On Privacy Preservation against Adversarial Data Mining*

Charu C. Aggarwal
IBM T. J. Watson Research Center, USA
charu@us.ibm.com

Jian Pei
Simon Fraser University, Canada
jpei@cs.sfu.ca

Bo Zhang
Zhejiang University, China
tczbzb@gmail.com

ABSTRACT

Privacy preserving data processing has become an important topic recently because of advances in hardware technology which have lead to widespread proliferation of demographic and sensitive data. A rudimentary way to preserve privacy is to simply hide the information in some of the sensitive fields picked by a user. However, such a method is far from satisfactory in its ability to prevent adversarial data mining. Real data records are not randomly distributed. As a result, some fields in the records may be correlated with one another. If the correlation is sufficiently high, it may be possible for an adversary to predict some of the sensitive fields using other fields.

In this paper, we study the problem of *privacy preservation against adversarial data mining*, which is to hide a minimal set of entries so that the privacy of the sensitive fields are satisfactorily preserved. In other words, even by data mining, an adversary still cannot accurately recover the hidden data entries. We model the problem concisely and develop an efficient heuristic algorithm which can find good solutions in practice. An extensive performance study is conducted on both synthetic and real data sets to examine the effectiveness of our approach.

Categories and Subject Descriptors:H.2.8 [**Database Applications**]: [Data Mining]

General Terms: Security, Algorithms, Performance **keywords:** Privacy preservation, data mining, association rules

1. INTRODUCTION

In recent years, large amounts of data about individuals have become available with corporations as well as public entities. This has led to serious concerns about the misuse and privacy of such data. Some interesting discourses on the nature of privacy in the context of recent trends in information technology may be found in [6]. This has led to a considerable amount of research on the subject, such as [1, 2, 3, 4, 8, 9, 10, 13].

The most basic model of privacy preserving data processing is one in which we erase the sensitive entries in data. These erased entries are usually particular fields which are decided by the user, who may either be the owner or the contributor of the data. The advantage of this approach is that it is extremely simple to implement in practice, and can be tailored easily to a variety of user preferences. As a result, many current applications use this straightforward method for privacy preservation. A variety of methods such as conceptual reconstruction [5] can be used to apply existing data mining algorithms on such data with missing values. A key weakness of this approach is that a data mining proficient adversary may use the correlations among the fields in the data to guess the sensitive fields from other (non-sensitive) fields.

EXAMPLE 1 (MOTIVATION). Consider a table `Employees` in Table 1. Suppose some users want to hide some information as follows. (1) Cathy wants to hide her salary level; (2) Frank wants to hide his education background; (3) Grace wants to hide her marriage status; (4) Ian wants to hide his gender; (5) Janet wants to hide her gender as well; and (6) All names should be hidden and replaced by random row-ids. The table after hiding is shown in Table 2.

Name	Title	Gdr	MStatus	Edu.	Sal-lvl
Alice	Assistant	F	Unmarr.	Coll.	SL-3
Bob	Assistant	M	Married	Coll.	SL-3
Cathy	Assistant	F	Married	Univ.	SL-3
Daniel	Manager	F	Unmarr.	Univ.	SL-5
Elena	Manager	F	Married	Univ.	SL-5
Frank	Manager	M	Married	Univ.	SL-5
Grace	Manager	F	Married	MBA	SL-7
Helen	Manager	F	Married	Ph.D.	SL-5
Ian	Accountant	F	Unmarr.	MBA	SL-5
Janet	Accountant	M	Married	Univ.	SL-4

Table 1: Table `Employee`.

Id	Title	Gdr.	MStatus	Educat.	Sal-lvl
1	Manager	F	Married	Univ.	SL-5
2	Assistant	F	Unmarr.	Coll.	SL-3
3	Manager	F	Married	Ph.D.	SL-5
4	Assistant	M	Married	Coll.	SL-3
5	Manager	M	Married	#	SL-5
6	Accountant	#	Unmarr.	MBA	SL-5
7	Manager	F	Unmarr.	Univ.	SL-5
8	Assistant	F	Married	Univ.	#
9	Manager	F	#	MBA	SL-7
10	Accountant	#	Married	Univ.	SL-4

Table 2: Table `Employee` after hiding some sensitive entries.

