure and the misses cost for all four algorithms, considering the minimum support threshold $\sigma = 10\%$. As can be observed, when ψ is 0%, no restrictive pattern is disclosed for all four algorithms. However, 90% of the legitimate patterns in the case of Naïve, 69% in the case of MaxFIA, 65% in the case of MinFIA, and 44% in the case of IGA are accidentally hidden.

When ψ is equal to 100%, all restrictive patterns are disclosed and no misses are recorded for legitimate patterns. What can also be observed is that the hiding failure for IGA is better than that for the other approaches. In addition, the impact of IGA on the database is smaller and the misses cost of IGA is the lowest among all approaches until $\psi = 40\%$. After this value, MinFIA and MaxFIA yield better results than IGA's.

Figures 6, 7, 8, and 9 show the effect of varying the support threshold for the patterns in the mining process. Notice that the higher the support, the more effective the hiding of patterns even with a more relaxed disclosure threshold. However, the misses are more recurrent. In practice, the support threshold for mining frequent patterns should always be set to 0% since before sanitizing the database one cannot know the support threshold that the user will select. Thus, for better privacy preservation, the security administrator should assume the lowest support threshold possible.

We could measure the dissimilarity between the original and sanitized databases by computing the difference between their sizes in bytes. However, we be-



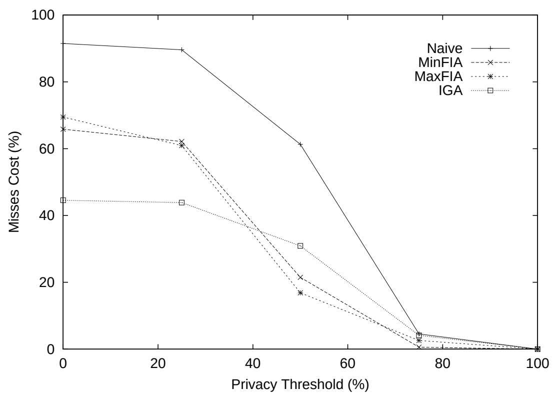
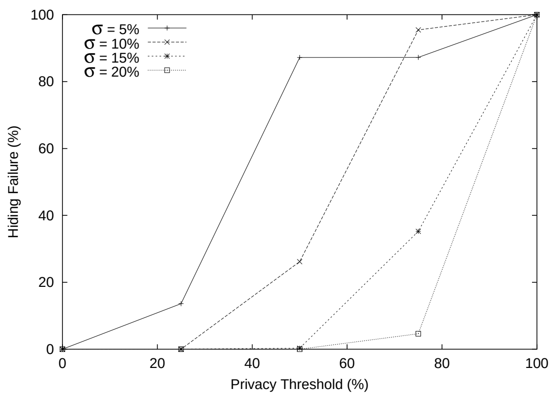


Figure 5: Effect of ψ on the hiding failure and the misses cost



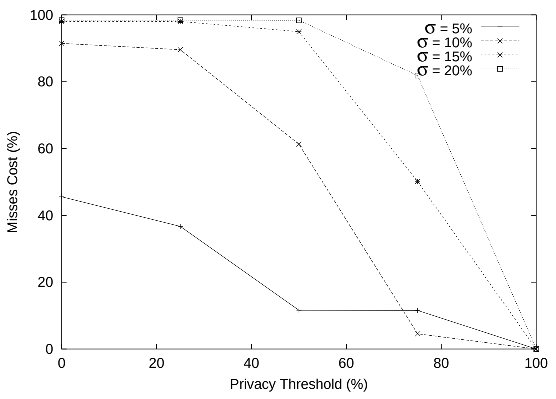


Figure 6: Effect of support threshold on privacy preservation (Naïve)

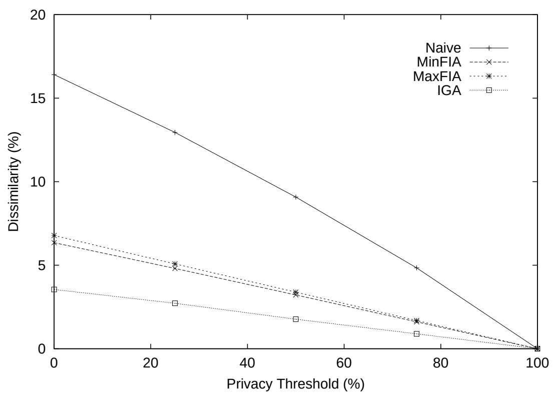


Figure 10: The difference in size between D and D'

lieve that this dissimilarity should be measured comparing their contents, instead of their sizes. Comparing their contents is more intuitive and gages more accurately the modifications made to the transactions in the database.

