Protecting Individual Information Against Inference Attacks in Data Publishing

Chen Li[1] Houtan Shirani-Mehr[1] Xiaochun Yang[2*]

[1] Department of Computer Science, University of California at Irvine, CA, USA
{chenli,hshirani}@ics.uci.edu
[2] School of Information Science and Engineering, Northeastern University, China
yangxc@mail.neu.edu.cn

Abstract. In many data-publishing applications, the data owner needs to protect sensitive information pertaining to individuals. Meanwhile, certain information is required to be published. The sensitive information could be considered as leaked, if an adversary can infer the real value of a sensitive entry with a high confidence. In this paper we study how to protect sensitive data when an adversary can do inference attacks using association rules derived from the data. We formulate the inference attack model, and develop complexity results on computing a safe partial table. We classify the general problem into subcases based on the requirements of publishing information, and propose the corresponding algorithms for finding a safe partial table to publish. We have conducted an empirical study to evaluate these algorithms on real data.

1 Introduction

As many database applications need to publish information on the Web or share information among different organizations, there is an increasing need for these applications to meet their security requirements. Often the data owner needs to protect sensitive information about individuals, such as the disease of a patient, the salary of an employee, or the ethnicity of a customer. On the other hand, given published data, an adversary could use the available information to infer the sensitive information. For example, common knowledge [1], regression models could be used to infer information. From the data owner's perspective, the method he uses to protect the sensitive information depends on what inference technique he believes an adversary could use to do the inference [2].

In this paper we study the problem of protecting sensitive information about individuals when an adversary does inference attacks using data distributions derived from published information. However, there are various ways the adversary can launch the attack. We study how to protect sensitive information about individuals against inference attacks. We focus on inference attacks using *association rules*. Several important factors need to be considered when deciding

* Supported by the Natural Science Foundation of China under Grant No.60503036, the Fok Ying Tong Education Foundation of China under Grant No. 104027.

what additional information needs to be hidden. First, the hidden information depends on the data owner's tolerance on the confidence the adversary can get about the real value of a sensitive entry. Secondly, the owner has to consider the fact that the adversary can do similar inferences by using other properties of this individual as well as a combination of his properties. Being conservative, a data publisher wants to make sure that the adversary does not obtain the real value with a high probability by using any combination of these properties. Thirdly, often there are application-specific requirements when publishing or sharing information. The application could require certain information be released. Furthermore, the application often requires the owner to release as much information as possible.

These challenges motivate our study, in which we make the following contributions: (i) We formulate a data-privacy framework in which a data owner needs to protect sensitive information against inference attacks using association rules; (ii) We study complexity issues of the problem; (iii) We classify the problem into subcases based on the requirements of publishing information, and develop algorithms for these cases; and (v) We conducted an empirical study to evaluate our algorithms on real data sets.

Due to space limitation, we include more results in the full version [13] of this paper.

1.1 Related Work

There have been many studies on data security and privacy. We briefly summarize some related topics in the context of data publishing and sharing.

k-anonymity [3]: The problem is to hide the quasi-identifiers of sensitive entities. Our goal is to hide the real values of individuals, not their identifiers.

Privacy-preserving data mining [4]: The problem is to hide distribution properties of a data set without revealing the information about each individual entity. The most commonly used approach is to perturb the data, say, using randomizing functions. In the applications we are considering, data can only be hidden, but cannot be modified. Evfimievski et al. [5] developed a privacy-preserving framework based on the notion of amplification.