*The research of Jian Pei and Bo Zhang was partially supported by NSERC Grants 312194-05 and 614067, and NSF Grant IIS-0308001. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies. This work was done in 2004 when the third author was an exchange student at Simon Fraser University.

KDD'06, August 20--23, 2006, Philadelphia, Pennsylvania, USA.

However, it is possible to generate the following association rules from Table 2:

R_1: *Assistant* $\rightarrow$ *SL-3* (support: 2, confidence 100%),

R_2: *Manager* $\wedge$ *SL-5* $\rightarrow$ *University* (support:3, confidence 66.7%), and

R_3: *Manager* $\wedge$ *Female* $\rightarrow$ *Married* (support:3, confidence 66.7%).

These rules have a revealing effect on the values of the individual records. For example, one may accurately predict that (1) the missing value in tuple 5 is "University" (by rule R_2); (2) the missing value in tuple 8 is "SL-3" (by rule R_1); and (3) the missing value in tuple 9 is "Married" (by rule R_3). However, the missing values in tuples 6 and 10 cannot be predicted accurately. ■

The inference of sensitive fields with the use of correlations is undesirable from a privacy preservation perspective. Therefore, in order to prevent such inference, it may be desirable to also hide some of the non-sensitive entries. The corresponding tradeoff here is that unnecessary hiding of entries loses information for the purpose of data analysis applications. Therefore, it is important to hide a *minimal* set of entries (i.e. a set of minimum size) in order to prevent such privacy violations.

We define the problem of *privacy preservation against adversarial data mining* as that of hiding a minimal set of entries so that the privacy of the sensitive fields are satisfactorily preserved.

Our study is critically different from the currently active studies on privacy-preserving data mining, such as [1, 2, 3, 8, 10, 13]. Privacy preserving data mining tries to transform the data in some way such that a certain types of data mining tasks can still be conducted with guaranteed privacy. Therefore, the focus is on effective mining using partially hidden or distorted data. In privacy preservation against adversarial data mining, we do not aim at any specific data mining tasks. Instead, we generally want to preserve the privacy against any attacks by abuse of data mining. Therefore, the hidden data cannot be recovered by data mining. Our study is also different from [14], which investigates how to hide a set of association rules. In particular, the method in [14] can only hide rules supported by disjoint frequent itemsets. This is done by decreasing their support or confidence, and can hide only a rule at a time.

We make several contributions. First, *we model the problem of privacy preservation against adversarial data mining concisely*. Our model is general and is independent of specific adversarial data mining techniques. *We also point out that finding an optimal solution to the problem of privacy preservation against adversarial data mining is NP-hard.* Second, *we develop an effective and efficient heuristic algorithm* to find practically effective solutions to the problem of privacy preservation against adversarial data mining. Last, *we conduct an extensive empirical evaluation on both real data sets and synthetic data sets to examine the performance of our method.* The results show that our method is effective and efficient in practice.

The remainder of the paper is organized as follows. We formulate the problem in Section 2. In Section 3, we overview the procedure of privacy preservation against adversarial data mining. The details of the proposed approach are developed in Sections 4 and 5. An empirical evaluation is reported in Section 6.

2. PROBLEM STATEMENT

Consider a database T of n records $t_1, \ldots, t_n$ and m attributes $D_1, \ldots, D_m$. Without loss of generality, we assume that the domains of dimensions are exclusive.

The value of record t_i on attribute D_j is denoted by $t_{i,j}$. We also refer to $t_{i,j}$ as an *entry*. We note that an entry is a *specific value in a tuple*, instead of an attribute value appearing in the table. At this moment, we assume that all attributes are categorical. We note that any continuous attribute can be transformed to a categorical domain by using the process of discretization. After the process of discretization, the methods discussed in this paper can be utilized for deciding which entries to remove. At the end of this process, the discretized attributes are replaced by the *mid-points* of the corresponding ranges. The process of discretization results is an *additional* level of approximation of the attribute values. All results in this paper will continue to hold for this discretized continuous case, except that these results need to be expressed in terms of the discretized attributes.