To measure the dissimilarity between the original and the sanitized datasets we could simply compare the difference of their histograms. In this case, the horizontal axis of a histogram contains all items in the dataset, while the vertical axis corresponds to their frequencies. The sum of the frequencies of all items gives the total of the histogram. So the dissimilarity between D and 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.

Figure 10 shows the differential between the initial size of the database and the size of the sanitized database with respect to the disclosure threshold ψ. To have the smallest impact possible on the database, the sanitization algorithm should not reduce the size of the database significantly. As can be seen in the first graph, IGA is the one that impacts the least on the database for all values of the disclosure threshold ψ. In the worst case, when $\psi = 0\%$, 3.55% of the database is lost. MinFIA and MaxFIA lose 6.35% and 6.78% respectively, and Naïve reduces 16.41% of the database, with the same threshold. This is due to the fact that Naïve removes all items of a restrictive pattern in its corresponding sensitive transactions, while the other algorithms only remove one item for each restrictive pattern. Thus, as can be seen, the four algorithms slightly alter the data in the original database, while enabling flexibility for someone to tune them.

5.2 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 patterns to hide. We varied the size of the original database D from 20K transactions to 100K transactions, while fixing the disclosure threshold ψ and the support threshold to 0%, and keeping the set of restrictive patterns constant (10 original patterns). Figure 11A shows that the four algorithms increase CPU time almost linearly with the size of the database. Note that Naïve, MinFIA, and MaxFIA yield almost the same CPU time since they are very similar. The I/O time (2 scans of the database) is also considered in these figures. This demonstrates

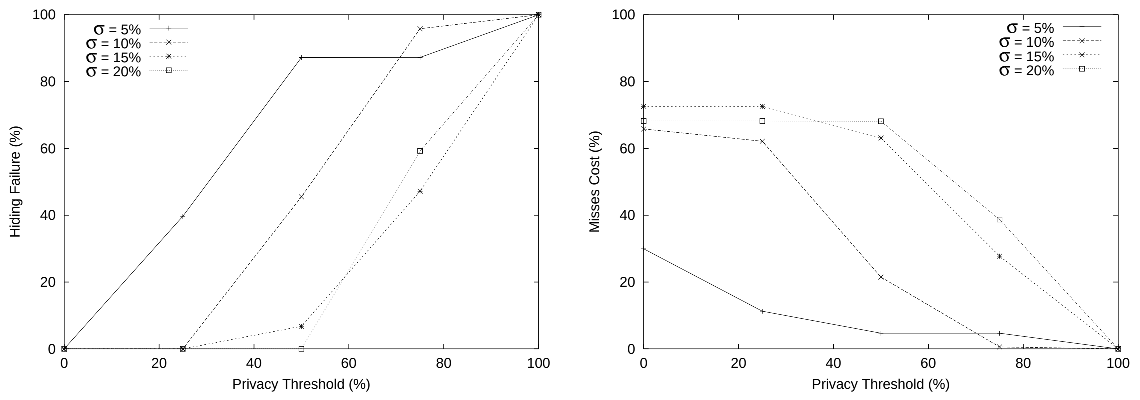


Figure 7: Effect of support threshold on privacy preservation (MinFIA)

