AN EXPERIMENTAL STUDY OF DISTORTION-BASED TECHNIQUES FOR ASSOCIATION RULE HIDING

Emmanuel D. Pontikakis, Achilleas A. Tsitsonis, and Vassilios S. Verykios

Abstract Data mining provides the opportunity to extract useful information from large databases. Various techniques have been proposed in this context in order to extract this information in the most efficient way. However efficiency is not our only concern in this study. The security and privacy issues over the extracted knowledge must be seriously considered as well. By taking this into consideration, we study the discovery of association rules in binary data sets and we propose algorithms for selectively hiding sensitive association rules. Association rule hiding is a well researched area in privacy preserving data mining and many algorithms have been proposed to address it. The algorithms that we introduce use a distortion-based technique for hiding the sensitive rules. The hiding process may introduce a number of side effects either by generating rules which were not previously existing (ghost rules) or by eliminating existing non-sensitive rules (lost rules). The proposed algorithms use effective data structures for the representation of the association rules and they strongly rely on the prioritization of the selection of the transactions to choose for falsification (victim transactions) by using weights. In this paper we show that our algorithms perform better than other similar algorithms in this field in eliminating non-sensitive rules without increasing the processing time significantly.

1. INTRODUCTION

Data mining techniques are often used by adversaries in order to attain sensitive information in public databases. Databases that hold huge amounts of information are not only difficult to manage, but also vulnerable to misuse. Unlike most of the related work in the field of privacy preserving data mining our goal is to hide a specific set of sensitive rules that we do not wish to be made public. There are many methods developed in order to deal with this problem. The most common are (a) the distortion of the transactions by altering their original values with false ones and (b) the

deletion of a certain number of transactions that contribute to the construction of the rules we want to hide. The deletion of transactions removes too much information from the database, which is problematic in most cases.

The algorithms developed in this paper, use a binary dataset as an input and find the association rules that hold in this dataset. In the sequel these algorithms hide a subset of these rules which are considered sensitive by the user, by selectively reversing the binary values (i.e., by replacing 1's by 0's). The hiding process can affect the set of rules which can be mined from the database either by hiding rules which are not sensitive *(lost rules),* or by introducing rules which were not supported by the original database (*ghost rules*).

In the rest of this paper, we present the theoretical implications of association rules, efficient algorithms for hiding these rules, as well as implementations and experiments with binary data sets. In section 2 we present the background and related work about association rule hiding. In section 3 the notation and the main definitions regarding sensitive association rules are given. In section 4 we present algorithms for hiding association rules and we propose two algorithms that hide the rules efficiently. In section 5 we compare the effectiveness of these two algorithms and we present the experimental results from their implementation. Finally in section 6 we make concluding remarks and we propose some future work related to the improvement of these techniques.

2. BACKGROUND AND RELATED WORK

Association rule hiding is one of the techniques used in the context of privacy preserving data mining [9,10,11,12,13,14,15]. The work presented in this paper builds upon a novel idea, which was first presented in [4]. Although the authors in [4] (and [2]) rely on the hiding of large itemsets, in this work we present our findings from the comparison of two proposed algorithms for hiding association rules.

Another technique for hiding association rules in a database is discussed in [1]. The algorithms presented there use the blocking approach which refers to the selective placing of unknown values (indicated by question marks) in the database in order to increase to the maximum degree the utility of the modified database after the hiding of the association rules.

Our work belongs to the value deletion/distortion family of rule hiding techniques. One of the most common techniques for data distortion is the deletion of some values in the transactions of a binary dataset (turning 1's to

0's). The deletion of some items in the database, which is the central theme of this work, has also been discussed in [2,3,5,6,7,8].

3. PRIVACY PRESERVING ASSOCIATION RULES

3.1 Basic Notation and Definitions

Let $I = \{i_1, i_2, \ldots, i_n\}$, be a set of literals called items. Let D be a database of transactions, where each transaction T is an itemset such that $T \subseteq I$. A unique identifier, called *TID,* is associated with each transaction. A transaction T supports an itemset A, a set of items in I, if $A \subseteq T$. The support *sup(A)* of an itemset A is $|A| / |D|$ (where $|A|$ is the number of transactions that support A and $|D|$ is the number of the database transactions). An itemset A is called *large* if its support is greater than a predefined support threshold.

