Available online at www.sciencedirect.com

ScienceDirect

ELSEVIER

Data & Knowledge Engineering 65 (2008) 75–89

DATA & KNOWLEDGE ENGINEERING

www.elsevier.com/locate/datak

A MaxMin approach for hiding frequent itemsets

George V. Moustakides [a], Vassilios S. Verykios [b,*]

[a] Department of Electrical and Computer Engineering, University of Patras, 26500 Rio, Greece
[b] Department of Computer and Communication Engineering, University of Thessaly, 38221 Volos, Greece

Received 5 June 2007; accepted 5 June 2007
Available online 18 July 2007

Abstract

In this paper, we are proposing a new algorithmic approach for sanitizing raw data from sensitive knowledge in the context of mining of association rules. The new approach (a) relies on the maxmin criterion which is a method in decision theory for maximizing the minimum gain and (b) builds upon the border theory of frequent itemsets. Experimental results indicate the effectiveness of the proposed methodology both with respect to the hiding results as well as with respect to the time performance compared to similar state of the art approaches.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Privacy preserving data mining; Knowledge hiding; Frequent itemset hiding; Association rule hiding

1. Introduction

Data mining is a rather new research area focusing on the development of algorithms for the intelligent analysis of data. This analysis goes beyond the testing of statistical hypotheses, all the way to the sophisticated extraction of knowledge that was previously unknown. Although the majority of the results created by data mining is, so far, positive, there are recently some major concerns regarding privacy breaches that incur in the data. These breaches into the privacy are the side effects of being able to discover hidden nuggets of information, and possibly use them against the privacy of information. For example, it is not uncommon even after having de-identified the data (which means that we have removed all the sensitive information like social security number, home addresses, mobile phone numbers, etc.) for a person to be re-identified by making use of certain patterns that closely characterize the individuals.

In response to the people's concerns about the processing that their personal information is undergone when this is collected in the data warehouses of different service providers and public organizations, legislation has been proposed. This legislation come into play for safeguarding the people's rights. More and more as we move forward into the information society, legal bodies amend the laws on the protection of data privacy in order to guarantee the confidentiality of data against new ways of data processing and file interconnection.

* Corresponding author.
E-mail address: verykios@inf.uth.gr (V.S. Verykios).

0169-023X/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2007.06.012

The specific aspect of data privacy that is investigated in this paper has to do with knowledge privacy when this knowledge is produced from the mining of the data. In particular, in this work we are dealing with knowledge in the form of frequent patterns and rules that are generated from association rule mining techniques. The task we are called for solving, known by the name *association rule hiding*, is to remove certain patterns that we consider sensitive from a database, by changing the data in such a way that everything but the sensitive knowledge remains intact. To elaborate on this idea, think of a scenario where customer sales data are collected from a big chain store and afterwards are distributed to various product providers. Even though the chain store has as its goal to help the providers in distributing their deliveries in the most effective way, a provider may use correlation among the sales of various products in order to stay ahead of the competition. For this reason, the chain store needs to remove certain information (correlations for example) from the data so that it will not be possible for a provider to engage in this illegitimate competition.

The solution proposed here builds upon the idea that we can minimize the impact of the changes in the data, that we apply in order to hide the sensitive knowledge, by considering only to minimize the impact on the positive border of the frequent patterns. As long as we succeed in maximizing the minimum gain (which is to keep the positive border itemsets above the frequency threshold), all the non-sensitive frequent patterns which are not in the positive border remain above the support threshold which means that they are preserved.

The rest of this paper is organized as follows. Section 2 presents related work which has been proposed for the association rule hiding problem. In Section 3, we state the specific problem, as well as the necessary background information for completeness purposes, and in Section 4, we expose our proposed maxmin technique for the association rule hiding problem. In Section 5, we present two algorithms which rely on the maxmin criterion along with an example which is outlined in Section 6 that sheds light on the workings of the MaxMin algorithms, and in Section 7, we present a thorough evaluation by indicating how the proposed algorithms behave in comparison with other similar algorithms proposed elsewhere. Finally, in Section 8, we conclude our study and we refer to some research threads for future consideration.

2. Related work

The problem of hiding sensitive knowledge either in the form of frequent patterns or in the form of sensitive rules appeared for the first time in an early paper by Atallah et al. [2]. The authors in this first work had investigated the problem of hiding sensitive frequent patterns (a) by providing a proof that an optimal solution to this problem is NP-hard and (b) by proposing a heuristic greedy approach that traverses the frequent itemset lattice for pinpointing the transactions and the items that they had to change, so that the support of a sensitive frequent pattern reduces and falls below the support threshold. Dasseni et al. [5] generalized the problem in the sense that they considered the hiding of both sensitive frequent itemsets and sensitive rules. The algorithms initially proposed in [5] were later on improved and evaluated for their performance under different sizes of input data sets and different sets of sensitive rules in [20] and it was seen to exhibit very good behavior both in their complexity and their effectiveness.

Saygin et al. [18,17] consider the problem of hiding frequent patterns and rules by using unknowns. Their idea was motivated by real life constraints with respect to the modification of the original database, where sometimes to tell a lie (turn a 0 to 1 or the opposite) is very different from expressing your ignorance (the value is not known). This work presents a fuzzification of the support and the confidence metrics, and considers both 0 and 1 values to use for hiding, in some proportion, so that it is not made easy for the data intruder to conclude upon the value hidden behind an unknown value.

An in depth experimentation and evaluation of distortion (turning zeroes to ones and the opposite) and blocking (using unknowns for hiding) techniques have been performed by Pontikakis et al. [15,16]. Bertino et al. [4] propose a complete evaluation framework for measuring the performance of association rule hiding techniques that can be utilized for comparing different association rule hiding algorithms. Zaiane et al. [13] present a new formalization of the association rule hiding problem by trying to remove/hide the inference channels created by rules that exist in a released rule base. A number of papers focusing on improving the effectiveness and complexity of the aforementioned algorithms have also appeared in literature [14,10–12,9, 8,13].

