Secure Association Rule Sharing

Stanley R. M. Oliveira[1], Osmar R. Zaïane[1], and Yücel Saygin[2]
{oliveira | zaiane}@cs.ualberta.ca, ysaygin@sabanciuniv.edu

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

[2]Faculty of Engineering and Natural Sciences
Sabanci University, Orhanli, 34956, Istanbul, Turkey

Abstract. The sharing of association rules is often beneficial in industry, but requires privacy safeguards. One may decide to disclose only part of the knowledge and conceal strategic patterns which we call restrictive rules. These restrictive rules must be protected before sharing since they are paramount for strategic decisions and need to remain private. To address this challenging problem, we propose a unified framework for protecting sensitive knowledge before sharing. This framework encompasses: (a) an algorithm that sanitizes restrictive rules, while blocking some inference channels. We validate our algorithm against real and synthetic datasets; (b) a set of metrics to evaluate attacks against sensitive knowledge and the impact of the sanitization. We also introduce a taxonomy of sanitizing algorithms and a taxonomy of attacks against sensitive knowledge.

keywords: Privacy preserving data mining, Protecting sensitive knowledge, Sharing association rules, Data sanitization, Sanitizing algorithms.

1 Introduction

Protecting against inference learning in data mining sense has begun to receive attention. In particular, the problem of privacy preservation, when sharing data while wanting to conceal some restrictive associations has been addressed in the literature [1, 2, 7, 4, 5]. The proposed solutions consist in transforming a transactional database to be shared in such a way that the restrictive rules cannot be discovered. This process is called data sanitization [1]. The effectiveness of the data sanitization is measured by the proportion of restrictive rules effectively hidden (hiding failure), the proportion of rules accidentally hidden (misses cost) and the amount of artifactual rules created by the process [4]. The problem we address here is different and more practical. It is the problem of *rule sanitization*. Rather than sharing the data, collaborators prefer to mine their own data and share the discovered patterns.

Let us consider a motivating example based on a case discussed in [3]. Suppose we have a server and many clients in which each client has a set of sold items (e.g. books, movies, etc). 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 association rules. In this context, the clients represent companies and the server is a recommendation system for an e-commerce application, for example, fruit of the clients collaboration. In the absence of rating, which is used in collaborative filtering for automatic recommendation building, association rules can be effectively used to build models for on-line recommendation.

When a client sends its frequent itemsets or association rules to the server, it sanitizes some restrictive itemsets according to some specific policies. The server then gathers statistical information from the sanitized itemsets and recovers from them the actual associations.

The simplistic solution to address the motivating example is to implement a filter after the mining phase to weed out/hide the restricted discovered rules. However, we claim that trimming some rules out does not ensure full protection. The sanitization applied to the set of rules must not leave a trace that could be exploited by an adversary. We must guarantee that some inference channels have been blocked as well.

This paper introduces the notion of rule sanitization. The main contribution of this paper is a novel framework for protecting sensitive knowledge before sharing association rules. This framework encompasses: (a) a sanitizing algorithm called Downright Sanitizing Algorithm (DSA). This algorithm sanitizes a set of restrictive rules while blocking some inference channels; (b) a set of metrics to evaluate attacks against sensitive knowledge and the impact of the sanitization. Another contribution is a taxonomy of existing sanitizing algorithms. Finally, we present a taxonomy of attacks against sensitive knowledge. To our best knowledge, the investigation of attacks against sensitive knowledge, notably in the context of data or rule sanitization, has not been explored in any detail.

This paper is organized as follows. Related work is reviewed in Section 2. The problem definition is stated in Section 3. In Section 4, we present our framework for protecting sensitive knowledge. In Section 5, we introduce our Downright Sanitizing Algorithm (DSA). The experimental results and discussion are presented in Section 6. 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 protecting sensitive knowledge in association rule mining by data sanitization. The existing sanitizing algorithms can be classified into two major classes: *Data-Sharing approach* and *Pattern-Sharing approach*, as can be seen in Figure 1A. In the former, the sanitization process acts on the data to remove or hide the 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 some noise, i.e., new items not originally present in such transactions. In the latter, the sanitizing algorithm acts on the rules mined from a database, instead of the data itself. The only known algorithm in this category is our DSA algorithm herein presented. The algorithm removes all restrictive rules before the sharing process.