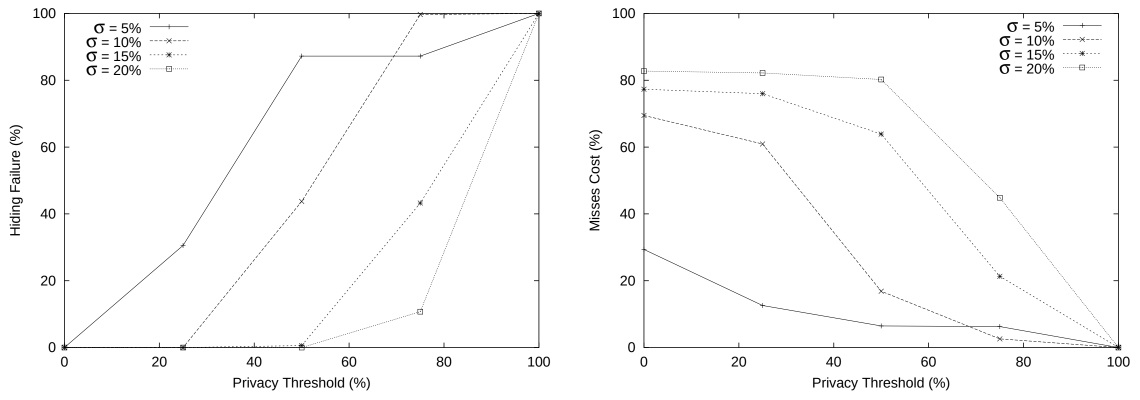


Figure 8: Effect of support threshold on privacy preservation (MaxFIA)

good scalability with the cardinality of the transactional database.

The slight inflection in the curve is due to the fact that the larger the database, the more restrictive patterns can be derived from the 10 original restrictive patterns. This significantly increases the number of sensitive transactions to be touched. The number of restrictive patterns actually increases more than the number of transactions, making most sensitive transactions conflicting ones (i.e. with many restrictive patterns).

We also varied the number of restrictive patterns to hide from approximately 1290 to 6600, while fixing the size of the database to 100K transactions and fixing the support and disclosure threshold as before. Figure 11B shows that our algorithms scale well with the number of patterns to hide. The figure reports the size of the original set of restricted patterns, which varied from 2 to 10. This makes the set of all restricted patterns range from approximately 1298 to 6608. Again, the algorithms Naïve, MinFia, and MaxFIA yielded results very similar.

This scalability is mainly due to the inverted files we use in our approaches for indexing the transactions per item and indexing the sensitive transactions per restrictive pattern. 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.

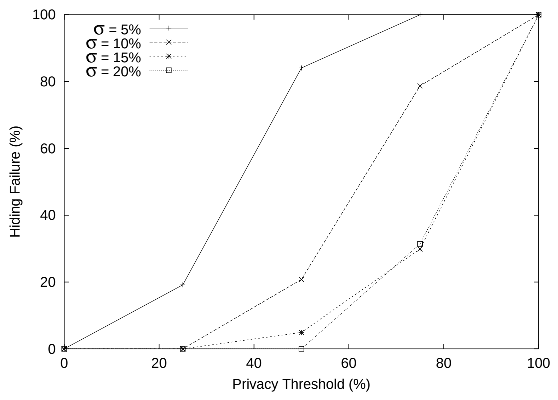
6 Related Work

Some effort has been made to address the problem of privacy preserving in data mining. Such investigation considers how much information can be inferred or calculated from large data repositories made available through data mining algorithms and looks for ways to minimize the leakage of information. This effort has been restricted basically to classification and association rules. In this work, we focus on the latter category.

Atallah et al. (Atallah et al. 1999) 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. This work was extended in (Dasseni et al. 2001), in which the authors investigated confidentiality issues of a broad category of association rules. This solution requires CPU-intensive algorithms and, in some way, modifies true data values and relationships.

In the same direction, Saygin et al. (Saygin et al. 2001) 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.

Related to privacy preserving in data mining, but in another direction, Evfimievski et al. (Evfimievski et al. 2002) proposed a framework for mining association rules from transactions consisting of categorical items in which the data has been randomized to preserve privacy of individual transactions. Although this strategy is feasible to recover association rules and preserve privacy using a straightforward uniform



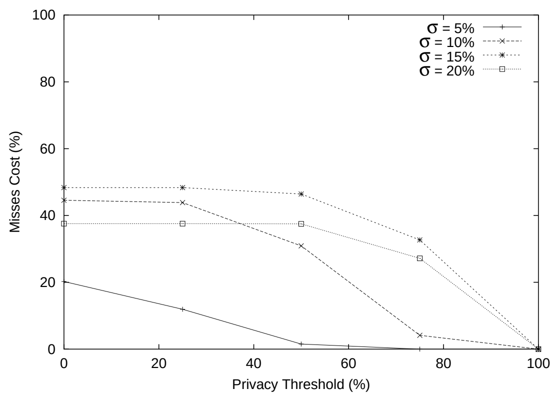
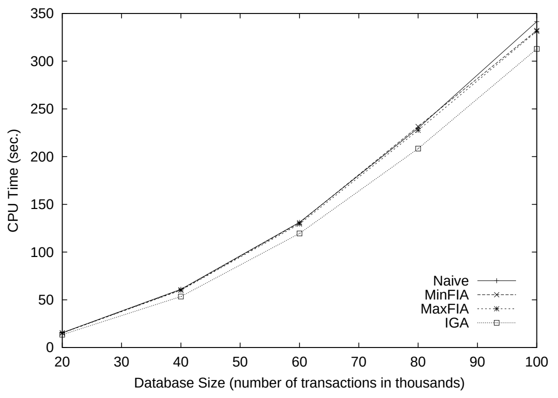