A set of entries in one record is called an *entry-set*. The set of privacy sensitive entries in the data is called the *directly private set*, denoted by P. In the most general case, the sensitivity of the data may be defined not only by the fields, but also by a combination of the data records as well as their attributes. For example, one user may wish to be private about his or her education level, whereas another user may be private about the age attribute. Furthermore, there may be other fields (such as salary) which may be defined as globally sensitive at the administrator level. Therefore, the directly private set is defined at the *entry level* rather than at the *attribute level*.

For the sake of simplicity, we call a value in an attribute an *item*, and a combination of multiple items an *itemset*. Clearly, an item or an itemset can appear in multiple tuples. In other words, an item can match multiple entries and an itemset can match multiple entry-sets. The use of this terminology helps us to leverage on existing machinery for association rules and large itemset generation.

It is further assumed that the information hiding is done at the server end. Thus, while the administrator at the server end is privy to the entire set of records, they do not make the entire data set publicly available. The advantage of information hiding at the server end is that *it is possible to use the inter-attribute correlations among the different records in order to decide which entries should be masked*.

The primary question in the problem of privacy preservation against adversarial data mining is *the choice of entries which should be hidden*. While it is clear that the entries in P should be hidden, it is also important to remove other entries which have predictive power. We use $t_{i,j} = \#$ to denote that the value is hidden. The table in which the privacy sensitive entries are blanked out is denoted by $(T - P)$.

An itemset X is said to *appear* in a tuple t if X matches a set of entries in t. Moreover, for a table T and a directly private set P, X is said to *publicly appear* in a tuple t if the entries in t matching X are not in P (i.e., not blanked out in $(T - P)$). Consider table `Employee` in Table 2. The set of sensitive entries is $P = \{t_{5,\text{Education}}, t_{6,\text{Gender}}, t_{8,\text{Salary-level}}, t_{9,\text{Married-or-not}}, t_{10,\text{Gender}}\}$.

As shown in Example 1, if we publicly publish Table 2, i.e., $(T - P)$, it may not preserve the privacy sufficiently. Therefore, only blanking out the sensitive entries is inadequate. If some of the fields have known correlations to the other fields, they may be used to predict the sensitive entries in the data. In other words, entries which have strong predictive power to any entry in set P need to be removed from the data. Therefore, the first step is to determine the entries which have strong predictive power to entries in set P.

A user may find correlation/association rules from $(T - P)$ and use the rules to predict the values in the blank entries. The problem of *privacy preservation against adversarial data mining* is to find a set of entries K such that predictive methods using only the information in $(T - P - K)$ cannot have an accuracy at least δ to predict the values of any entries in P, where $\delta \in (0, 1]$ is a user-specified parameter. To make the information loss as little as possible, we want to minimize the size of K. We will refer to the set K as the *derived private set*.

One way to select entries for the derived private set is to use those entries to invalidate confident rules.

EXAMPLE 2 (RULE INVALIDATION). Consider rule R_2: *Manager* $\wedge$ *SL-5* $\rightarrow$ *University* discussed in Example 1. In order to in-

validate this rule, we can blank some occurrences of "Manager", "SL-5" and/or "University" in tuples 5 and 7. Note that we *do not* have to blank *all* those occurrences. For example, if $t_{1,\text{Education}} =$ University is removed, then the confidence of the rule R_2: *Manager* $\wedge$ *SL-5* $\rightarrow$ *University* is lowered down to 50%.

Moreover, we note that removing $t_{1,\text{Title}} =$ Manager is more advantageous than removing $t_{1,\text{Education}} =$ University since $t_{1,\text{Title}}$ affects two rules R_2 and R_3 rather than just one.

If the minimum support of a rule is 2 and the confidence threshold is 60%, then by blanking out only the entry $t_{1,\text{Title}}$, we can hide both rules R_2 and R_3. ∎

The process of removing derived private entries in order to reduce the confidence level of the rules in the data is called *rule invalidation*.