An association rule is an implication of the form $A \Rightarrow B$, where $A \subset I$, $B \subset I$ and $A \cap B = \emptyset$. The *support* of the rule $A \Rightarrow B$ is $sup(A \Rightarrow B) = |A \cup B| / |D|$ and the *confidence* of the rule $A \Rightarrow B$ is $conf(A \Rightarrow B) = |A \cup B| / |A|$. The *Minimum Support Threshold (MST)* and the *Minimum Confidence Threshold (MCT)* are defined in order to trim those rules with support and confidence below these thresholds correspondingly.

Given a set of literals $I = \{i_1, i_2, \ldots, i_n\}$, a transaction $T \subseteq I$ can also be represented as *a bitmap vector* $(t_1, t_2, \ldots, t_n)$, where $t_j = 1$ if and only if $i_j \in T$ otherwise $t_j = 0$. Using this representation for transactions and itemsets, we can easily compute whether a transaction T supports an itemset A $(A \subseteq T)$ by testing if $A \cap T = A$.

3.2 Sensitive Rules Terminology

For a database D a user defines the *MCT* and *MST* and then mines the database to find association rules. We call the set of these rules R. The user then will select to hide a subset $R_H \subseteq R$ that he considers sensitive to be public. By sensitive we mean that a certain rule in $R_H \subseteq R$ should not be made public, either because this is enforced by a privacy policy or because if it is disclosed we may provide our competitor with a business advantage. We do not define sensitivity formally in this paper but we associate sensitivity with the decrease of the support or the confidence of a rule R below the specified thresholds. In this study we accept that the user decides to decrease the confidence of a sensitive rule by a *Safety Margin* threshold *(SM)* below the MCT in order to hide the sensitive rule. The modified

database D_M after the hiding process does not include the sensitive set of rules R_H.

3.3 Data Loss Definitions

We define the Data Loss (side effects) which results after the hiding process by using the four statements below:

1. If a rule R has had *conf(R)>MCT* before the hiding process and after the hiding process has *conf(R)<MCT* then we say that this rule has been *lost* or *eliminated.*
2. If a rule R has had *conf(R)<MCT* before the hiding process and after the hiding process has *conf(R)>MCT* then we say that this rule *(ghost rule)* has been created.
3. If an Itemset *I* was large before the hiding process (*sup(I)>MST*) and after the hiding process its *sup(I)<MST* then we say that this Itemset has been eliminated.
4. If an Itemset *I* was not large before the hiding process (*sup(I)<MST*) and after the hiding process its *sup(I)>MST* then we say that this Itemset has been created.

4. DISTORTION-BASED ALGORITHMS FOR RULE HIDING

We will present two new algorithms, which use the distortion technique for hiding association rules, in sections 4.5 and 4.6. An algorithm for hiding rules in a database has to achieve the following main goals in order to be considered effective:

1. To minimize the changes in the values of the database.
2. To minimize the side effects (rules and itemsets lost or created) by selecting the items in the appropriate transactions to change.
3. To modify the database in a way that an adversary cannot recover the original values of the database.

4.1 Algorithms Notation

Let T be a set of transactions in the database, and let $|T|$ be the cardinality of set T. T_R is the set of transactions that support the rule R ($I_L \Rightarrow I_R$) with I_L the left itemset and I_R the right itemset of this rule. Let $T_{L'Rp}$ be the transactions that partially support I_R but do not support I_L. Furthermore, let T_{LpR} be the transactions that partially support I_L and fully support I_R. The number N_1 denotes in how many transactions 1's must be

replaced with 0's to decrease either the *sup(R)* or the *conf(R).* These transactions are called *victims.*

In addition, $/I_{Ti}/$ is the number of items of transaction T_i. Let *conf(R)* be the initial confidence of the rule *R.* For each rule, we assign a number $A_{MCT}(R)$ that denotes how many transactions this rule has above the *MCT.* This is computed easily as follows: $A_{MCT}(R)=(conf(R)-MCT)|I_L|$. For example if the *MCT=70%* and *conf(R)=85%* and $|I_L|=560$ transactions then $A_{MCT}(R)=84$ transactions. Moreover, R_{common} denotes the rules that have common items in their generating itemset with a rule R_i. $R_{eliminated}$ denotes the rules that will be lost after a deletion of an item in the database. Finally, let $|T_R|_{before}$ be the number of transactions that support *R* before the hiding process and let $|T_R|_{after}$ be this number after the hiding process.