Figure 9: Effect of support threshold on privacy preservation (IGA)



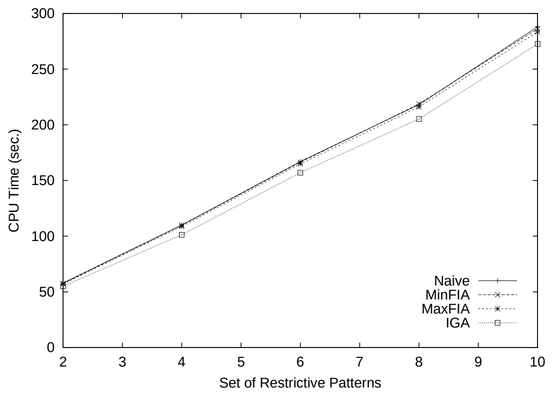


Figure 11: (A): CPU time wrt database size (B): CPU time wrt number of restrictive patterns

randomization, it introduces some false drops and may lead a miner to find associations rules that are not supposed to be discovered.

In the context of distributed data mining, Kantarcioglu and Clifton (Kantarcioglu & Clifton 2002) addressed secure mining of association rules over horizontally partitioned data. This approach considers the discovery of associations in transactions that are split across sites, without revealing the contents of individual transactions. This method is based on secure multi-party computation (Du & Atallah 2001) and incorporates cryptographic techniques to minimize the information shared, while adding little overhead to the mining task.

In (Vaidya & Clifton 2002), Vaidya and Clifton addressed the problem of association rule mining in which transactions are distributed across sources. In this approach, each site holds some attributes of each transaction, and the sites wish to collaborate to identify globally valid associations rules. This technique is also based on secure multi-party computation.

Although the papers mentioned above deal with privacy preserving in association rules, in this paper, our work is directly related to (Atallah et al. 1999, Dasseni et al. 2001, Saygin et al. 2001). Our work 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 patterns but also on maximizing the discovery of patterns after sanitizing a database. More importantly, our sanitizing algorithms select sensitive transactions with the lowest degree of conflict and remove from them the victim item with specific criteria, while the algorithms in related work remove and/or add items from/to transactions without taking into account the impact on the sanitized database. Third, our framework can achieve a reasonable performance since it is built on indexes. Another difference of our framework from the related work is that we “plug” a transaction retrieval search engine for searching transaction IDs through the transactional database efficiently.

7 Conclusions

In this paper, we have introduced a new framework for enforcing privacy in mining frequent patterns, which combines three advances for efficiently hiding restrictive rules: inverted files, one for indexing the transactions per item and a second for indexing the sensitive transactions per restrictive pattern; a transaction retrieval engine relying on Boolean queries for retrieving transaction IDs from the inverted file and combining the resulted lists; and a set of sanitizing algorithms. This framework aims at meeting a balance between privacy and disclosure of information.

In the context of our framework, the integration of the inverted file and the transaction retrieval engine are essential to speed up the sanitization process. This is due to the fact that these two modules feed the sanitizing algorithms with a set of sensitive transactions to be sanitized. It should be noticed that this index schema and the transaction retrieval engine are simple to be implemented and can deal with large databases without penalizing the performance since these two techniques are scalable.

The experimental results revealed that our algorithms for sanitizing a transactional database can achieve reasonable results. Such algorithms slightly alter the data while enabling flexibility for someone to tune them. In particular, the IGA algorithm reached the best performance, in terms of dissimilarity and preservation of legitimate frequent patterns. In addition, the IGA algorithm also yielded the best response time to sanitize the experimental dataset.