A closely related topic is association-rule hiding (ARH for short) [6, 7], in which the goal is to prevent sensitive association rules from being disclosed. For instance, Atallah et al. [6] proved that it is NP-hard to hide a set of association rules while minimizing the effect on other association rules. This effect is measured as the number of rules lost and the number of rules newly introduced in the hiding process. In [6, 7] several heuristics were proposed to hide association rules. Although our approach is also based on hiding sensitive association rules, there are several differences. (1) Our goal is to hide sensitive entries (the same with [8]), and we use hiding association rules as an intermediate step, while ARH's goal is to hide rules. (2) Most ARH works are dealing with transactional data with values of 1 and 0, while we are dealing with more general data, mostly categorical data. (3) Most ARH works try to minimize the side effect of information hiding on other rules, while we focus on minimizing information loss in terms of

the number of hidden entries. (4) We consider application-specific requirements about what information needs to be published, which are not considered either in ARH works nor [8]. In order to experimentally show the difference between our algorithms and those ARH algorithms, we also implemented one of their approaches to compare the results.

Other related works include the following. [9] developed data-dependent approaches to data privacy. [10] developed an encryption-based approach to access control on publishing XML documents. There are also studies on inference control in relational databases [11], and studies on XML security control (e.g., [12]).

2 Data-Privacy Framework

We consider applications in which a data owner has data stored in a relational table $R(A_1, \ldots, A_k)$, which has the following three types of entries.

- A **positive entry** is an entry the data owner has to release, and it is marked with a positive sign ("+") in the table.
- A **negative entry**, also called a *sensitive entry*, is an entry the owner wants to hide, and it is marked with a negative sign ("−"). The owner hides the value of each negative entry by replacing it with a NULL. A record with a sensitive entry is called a *sensitive record*. In this study we focus on the case where all the negative entries are from the same attribute, called the *sensitive attribute*, denoted by S. Our results can be extended to the case of multiple sensitive attributes.
- An **unmarked entry** is an entry without a sign. It means that the owner can publish its original value, or hide it, in order to protect other sensitive entries against inference attacks.

2.1 Association Rules

A *partial table* is a table in which some entries have been hidden by the data owner. Given a partial table, there are various ways the adversary could do inference. In this work, we focus on the case where the adversary utilizes the derived *association rules* [6, 7] from the partial table to do inference. A *pattern* for the relation R is a set of attribute-value pairs: $\{(D_1, v_1), (D_2, v_2), \ldots, (D_k, v_k)\}$, in which each D_i is an attribute in the relation (possibly the sensitive attribute), and each v_i is a value for the attribute D_i. The *support* of this pattern p, denoted by $supp(p)$, is the number of records in the table that satisfy the following condition: for each $i = 1, \ldots, k$, the value of the record on attribute D_i is v_i. We say such records *have the pattern* p. An *association rule* is in the format of $r : p \rightarrow s$, in which p is a pattern that does not use the sensitive attribute, and s is a value for the sensitive attribute S. We call p the *condition pattern* of the rule r. The *support* of r, denoted by $supp(r)$, is the support of the pattern $p \cup \{(S, s)\}$ (or "$p \cup \{s\}$" for short). The confidence of r is

$$conf(r) = \frac{supp\big(p \cup \{(S, s)\}\big)}{supp(p)}.$$

Each record with the pattern $p \cup \{(S, s)\}$ is called *relevant* to the rule r.

2.2 Inference Attacks Using Association Rules

Let t be a sensitive record. Its sensitive entry, e, for the sensitive attribute S, is hidden by the data owner. The adversary uses a set of *condition attributes* to infer the original value of the sensitive entry e. Based on the assumption that a set of condition attributes are given, our techniques can provide the corresponding privacy guarantees.

Let $\{D_1, \ldots, D_m\}$ be a nonempty subset of the condition attributes of the sensitive record t. Consider the corresponding association rule r: $\{(D_1, t[D_1]), \ldots, (D_m, t[D_m])\} \rightarrow t[S]$. Here "$t[D_i]$" denotes the value of attribute D_i in record t. The adversary could use r to infer the sensitive value $t[S]$ of record t. We assume that the data owner provides a minimum support *minsupp* and a minimum confidence *minconf* to specify his tolerance of information leakage. We say that the association rule r *leaks* the sensitive information in record t if (i) the support of r is greater than *minsupp*, and (ii) the confidence of r is greater than *minconf*. In this case, we call rule r an *unsafe* association rule. So a rule is safe if its support is within *minsupp*, *or* its confidence is within *minconf*.