Among the algorithms of the Data-Sharing approach, we classify the following categories: *Item Restriction-Based*, *Item Addition-Based*, and *Item Obfuscation-Based*.

Item Restriction-Based: These algorithms [2] remove one or more items from a group of transactions containing restrictive rules. In doing so, the algorithms hide restrictive rules by reducing either their support or confidence below a privacy threshold. Other algorithms [4–6], that lie in this category, hide rules by satisfying 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.

Item Addition-Based: Unlike the previous algorithms, item addition-based algorithms modify existing information in transactional databases by adding some items not originally present in some transactions. The items are added to the antecedent part X of a rule $X \rightarrow Y$ in transactions that partially support it. In doing so, the confidence of such a rule is decreased. This approach [2] may generate artifacts such as artificial association rules that would not exist in the original database.

Item Obfuscation-Based: These algorithms [7] hide rules by placing a mark "?" (unknowns) in items of some transactions containing restrictive rules, instead of deleting such items. In doing so, these algorithms obscure a given set of restrictive rules by replacing known values with unknowns. Like the item reduction-based algorithms, these algorithms reduce the impact in the sanitized databases protecting miners from learning "false" rules.

The work presented here differs from the related work in some aspects, as follows: first, our algorithm addresses the issue of pattern sharing and sanitizes rules, not transactions. Second, we study attacks against sensitive knowledge in the context of rule sanitization. This line of work has not been considered so far. Most importantly, our contribution in rule sanitization and the existing solutions in data sanitization are complementary.

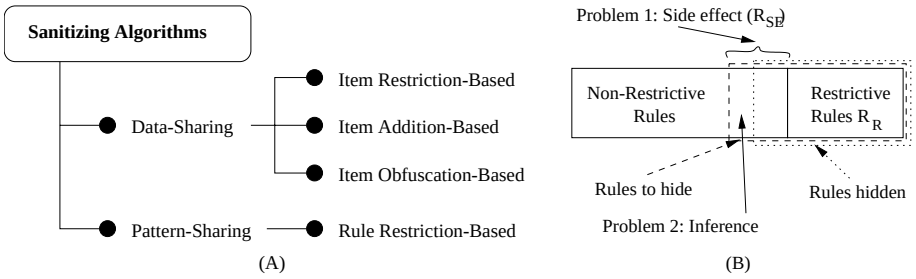


Fig. 1. (A): A Taxonomy of Sanitizing Algorithms. (B): Rule Sanitization problems.

3 Problem Definition

The specific problem addressed in this paper can be stated as follows: Let D be a database, R be the set of rules mined from D based on a minimum support threshold σ, and R_R be a set of restrictive rules that must be protected according to some security/privacy policies. The goal is to transform R into R', where R' represents the set of non-restrictive rules. In this case, R' becomes the released set of rules that is made available for sharing.

Ideally, $R' = R - R_R$. However, there could be a set of rules r in R' from which one could derive or infer a restrictive rule in R_R. So in reality, $R' = R - (R_R + R_{SE})$, where R_{SE} is the set of non-restrictive rules that are removed as side effect of the sanitization process to avoid recovery of R_R.

Figure 1B illustrates the problems that occur during the rule sanitization process. Problem 1 conveys the non-restrictive rules that are removed as a side effect of the process (R_{SE}). We refer to this problem as *side effect*. It is related to the misses cost problem in data sanitization [4]. Problem 2 occurs when using some non-restrictive rules, an adversary may recover some restrictive ones by inference channels. We refer to such a problem as *recovery factor*.

4 Framework For Protecting Sensitive Knowledge

Before introducing our framework for protecting sensitive knowledge, we briefly review some terminology from graph theory. We present our new sanitizing algorithm in Section 5.2.