Another contribution of this work includes three performance measures that quantify the fraction of mining patterns which are preserved in the sanitized database. The Hiding Failure measures the amount of restrictive patterns that are disclosed after sanitization. Misses Cost measures the amount of legitimate patterns that are hidden by accident after sanitization, and Artifactual Patterns measure the artificial patterns created by the addition of noise in the data. We evaluated such metrics by testing different values of the disclosure threshold ψ for our algorithms.

The work presented herein addresses the issue of hiding some frequent patterns from transactional databases. All association rules derivable from these frequent patterns are thus also hidden. This could make the approach sometimes restrictive. For instance, if the pattern ABC is restricted, the pattern $ABCD$ would also be restricted since it includes the previous one, and the association rule $ABC \rightarrow D$ would be hidden even though initially there was no restrictions on D. There is no means to specify the constraints on the association rules rather than the frequent patterns. One may want to express that $AB \rightarrow C$ is restricted but not $C \rightarrow AB$. However, this is not feasible at the frequent patterns level since both rules are derived from the same frequent pattern ABC. We are investigating new optimal sanitization algorithms that minimize the impact in the sanitized database. We are also investigating, in the context of privacy in data mining, association rules or other patterns, the integration of role-based access control in relational databases with rule-based constraints specifying privacy policies.

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. The authors would like to acknowledge the helpful comments made by the anonymous reviewers of this paper.

References

Atallah, M., Bertino, E., Elmagarmid, A. K., Ibrahim, M. & Verykios, V. S. (1999), Disclosure Limitation of Sensitive Rules, *in* 'IEEE Knowledge and Data Engineering Workshop', Chicago, Illinois, USA, pp. 45–52.

Baeza-Yates, R. & Ribeiro-Neto, B. (1999), *Modern Information Retrieval*, Addison Wesley Longman.

Brankovic, L. & Estivill-Castro, V. (1999), Privacy Issues in Knowledge Discovery and Data Mining, *in* 'Australian Institute of Computer Ethics Conference (AICEC99)', Melbourne, Australia, pp. 89–99.

Clifton, C. & Marks, C. (1996), Security and Privacy Implications of Data Mining, *in* 'Workshop on Data Mining and Knowledge Discovery', Montreal, Canada, pp. 15–19.

Clifton, C. (2000), 'Using Sample Size to Limit Exposure to Data Mining', *Journal of Computer Security* **8**(4), 281–307.

Clifton, C., Du, W., Atallah, M., Kantarcioglu, M., Lin, X. & Vaidya, J. (2001), Distributed Data Mining to Protect Information Privacy, Proposal to the National Science Foundation, December 2001.

Dasseni, E., Verykios, V. S., Elmagarmid, A. K. & Bertino, E. (2001), Hiding Association Rules by Using Confidence and Support, *in* '4th Information Hiding Workshop', Pittsburg, PA, USA, pp. 369–383.

Dietzfelbinger, M., Karlin, A. R., Mehlhorn, K., auf der Heide, F. M., Rohnert, H. & Tarjan, R. E. (1994), 'Dynamic Perfect Hashing: Upper and Lower Bounds', *SIAM Journal on Computing* **23**(4), 738–761.

Du, W. & Atallah, M. J. (2001), Secure Multi-Party Computation Problems and their Applications: A Review and Open Problems, *in* '10th ACM/SIGSAC New Security Paradigms Workshop', Cloudcroft, New Mexico, pp. 13–22.

Evfimievski, A., Srikant, R., Agrawal, R. & Gehrke, J. (2002), Privacy Preserving Mining of Association Rules, *in* '8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', Edmonton, AB, Canada, pp. 217–228.

Han, J. & Kamber, M. (2001), *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, San Francisco, CA, USA.

Johnsten, T. & Raghavan, V. V. (1999), Impact of Decision-Region Based Classification Mining Algorithms on Database Security, *in* '13th Annual IFIP WG 11.3 Working Conference on Database Security', Seattle, USA, pp. 177–191.

Kantarcioglu, M. & Clifton, C. (2002), Privacy-Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data, *in* 'ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery', Madison, Wisconsin, USA.

Oliveira, S. R. M. & Zaïane, O. R. (2002), A Framework for Enforcing Privacy in Mining Frequent Patterns, TR02-13, Department of Computing Science, University of Alberta, Canada.

Rizvi, S. J. & Haritsa, J. R. (2002), Maintaining Data Privacy in Association Rule Mining, *in* '28th International Conference on Very Large Data Bases', Hong Kong, China.