For the sensitive record t, there are different subsets of its condition attributes. A sensitive entry is *leaked* if one of these association rules is unsafe. Otherwise, the sensitive entry is *safe*. The data owner, being conservative, wants to make sure that the sensitive entry of this record is not leaked, i.e., none of its association rules is unsafe. A partial table is called *safe* if each of its sensitive entries is safe, i.e., it does not have an unsafe association rule.

In order to make sure that each sensitive entry is safe in a partial table, the data owner needs to decide which of other (unmarked) entries need to be hidden, so that each association rule for the sensitive entries is safe. While there are many such safe partial tables, we are interested in finding one that has the "maximum" amount of information. In this study we consider information amount measured by the number of entries that are not hidden. Intuitively, the more entries that are hidden, the less information is published. A safe partial table T_p is called *optimal* if there is no other partial table whose information amount is greater than that of T_p. In the following sections we study how to compute a safe partial table while releasing as much information as possible. For limited space, we focus on the case where the minimum support is 0. We discuss how to extend the results to the case where $minsupp > 0$ in [13].

3 Complexity Results

In this section we study complexity issues related to the problem of computing a safe partial table. All the proofs are in [13].

Theorem 1. *The problem of deciding whether there exists a safe partial table is NP-hard.*

Theorem 2. *The problem of computing an optimal safe partial table is NP-hard.*

The complexity results above are developed for the general cases, where the positive entries can appear in any attribute. Often the application requires the positive entries be for certain attributes. Based on where the positive entries are allowed in the table, we study various subcases. A table can be divided into four regions: Condition entries of the nonsensitive records, sensitive entries (those entries of the sensitive attribute) of the nonsensitive records, condition entries of the sensitive records, and sensitive entries of the sensitive records which are always marked negative. Based on whether the application allows each region to be hidden, there are eight subcases as shown in Fig. 1. Besides the negative mark, only entries marked with ◯ in Fig. 1 can be hidden.

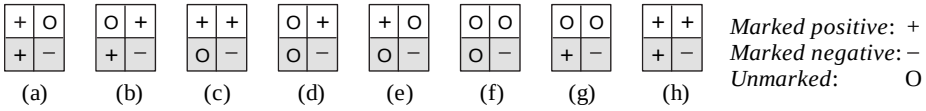


Fig. 1. Subcases (the shaded region represents sensitive records).

We study the complexity issues in these subcases. We have proved that the problem of computing an optimal safe partial table is NP-hard for each of the subcases (a)-(g) in Fig. 1. For each subcase, there is always a safe partial table, since there is one unmarked region, and hiding all the entries in this region can make the table become safe. For subcase (h), the problem of computing an optimal solution is not defined since no entry can be hidden.

4 Computing a Safe Partial Table Efficiently

In this section, we study how to compute a safe partial table efficiently. We first analyze the subcases in Fig. 1, and identify two cases, cases (a) and (b), on which our algorithms for other remaining cases are based. We develop an algorithm for finding a solution for case (a). In the next section, we will study case (b). Algorithms for finding solutions of remaining cases are developed in [13].

Lemma 1 tells us that, in cases (f) and (g), we do not need to hide the condition entries of those nonsensitive records to find a solution. As a consequence, an algorithm for case (a) is applicable for case (g), and an algorithm for case (e) is applicable for case (f) as well.

Lemma 1. *In cases (f) and (g), for each safe partial table T that has hidden condition entries of the nonsensitive records, there is another safe partial table T' that does not hide more entries than T, and none of the condition entries of the nonsensitive records is hidden in T'.*

4.1 Algorithm for Case (a)

The main idea of the algorithm is to reduce the problem of finding an optimal safe partial table to the Set Multi-Cover problem [14].