3. Problem formulation

The problem we are dealing with in this paper is to protect the implied knowledge that is hidden in a database by slightly modifying the data, in such a way that (a) the sensitive knowledge is obscured and (b) the non-sensitive knowledge as well as the raw data remain to a maximum degree intact. We consider this problem in the context of frequent itemset mining of the association rule discovery framework. In a database D, a frequent itemset is a set of items A, chosen from the universe of possible items $I = \{i_1, i_2, i_3, \ldots, i_n\}$, that appears frequently enough in the database. In other words, a frequent itemset is one that its frequency of occurrence in the database is above a minimum frequency threshold specified by the user. The frequency of occurrence in this context is known as *support* and it is denoted by *supp*. By specifying a support threshold σ, in the first phase of an association rule discovery algorithm, all frequent itemsets are discovered in an increasing order of cardinality (number of items). Let us call the set of all frequent itemsets as F.

Given a privacy policy, as exemplified by a directive released by a data protection authority, we can assume that part of these frequent itemsets might be sensitive, or else it should not be disclosed to the public or to some competitor. Let us call the set of frequent itemsets which are sensitive as S. What we should do at this step is to minimally change or perturb our database in such a way that the sensitive frequent itemsets become infrequent. Apparently, we are assuming a privacy policy which is completely related to the fact that an itemset is frequent. We argue that such a policy is legitimate since the frequency of an itemset makes it interesting from a decision maker point of view. By turning a certain frequent itemset to infrequent, we are automatically making it uninteresting for further consideration, which implies that this itemset will escape the scrutiny or attention of the data terrorists.

The formalism of the border presented in [7] is important in our problem formulation and for completeness we include it here. Let P be a set of patterns, and $\prec$ a partial order on P. Further, let F be closed downwards under the relation $\prec$. The *border* $Bd(F)$ of F consists of those patterns ϕ such that all more general patterns than ϕ are in F and no patterns more specific than ϕ is in F: $Bd(F) = \{\phi \in P \mid \text{for all } \gamma \in P \text{ such that } \gamma \prec \phi \text{ we have } \gamma \in F, \text{ and for all } \theta \in P \text{ such that } \phi \prec \theta \text{ we have } \theta \notin F\}$. Those patterns ϕ in $Bd(F)$ that are in F are called the *positive border* $Bd^+(F) = \{\phi \in F \mid \text{for all } \theta \in P \text{ such that } \phi \prec \theta \text{ we have } \theta \notin F\}$, and those patterns ϕ in $Bd(F)$ that are not in F are the *negative border* $Bd^-(F) = \{\phi \in P \backslash F \mid \text{for all } \gamma \in P \text{ such that } \gamma \prec \phi \text{ we have } \gamma \in F\}$. In other words, the positive border consists of the most specific patterns in F, the negative border consists of the most general patterns outside F, and the border is the union of these two sets.

Let us also consider a subset S of F which includes the sensitive rules in F, that needs to become safe based on some privacy policy, or in other words to lose so much of its frequency as it is required to just become infrequent. Such a requirement first of all require S to move to $P \backslash F$ and then it will necessitate a revision of both the positive and negative border. The same idea was proposed initially by Sun and Yu [19] who claimed that in order to formalize the work performed by a hiding algorithm so that it achieves the minimum impact on the database, there is a need to control the impact of the hiding algorithm on the expected positive border, which is the positive border after this has been shaped up with the removal of the sensitive itemsets. By following a similar approach, we need to prune the frequent itemset lattice, starting from the sensitive itemsets and moving upwards from there till we reach the original positive border. The removal of the sensitive itemsets from the old frequent itemset lattice along with their super-itemsets (supersets of itemsets) creates a new (revised) positive border and a new negative border. The nature of the proposed MaxMin algorithms is that they try to remove from the database all the sensitive itemsets that belong to the new (revised) negative border, while they try to keep the supports of the frequent itemsets in the new (revised) positive border above the support threshold. More information about the revision theory of the border can be found in Section 4.1.

4. A MaxMin approach for hiding sensitive itemsets

The MaxMin approach that we propose, relies on the fact that we succeed in the hiding of the sensitive itemsets while at the same time we minimize the impact of the hiding process to the non-sensitive information. We achieve this by considering only the effects of the hiding process to the itemsets on the positive border, after the border has been revised by taking into consideration the sensitive large itemsets. In the following subsections, we present the necessary theory which underlies the revision of the border, and a number of theorems that prove the minimum impact of the changes in the database imposed by the MaxMin approach. In Section 5, we present two algorithms that rely on these theorems and demonstrate the minimum impact behavior.

4.1. Border revision theory

We follow here the approach first proposed in [19]. We should emphasize that the border revision technique that we implemented is not documented in [19], apart from an example that indicates the workings of the border revision approach. Also, since the software implementing the algorithms in [19] is proprietary we had to implement the border revision technique by ourselves. Given a set of sensitive itemsets for hiding, it follows that the border of the frequent itemsets needs to be modified. The idea is that since some of the old (before hiding) frequent itemsets will eventually become hidden or else their support gets lowered below the support threshold, the border needs to be modified in order to accommodate the transition of the sensitive itemsets from frequent to infrequent. The revision of the border has to do with keeping track of what frequent itemsets need to be protected from hiding (must not lose support) and it is a process that takes place only in the data structure holding the frequent itemset lattice and not in the actual database.

The algorithm that revises the positive border, starts from the minimum level of the frequent itemset lattice corresponding to the shorter (in number of items) sensitive itemsets. It then proceeds upwards to the itemset lattice by removing from the lattice, all the sensitive itemsets along with their super itemsets, until it reaches the upper limit of the lattice, where it stops. After that, it moves down the new frequent lattice (produced in the previous phase) and checks whether a frequent itemset has a cover. An itemset cover is a super itemset of this itemset that is also frequent. If a frequent itemset in the revised frequent lattice does not have a cover, then it belongs to the new positive border. The algorithm stops, when it reaches the minimum level of the revised frequent itemset lattice.

In a similar manner, the algorithm that revises the negative border, starts from the bottom of the revised frequent itemset lattice, which has been produced after the revision of the positive border. It then, in each level of the lattice, produces joins of frequent itemsets from this level similarly with the joins of algorithms made by Apriori algorithm. For each joined itemset created, checks whether all its sub itemsets belong to the revised itemset lattice. If this is the case, it checks whether the joined itemset belongs to the revised itemset lattice by itself. If it belongs, then it continues with generating new joined itemsets further. If the joined itemset does not belong to the revised itemset lattice while all of its sub itemsets belong to this, then it places the joined itemset to the revised negative border. This process ends when the algorithm arrives at the upper limit of the revised itemset lattice.