4.1 Basic Definitions

The itemsets in a database can be represented in terms of a directed graph. We refer to such a graph as *frequent itemset graph* and define it as follows:

Definition 1 (Frequent Itemset Graph). *A frequent itemset graph, denoted by $G = (C, E)$, is a directed graph which consists of a nonempty set of frequent itemsets C, a set of edges E that are ordered pairings of the elements of C, such that $\forall u, v \in C$ there is an edge from u to v if $u \cap v = u$ and if $|v| - |u| = 1$ where $|x|$ is the size of itemset x.*

Figure 2b shows a frequent itemset graph for the sample transactional database depicted in Figure 2a. In this example, the minimum support threshold σ is set to 2. As can be seen in Figure 2b, in a frequent itemset graph G, there is an ordering for each itemset. We refer to such an ordering as *itemset level* and define it as follows:

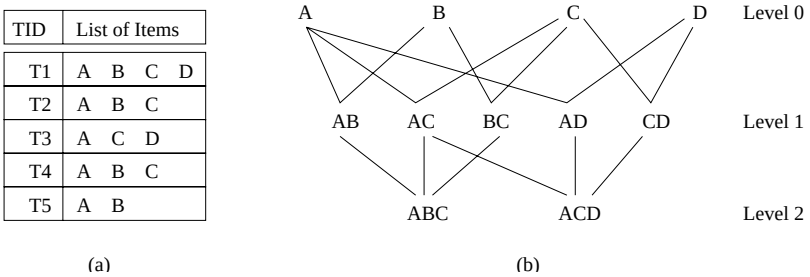


Fig. 2. (a) A transactional database. (b) The corresponding frequent itemset graph.

Definition 2 (Itemset Level). *Let $G = (C, E)$ be a frequent itemset graph. The level of an itemset u, such that $u \subset C$, is the length of the path connecting an 1-itemset to u.*

Based on Definition 2, we define the level of a frequent itemset graph G as follows:

Definition 3 (Frequent Itemset Graph Level). *Let $G = (C, E)$ be a frequent itemset graph. The level of G is the length of the maximum path connecting an 1-itemset u to any other itemset v, such that $u, v \in C$, and $u \subset v$.*

In general, the discovery of itemsets in G is the result of top-down traversal of G constrained by a minimum support threshold σ. The discovery process employs an iterative approach in which k-itemsets are used to explore $(k + 1)$-itemsets.

4.2 Taxonomy of Attacks

An attack occurs when someone mines a sanitized set of rules and, based on non-restrictive rules, deduce one or more restrictive rules that are not supposed to be discovered. We have identified some attacks against sanitized rules, as follows:

Forward-Inference Attack: Let us consider the frequent itemset graph in Figure 3A. Suppose we want to sanitize the restrictive rules derived from the itemset ABC. The naïve approach is simply to remove the itemset ABC. However, if AB, AC, and

BC are frequent, a miner could deduce that ABC is frequent. A database owner must assume that an adversary can use any inference channel to learn something more than just the permitted association rules. We refer to this attack as forward-inference attack. To handle this attack, we must also remove at least one subset of ABC in the level 1 of the frequent itemset graph. This complementary sanitization is necessary. In the case of a deeper graph, the removal is done recursively up to level 1. We start removing from level 1 because we assume that the association rules recovered from the itemsets have at least 2 items. Thus, the items in level 0 of the frequent itemset graph are not shared with a second party. In doing so, we reduce the inference channels and minimize the side effect.

Backward-Inference Attack: Another type of attack occurs when we sanitize a non-terminal itemset. Based on Figure 3B, suppose we want to sanitize any rule derived from the itemset AC. If we simply remove AC, it is straightforward to infer the rules mined from AC since either ABC or ACD is frequent. We refer to this attack as backward-inference attack. To block this attack, we must remove any superset that contains AC. In this particular case, ABC and ACD must be removed as well.