Definition 1. *(Set Multi-Cover) Given a universal set* $\mathcal{U} = \{e_1, e_2, \ldots, e_n\}$ *and a set* $\mathcal{L}$ *of some subsets of* $\mathcal{U}$, $\mathcal{L} = \{S_i | S_i \subseteq \mathcal{U}\}$, *our goal is to find a smallest sub-collection* $\mathcal{C} \subseteq \mathcal{L}$ *that covers each element* $e \in \mathcal{U}$ *at least* r_e *times.*

Consider a sensitive record with a sensitive value s. It has an unsafe rule $r : p \rightarrow s$, whose confidence is greater than the confidence threshold *minconf*. In case (a), to reduce this confidence, we can only hide the sensitive entries of those nonsensitive records. In particular, we need to consider those records relevant to the rule (i.e., they have the pattern $p \cup \{s\}$), and choose some to hide their sensitive entries. The minimum number β_p of such entries that require to be hidden should satisfy the following inequality:

$$\frac{supp(p \cup \{s\}) - \beta_p}{supp(p)} \leq minconf.$$

For each nonsensitive record t with an s value and with at least one of these patterns, let S_t be the set of the patterns that the record t has. S_t is called the *pattern set* of t. Now we convert our problem of finding an optimal partial table to an instance of the set multi-cover problem. Let the universal set $\mathcal{U}$ consist of the condition patterns in the unsafe association rules of the record. Let the set $\mathcal{L}$ consist of the pattern sets of the records with an s value. Our goal is to find a cover for these sets that covers each condition pattern p at least $r_p = \beta_p$ times.

Our algorithm works as follows. In each iteration, we select a set S_t with the largest number of unsafe patterns, and add it to the cover. Correspondingly, we hide the sensitive entry of the record t. Thus we decrease the r_p value of each pattern in the set S_t by one.

The performance ratio of the algorithm is known to be H_k, in which k is the size of largest set, and H_k is the harmonic number of order k [14]. k is at most $2^{(n-1)}$, where n is the number of condition attributes. If $\mathcal{C}$ is the cover obtained by our algorithm, and $\mathcal{C}^*$ is the optimal cover, then

$$\frac{|\mathcal{C}|}{|\mathcal{C}^*|} \leq H_{2^{(n-1)}} < (n-1)\ln 2 + 1.$$

The upper bound for performance ratio is computed using the upper bound for H_k in [15].

Multiple Sensitive Values: For case (a), there could be more than one sensitive value. Notice that since we only hide entries of the sensitive attribute, hiding the entries with one sensitive value does not affect the rules of a sensitive record with a different sensitive value. To see the reason, consider the confidence formula for an association rule $p \rightarrow s$, $conf(p \rightarrow s) = \frac{supp(p \cup \{s\})}{supp(p)}$. Hiding a sensitive value s will only reduce $supp(p \cup \{s\})$, and has no effect on other association rules $p \rightarrow t$, where $t \neq s$. Therefore we can group the sensitive records according to their sensitive values, and run the algorithm above for each group of records to find a safe partial table.

5 Algorithm for Case (b)

In this section we study how to compute a safe partial table for case (b) in Fig. 1, in which we can only hide the condition entries of those nonsensitive records. Similarly to the way we deal with case (a), we first develop an algorithm for the case where the sensitive records have the same sensitive value. We can extend it ([13]) to the case where the sensitive records can have different sensitive values.

5.1 R-Graph

Consider the case where we have a set $H = \{t_1, \ldots, t_m\}$ of sensitive records with the same sensitive value s with k condition attributes. Some of these records are not safe in the original table, i.e., they have unsafe association rules. Our algorithm for finding a safe partial table is based on a directed graph called *R-Graph*. It defines a partial order among the condition patterns of these sensitive records. It is constructed as follows. For each sensitive record t_i, consider its 2^k-1 association rules corresponding to the nonempty subsets of the k condition values of t_i. For each rule r, let $p(r)$ be its condition pattern. There is a node, denoted by $v(r)$, in the R-graph, which corresponds to rule r. For example, for a record $\langle a, b, s\rangle$ with the sensitive value s, the graph has three nodes corresponding to the rules $\{a\} \rightarrow s$, $\{b\} \rightarrow s$, and $\{a, b\} \rightarrow s$, respectively.