4.2 Association Rule Hiding based on a Distortion Approach

The algorithms which use the distortion technique in order to achieve the hiding of a sensitive rule, they selectively alter some values of the database in a way that an adversary cannot infer the original state of the database. Every change that is done through the hiding process cannot be tracked by an adversary because the database is being changed permanently and this is a great advantage of this technique.

4.3 A Distortion Approach: Reversing 1's to 0's

Let us assume that we want to hide the rule $R\ (I_L \Rightarrow I_R)$ with I_L the left itemset of this rule with items $(I_{L1}, I_{L2},..., I_{Li})$ and I_R the right itemset of this rule with items $(I_{R1}, I_{R2},..., I_{Rj})$. The confidence of this rule is given by

$$conf(R) = \frac{\sup(I_L \cup I_R)}{\sup(I_L)}$$

and the support is given by $\sup(R) = \sup(I_L \cup I_R)$. We can either reduce the confidence below the MCT or the support below the MST. When we want to reduce the support we have to select some items of I_L or I_R and delete them from the transactions that support the sensitive rule. If we select those items from the I_R then both the support and the confidence will be decreased. Otherwise, if we select items from I_L to delete the confidence of the rule may not be decreased because both the numerator and the denominator are decreased. So it is more effective to delete items from the I_R since by doing that we reduce both the support and the confidence of the sensitive rule.

4.4 Important Data Structures

The distortion-based algorithms that we will introduce in this paper use extensively some data structures in order to find, save and access the rules in the database. In order to reduce time and space complexity in these operations we have implemented the following data structures:

- An *inverted index* between the items and the transactions in the database. For every item in the database there is a list of transactions that supports it. So, in order to find each large itemset we use the *mergesort* algorithm to sort the items that create the itemset. In that way we can find the support value of an itemset I in O(|I|) time without having each time to scan the database from the beginning.
- The rules are stored in an index and for each one of them we store the confidence value and the $A_{MCT}(R)$ value. So, if we want to restore the confidence of a rule we can do that in O(logn) time where n is the total number of rules in the database by applying a binary search in that index.

4.5 Priority-based Distortion Algorithm

4.5.1 Priority-based Distortion Algorithm Definition

The first algorithm we propose is the Priority-based Distortion Algorithm. This algorithm reduces the confidence of the rule we hide by reversing 1's to 0's in items in the right itemset of each sensitive rule. First, it finds all the transactions that support a rule *Ri,* and all the possible items in them that can be hidden in each step. Then it finds how many rules will be lost if a specific item will be reversed. After that it chooses to hide the item that will cause the least or no side effects to other rules. In order to compute these rules correctly, the algorithm holds in memory a hash table with all the rules with common items with the rule we hide. Further explanations are given in section 4.5.2.

4.5.2 Explanation of the steps of PDA

At step 1, the Priority-based Distortion Algorithm recovers the transactions set T_R.

At step 2, the algorithm scans this set of transactions to collect the rules which may be affected by the hiding process. These are the rules that have at least one common item with the items of the rule we currently hide. Each one of the rules that is collected has $A_{MCT}(R)$ transactions above *MCT*.

So, for example if one rule *R* has $A_{MCT}(R)=+1$ then by just inverting an item in a transaction that supports the rule R, this rule can be eliminated. In this way the algorithm tries to exclude the transactions that will immediately eliminate or create new rules.

Priority-based Distortion Algorithm

Input: Transactions **T∈ D,** Rules Set **RS,** Rules to hide $\mathbf{R_H}$, Threshold **SM,** Thresholds **MCT** and **MST.**

Output: Modified Database $\mathbf{D_M}$

For each $R_i \in R_H$
Do {
Step 1: Find T_R, for *Ri*.
Step 2: For each transaction in T_R find the rules $R_{common} \in RS$ and their $A_{MCT}(R)$, with at least one common item with I_R

Do until conf(Ri)<(MCT-SM)
{
Step 3: For each T∈ T_R find how many R_{common} will be eliminated if any item i∈ I_R will be reversed.
Step 4: Assign a Pi_T for each i∈ I_R in every $T \in T_R$ such as $Pi_T = |R_{eliminated}|$.
Step 5: Reverse the item with the lowest Pi_T in the transaction which it belongs.
Step 6: Update $conf(R_i)$, $A_{MCT}(R)$ for the rules that have been affected.
}
}