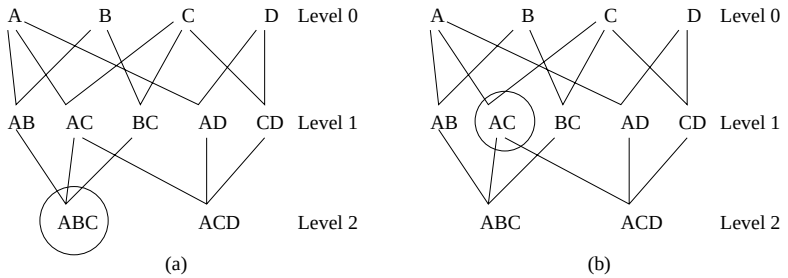


Fig. 3. (a) An example of forward-inference. (b) An example of backward-inference.

4.3 Metrics

In this section, we introduce two metrics related to the problems illustrated in Figure 1B: The *side effect* and the *recovery*.

Side Effect Factor (SEF): Measures the amount of non-restrictive association rules that are removed as side effect of the sanitization process. The side effect factor is calculated as follows: $SEF = \frac{(|R|-(|R'|+|R_R|))}{(|R|-|R_R|)}$ where R, R', and R_R represent the set of rules mined from a database, the set of sanitized rules, and the set of restrictive rules, respectively, and $|S|$ is the size of the set S.

Recovery Factor (RF): This measure expresses the possibility of an adversary recovering a restrictive rule based on non-restrictive ones. The recovery factor of one pattern takes into account the existence of its subsets. The rationale behind the idea is that all nonempty subsets of a frequent itemset must be frequent. Thus, if we recover all subsets of a restrictive itemset (rule), we say that the recovery factor for such an itemset is possible, thus we assign it the value 1. However, the recovery factor is never certain, i.e., an adversary may not learn an itemset even with its subsets. On the other hand, when not all subsets of an itemset are present, the recovery of the itemset is improbable, thus we assign value 0 to the recovery factor.

5 Rule Sanitization: The DSA Algorithm

5.1 The General Approach

Our approach has essentially three steps as follows. These steps are applied after the mining phase, i.e., we assume that the frequent itemset graph G is built. The set of all itemsets that can be mined from G, based on a minimum support threshold σ, is denoted by C.

Step1: Identifying the restrictive itemsets. For each restrictive rule in R_R, convert it to an itemset $c_i \in C$ and mark it to be sanitized.

Step2: Selecting subsets to sanitize. In this step, for each itemset c_i to be sanitized, we compute its item pairs from level 1 in G, subsets of c_i. If none of them is marked, we randomly select one of them and mark it to be removed.

Step3: Sanitizing the set of supersets of marked pairs in level 1. The sanitization of restrictive itemsets is simply the removal of the set of supersets of all itemsets in level 1 of G that are marked for removal. This process blocks inference channels.

5.2 The Downright Sanitizing Algorithm

In this section, we introduce the Downright Sanitizing Algorithm, denoted by DSA. The inputs for DSA are the frequent itemset graph G, the set of all association rules R mined from a database G, and the set of restrictive rules R_R to be sanitized. The output is the set of sanitized association rules R'.

Downright_Sanitizing_Algorithm
Input: G, R, R_R
Output: R'
Step 1. **For each** association rule $rr_i \in R_R$ **do**
  1.1. $pattern_i \leftarrow rr_i$ //Convert each rr_i into a frequent itemset $pattern_i$
Step 2. **For each** $pattern_i$ in the level k of G, where $k > 1$ **do** {
  2.1. Pairs($pattern_i$) //Compute all the item pairs of $pattern_i$
  2.2. **If** (Pairs($pattern_i$) $\cap$ $MarkedPair = \emptyset$) **then** {
    2.2.1 $p_i \leftarrow$ random(Pairs($pattern_i$)) //Select randomly a pair $p_i \in pattern_i$
    2.2.2 $MarkedPair \leftarrow MarkedPair \cup p_i$ //Update the list $MarkedPair$
  }
}
Step 3. $R' \leftarrow R$
  3.1. **For** each itemset $c_i \in G$ **do** {
    3.1.1. **If** $\exists$ a marked pair p, such that $p \in MarkedPair$ and $p \subset c_i$ **then**
      3.1.1.1. Remove(c_i) from R' //c_i belongs to the set of supersets of p
  }
End Algorithm

6 Experimental Results

In this section, we study the efficiency and scalability of DSA and the similar counterparts in the literature. There are no known algorithms for rule sanitization. However, transaction sanitization algorithms can be used for this purpose. Indeed, in order to sanitize a set of rules R to hide R_R, one can use data sanitization to transform the database D into D' to hide R_R and then mine D' to get the rules to share. We used this idea to compare our algorithm to existing data sanitization approaches.