After the formation of the revised positive and negative border, the algorithms proposed in this paper can take over the hiding of the sensitive itemsets by actually modifying the database. Both of the algorithms accept as input, the revised positive border, as well as the intersection of the revised negative border with the list of sensitive itemsets. It is only the intersection that needs to be tested, because (a) the itemsets that belong to the revised negative border but not in the intersection used to be in the original negative border (it was hidden before) and (b) the itemsets that belong to the sensitive itemsets but do not belong in the revised negative border, have a sub itemset that belongs to the revised negative border.

4.2. Hiding of a sensitive itemset

The main idea behind the MaxMin optimization criterion as this is applied to the hiding of the sensitive itemsets is to reduce the support of the sensitive itemsets while at the same time it tries to maintain the support of a non-sensitive itemset intact, if possible. Below we introduce the necessary terminology and the accompanying theorems before we describe the basic algorithms.

Let us assume that a sensitive itemset *abd* needs to be hidden. For every item that belongs to a sensitive itemset we list the set of itemsets in the revised positive border that depend on it. We assume that the revised positive border is the set $\{ab, bd, acd, cde\}$ and we create a data structure which we call *affinity list* that maintains the lists of revised positive border itemsets that depend on every item in a sensitive itemset. The use of the affinity lists in the course of the MaxMin algorithms will be demonstrated in Section 6.

a	$ab/4, acd/4$
b	$ab/4, bd/5$
d	$bd/5, acd/4, cde/3$

We call an item (for example a) that belongs to a sensitive itemset a *tentative victim item*, and a large itemset (for example *acd*) that belong to the revised positive border and is affected by a tentative victim item, as a *tentative victim itemset*. Note that in the following discussion, both the positive and the negative borders are the revised (new) ones. We also call the set of tentative victim itemsets that depend on the same tentative victim item *vi* as the *vi*-list. For example the a-list for the tentative victim item a of sensitive itemset *abd* is the set $\{ab, acd\}$. In every *vi*-list, we select the itemset(s) with the minimum support which we call *minimum support itemsets*. Itemset *cde* is the minimum support itemset in the d-list of the sensitive itemset *abd*. A minimum support itemset is the most sensitive one among the itemsets in each *vi*-list since it is the closest to the borderline between the positive and the negative border.

From among all the minimum support positive border itemsets ($\{ab, acd\}$ for a-list, $\{ab\}$ for b-list, and $\{cde\}$ for the d-list) we select the itemset(s) with the highest (maximum) support. We call such an itemset the *maxmin* itemset – there might be a set of maxmin itemsets, and in this case we call this set, the maxmin set – since this is the only itemset among the different minimum support itemsets which is the maximum distance away from the border. In the affinity list of sensitive itemset *abd* there are two maxmin itemsets $\{ab, acd\} \cup \{ab\} = \{ab, acd\}$. The maxmin itemset determines the tentative victim item through which the hiding of the sensitive itemset will take place. We call such an item a *victim* item. For example the maxmin set $\{ab, acd\}$ indicates that either a or b should be the victim item, since the itemsets in the maxmin set belong to the *vi*-lists of items a and b. The proposed algorithms modify the victim item indicated by the maxmin itemset in such a way that the value of the support of the maxmin itemset, if possible, not to be modified. The reduced complexity of the proposed MaxMin algorithms compared to other similar approaches, emanates from the fact that it is the selection of one and only itemset that determines the victim item, through which a sensitive itemset will lose support. This selection is taking place in a very small set of itemsets which are both affected by the sensitive itemset and belong to the revised positive border.

In the following discussion, we present a number of theorems which are cornerstone in the functionality of the proposed MaxMin algorithms. The first theorem is concerned with two tentative victim itemsets that achieve the minimum support value for their corresponding *vi*-lists. The theorem indicates that if the support values for two minimum support itemsets are different, then by modifying the tentative victim item that corresponds to the larger minimum support itemset, the support of the smaller minimum support itemset remains unaffected. This is true independently of whether the larger minimum support itemset is affected or not by the modification. We state the theorem more formally next.

Theorem 1. *If two minimum support itemsets S_A and S_B contain the tentative victim items A and B, respectively, and have supports* $\mathrm{supp}(S_A) > \mathrm{supp}(S_B)$, *then the minimum support itemset S_B does not contain A.*

Proof. Let us assume that S_A and S_B are two minimum support itemsets corresponding to the tentative victim items A and B respectively. We also assume that $\mathrm{supp}(S_A) > \mathrm{supp}(S_B)$ and that S_B contains the tentative victim item A. Since S_B contains the tentative victim item A, it should also appear in the list of tentative victim itemsets of tentative victim item A along with S_A. Because $\mathrm{supp}(S_A) > \mathrm{supp}(S_B)$, it means that the S_B is the minimum support itemset for the tentative victim item A. But we claimed at the beginning that the minimum support itemset for A is S_A. For this reason, we have proved that S_B cannot contain sensitive item A. $\square$

The direct consequence of Theorem 1 is that if we modify the victim item A, and this has as a side effect that the maxmin decreases by one (in the worst case), no other minimum support value will be affected by this change.

The second theorem and its accompanying lemma apply to the case when two sets of minimum support itemsets (each is a subset of its corresponding *vi*-list) corresponding to two different tentative victim items happen to have the same support. Below we state formally a theorem that ensures that if the victim item corresponding to one set of minimum support itemsets can change without affecting the support of its minimum support itemsets, then the other set of minimum support itemsets will not be affected either.

Theorem 2. *Let $L_A = \{A_1, A_2, \ldots, A_K\}$ and $L_B = \{B_1, B_2, \ldots, B_M\}$, the vi-lists for two tentative victim items A and B correspondingly which are contained in sensitive itemset S. Let also $L_{S_A} = \{A_{i_1}, A_{i_2}, \ldots, A_{i_k}\}$ and $L_{S_B} = \{B_{j_1}, B_{j_2}, \ldots, B_{j_m}\}$, the sets of minimum support itemsets that correspond to the tentative victim items, such that $s_A = \mathrm{supp}(A_{i_1}) = \mathrm{supp}(A_{i_2}) = \cdots = \mathrm{supp}(A_{i_k}) < \mathrm{supp}(A_i)$ where $i \notin \{i_1, i_2, \ldots, i_k\}$ and $s_B = \mathrm{supp}(B_{j_1}) = \mathrm{supp}(B_{j_2}) = \cdots = \mathrm{supp}(B_{j_m}) < \mathrm{supp}(B_j)$ where $j \notin \{j_1, j_2, \ldots, j_m\}$. If $s_A = s_B$ and the support of S is decreased through A without the border itemsets in L_{S_A} to be affected, then no itemset in L_{S_B} will be affected either.*