A second way of protecting the sensitive entries is to *prevent rules from being fired* by blanking out the entries in the sensitive records corresponding to the antecedents of the rules. In such a case, even though the rules may continue to have high relevance (confidence level), the entries in the antecedents of the rules may get blanked out for the sensitive records only. This process is referred to as *rule marginalization*.

EXAMPLE 3 (RULE MARGINALIZATION). Consider rule R_1: *Assistant* $\rightarrow$ *SL-3* in Example 1. In Table 2, only sensitive entry $t_{8,\text{Salary-level}}$ can be predicted using this rule. Therefore, instead of invalidating the rule, we can simply blank out entry $t_{8,\text{Title}} =$ Assistant. Then, sensitive entry $t_{8,\text{Salary-level}}$ cannot be predicted accurately. ∎

Rule marginalization refers to the fact that the rules may continue to have high confidence level, but get marginalized because they are no longer relevant to any sensitive entry in the data. Therefore, the predictive power of the rule is effectively removed. Rule marginalization is especially useful when only a small number of users have chosen to keep their records private for a particular column. In such a case, it is possible to block a small number of antecedents from the rules in order to keep the entries private.

Generally, it is a tricky question to determine whether it is more useful to remove entries in order to prevent rules from being fired or whether it is more useful to aim for reducing the confidence level of rules. Moreover, blanking out an entry may simultaneously prevent some rules from firing and reduce the confidence level of some other rules. In a later section, we will explore this tradeoff, and discuss an effective strategy for balancing rule invalidation and marginalization.

We note that the predictive nature of the problem has some similarity to the problem of adversarial classification [7], but with some critical differences. First, while the adversarial classification problem concentrates on the prediction of a single field, *the problem of privacy preservation against adversarial data mining studied here may involve prediction of any sensitive field in the data*. This makes the problem much more general and more difficult to solve than a standard classification problem. At the same time, it is also more difficult for an adversary who may need to be able to make prediction on multiple fields in the data from an incomplete data set. Second, while the adversarial classification problem concentrates on the method of data perturbation as an adversarial measure, *the problem of privacy preservation against adversarial data mining concentrates on the issue of information hiding as a measure to thwart privacy attacks*.

In this paper, we utilize association rules to construct the model which decides the entries to be hidden. This is because association based methods are easy to generalize to the case where the prediction may be performed over any fields in the data. Furthermore, such methods are relatively robust for a large number of applications in which the data records have high dimensionality. In such cases, there exist an exponential number of subspaces that such a predictor can explore over the data. The related work in [15] compares the effects of different schemes for creating privacy leakage in the context of learning based systems. This is orthogonal to the goal of our paper, which uses attribute suppression in order to preserve privacy in a more general setting.

3. OVERVIEW OF THE APPROACH

We want to determine association rules with strong predictive power. However, in this case it is more important to find association rules with a high confidence level than those with a high support level. In fact, it is important to be exhaustive in the association rule generation process in order to ensure that none of the sensitive fields are divulged. The overall algorithm for adversarial data suppression is in the following two steps:

Step 1: Mining adversarial rules. We mine all association rules from $(T - P)$ in the form of $X \rightarrow y$ where X is a set of attribute values and y is a value on an attribute Y such that (1) the confidence of the rule is no less than δ in $(T - P)$; and (2) for tuples t where X publicly appears and the value of t on attribute Y is blanked out, t has value y on Y with a probability of at least δ. Such confident association rules related to the adversarial data mining are called *adversarial rules*.

Step 2: Determining derived private set. We select a set of entries $K \subseteq (T - P)$ such that by deleting the entries corresponding to $P \cup K$, no adversarial rules can be fired to predict any value in P accurately. We note that the process of blanking out entries in K reduces the amount of information available in the data. This is an unfortunate consequence of the privacy preservation process. These effects need to be minimized. Therefore, we would like to minimize the cardinality of the derived private set K.

We establish the correctness of the above algorithm as follows. For a table T and a directly private set P, consider a rule $R : X \rightarrow y$, where X is an itemset and y is a value on attribute Y. All tuples t in which X publicly appears and the attribute value on Y is not blanked out form the *public set* of R, denoted by $pub(R)$. That is, $pub(R) = \{t | t \in T, X \text{ publicly appears in } t, t.Y \notin P\}$. A user can derive the confidence of R from its public set. The confidence of R in the public set is called the *public confidence* of R.