Figure 1. Pseudocode for Priority-based Distortion Algorithm

At step 3, for each possible switching of an 1 to 0 that can be done in $i \in I_R$ in all $T \in T_R$ the algorithm finds how many rules will be eliminated. This can be achieved by recovering the $A_{MCT}(R)$ value of each rule and checking if this is equal to +1. So, at step 4 for each item that it could be hidden in $T \in T_R$ the algorithm finds how many rules with $A_{MCT}(R)=1$ will be eliminated and assigns a Priority P_{IT} in each item that may be reversed.

At step 5, the Priority-based Distortion Algorithm simply reverses the item with the lowest P_{IT} and finally at step 6 it updates the $A_{MCT}(R)$ values for the remaining rules in R_{common}.

4.6 Weight-based Sorting Distortion Algorithm

We present here a detailed description of the second algorithm that we propose along with an extended discussion about its steps. The main difference between the Priority-based Distortion Algorithm and the Weight-based Sorting Distortion Algorithm is that the Weight-based Sorting Distortion Algorithm uses a sorting technique to sort the victim transactions, instead of scanning all of them each time that it hides a simple item in the database.

WEIGHT-BASED SORTING DISTORTION ALGORITHM
Input: Transactions **T∈ D,** Rules Set **RS,** Negative Border Rules Set **NBRS,** Rules to hide $\mathbf{R_H}$, Threshold **SM,** Thresholds **MCT** and **MST**, Proportion $\mathbf{A_{01}}$.
Output: Modified Database $\mathbf{D_M}$

For each $R_i \in R_H$
Do {
Step 1: Compute N_1.
Step 2: 2.1 Find T_R for *Ri*.
2.2 For each transaction in T_R find the rules $R_{common} \in RS$ with at least one common item with I_R.
2.3 Assign a weight *w* for each R_{common}.
2.4 Assign a P_T for each T such as $P_{Ti} = \sum_{i=0}^{|Rcommon|} w_i$
Step 3: Sort $T \in T_R$ starting from them with lowest P_{Ti}.
Step 4: For the first N_1 sorted $T \in T_R$ reverse an item $i \in I_R$
Step 5: Update $conf(R_i)$, $A_{MCT}(R)$, for all the other rules that have been affected.
}

Figure 2. Pseudocode for Weight-based Sorting Distortion Algorithm

4.6.1 Weight-based Sorting Distortion Algorithm Definition

The algorithm in Figure 2 concentrates on the optimization of the hiding techniques of the hiding process so as to achieve the least side effects and the minimum complexity. The Weight-based Sorting Distortion Algorithm finds the transactions which support the rule we hide, assigns them a priority and sorts them in ascending order according to the priority value that each transaction has. The algorithm assigns a weight to each rule in the database according to how close to the *MCT is* the confidence of each

rule. Then uses these weights to compute the priority value for each transaction according to how weak are the rules that a transaction supports.

4.6.2 Explanation of the Hiding Process in the WSDA

At step 1, the N_I is calculated. The algorithm must make N_I transactions no longer support the R_i. So if

$$conf(R_i) = \frac{|T_{Ri}|}{|T_{IL}|} \Rightarrow \frac{|T_{Ri}| - N_1}{|T_{IL}|} \leq (MCT - SM) \quad (1)$$

Solving equation (1) for N_I we have:

$$N_1 = \lceil |T_{Ri}| - (MCT - SM) * |T_{IL}| \rceil$$

At step 2.1, the Weight-based Sorting Distortion Algorithm recovers the transactions that support the rule R_i which are denoted as T_R. More specifically, at step 2.2 the algorithm scans this set of transactions to collect the rules which may be affected by the hiding process. Each one of the rules that is collected has $A_{MCT}(R)$ transactions above *MCT*. Then, at step 2.3 the algorithm assigns a w_i for each rule in R_{common}.

Each w_i is assigned as follows:

$$w_i = \frac{A_{MCT}(R)_{\max}}{A_{MCT}(R)_{\min} A_{MCT}(R)_i}$$

where $A_{MCT}(R)_{\max}$ the maximum $A_{MCT}(R)$ value between R_{common} rules and $A_{MCT}(R)_{\min}$ the minimum.