Proof. If an itemset in L_{S_B} is affected by the decrease of the sensitive itemset through A, then this means that the affected itemset contains A. If it contains A, it should also belong to L_{S_A}. But the theorem claims that no itemset from L_{S_A} is affected from the change in A. Because of that, no itemset in L_{S_B} will be affected either. □

The lemma below applies to the cases in complement to those covered by Theorem 2 that the change in a victim item causes a minimum support itemset from its corresponding *vi*-list to lose support. If it so happens that the minimum support itemset which loses support is the only one affected in the *vi*-list by the change in the victim item, and at the same time it is not included in the other set of minimum support itemsets, then no minimum support itemset from the second set will be affected. The statement and its proof follows more formally below.

Lemma 1. *Let $s_A = s_B$ and the support of S is decreased through A. Let also $A_{i_1}, A_{i_2}, \ldots, A_{i_r}$ be a set of tentative victim itemsets in L_{S_A} that are both different from all the tentative victim itemsets in L_{S_B} and are also affected by the decrease in the support of S through A, while the rest of the border itemsets in L_{S_A} are not affected. If this holds, then no tentative victim itemset in L_{S_B} is affected from the decrease of S through A.*

Proof. Since $A_{i_1}, A_{i_2}, \ldots, A_{i_r} \in L_{S_A}$ are both the only border itemsets in L_{S_A} which are affected by the change in S through A, and they are different from all the border itemsets in L_{S_B}, no border itemset in L_{S_B} will be affected. If a border itemset $B_{i_s} \in L_{S_B}$ is being affected by a change in A, then this border itemset should contain A and also belong to L_{S_A}. Therefore we would have an additional itemset besides $A_{i_1}, A_{i_2}, \ldots, A_{i_r} \in L_{S_A}$ affected by the change, which leads to a contradiction. □

4.3. Hiding of sets of sensitive itemsets

We turn now our discussion to the methodology which will be followed by the MaxMin algorithms in order to hide a set of sensitive itemsets. In the discussion above, we presented the foundations underlying the hiding of a single sensitive itemset.

The assumption that it is made in the frequent set hiding problem, is that the privacy administrator is presented with a set of sensitive itemsets for hiding. The heuristic which is used by the MaxMin algorithms for the hiding of a set of sensitive itemsets is that the algorithm performs a sorting of the sensitive itemsets based on their support in an increasing order of support, and starts the hiding process from the sensitive itemset with the minimum support. After the algorithm finishes the hiding of the minimum support itemset, it continues with the next itemset in the order indicated until it is done with the hiding of every sensitive itemset.

By adopting a heuristic like that we enforce the requirement of the minimum impact adopted by the proposed MaxMin algorithms, since the sensitive itemset with the minimum support is the one that is closer to the border, and it is hidden in the smaller number of iterations.

We can also easily prove that the support of the minimum support itemset remains constantly smaller than the support of all the other sensitive itemsets that remain to be hidden. This has as a result that the complete hiding of a certain sensitive itemset can be considered in isolation from the hiding of all the other sensitive itemsets. The following theorem makes the above idea concrete.

Theorem 3. *Let $S_1, S_2, \ldots, S_n$ be the sensitive itemsets which are sorted in increasing order of their supports. The support of the minimum support sensitive itemset S_1 will be constantly smaller than the supports of all the other sensitive itemsets $S_2, S_3, \ldots, S_n$ during the hiding of this itemset.*

Proof. In every iteration of a MaxMin algorithm, the sensitive itemset in the foreground (the itemset selected each time by the algorithm for hiding) loses one point from its support. For any other sensitive itemset, the algorithm may reduce the support of this itemset by at most one. For this reason, the difference in the support of the itemset selected for hiding with all the rest, after the application of the MaxMin algorithm, will be at least as great as it was at the beginning of the hiding process. □

Because of the fact that different sensitive itemsets (except the one on which the algorithm is applied each time) may lose different amounts of support, after the hiding of each sensitive itemset, the remaining (not hidden yet) itemsets need to be sorted again in the increasing order of their supports.

A MaxMin algorithm needs to decide also which sensitive itemset to select next for hiding in case there is a tie in the supports of different itemsets. In this case, the algorithm considers the longer sensitive itemset first because this itemset offers to the algorithm the higher degree of freedom with respect to the selection of the victim items.

5. MaxMin algorithms

In the following discussion, we present two MaxMin algorithms that hide sensitive itemsets in an increasing order of sophistication.

5.1. Algorithm MaxMin 1

The first algorithm MaxMin 1 attempts to hide a sensitive itemset by selecting the victim item in such a way that the side effects to the minimum support itemsets in the positive border are minimized. To achieve this, it tries to hide the victim item from a transaction which supports the corresponding sensitive itemset, so as the support of the minimum support itemsets except possibly the support of the maxmin itemsets remains unchanged. MaxMin 1 algorithm assures that if the value of the maxmin is unique, then by modifying the maxmin itemset, no other minimum border itemset is modified. This property holds because of Theorem 1.

In case the maxmin value is attained by more than one tentative victim itemsets where each one corresponds to different tentative victim items, MaxMin 1 randomly selects the victim item to use for hiding. It is also indifferent to the algorithm the number of different border itemsets that attain the maxmin value, because all of them indicate the same victim item.

Algorithm 1: The MaxMin 1 Algorithm

```
Input: A database $D$, the positive border, the support threshold $\sigma$ and the
       set $S$ of sensitive itemsets.
Output: A database $D'$, that protects the privacy of rules.

Method:
while $S \neq \emptyset$ do
    $s = argmin_{a \in S}[supp(a)]$;
    while $supp(s) \geq \sigma$ do
        $M = \{b | b \in a\text{-}list$ and $a \in s\}$;
        $maxmin = \{c | argmin_{c \in M}[supp(c)]\}$;
        $b$ = victim item corresponding to an itemset in $maxmin$;
        set $b = 0$ in an $l \in L_S$;
    $S = S - s$;
```

The pseudocode of the proposed MaxMin 1 algorithm is shown in Algorithm 1. The algorithm accepts as input the positive border itemsets from the revised border, as well as the list of sensitive itemsets and it produces as output the sanitized database. The processing starts with the housekeeping for the data structures. The outer while loop goes through the list of sensitive itemsets starting from the one with the minimum support. In case there is a tie in the support of two or more itemsets, the algorithm breaks the tie by considering the longest itemset first. From among sensitive itemsets of equal support and length the algorithm processes the itemsets in lexicographic order. The sensitive itemset which is selected in this way undergoes a certain processing that creates an affinity list containing for each tentative victim item, the corresponding tentative victim border set.