The graph has a directed edge from a vertex $v(r)$ to $v(r')$ if the pattern $p(r) \supset p(r')$, and $|p(r)| = |p(r')| + 1$. That is, the pattern $p(r)$ is obtained by adding one more value to pattern $p(r')$. We call $v(r)$ a *parent* of $v(r')$. Intuitively, hiding an entry of a record relevant to the pattern of the child node will also reduce the confidence of the rule of the pattern of the parent node.

Each node $v(r)$ in the R-graph is associated with two numbers. The first number, called the *MinHide* of the rule r, represents the minimum number of records relevant to the rule r that need to be hidden in order to make rule r safe. By "hidden" we mean each such record should have at least one of its condition values hidden. That is, MinHide is the minimum integer x that satisfies:

$$\frac{supp(p(r) \cup \{s\}) - x}{supp(p(r)) - x} \leq minconf.$$

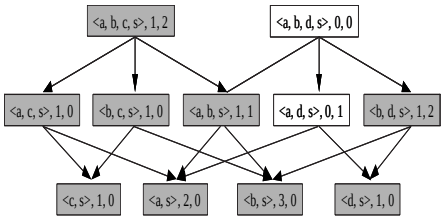
Given a partial table, a node is safe iff its MinHide value is 0. The second number, called the *ExactMatch* number of the rule, is the number of records relevant to the rule, but not relevant to the rule of any ancestor of this node.

For instance, let $minconf = 0.55$. Then the constructed R-graph for the table in Fig. 2(a) is shown in Fig. 2(b). There are two sensitive records $\langle a, b, c, s\rangle$ and $\langle a, b, d, s\rangle$. The graph has 11 nodes corresponding to the 11 different condition patterns of the records, such as $\{a, b, c, s\}$, $\{a, b, s\}$, $\{b, c, s\}$, etc. There is an edge from the node $\{a, b, c, s\}$ to the node $\{a, b, s\}$, since the former can be obtained by adding a value c to the latter. Consider the rule $r : \{b, d\} \rightarrow s$ and the corresponding pattern $p(r) : \{b, d\}$. The node $p(r)$ is associated with two values. The MinHide value of the rule is 1, meaning that we need to hide at least

ID	A_1	A_2	A_3	A_4
$t1$	a+	b+	c+	s$^-$
$t2$	a+	b+	c+	s
$t3$	a+	b+	c+	s
$t4$	a+	b+	*+	s
$t5$	a+	b+	d+	s$^-$
$t6$	a+	*+	d+	s
$t7$	*+	b+	d+	s
$t8$	*+	b+	d+	s
$t9$	*+	*+	d+	*

(a)

Each $*$ represents distinct value that is different from the other values.



(b)

Fig. 2. An example of a table with two sensitive records and its R-Graph. Each shaded node represents an unsafe rule, i.e., its MinHide value is greater than zero.

one record relevant to this rule, in order to make the rule safe (in the current partial table). The ExactMatch value of this node is 2, since there are two records relevant to this rule, and not relevant to the rule of any of its ancestors. Notice that as we run the algorithm to hide more entries, these values of the nodes in the graph can change, since we are hiding more entries.

5.2 Algorithm

Our approximation algorithm works iteratively. In each step, it identifies the records of a pattern, then decides their entries to hide. The algorithm modifies the R-graph after each iteration. It repeats until all the nodes in the R-graph are safe.

Choosing Records to Hide If the ExactMatch number of a node is not zero, we call this node an *active node*. We could choose such a node to hide its relevant records in order to reduce the confidences of its unsafe descendant nodes. The algorithm chooses one of the "top-level" active nodes, i.e., those that do not have active ancestors. For instance, in Fig. 2(b), node $\{a, b, c, s\}$ is a top-level active node. By hiding a condition value of a record relevant to this rule, we can reduce the confidence of the rule, and possibly the confidences of its unsafe descendant nodes. In the case where we have more than one top-level active node, we can use one of the following methods to choose one to hide its entries.