On the other hand, X may publicly appear in some tuples in T but the Y attribute values in those tuples are blanked out. Then, R can be used to predict the value of Y in those tuples. Such tuples are called the *hidden set* of R, denoted by $hid(R)$. That is, $hid(R) = \{t | t \in T, X \text{ publicly appears in } t, t.Y \in P\}$. The confidence of R in the hidden set is called the *hidden confidence* of R. Clearly, an external user can only calculate the public confidence from $(T - P)$. Since we assume that information hiding is done at the server end, only the owner/administrator of the original data can calculate the hidden confidence.

Generally, let δ be a user specified confidence threshold. For a table T and a directly private set P, an *adversarial rule* (with respect to δ) is an association rule $R : X \rightarrow y$ such that both the public and hidden confidence of R are at least δ.

An adversarial rule is potentially revealing when it has high public confidence and high hidden confidence. Otherwise, it cannot be used to predict accurately. That is, if a rule has high public confidence but low hidden confidence, then it cannot accurately predict the blanked entries. On the other hand, if a rule has high hidden confidence but low public confidence, then a user who can only read $(T - P)$ cannot identify the rule from the public data.

THEOREM 1 (CORRECTNESS). *For a table T and a directly private set P, any entry $z \in P$ on attribute Y cannot be predicted with a confidence δ or higher using only the information from $(T - P)$ if and only if there exists no adversarial rule $R : X \rightarrow y$ in $(T - P)$ such that y is on attribute Y and X publicly appears in*

the tuple t containing z. ■

4. MINING ALL ADVERSARIAL RULES

We will mine all adversarial rules in a depth-first search framework. The complete set of itemsets can be enumerated using a set enumeration tree [11]. A set enumeration tree employs a total order on the set of all items. Each itemset is treated as a string such that all items in the itemset are sorted in the total order. Then, an itemset X is an ancestor of another itemset Y in the set enumeration tree if X is a prefix of Y.

The general idea of our algorithm to mine the complete set of adversarial rules is to conduct a depth-first search of the set enumeration tree. The nodes in the tree are the itemsets publicly appearing in some tuples. We check whether such itemsets can be an antecedent of some adversarial rules.

Clearly, for any adversarial rule $R : X \rightarrow y$, X must publicly appear in some tuples. Hence, the search of adversarial rules is complete if all the itemsets publicly appearing in some tuples can be enumerated completely. The correctness and the completeness of the mining of adversarial rules by a depth-first search of a set enumeration tree immediately follow the following property.

THEOREM 2 (ANTI-MONOTONICITY). *If an itemset X does not publicly appear in any tuple, then any superset of X cannot publicly appear in any tuple.* ■

Often, a complete set enumeration tree can be huge in real applications, when there are hundreds or even thousands of items. Therefore, it is important to prune significant portions of the set enumeration tree.

First of all, we can prune all those itemsets that do not publicly appear in a data set.

PRUNING RULE 1. (PRUNING NOT PUBLICLY APPEARED ITEMSETS) *In the depth-first search of the set enumeration tree, an itemset X which does not publicly appear in any tuple, as well as all the supersets of X can be pruned.* ■

Not every itemset publicly appearing in some tuples is an antecedent of some adversarial rule. This fact can be used to improve the effectiveness of the depth-first search algorithm. This means that if we can determine that all nodes in a subtree cannot be antecedents of any adversarial rule, then the subtree can be pruned and the search space is narrowed. We can examine whether an itemset and its supersets are antecedents of some adversarial rules from the *projected databases*.

Let X be an itemset, T be a table, and P be a directly private set. For a tuple t in T, if X publicly appears in t, then the *projection* of t with respect to X, denoted by $t|_X$, is the set of entries in t that are not matched by X. If X does not publicly appear in t, then $t|_X = \emptyset$.

The *projected database* with respect to X is the set of nonempty projections with respect to X in T.

The concept of projected databases and its utilization are illustrated in the following example.