The next step of the MaxMin 1 algorithm is to remove support from the current sensitive itemset, so that it becomes hidden. This is done through an inner loop that iterates as many times as the difference between the initial support of the sensitive itemset and the support threshold plus one. In each iteration of the inner loop, the algorithm makes use of the affinity list to find out the maxmin itemset and consequently the corresponding victim item. After the victim item has been pinpointed the algorithm makes a scan through the database to find the first transaction that supports the sensitive itemset. In this transaction, the algorithm turns the one that corresponds to the victim item to a zero. After this modification the algorithm checks whether the support of the sensitive itemset has been appropriately reduced below the minimum support threshold. If this is not the case, it modifies the affinity list and iterates once more.

5.2. Algorithm MaxMin 2

Algorithm MaxMin 2 improves over the MaxMin 1 algorithm in a number of ways. For those cases where there is a maxmin value which is assumed by some maxmin itemsets in the same *vi*-list, the algorithm checks whether the reduction in the support of the sensitive itemset can be achieved without modifying the support of any maxmin itemset. By using Theorem 1 we know in advance that no other minimum support itemset will be affected in this case.

To reduce the support of the sensitive itemset without affecting the support of the maxmin itemsets, we need to ensure that each of the maxmin itemsets is not a subset of the sensitive itemset, and at the same time that there are transactions that provide support to the sensitive itemset without supporting the maxmin itemset. In order to be able to get this information we need to maintain for every sensitive itemset s and every maxmin itemset *maxmin* in the *vi*-list, the list of transactions that support them. If we denote the lists of these transactions as L_s and L_{maxmin}, respectively, then by computing the difference between these two lists (sets) we will know whether we can reduce the support of the sensitive itemset without affecting the support of the maxmin itemset.

In those cases where there are more than one maxmin itemsets that correspond to different tentative victim items we go through all the *vi*-lists containing maxmin itemsets and check whether we can reduce the support of the sensitive itemset without affecting any of the maxmin itemsets in each *vi*-list. If we can do this, then by making use of Theorem 2 we can ensure that no other maxmin itemset in any other *vi*-list will be affected. In order to do this, we need to find transactions that support the sensitive itemset but do not support any of the maxmin itemsets in the *vi*-list. For this, we compute again the lists of transactions that support both the sensitive itemset and the set of maxmin itemsets in every *vi*-list and we compute the difference of the lists that correspond to the maxmin itemsets from the list of the sensitive itemset. If the result is not empty, then we can reduce the support of the sensitive itemset without affecting any other itemset.

In case where the previous scenario is not feasible we consider pairs of *vi*-lists in every iteration of the algorithm. We call the first list in the pair as *vi*1-list and the second *vi*2-list. Then what we try to do is to find a set of minimum support itemsets in a *vi*1-list that will lose support by a change in the corresponding victim item, which are different from all the minimum support itemsets of another *vi*2-list that attain the maxmin value, while we can maintain the support of the rest of the minimum support itemsets in *vi*1-list, then by Lemma 1 we can ensure that the maxmin value in *vi*2-list will remain the same.

Algorithm 2: The MaxMin 2 Algorithm

```
Input: A database $D$, the positive border, the support threshold $\sigma$ and the
       set $S$ of sensitive itemsets.
Output: A database $D'$, that protects the privacy of rules.

Method:

while $S \neq \emptyset$ do
    $s = argmin_{a \in S}[supp(a)]$;
    while $supp(s) \geq \sigma$ do
        $M = \{b | b \in a\text{-}list$ and $a \in s\}$;
        $maxmin = \{c | argmin_{c \in M}[supp(c)]\}$;
        if $\exists b \in s : maxmin \subseteq b\text{-}list$ and $\forall c \neq b$ $c\text{-}list \cap maxmin = \emptyset$ then
            $vi = b$;
            $L = \emptyset$;
            $L = L_s - L_{maxmin}$;
            if $L \neq \emptyset$ then
                set $b = 0$ in an $l \in L$;
            else
                set $b = 0$ in an $l \in L_s$;
        else
            $K = \{d \in s | d\text{-}list \cap maxmin \neq \emptyset\}$;
            foreach $k \in K$ do
                $kmaxmin = maxmin \cap k\text{-}list$;
                $L = \emptyset$;
                $L = L_S - L_{kmaxmin}$;
                if $L \neq \emptyset$ then
                    set $k = 0$ in an $l \in L$;
                    break;
            if $L = \emptyset$ then
                foreach $k_1 \in K$ do
                    foreach $k_2 (\neq k_1) \in K$ do
                        $kmaxmin_1 = maxmin \cap k_1\text{-}list$;
                        $kmaxmin_2 = maxmin \cap k_2\text{-}list$;
                        $L = \emptyset$;
                        $L = (L_{kmaxmin_1} \cap L_s) - (L_{kmaxmin_2} \cap L_s)$;
                        if $L \neq \emptyset$ then
                            set $k_1 = 0$ in an $l \in L$;
                        else
                            set $k_1 = 0$ in an $l \in L_{kmaxmin_1} \cap L_s$;
    $S = S - s$;
```

Finally, there is no other option than to reduce the support of the sensitive itemset while we reduce the support of the maxmin itemset as well. The sketch of the MaxMin 2 algorithm is given in Algorithm 2.