Saygin, Y., Verykios, V. S. & Clifton, C. (2001), 'Using Unknowns to Prevent Discovery of Association Rules', *SIGMOD Record* **30**(4), 45–54.

Vaidya, J. & Clifton, C. (2002), Privacy Preserving Association Rules Mining in Vertically Partitioned Data, *in* '8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', Edmonton, AB, Canada, pp. 639-644.

9 Rejoinder to Reviewers Remarks

In this section, we address some technical questions raised by the reviewers of this paper. We believe that this kind of discussion is fruitful to advance the state of the art in privacy and data mining. We realize that this is only the starting point, and encourage others to address these technical questions.

9.1 Indicate applications of the sanitizing procedure

The idea behind data sanitization comes from the procedure of data cleaning, typically part of data pre-processing. While data cleaning routines operate on the data to clean it by filling in missing values, smoothing noisy data, and identifying or resolving inconsistencies; data sanitization procedures act on the data to remove or hide, or even alter sensitive information from the data.

The scenario we address in this paper is one which deals with two parties A and B, A owning a transactional database and B wanting to mine it for frequent itemsets (or association rules). If A is willing to make the database available to B provided that B cannot find some specific restricted patterns r_p, whether r_p exist or not is not really the issue, the question is how can we make these patterns r_p hidden regardless of which minimum support threshold B would use. Note that A does not know which frequent itemset mining algorithm or support threshold B would use. Sanitization is the process by which A's transactional database is altered to generate a new database which in turn is provided to B and guarantees that B would not discover any r_p.

There are many examples of applications that motivated the research on sanitization in previous publications. For example, in (Clifton & Marks 1996) the authors discussed the confidentiality of sensitive information that is not only limited to patterns of individuals, but also to other scenarios where data mining techniques can be used to provide business competitors with an advantage. They emphasized that data mining technology provides a whole new way of exploring large databases. However, in the wrong hands this may be a problem. Among the scenarios discussed in that paper, the authors provided a well designed scenario of how different data mining techniques can be used in a business setting to provide business competitors with an advantage.

This scenario is stated as follows: suppose a situation exists in which one supplier offers products in reduced price 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 highly sensitive knowledge from unclassified data that is 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. This application is also mentioned in (Dasseni et al. 2001) as a motivating example.

In (Atallah et al. 1999), the authors argued that data sanitization could be any interesting building block solution to the problem of data mining and security. In particular, the authors identified several scenarios in which data sanitization could provide a reasonable solution. For example, data products (e.g. macrodata or tabular data and microdata or raw data records), are designed to inform public or business policy, and research or public information. Solutions to this problem require combining various techniques and mechanisms, such as suppression, generalization, data swapping, among others. Among these techniques, data sanitization seems an interesting approach to hide sensitive information from public access. In that particular paper, Atallah et al. attempted to selectively hide some association rules from large databases with as little as possible impact on other non-sensitive ones. The authors also mentioned that one important extension of their work should be metrics to quantify the difference between the released database and the original one. In our paper, we provide such metrics.

Apart from these papers, in (Saygin et al. 2001) the authors also presented a method for selectively removing individual values from a database to prevent the discovery of a set of rules. This method relies on data sanitization and it works by replacing select attribute values with unknowns. This technique was designed for applications where it is necessary to store imprecise or unknown values for some attributes, such as when actual values are confidential or not available.

9.2 How can users know the patterns they want to restrict, specially if they can not imagine them? (If we know which patterns to deny, in a sense we already know the patterns)

Indeed, the discovery of patterns in large databases is an exploratory task in nature. Users look for hidden patterns without a predetermined idea or hypothesis about what patterns may be. However, in the context of enforcing privacy in data mining some constraints are necessary. To guarantee privacy, organizations have to pay an extra cost to meet security requirements.

In the scenario we described in Section 9.1, A, the owner of the transactional database, has full access to the database and would know what should be restricted based on the application and the database content, whether these patterns to restrict exist in the database or not. The objective of the sanitization is to hide these patterns if they exist. A does not know if the restricted patterns exist and does not have to look for them. A only knows that if these patterns exist they should not be disclosed to B. The user B has no knowledge that some patterns were hidden. Once B gets access to the sanitized database, B can mine any available pattern. The restricted patterns, if they existed in the original database, are supposedly removed by the sanitization process. In other words, the user B doesn't have to know about the patterns, and the database owner A

doesn't have to know whether the patterns exist or not in the original database. A only needs to know which patterns (existing or not) should not be disclosed.