6.1 Datasets

We validated DSA against real and synthetic datasets. The real dataset, BMS-Web-View-2 [8], placed in the public domain by Blue Martini Software. The dataset contains 22,112 transactions with 2717 distinct items, and each customer purchasing has four items on average. The synthetic dataset was generated with the IBM synthetic data generator. This dataset contains 100,000 transactions with 500 different items, and each customer purchasing has ten items on average.

6.2 Sanitizing Algorithms

For our comparison study, we selected the best sanitizing algorithms in the literature: (1) Algo2a hides restrictive rules by reducing support [2]. (2) Item Grouping Algorithm (IGA) [4] which groups restricted association rules in clusters of rules sharing the same itemsets. The shared items are removed to reduce the impact on the sanitized dataset; (3) Sliding Window Algorithm (SWA) [6] scans a group of K transactions, at a time and sanitizes the restrictive rules present in such transactions based on a disclosure threshold ψ defined by a database owner. We set the window size of SWA to 20000 transactions in both datasets; (4) An algorithm similar to DSA, called Naïve, which sanitizes restrictive itemsets and their supersets, i.e., Naïve blocks the forward-inference attack without considering blocking the backward-inference attack.

6.3 Methodology

We performed two series of experiments: the first to evaluate the effectiveness of DSA, and the second to measure its efficiency and scalability.

Our comparison study was carried out through the following steps: (1) We used the algorithms IGA, SWA, and Algo2a to sanitize both initial databases; (2) We applied the Apriori algorithm on the sanitized data to extract the rules to share. For DSA, also two steps were necessary: (1) Apply Apriori algorithm to extract rules from the two initial datasets; (2) Use DSA to sanitize these rules. The effectiveness is measured in terms of restrictive associations that can be recovered by an adversary, as well as the proportion of non-restrictive rules hidden due to the sanitization.

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. In our experiments, we selected a set of 20 restrictive association rules for the real dataset and 30 for the synthetic dataset, with rules ranging from 2 to 5 items. The real dataset has 17,667 association rules with support $\geq$ 0.2% and confidence $\geq$ 50%, while the synthetic dataset has 20,823 rules with support $\geq$ 0.1% and confidence $\geq$ 50%.

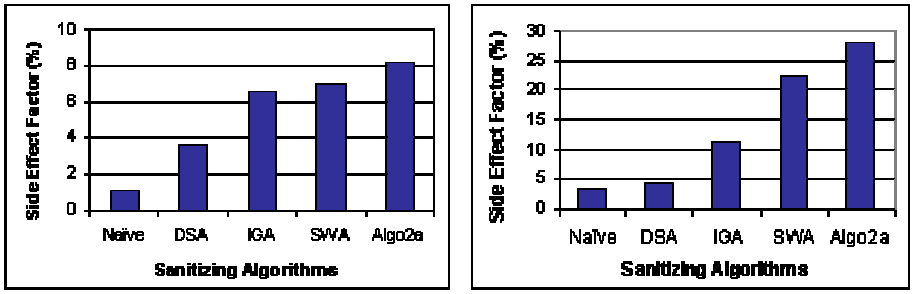
6.4 Measuring Effectiveness

In this section, we measure the effectiveness of DSA, IGA, SWA, and Algo2a considering the metrics introduced in Section 4.3.