6. Example

Let us assume the database that is shown in the left hand side of Fig. 1. Let also the support threshold σ be 3. The set of all frequent itemsets along with their supports is given in the right hand side of Fig. 1. The positive border in this way turns out to be $Bd^+ = \{abd, acd, bcd, cde\}$. Assuming that we are told that the sensitive items belong to the set $S = \{bc, abd, bcd\}$, the expected positive border becomes $\widetilde{Bd}^+ = \{ab, bd, acd, cde\}$ and the expected negative border is thus $\widetilde{Bd}^- = \{abd, bc\}$. In order to hide the sensitive itemsets, we need to mod-

TID	$Items$
1	$abcde$
2	acd
3	$abdfg$
4	$bcde$
5	abd
6	$bcdfh$
7	$abcg$
8	$acde$
9	$acdh$

Frequent itemsets : Support

$abd : 3, acd : 4, bcd : 3, cde : 3$

$ab : 4, ac : 5, ad : 6, bc : 4, bd : 5, cd : 6, ce : 3, de : 3$

$a : 7, b : 6, c : 7, d : 8, e : 3$

Expected Frequent Itemsets on D'

acd, cde

$ab, ac, ad, bd, cd, ce, de$

a, b, c, d, e

Fig. 1. An example.

ify the database in such a way that the itemsets in the expected negative border lose support while at the same time we keep the support of the itemsets in the expected positive border above the support threshold.

Let us first see what happens if we try to hide the sensitive itemsets in $\widetilde{Bd}^-$ by making use of algorithm MaxMin 1. We first sort the negative border itemsets in increasing order of support and we select for hiding the minimum support itemset which is itemset *abd*. We then create the necessary affinity list as it is indicated below containing in the first column the tentative victim itemsets that belong to the sensitive itemset *abd* and for each tentative victim item we have included the *vi*-lists in a row. For each tentative victim itemset we have included its support after the slash symbol.

a	$ab/4, acd/4$
b	$ab/4, bd/5$
d	$bd/5, acd/4, cde/3$

Since the support of *abd* is 3 and the minimum support threshold is also 3, we need to reduce the support of *abd* by one in order to hide it. For this, the inner loop in Algorithm 1 will be executed only once. Having created the data structure above, we can easily compute the maxmin itemset which is actually a set in this case containing itemsets *ab* from *a*-list and *b*-list as well as *acd* from *a*-list. The algorithm randomly selects the victim maxmin itemset from among the ones that attain the maxmin value (this is 4). Let us assume that randomly selects itemset *ab* from *a*-list. This selection determines automatically that the victim item is item *a*. In the sequel the algorithm scans the database in order to find the first transaction that supports itemset *abd* and removes from this transaction item *a*. The transaction with TID 1 qualifies to this and so the 1 on *a* in the first transaction becomes 0. Since the sensitive itemset *abd* has been hidden, we remove it from the list of sensitive itemsets and we continue with the next sensitive itemset which is itemset *bc*.

We then build an affinity list for the sensitive itemset *bc* which is shown below.

b	$ab/3, bd/5$
c	$acd/3, cde/3$

The inner loop of MaxMin 1 will iterate two times since the support of the sensitive itemset *bc* is 4. In the first iteration we compute the maxmin itemset which randomly again is selected among the three that qualify to be the itemset *ab*. This sets *b* to be the victim item, and so the algorithm scans the database to find the first transaction that supports the sensitive itemset *bc* and removes from this transaction the item *b*. The first transaction that supports *bc* is again transaction with TID 1, and so *b* in this transaction becomes 0. Since we need one more time the inner loop to iterate, we recompute the affinity list, which is illustrated below.

b	$ab/3, bd/4$
c	$acd/3, cde/3$

Interestingly enough we note that no positive border itemset became hidden in the previous iteration. In the second iteration we compute again the maxmin itemset, which is also selected randomly among the three that qualify. Let us assume that *cde* is selected this time which makes *c* the victim itemset. Another scan through the database identifies transaction with TID 4 that supports the sensitive itemset, and sets item *b* in this transaction to 0. The sensitive itemset is hidden in this way.

We turn now to the illustration of the workings of the second proposed algorithm MaxMin 2. By following the same reasoning as above we build the affinity list for the sensitive itemset *abd* as it is shown below and we find the maxmin itemset.

a	$ab/4, acd/4$
b	$ab/4, bd/5$
d	$bd/5, acd/4, cde/3$

The maxmin value is attained by three itemsets. What we do next is to check whether we can reduce the support of the sensitive itemset without affecting the support of the minimum support itemsets that attain the maxmin value. For this reason we compute the set difference $L_{abd} - L_{ab} - L_{acd}$ which we find to be the empty list. For this reason, it is obvious that we cannot reduce the support of the sensitive itemset *abd* without reducing the support of at least one maxmin itemset. The next check is to see whether we can still reduce the support of a maxmin itemset, while leaving the support of other maxmin itemsets intact. In this way, we will have maintained the maxmin value intact for the next iteration.

By following the steps of the algorithm MaxMin 2 since *ab* is in both *a*-list and *b*-list, a possible reduction in its support will also affect the maxmin value. Since *ab* is also a subset of the sensitive itemset, no matter what the change will be, the maxmin will be reduced as well. In this case, we select any transaction that supports *abd* and we change randomly either *a* or *b*. Let us change *a* in the transaction with TID 1. The sensitive item *abd* has been hidden then, and we continue with the next sensitive itemset *bc* where the algorithm needs to iterate twice. We first build the affinity list structure for *bc* which is shown below.

b	$ab/3, bd/5$
c	$acd/3, cde/3$

The maxmin value is taken by three minimum support itemsets. We first check whether we can reduce the sensitive itemset without affecting any of the minimum support itemsets. For the tentative victim item *b* we compute $T_{bc} - T_{ab}$ which is $\{1,4,6\}$ and it implies that we can change *b* in transactions with TIDs $\{1,4,6\}$ without affecting *ab*. By Theorem 2, we know that the minimum support itemsets in the *c*-list will not be affected. Let us change *b* in transaction with TID 1 and continue.

The new affinity list structure is shown below.

b	$ab/3, bd/4$
c	$acd/3, cde/3$

Once again we compute $T_{bc} - T_{ab}$ which is $\{4,6\}$ and once again we can modify *b* in either transaction with TIDs $\{4,6\}$ without affecting any other minimum support itemset that achieve the maxmin value. We finally change *b* in transaction with TID 4 and the sensitive itemset is hidden without affecting any maxmin itemset.

7. Experiments and evaluation

We have compared our algorithms with the heuristic algorithm 2.b from [20] which has been selected for comparison by Sun and Yu [19], as well as with the border based approach which proposed by Sun and Yu [19]. Since Sun and Yu indicated that their algorithm performs constantly better than the heuristic algorithm, we present our findings from the comparison of the proposed algorithms with the border based algorithm.