The closer the confidence of a rule is above the *MCT* (or the number of transactions $A_{MCT}(R)$ above *MCT*) the more possible is for that rule to be eliminated if some items are turned to 0 in the victim transactions, so this rule gets a higher weight.

At step 2.4, the algorithm computes the priority value for each transaction. The reason for this assignment is to give lower P_T to transactions that support fewer and stronger rules than others, in order to implicitly minimize the rules that will be eliminated.

At step 3, the algorithm sorts the transactions in ascending order according the *Pi* value of each transaction. In that way the transactions with the lowest P_T will come on top. For the sorting process it can be used an already known algorithm such as *quicksort,* or *bubblesort.*

At step 4, N_I items of I_L in $T \in T_R$ are reversed in order to reduce the $conf(R_i)$ as we explained in 4.3.

At step 5, the algorithm updates the data structures with the confidence of each rule and removes the rules that have been eliminated from the transactions that were supporting them. After that, the algorithm returns to step 1 to hide the next rule.

A thorough comparison between the efficiency of the above two algorithms is presented in Section 5 of the paper.

5. EFFECTIVENESS OF PDA AND WSDA ALGORITHMS

We will compare the two algorithms by using both a complexity analysis and an experimental framework in section 5.1 and section 5.2 correspondingly.

5.1 Complexity Analysis

The Priority-based Distortion Algorithm first finds T_{Ri}. This can be done in $O(|D|)$ time. At step 2 the algorithm has to find R_{common}. Since for every transaction we maintain a list of rules represented by their numbers, we can find in $O(log(|R|))$ each rule by its number and in O(1) the items and the $A_{MCT}(R)$ for each rule. So all the common rules for all the transactions that support R_i can be found in $O(|T_{Ri}|*|R|*log(|R|))$. At step 3, for each item that is candidate to be hidden we count how many rules which include this item in each transaction have $A_{MCT}(R) = 1$. So the total time for that is $O(|I_{Ri}|*|T_{Ri}|*|R|)$ where $|I_{Ri}|$ is the number of items in the right itemset of the rule R_i that is currently being hidden. Furthermore, step 4 is a single assignment and can be done in O(1). At step 5 there is a list with $|I_{Ri}|*|T_{Ri}|$ numbers and with a serial scanning the algorithm finds the lowest of them in $|I_{Ri}|*|T_{Ri}|$ time. Finally, in step 6 some updates are made in $O(|R|)$ time.

The Weight-based Sorting Distortion Algorithm tries to reduce the side effects without having to spend significant processing time to hide the rules. At step 1 it does some quick computations in $O(1)$ time. At step 2, and more specifically at step 2.1 the algorithm has to find T_{Ri}. This can be done in $O(|D|)$ time. But if we use the inverted index with items and transactions, an algorithm like *mergesort* could done it in $O(|i_1|+|i_2|+\ldots|i_n|)$ where i_k the items of the rule R_i. In step 2.2 the algorithm has to find R_{common}. Since for every transaction we maintain a list of rules, represented by their numbers, we can find in O(log(|Rl)) each rule by its number and in O(1) the items and the $A_{MCT}(R)$ for each rule. After that Weight-based Sorting Distortion Algorithm compares the rules found (R_F) with I_R to find common items in $O(|I_R|*|R_F|)$. In step 2.3 the algorithm has to assign weights in the set of

rules that have been found. First, the $A_{MCT}(R)_{min}$, $A_{MCT}(R)_{max}$ are found in $O(2*|R_F|)$. Then in $O(|R_F|)$ the weights w are computed. In addition in step \2.4 all transactions get a Priority value in $O(|T_R|* |R|)$ where |*R*| the maximum number of rules that a transaction supports. At step 3 sorting the sets T_R needs $O(|T_R|log|T_R|)$. In addition at step 4 reversing N_1 items is done in $O(N_1)$. Finally step 5, needs *O(/R/)* where |R| the number of rules in the database.