- RANDOM: We randomly select one of them.
- RANDOM-MAX: Among those active top-level nodes with the maximum number of attributes, we randomly select one.
- MAX-DESCENDANT: We choose the node with the largest number of unsafe descendant nodes. Intuitively, hiding a condition value of a record relevant to such a pattern can help reduce the confidence of many unsafe nodes.

Hiding a Condition Entry in Selected Records When choosing an entry for a record relevant to a node to hide, we want to use this value to reduce the confidence of as many unsafe descendant nodes as possible. Based on this

observation, we choose an entry that, among all the entries of this node, appears in the most unsafe descendant nodes of the node.

Deciding the Number of Records to Hide After selecting a top-level active node $v(r)$ and its entry e, we need to decide the number m of records relevant to this rule whose e value we want to hide. One naive way is to use the ExactMatch value of this rule. This method does not consider the MinHide values of the descendants of this node that share this e value, thus may hide too many records. (Notice that an active node could have a MinHide value of 0.) To solve this problem, we decide value m as follows. We consider all the unsafe descendant nodes of this node $v(r)$ that share this e value, and let α to be the minimum value of these MinHide values. Then we choose m to be the minimum of α and the ExactMatch number of this rule r.

6 Experiments

We have conducted experiments to evaluate our proposed algorithms. Based on the analysis in Section 4, we compare cases (a) and (b) in this paper and cases (c) to (g) in [13]. For case (h), no entries can be hidden, except those sensitive entries. We used two real relational data sets. The first one is the "adult" database from the UC Irvine machine learning repository.[3] The data set has $32,561$ records in its training set, and we randomly chose $30,000$ of them in our experiments. We used the 7 categorical attributes as the condition attributes, including gender, marital status, level of employment, highest education degree, number of education degrees, and salary (low or high). We used its occupation attribute as the sensitive attribute. The second data set contains information about customers of a bank company. It has $30,000$ records. We used 7 categorical attributes: background (award or bad records), loan type (such as house purchase or education), amount of savings (low, medium, or high), occupation, marital status, number of apartments/houses, and resident type. We used the credit rating attribute as the sensitive attribute.

In the experiments, we evaluated how the following factors affect the number of entries hidden by the algorithms and their efficiency. (1) The number of condition attributes; (2) the number of sensitive records; (3) the number of records; (4) the value of the minimum confidence *minconf*; (5) the value of minimum support *minsupp*, and (6) the number of unsafe rules. We also compared our algorithms with one of the association-rule-hiding algorithms called the CR algorithm [7]. All the algorithms were implemented in C++, and run on an Intel Pentium IV 2.4GHz PC with 512MB RAM. For each setting, we tested an algorithm by running it 10 times to compute the average values.

6.1 Algorithm Implementation for Case (a)

We randomly selected a certain number of records from each data set as sensitive records, and let this number vary between 2 and 35. We set the owner's minimum

[3] Downloaded from http://www.ics.uci.edu/∼mlearn/MLSummary.html

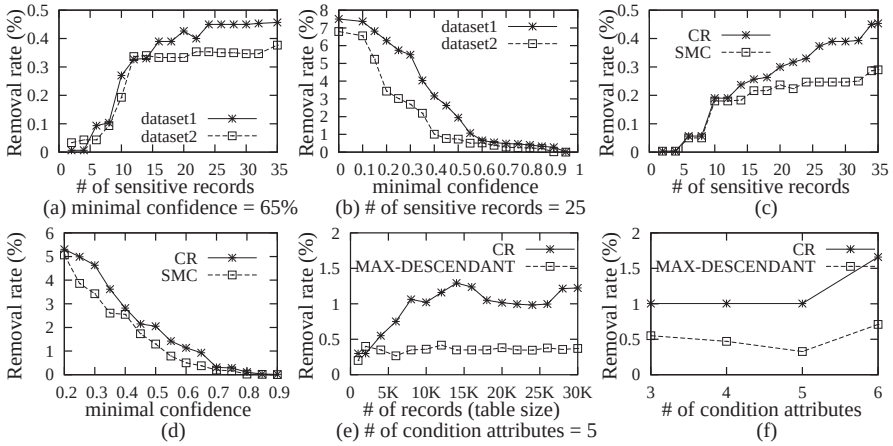


Fig. 3. Test results.