EXAMPLE 4 (PROJECTED DATABASE). Consider the example illustrated in Table 2. The projected database for itemset $X_1 = \{\text{College}\}$ is $T|_{X_1} = \{$(Assistant, Female, Unmarried, SL-3), (Assistant, Male, Married, SL-3)$\}$. The projected database for itemset $X_2 = \{\text{Accountant}\}$ is $T|_{X_2} = \{$(#, Unmarried, MBA, SL-5), (#, Married, University, SL-4)$\}$. The projected database for itemset $X_3 = \{\text{Unmarried, SL-5}\}$ is $T|_{X_3} = \{$(Accountant, #, MBA), (Manager, Female, University)$\}$. The projected database for itemset $X_4 = \{\text{Manager, Female}\}$ is $T|_{X_4} = \{$(Married, University, SL-5), (Married, Ph.D., SL-5), (Unmarried, University, SL-5), (#, MBA, SL-7)$\}$. ■

The itemsets can be divided into five categories according to the projected databases of the itemsets. The categorization of itemsets is very useful from the perspective of rule pruning. For some categories of itemsets, it is possible to design efficient pruning rules by using specific properties of those categories.

An itemset is *privacy-free* if its projected database does not contain any directly private entry at all.

EXAMPLE 5 (PRIVACY-FREE ITEMSETS). Consider itemset $X_1 = \{\text{College}\}$ from Example 4. Since its projected database contains no directly private entry, X_1 cannot be the antecedent of any adversarial rule. Moreover, any superset of X_1, such as $\{\text{Assistant, College}\}$ and $\{\text{Assistant, Male, College}\}$, cannot be the antecedent of any adversarial rule, either. ■

PRUNING RULE 2 (PRUNING PRIVACY-FREE ITEMSETS). *In the depth-first search of the set enumeration tree, a privacy-free itemset and all supersets of it can be pruned.* ■

An itemset X is *non-discriminative* if every tuple in the projected database of X contains directly private entries in the same attribute(s).

EXAMPLE 6 (NON-DISCRIMINATIVE ITEMSETS). Consider itemset X_2 in Example 4. Every tuple in $T|_{X_2}$ has a blanked value in attribute *Gender*. Therefore, any association rule having X_2 or any superset of X_2 cannot make an accurate prediction of the gender of accountants. In other words, those itemsets cannot be the antecedents of any adversarial rules with respect to gender. ■

PRUNING RULE 3. (PRUNING NON-DISCRIMINATIVE ITEMSETS) *In the depth-first search of the set enumeration tree, if an itemset X is non-discriminative with respect to Y, then any superset of X is also non-discriminative with respect to Y. Y can be pruned from the projected databases of X and any superset of X. Moreover, if an itemset X is non-discriminative with respect to all other attributes that contain some entries in the directly private set, then X and its supersets can be pruned.* ■

An itemset X is said to be a *contrast itemset* if for any entry $y \in P$ such that $X \cup y$ appears in some tuples in T, $X \rightarrow y$ has a public confidence of 0. Clearly, X as well as any supersets of X cannot be used to accurately predict y.

EXAMPLE 7 (CONTRAST ITEMSETS). Consider itemset X_3 in Example 4. In the projected database, there is one blanked entry in attribute *Gender*, whose value is "Female". However, in the public set of X_3, rule "$X_3 \rightarrow$ Male" has a public confidence of 100%. Thus, X_3 or any of its supersets cannot be used to predict the gender accurately. ■

PRUNING RULE 4 (PRUNING CONTRAST ITEMSETS). *In the depth-first search of the set enumeration tree, a contrast itemset and all supersets of it can be pruned.* ■

An itemset X is *discriminative* if X is the antecedent of some adversarial rules. This can be determined by checking the projected database of X. Technically, if there is a value y such that $X \rightarrow y$ has public and hidden confidence of at least δ with respect to the projected database of X, then X is discriminative.