5.2 Experimental results

We have performed extensive experiments in order to compare the effectiveness of the algorithms presented in above. We run these algorithms in Linux operating system at a Pentium 4 at 2,4 GHz with 512MB RAM. We used a synthetic database with 5k transactions that we generated with *IBM Synthetic Data Generator.* The database had 50 items and the average number of items per transaction was 13. We converted the results into a binary database appropriate for our implementation. In order to decrease the execution time we created an inverted index with the items and the transactions that support each item. In addition, during the discovery of the rules in the database we created an index with the association rules and the items that each rule contains. In the 5k database we chose to hide 5 rules out of 299. The MST was 9% and the MCT was 65%. The following rules were chosen randomly for hiding:

5k Database	
Rules	*Confidence*
1 ⇒ 33	80.6%
34 ⇒ 3	68.9%
1 33 ⇒ 48	100%
6 22 ⇒ 11	78.1%
7 41 ⇒ 3	98%

We will compare the two presented algorithms using some metrics of their effectiveness. We also use in the comparison, an already published algorithm in [5] for rule hiding that we also implemented. This algorithm is called 1.b in [5].

5.2.1 Rules lost or created in the database and Large Itemsets that remained unaffected

At Figure 3 we present the side effects for different *safety margin* values (10%, 20%, 40%, 60%). As we see, PDA performs slightly better in

eliminating or creating new rules than WSDA. Both of the algorithms that we propose perform better than algorithm 1.b in reducing the side effects in other rules in the database. We can see clearly the trade-off between Privacy and Data Loss, because if we raise the *safety margin* then more side effects will be created. At Figure 4 we see how many Large Itemsets have been remained after the hiding process. The results show a similar performance for all algorithms that we experimented with.

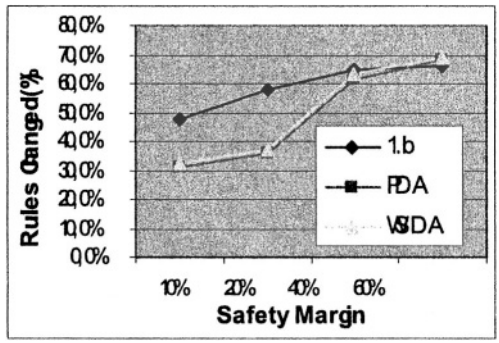

Figure 3. Rules lost or created after the hiding process.

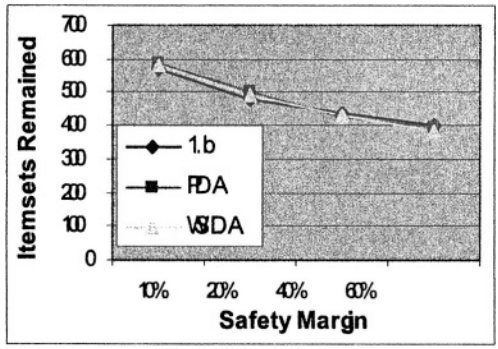

Figure 4. Large Itemsets remained after the hiding process.

5.2.2 Time experiments

We compared the execution time of PDA, WSDA and 1.b algorithm in 8 different databases. The average number of items in each transaction in the first 4 databases was 13 and in the other 4 databases were 20. The

smaller database had 2.5k transactions and the largest had 10k transactions. WSDA was faster than PDA and this is because WSDA sorts the transactions at step 3 and then consecutively hides the items in those transactions, and on the other hand PDA scans all the set of victim transactions each time it hides an item. 1.b was faster than the algorithms we propose because it is simpler to implement. Namely, WSDA needs more time to execute than 1.b but both 1.b and WSDA run in linear time according to the input. In contrast, PDA seems to consume much more time as the input transactions or *safety margin* increase.

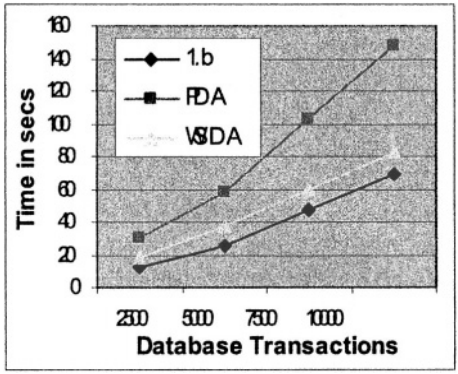

Figure 5. Running time for each algorithm for average 13 items in each transaction