Initially, we present the hiding achieved by the border based approach on the example presented in Section 6. The algorithm proceeds in changing 3 items, namely item *a* from transaction TID 3 for achieving the hiding of sensitive itemset *abd*, and item *b* from transactions TID 6 and TID 7 for the hiding of sensitive itemset *bc*. The hiding process creates a side effect, in the sense of causing the itemset *ab* from the expected positive border, to lose support and become infrequent. Recall that both of the MaxMin techniques proposed in this paper when applied to the example took three changes while they created no side effect, which is an improvement over the border based approach.

For comparison purposes we have created a table that indicates the performance of our algorithms versus the performance of the border based approach for hiding different sets of itemsets from the itemset lattice of the same example.

Fig. 2 contains four columns. The first presents the set of sensitive itemsets which are selected for hiding, while the rest three columns present the performance of each one of the compared algorithms, the Border based, the MaxMin 1 and the MaxMin 2. The performance of each algorithm is measured by using the notation *m*/*n* where *m* indicates the number of changes in raw data and *n* indicates the number of side effects (number of non-sensitive frequent itemsets which became hidden through the hiding process). Depending on the application, either the first or the second performance measure may have the major significance. Intuitively, the number of raw data changes is bounded by the number of iterations needed for decreasing the support of all the sensitive itemsets while the number of side effects is bounded by the number of frequent itemsets in the revised positive border. By giving the same weight to both performance metrics, we can consider that the algorithm that achieved the minimum sum over these two numbers in each row in the comparison table, is the algorithm that wins. In this respect, we have indicated the winning algorithm in each row by underlining

Sensitive Itemsets	Border Based	MaxMin 1	MaxMin 2
ab	2/0	2/0	2/0
ad	4/1	4/1	$\underline{4/0}$
cd	$\underline{4/2}$	4/3	4/3
abd	1/0	1/0	1/0
cde	$\underline{1/1}$	1/2	$\underline{1/1}$
ab, *acd*	4/1	$\underline{3/0}$	$\underline{3/0}$
ac, *abd*	4/1	3/1	$\underline{3/0}$
ad, *bcd*	5/1	5/1	$\underline{5/0}$
bc, *cde*	2/1	2/1	2/1
ce, *abd*	$\underline{2/0}$	2/1	$\underline{2/0}$
ac, *abd*, *cde*	4/1	4/2	$\underline{3/1}$
ab, *de*, *acd*	5/2	4/1	$\underline{3/0}$
ac, *ad*, *bcd*	$\underline{5/0}$	6/1	$\underline{5/0}$
abd, *acd*, *cde*	4/2	$\underline{3/2}$	$\underline{3/2}$
abd, *acd*, *bcd*	4/0	$\underline{3/0}$	$\underline{3/0}$
ab, *bc*, *cd*, *de*	9/2	8/2	$\underline{7/0}$

Fig. 2. Comparison of Border based, MaxMin 1 and MaxMin 2 algorithms for the example of Section 6.

the performance measures of this algorithm. The information listed in Fig. 2 indicates that for the reference example, as the number of sensitive itemsets grows, MaxMin 2 algorithm outperforms both the Border based and the MaxMin 1 algorithm.

The results presented above are consistent with similar comparisons performed with larger data sets generated by the IBM Synthetic Data Generator [1]. We have also experimented with a number of real datasets which are publicly available through the FIMI repository (http://fimi.cs.helsinki.fi/). In this paper, we report results on comparing the reference border based approach with our proposed algorithms on two datasets from Blue Martini Software Inc. namely BMS-WebView-1 (BMS1 for short) and BMS-WebView-2 (BMS2 for short) that were used for the KDD Cup of 2000 [6] and contain click stream data, and another dataset, the mushroom dataset (Mushroom) that was made available by Roberto Bayardo from the University of California, Irvine [3]. These three datasets was selected on purpose for this evaluation study because they exhibit varying characteristics with respect to the number of transactions and items that they contain, as well as with respect to the average transaction length.

Fig. 3 presents the most indicative results from comparing the three algorithms under evaluation on the three real datasets. The "Hiding Scenario" column in this table indicates the nature and the number of sensitive itemsets which were selected for hiding. In particular, we make use of the notation (i,j) to indicate by i the number of sensitive itemsets that are hidden each time, as well as, by j the length of the sensitive itemset. The rest of the columns are either self explanatory or they have the same semantics as in Fig. 2. We should note here that the selection of sensitive itemsets to hide in each hiding scenario was completely random.

More specifically the evaluation of the three algorithms on the three real datasets indicated the MaxMin 2 performed constantly better than the other two algorithms (exhibiting only a minimum of times similarly with the Border approach) when we consider the sum of the two performance metrics which consists of the number

Data Set	Hiding Scenario	Border Based	MaxMin 1	MaxMin 2
BMS1	(4,10)	286/34	275/30	270/27
	(2,10)	255/30	255/29	234/28
	(1,10)	250/29	246/27	233/25
	(4,5)	361/47	380/44	355/43
	(2,5)	352/45	361/41	354/43
	(1,5)	341/45	362/38	332/38
BMS2	(4,10)	272/27	279/31	271/27
	(2,10)	270/26	280/28	269/25
	(1,10)	262/20	266/22	260/19
	(4,5)	347/42	354/45	348/41
	(2,5)	320/36	340/34	322/31
	(1,5)	321/32	335/31	314/30
Mushroom	(4,10)	299/32	302/31	302/29
	(2,10)	299/33	291/33	293/20
	(1,10)	286/26	287/29	280/19
	(4,5)	324/56	345/54	329/55
	(2,5)	332/46	347/41	333/41
	(1,5)	335/44	343/42	322/37

Fig. 3. Comparison of Border based, MaxMin 1 and MaxMin 2 algorithms for the BMS1, BMS2, and Mushroom.

of changes and the number of side effects. We should also note here the overwhelming superiority of both of the MaxMin approaches with respect to execution times compared to the Border approach. To give an insight into the execution times of the algorithms we report that for both MaxMin algorithms the range of execution times was in between 10 and 20 min, while for the Border based approach the execution time was in the range of 70 and 80 min.

8. Conclusions

We have presented two new algorithms which rely on the maxmin criterion for the hiding of sensitive itemsets in an association rule hiding framework. Both algorithms apply the idea of the maxmin criterion in order to minimize the impact of the hiding process to the revised positive border which is produced by removing the sensitive itemsets and their super itemsets from the lattice of frequent itemsets. By restricting the impact on the border, we can be very efficient in the selection of items which must be selected for hiding, while at the same time it can be ensured that non-border itemsets are protected from hiding. A thorough experimental evaluation indicated that the proposed algorithms (especially MaxMin 2) being at the same time much less computationally demanding, outperform other similar approaches.