Let us analyze different approaches proposed in the literature. For example, in (Kantarcioglu & Clifton 2002) the authors incorporate cryptography techniques to minimize the information shared, while adding overhead to the mining task. The same technique, i.e., multi-party computation, is considered in the solution proposed in (Vaidya & Clifton 2002). Note that the extra cost in these approaches is caused by the incorporation of cryptography, which implies an overhead to the discovery process.

In the recent International Conference on Very Large Data Bases (28th VLDB), Rizvi and Haritsa (Rizvi & Haritsa 2002) proposed a scheme, based on probabilistic distortion of used data. Although this framework provides a high degree of privacy to the user and retains a high level of accuracy in the mining results, mining the distorted database can be, apart from being error-prone, significantly more expensive in terms of both time and space as compared to mining the original database. Again, solutions to address privacy and data mining always deal with a trade-off.

Another approach, based on data obfuscation, was proposed in (Evfimievski et al. 2002). Evfimievski et al. showed that their technique is feasible to recover association rules and preserve privacy using a straightforward uniform randomization, but there is a cost to be paid: their technique introduces some false drops to the discovered process.

In the context of data sanitization, we also have to pay an extra cost. In this particular case, we need to look ahead for some sensitive patterns in order to prevent them and guarantee privacy. However, this approach does not introduce false drops to the data and does not require any overhead to the mining process after sanitizing a database. In addition, we pay an extra cost because some patterns will be accidentally removed since there are functional dependencies between restricted and non-restricted patterns.

9.3 Users would be required to submit the patterns they do not want to be mined, but who would store this information? Users would have to strongly trust the archiver of transactions?

There is no requirement to submit patterns to hide or archive them with a trusted archiver. The database owner simply knows the patterns not to disclose to a certain user and sanitizes the database to hide those patterns. If two users request access to the database for mining, the database owner could sanitize the database twice differently to hide different patterns for each user.

The problem considered in the Introduction Section of this paper could be addressed in different ways. For example, in (Evfimievski et al. 2002) the authors suggest that the clients should send patterns instead of the data. To do so, the data has been randomized to preserve the privacy of individuals in transactions. Although this strategy is feasible to recover some original patterns and preserve privacy, this introduces some false drops and may lead a miner (the server) to find patterns that are not supposed to be discovered. In this case, some recommendations will be misleading users. A more devastating situation would be to use an approach like this with in patient's record and jeopardize the life of the patients by misleading the medical decision makers with wrong patterns.

Note that in our motivating example, we consider the situation in which the clients will be sending a sanitized dataset, instead of patterns. We believe that this approach is reasonable for the following reasons: (1) We do not know the thresholds that the user would want to apply during the mining; (2) The sanitized dataset will not create false drops, so that the server will recommend trusted information, although some recommendations will not be considered, since some patterns will be removed in the sanitization process.

9.4 Why not obtain all patterns, remove those that users do not want and publish all the remaining ones?

The simplistic solution to address our motivating example is to implement a filter after the mining phase to weed out/hide the restricted discovered patterns. In this case, we could remove the restrictive patterns and send only the non-restrictive ones. Again, we argue that sending the sanitized database is more efficient than sending patterns. The database owner does not know the type of patterns that the user is looking for. In other words, the user could use any support thresholds. The database owner cannot guess apriori the thresholds to be used.

9.5 How can we avoid sanitizing to a threshold ψ, but the user may apply a lower value of ψ and still retrieve and disclose the restricted patterns. This clearly is a weakness of the algorithm. How much does the user has to reduce ψ to de-sanitize the sanitized database?

We argue that the disclosure threshold proposed in our hiding strategies is not a weakness, but a strength of our approach. First of all the threshold ψ is set only by the database owner before sanitization. This threshold represents the tradeoff between hiding all restrictive patterns as well as accidentally other legitimate patterns, and disclosing some restrictive patterns but hiding less of the legitimate patterns. The user has no knowledge of ψ as the user does not participate in the sanitization process. Moreover, by setting the disclosure threshold ψ to zero, our algorithms guarantee that no restrictive pattern is disclosed. It is up to the database owner, who sets ψ, to decide about the acceptable tradeoff. It is important to note that our sanitization methods are lossy in the sense that there is no de-sanitization possible. The alteration 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.