In order to compare the algorithms under the same conditions, we set the disclosure thresholds ψ of the algorithms IGA and SWA, and the privacy threshold λ of algorithm Algo2a to 0%. In this case, all restrictive rules are completely sanitized. We purposely set these thresholds to zero because DSA always sanitizes all the restrictive rules. However, the value for the side effect factor differs from one algorithm to another. For instance, Figure 4A shows the side effect factor on the synthetic dataset. The lower the result the better. For this example, 1.09% of the non-restrictive association rules in the case of Naïve, 3.58% in the case of DSA, 6.48% in the case of IGA, 6.94% in the case of SWA, and 8.12% in the case of Algo2a are removed by the sanitization process.

Similarly, Figure 4B shows the side effect of the sanitization on the real dataset. In this situation, 3.2% of the non-restrictive association rules in the case of Naïve, 4.35% in the case of DSA, 11.3% in the case of IGA, 22.1% in the case of SWA, and 27.8% in the case of Algo2a are removed.

In both cases, Naïve yielded the best results, but we still need to evaluate how efficient Naïve is to block inference channels. We do so below. DSA also yielded promising results, while the sanitization performed by Algo2a impacts the database more significantly. An important observation here is the results yielded by SWA and IGA. Both algorithms benefit from shared items in the restrictive rules during the process of sanitization. By sanitizing the shared items of these restrictive rules, one would take care of hiding such restrictive rules in one step. As a result, the impact on the non-restrictive rules is minimized. In general, the heuristic of IGA is more efficient than that one in SWA. This explains the better performance of IGA over SWA in both datasets.



(A) (B)

Fig. 4. (A): Side effect on the synthetic dataset. (B): Side effect on the real dataset.

After identifying the side effect factor, we evaluated the recovery factor for Naïve and DSA. This measure is not applied to IGA, SWA, and Algo2a since these algorithms rely on data sanitization instead of rule sanitization. Thus, once the data is shared for mining, there is no restriction about the rules discovered from a sanitized database.

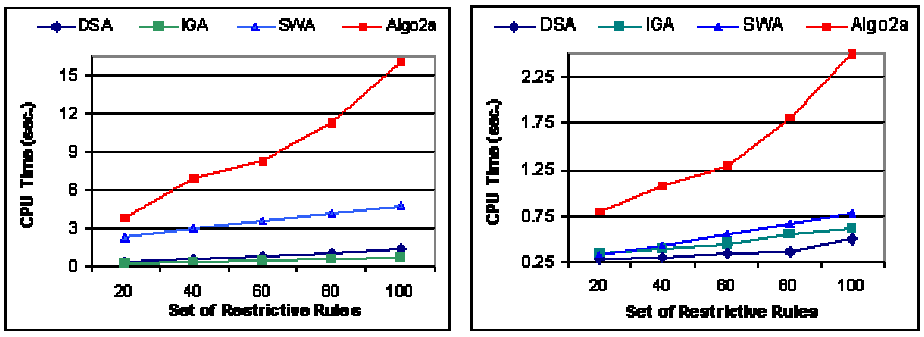
In the case of rule sanitization, some inference channels can occur, as discussed in Section 4.2. We ran a checklist procedure to evaluate how efficient is the sanitization performed by Naïve and DSA. We check the existence of any subset of the restrictive rules removed in order to identify the recovery factor. If all subsets of a rule are found, we assume the rule could be recovered. As expected, Naïve blocked well the forward-inference attacks but failed to block backward-inference attacks in both datasets. On the contrary, DSA yielded the best results in all cases, i.e., DSA blocked both forward-inference and the backward-inference attacks. The results suggested that hardly can an adversary reconstruct the restrictive rules after the sanitization performed by DSA.

6.5 CPU Time for the Sanitization Process

We tested the scalability of DSA and the other algorithms vis-à-vis the number of rules to hide. We did not plot Naïve because its CPU time is very similar to that one of DSA. We varied the number of restrictive rules to hide from 20 to 100 and set the disclosure thresholds to $\psi = 0\%$. The rules were randomly selected from both datasets. Figures 5A and 5B show that the algorithms scale well with the number of rules to hide. Note that IGA, SWA, and DSA increase CPU time linearly, 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 both DSA and SWA require only one.