confidence $minconf = 0.6$, and minimum support $minsupp = 60$, which is 0.2% of the table size.

It is computationally prohibitive to implement the exhaustive search algorithm. We implemented the algorithm based on Set Multi Cover, called the SMC algorithm, as described in Section 4. Fig. 3(a) shows how the number of sensitive records affects the number of entries hidden by this algorithm. The y-axis, called "Removal rate," is the percentage of the total number of entries in the table that are hidden. It shows that as the number of sensitive records increased, the number of removals in the sensitive attribute increased, since more records could have unsafe association rules, resulting in hiding more entries. For dataset 1, when there were 4 sensitive entries, the algorithm decided to hide about 0.006% of all the entries. When there were 35 sensitive entries, the algorithm decided to hide about 0.45% of all the entries. Next, we selected 20 different sensitive records, and let the minimum confidence *minconf* vary from 0 to 1.0. Fig. 3(b) shows that as the minimum confidence increases, there are fewer unsafe rules, resulting in hiding fewer entries.

Comparison with ARH algorithms: We experimentally compared our algorithms with those association-rule-hiding (ARH) algorithms [6, 7]. We chose the CR algorithm [7] as a representative. As explained in Section 1, our problem is different from the problem of hiding association rules in various aspects. The goal of this comparison is to show how many entries we need to remove if we were to use one of these ARH algorithms to solve our problem. When adopting the CR algorithm to our setting, we used it to hide those unsafe rules introduced by the sensitive records. We followed the algorithm's way of deciding the sequence of hiding the association rules, and the way of choosing records to hide. Since in case (a), we can only remove values of the sensitive attribute, we ignored the last step in the CR algorithm that chooses an entry of a record to hide. Figs. 3(c) and (d) show the results of our SMC algorithm and the CR algorithm for case (a), using

the same setting as described above. Fig. 3(c) shows that our SMC algorithm hides much fewer additional entries than the CR algorithm. Fig. 3(d) shows that the number of removals decreased for both algorithms, as the minimum confidence *minconf* decreased. Again, our SMC algorithm outperformed the CR algorithm.

6.2 Algorithm Implementation for Case (b)

We also implemented the algorithm described in Section 5 for case (b), in which we can only hide the condition entries of those nonsensitive records. We used data set 1. The minimum confidence $minconf = 0.60$ and the minimal support $minsupp = 60$, which is about 0.2% of the table size. We randomly chose some number of records (table size) and categorical attributes. We increased the number of records and condition attributes. We randomly chose 10 entries (with the same value) as sensitive entries. Three heuristics were implemented to choose a top-level node in the R-Graph, namely, RANDOM, RANDOM-MAX, and MAX-DESCENDANT, as described in Section 5.2. Their results were very similar, so we mainly reported the results of the MAX-DESCENDANT heuristic. In addition, we modified the last step of the CR algorithm slightly so that it can choose a condition entry of a record to hide. Figs. 3 (e) and (f) show how the removal rate changed as the number of records in the table increased up to $30,000$. As the table size becomes larger, the support of the association rule of the sensitive records also increased, which resulted in more values hidden in order to obtain a safe partial table. Our algorithm ("MAX-DESCENDANT") removes fewer entries than the CR algorithm.