References

[1] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: Proceedings of the 20th VLDB, 1994.

[2] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, V.S. Verykios, Disclosure limitation of sensitive rules, in: Proceedings of the 1999 IEEE Knowledge and Data Engineering Exchange Workshop (KDEX'99), 1999, pp. 45–52.

[3] Roberto Bayardo, Efficiently mining long patterns from databases, in: SIGMOD '98: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, 1998.

[4] E. Bertino, I.N. Fovino, L.P. Povenza, A framework for evaluating privacy preserving data mining algorithms, Data Mining and Knowledge Discovery 11 (2) (2005) 121–154.

[5] E. Dasseni, V.S. Verykios, A.K. Elmagarmid, E. Bertino, Hiding association rules by using confidence and support, in: Proceedings of the Fourth International Workshop on Information Hiding, 2001, pp. 369–383.

[6] R. Kohavi, C. Brodley, B. Frasca, L. Mason, Z. Zheng, KDD-Cup 2000 organizers' report: peeling the onion, ACM SIGKDD Exploration Newsletter 2 (2) (2000) 86–98. Available at: http://www.ecn.purdue.edu/KDDCUP.

[7] H. Mannila, H. Toivonen, Levelwise search and borders of theories in knowledge discovery, Data Mining and Knowledge Discovery 1 (3) (1997) 241–258.

[8] S.R.M. Oliveira, O.R. Zaïane, Privacy preserving frequent itemset mining, in: Proceedings of the 2002 IEEE International Conference on Privacy, Security and Data Mining (CRPITS 2002), 2002, pp. 43–54.

[9] S.R.M. Oliveira, O.R. Zaiane, Protecting sensitive knowledge by data sanitization, in: Proceedings of the Third IEEE International Conference on Data Mining (ICDM 2003), 2003, pp. 211–218.

[10] S.R.M. Oliveira, O.R. Zaiane, A framework for enforcing privacy in mining frequent patterns, Technical Report, Computer Science Department, University of Alberta, Canada, 2002.

[11] S.R.M. Oliveira, O.R. Zaiane, Algorithms for balancing privacy and knowledge discovery in association rule mining, in: Proceedings of the 2003 International Database Engineering and Applications Symposium (IDEAS 2003), 2003, pp. 54–63.

[12] S.R.M. Oliveira, O.R. Zaiane, A unified framework for protecting sensitive association rules in business collaboration, International Journal of Business Intelligence and Data Mining 1 (3) (2006) 247–287.

[13] S.R.M. Oliveira, O.R. Zaïane, Y. Saygin, Secure association rule sharing, in: Proceedings of the Eighth Pacific-Asia Conference (PAKDD 2004), 2004, pp. 74–85.

[14] Stanley Oliveira, Osmar Zaiane, Privacy preserving frequent itemset mining. in: CRPITS'14: Proceedings of the IEEE International Conference on Privacy, Security, and Data Mining, 2002, pp. 43–54.

[15] E.D. Pontikakis, A.A. Tsitsonis, V.S. Verykios, An experimental study of distortion-based techniques for association rule hiding, in: Proceedings of the 18th Conference on Database Security (DBSEC 2004), 2004, pp. 325–339.

[16] E.D. Pontikakis, V.S. Verykios, Y. Theodoridis, On the comparison of association rule hiding heuristics, in: Hellenic Database Management Symposium, 2004.

[17] Y. Saygin, V.S. Verykios, C. Clifton, Using unknowns to prevent discovery of association rules, ACM SIGMOD Record 30 (4) (2001) 45–54.

[18] Y. Saygin, V.S. Verykios, A.K. Elmagarmid, Privacy preserving association rule mining, in: Proceedings of the 2002 International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems (RIDE 2002), 2002.

[19] X. Sun, P.S. Yu, A border-based approach for hiding sensitive frequent itemsets, in: Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM 2005), 2005, pp. 426–433.

[20] V.S. Verykios, A.K. Emagarmid, E. Bertino, Y. Saygin, E. Dasseni, Association rule hiding, IEEE Transactions on Knowledge and Data Engineering 16 (4) (2004) 434–447.

George V. Moustakides was born in Drama, Greece, in 1955. He received the diploma in Electrical Engineering from the National Technical University of Athens, Greece, in 1979; the M.Sc. in Systems Engineering from the Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, in 1980, and the Ph.D. in Electrical Engineering and Computer Science from Princeton University, Princeton NJ, in 1983. From 1983 to 1986 he was with INRIA, France and from 1987 to 1990 he held a research position at the Computer Technology Institute of Patras, Greece. In 1991, he joined the Computer Engineering and Informatics department, University of Patras, Greece, as associate professor and in 1996 he became professor at the same department. In 2001, he joined the department of Computer and Communication Engineering, University of Thessaly, Volos, Greece until 2006. Since 2007, he is a professor with the department of Electrical and Computer Engineering, University of Patras, Greece. His interests include adaptive signal processing algorithms, sequential detection techniques and probabilistic and statistical methods for databases.

Vassilios S. Verykios received the diploma degree in Computer Engineering from University of Patras, Greece, the M.Sc. and Ph.D. degrees from Purdue University in 1992, 1997, and 1999, respectively. In 1999, he joined the faculty of Information Systems in the College of Information Science and Technology at Drexel University, Pennsylvania, as a tenure track assistant professor. Since 2005, he is an assistant professor in the Department of Computer and Communication Engineering at the University of Thessaly, in Volos, Greece. He has also served on the faculty of Athens Information Technology Center, Hellenic Open University and University of Patras, Greece. His main research interests include knowledge based systems, privacy and security in advanced database systems, data mining, data reconciliation, parallel computing and performance evaluation of large scale parallel systems. He has published over 40 papers in major referred journals and in the proceedings of international conferences and workshops, and he has served in the program committees of several international scientific events. He has consulted for Telcordia Technologies, ChoiceMaker Technologies, Intracom SA and LogicDIS SA. He has also been a visiting researcher for CERIAS, the Department of Computer Sciences at Purdue University, the US Naval Research Laboratory and in the Research and Academic Computer Technology Institute in Patras, Greece.