Although IGA requires 2 scans, it is faster than SWA in most cases. The main reason is that SWA performs a number of operations in main memory to fully sanitize a database. IGA requires on scan to build an inverted index where the vocabulary contains the restrictive rules and the occurrences contain the transaction IDs. In the second scan, IGA sanitizes only the transactions marked in the occurrences. Another important result is that IGA and DSA yielded very similar CPU time for both datasets. In particular, IGA was better in the synthetic dataset because the transactions contain more items and IGA requires less operations in main memory.



(A) (B)

Fig. 5. (A): CPU time for the synthetic dataset. (B): CPU time for the real dataset.

6.6 General Discussion

Our experiments demonstrated the evidence of attacks in sanitized databases. The figures revealed that DSA is a promising solution to protect sensitive knowledge before sharing association rules, notably in the context of rule sanitization. DSA has a low value for side effect factor and very low recovery factor. We have identified some advantages of DSA over the previous data sanitizing algorithms in the literature as follows: (1) Using DSA, a database owner would share patterns (only rules) instead of the data itself; (2) Sanitizing rules, one reduces drastically the possibility of inference channels since the support threshold and the mining algorithm are selected previously by the database owner; and (3) Sanitizing rules instead of data results in no alteration in the support and confidence of the non-restrictive rules, i.e., the released rules have the original support and confidence. As a result, the released rules seem more interesting for practical applications. Note that the other approaches reduce support and confidence of the rules as a side effect of the sanitization process.

On the other hand, DSA reduces the flexibility of information sharing since each time a client (party) wants to try a different set of support and confidence levels, it has to request for the rules from the server.

7 Conclusions

In this paper, we have introduced a novel framework for protecting sensitive knowledge before sharing association rules.

Our contributions in this paper can be summarized as follows: First, a sanitizing algorithm called Downright Sanitizing Algorithm (DSA). DSA blocks some inference channels to ensure that an adversary cannot reconstruct restrictive rules from the non-restrictive ones. In addition, DSA reduces drastically the side effect factor during the sanitization process. Our experiments demonstrated that DSA is a promising approach for protecting sensitive knowledge before sharing association rules. Second, the framework also encompasses metrics to evaluate the effectiveness of the rule sanitization process. Another contribution is a taxonomy of existing sanitizing algorithms. We also introduced a taxonomy of attacks against sensitive knowledge.

The work presented here introduces the notion of rule sanitization, which complements the idea behind data sanitization. While data sanitization relies on protecting sensitive knowledge before sharing data, rule sanitization is concerned with the sharing of patterns. Currently, we are investigating the existence of new type of attacks against sanitized databases, and the effective response to such attacks. Another interesting issue to address is the problem of hiding rules in collective data. In our previous motivating example, if all clients share their rules in the server, but want to hide some global rules, i.e. rules that become confident with the collective support, our algorithm seems vulnerable in such context and a collaborative sanitization should be explored.

8 Acknowledgments

Stanley Oliveira was partially supported by CNPq, Brazil, under grant No. 200077/00-7. Osmar Zaïane was partially supported by a research grant from NSERC, Canada. Yücel Saygin was partially supported by European Commission under IST-2001-39151 CODMINE project. We would like to thank Elena Dasseni for providing us with the code of her algorithm for our comparison study.

References

1. 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.
2. 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.
3. A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy Preserving Mining of Association Rules. In *Proc. of the 8th ACM SIGKDD Intl. Conf. on Knowlegde Discovery and Data Mining*, pages 217–228, Edmonton, AB, Canada, July 2002.
4. 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.
5. 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)*, pages 54–63, Hong Kong, China, July 2003.
6. S. R. M. Oliveira and O. R. Zaïane. Protecting Sensitive Knowledge By Data Sanitization. In *Proc. of the 3rd IEEE International Conference on Data Mining (ICDM'03)*, pages 613–616, Melbourne, Florida, USA, November 2003.
7. Y. Saygin, V. S. Verykios, and C. Clifton. Using Unknowns to Prevent Discovery of Association Rules. *SIGMOD Record*, 30(4):45–54, December 2001.
8. Z. Zheng, R. Kohavi, and L. Mason. Real World Performance of Association Rules Algorithms. In *Proc. of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, August 2001.