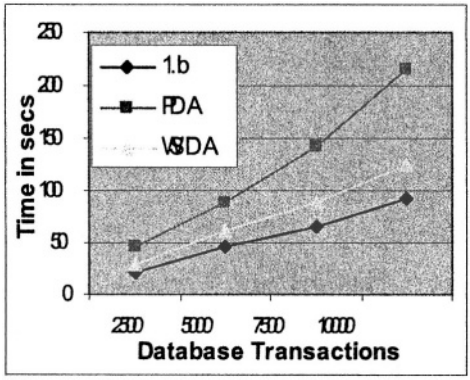

Figure 6. Running time for each algorithm for average 20 items in each transaction

6. CONCLUSIONS AND FUTURE WORK

PDA algorithm designed to concentrate on the minimization of the side effects of the hiding process though it provided the least side-effects in the database it was proved that this algorithm needs much time to be completed. WSDA performed similarly in minimizing the side effects, running in less time than PDA. This makes the WSDA the best algorithm in overall related to the other two algorithms.

In the future we plan to test the above techniques in real datasets that differ in the dependency of their itemsets. In addition, we plan to test techniques that reverse 0's to 1's in the database instead of reversing only 1's to 0's. At last, we plan to construct new algorithms that will concentrate on the minimization of the data loss by sorting the victim transactions according to a specific criterion of data loss each time.

References

[1] Yucel Saygin, Vassilios Verykios, and Chris Clifton, *Using Unknowns to Prevent Discovery of Association Rules,* SIGMOD Record **30** (2001), no. 4, 45–54.

[2] Stanley R. M. Oliveira and Osmar R. Zaiane, *Privacy Preserving frequent Itemset Mining,* In Proceedings of the IEEE ICDM Workshop on Privacy, Security and Data Mining (2002), 43–54.

[3] Vassilios S. Verykios, Ahmed K. Elmagarmid, Bertino Elisa, Yucel Saygin, and Dasseni Elena, *Association Rule Hiding,* IEEE Transactions on Knowledge and Data Engineering (2003).

[4] Mike J. Atallah, Elisa Bertino, Ahmed K. Elmagarmid, Mohamed Ibrahim, and Vassilios S. Verykios, *Disclosure Limitation of Sensitive Rules,* In Proceedings of the IEEE Knowledge and Data Engineering Workshop (1999), 45–52.

[5] Elena Dasseni, Vassilios S. Verykios, Ahmed K. Elmagarmid, and Elisa Bertino, *Hiding Association Rules by using Confidence and Support,* In Proceedings of the 4th Information Hiding Workshop (2001), 369–383.

[6] Yucel Saygin, Vassilios S. Verykios, and Ahmed K. Elmagarmid, *Privacy Preserving Association Rule Mining,* In Proceedings of the 12th International Workshop on Research Issues in Data Engineering (2002), 151–158.

[7] Johnsten, T., Raghavan, V., Hill K., 2002. *The Security Assessment of Association Mining Algorithms* in "Proceedings of the Sixteenth Annual IFIP WG 11.3 Working Conference on Database Applications Security.

[8] Vassilios S. Verykios, Elisa Bertino, Igor Nai Fovino, Loredana Parasiliti Provenza, Yucel Saygin, Yannis Theodoridis, *State of the Art in Privacy Preserving Data Mining.* To appear in SIGMOD Record 2003.

[9] Alexandre Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, Johannes Gehrke. *Privacy Preserving Mining of Association Rules.* SIGKDD 2002, Edmonton, Alberta Canada.

[10] Murat Kantarcioglou and Chris Clifton, *Privacy Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data,* In Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (2002), 24–31.

[11] Jaideep Vaidya and Chris Clifton, *Privacy Preserving Association Rule Mining in Vertically Partitioned Data,* In the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2002), 639–644.

[12] R. Agrawal and R. Srikant. *Privacy-Preserving Data Mining.* In Proceedings of the 2000 ACM SIGMOD Conference on Management of Data, Dallas, TX, May 14-19 2000. ACM.

[13] C. Clifton and D. Marks. *Security and Privacy Implications of Data Mining.* In ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, pages 15-19, May 1996.

[14] Y. Lindell and B. Pinkas. *Privacy Preserving Data Mining.* Lecture Notes in Computer Science, 2000.

[15] LiWu Chang and Ira S. Moskowitz, *Parsimonious Downgrading and Decision Trees Applied to the Inference Problem,* In Proceedings of the 1998 New Security Paradigms Workshop (1998), 82–89.

---

* This work has been supported by European Commission under the PANDA and CODM1NE IST/FET Projects.