We next evaluated how the number of condition attributes affects the removal rate. We randomly chose $30,000$ records in dataset 1. We set $minsupp = 60$, which is 0.2% of the table size. We let the number of condition attributes vary from 3 to 6. Fig. 3(f) shows that when we increased the number of condition attributes, more values need to be hidden since more condition values can be used to do inference. There are two reasons that our algorithm outperforms the CR algorithm. First, our algorithm chooses a top-level node that can affect more unsafe rules, while CR chooses a record that covers fewer rules, trying to minimize the "side effect" on other rules (e.g., making a safe rule unsafe). Second, our algorithm chooses hiding an entry that occurs in more unsafe descendant patterns of the selected record pattern in the R-graph (i.e., it affects more unsafe rules), whereas CR chooses hiding an entry whose support is maximum in the table. Our experiments [13] show the results when there are multiple sensitive values. We set $minsupp = 60$ and $minconf = 0.7$ for data set 1. We chose 20 sensitive records, and tested the effect of different numbers of sensitive values. We increased the number of different sensitive values from 1 to 9. The results show that when the number of sensitive values increased, we need more removals to get a safe partial table. The reason is that removing conditions of one unsafe rule may increase the confidence of another unsafe rule, which needs more removals to make it safe.

7 Conclusions

In this paper, we studied an important privacy problem in data-publishing or sharing environments: how to protect sensitive information about individuals against inference attacks using association rules? In deciding what data should be released, we need to consider application-specific requirements, e.g., some information has to be released. We formulated the problem, and developed complexity results of the problem. We classified the different cases of the problem, based on what information has to be released. We developed efficient algorithms for computing a safe partial table. We have conducted an empirical study on real data sets to evaluate our techniques.

References

1. Yang, X., Li, C.: Secure XML publishing without information leakage in the presence of data inference. In *VLDB*, 2004.
2. Verykios, V.S., Bertino, E., Fovino, I.N., et al.: State-of-the-art in privacy preserving data mining. *SIGMOD Record*, 33(1):50–57, 2004.
3. Sweeney, L.: k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, 2002.
4. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In *SIGMOD Conference*, pages 439–450, 2000.
5. Evfimievski, A., Srikant, R., Agrawal, R., Gehrke, J.: Privacy preserving mining of association rules. In *KDD*, pages 217–228, 2002.
6. Atallah, M., Bertino, E., Elmagarmid, A., Ibrahim, M., Verykios, V.: Disclosure limitation of sensitive rules, 1999.
7. Saygin, Y., Verykios, V.S., Clifton, C.: Using unknowns to prevent discovery of association rules. *SIGMOD Record*, 30(4):45–54, 2001.
8. Aggarwal, C., Pei, J., Zhang, B.: On Privacy Preservation against Adversarial Data Mining. In *Proceedings of ACM SIGKDD 2006*, 2006.
9. Damiani, E., Vimercati, S.D.C.D., Paraboschi, S., Samarati, P.: A Fine-Grained Access Control System for XML Documents. *ACM Transaction on Information and System Security*, 5(2):169–202, 2001.
10. Miklau, G., Suciu, D.: Controlling Access to Published Data using Cryptography. In *VLDB*, 2003.
11. Brodskyand, A., Farkas, C., Jajodia, S.: Secure Databases: Constraints, Inference Channels, and Monitoring Disclosures. *TKDE*, 12(6):900–919, 2000.
12. Bertino, E., Carminati, B., Ferrari, E.: A Secure Publishing Service for Digital Libraries of XML Documents. In *ISC*, pages 347–362, 2001.
13. Li, C., Shirani-Mehr, H., Yang, X.: Protecting Individual Information Against Inference Attacks in Data Publishing. UCI Technical Report, 2006.
14. Rajagopalan, S., Vazirani, V.V.: Primal-dual RNC approximation algorithms for set cover and covering integer programs. *SIAM Journal on Computing*, 28(2):525–540, 1999.
15. Graham, R.L., Knuth, D.E., Patashnik, O.: *Concrete mathematics: a foundation for computer science*. Addison-Wesley Longman Publishing Co., USA, 1989.