EXAMPLE 8 (DISCRIMINATIVE ITEMSETS). Consider itemset X_4 in Example 4. In the projected database, there is one blanked entry in attribute *Married-or-not*, whose value is "Married". Moreover, two out of the three tuples in the projected database have non-blanked value "Married" in attribute *Married-or-not*. Thus, we can generate a confident association rule *Manager* $\wedge$ *Female* $\rightarrow$ *Married*, which is the adversarial rule R_3 discussed in Example 1. ■

Input: a table T, a directly private set P, a confidence threshold δ, and an optional minimum support threshold γ (default $\gamma = 1$)
Output: the complete set of adversarial rules;
Method:
1: let $A_1, \ldots, A_n$ be an order of attributes, extend the order to an order over all items: item x in A_i is after item y in A_j if $i \leq j$; items from the same attributes are sorted alphabetically;
2: conduct a depth-first search on the set enumeration tree of itemsets, for each itemset X (i.e., at each node of the tree);
3: if the support of X is less than γ then return;
4: create the projected database for X;
5: // applying Pruning Rules 2, 3 and 4
if X is privacy-free, non-discriminative or contrast then return;
6: if X is discriminative then output an adversarial rule;
7: remove unpredictable attributes with respect to X;
8: for each item z appearing in the projected database in any attribute that has not been considered yet, form itemset $X \cup \{z\}$ and recursively call the depth-first search procedure.

Figure 1: Algorithm *FAiR*.

Input: a table T, a directly private set P, a confidence threshold δ, and a set $\mathcal{R}$ of adversarial rules
Output: a set of derived entry sets;
Method:
1: set $c_{i,j} = 0$ for all entries;
2: **while** rule set $\mathcal{R}$ is not empty **do**
3: for each rule $r \in \mathcal{R}$ and entry $t_{i,j} \in T$ **do**
4: compute the contribution of blanking $t_{i,j}$ to invalidation or marginalization of rule R;
5: add the contribution to $c_{i,j}$;
6: blank a fraction f of the original number of entries with the largest weight;
7: remove the rules invalidated or marginalized;

Figure 2: Algorithm *GraDeS*.

We note that an itemset may be the antecedent of more than one adversarial rule. However, all adversarial rules having the same antecedent can be generated by only one scan of the projected database. We only have to maintain a set of counters which track the number of occurrences of different attribute values in their public and hidden sets respectively. During the scan the counters can be updated by examining each record sequentially.

The supersets of discriminative itemsets should still be checked. This is because we may find confident adversarial rules among these supersets. Note that we have to either invalidate or marginalize *all* adversarial rules.

An itemset is *undetermined* if it is not in any of the previous four categories. For such an itemset, we can neither prune it, nor generate adversarial rules. Therefore, the depth-first search needs to be continued at such nodes in order to make judgements about the itemsets in the corresponding subtrees.

The efficiency of the depth-first search method can be improved further by utilizing the following observation. *If the support of an adversarial rule is very low, then the adversarial rule can be statistically insignificant*. In a real application, a user may often specify a minimum support threshold. We only determine adversarial rules whose support is at least equal to this threshold. Therefore, any itemset whose support is less than this threshold can be pruned, and so can its supersets. The corresponding algorithm *FAiR* (Finding Adversarial Rules) is illustrated in Figure 1.

5. DETERMINING OPTIMAL DERIVED PRIVATE SET

We can use the set of adversarial rules to determine the set of entries which need to be removed from the data. Unfortunately, the problem of finding the smallest derived private set is NP-hard (limited by space, we omit the formal result here). Thus, we need to design a heuristic algorithm to find practically effective solutions.

We need to find a good tradeoff between rule invalidation and rule marginalization. we can construct an effective solution for the task by quantifying the level of information revealed by the different entries. The greater the level of information revealed by an entry, the greater the weight of the corresponding entry. This weight is denoted by $c_{i,j}$ for entry $t_{i,j}$.

Initially, we set $c_{i,j} = 0$ for all the entries. Let us consider a database containing N entries. We note that when an entry is deleted, it could either prevent a rule from being *fired* because of rule marginalization, or it could prevent a rule from being *found* because of rule invalidation.

Consider an entry $t_{i,j}$ and a rule $R : X \rightarrow y$ where $y \in Y$. Three cases may arise: tuple t_i (the tuple containing entry $t_{i,j}$) is in the public set $pub(R)$, t_i is in the hidden set $hid(R)$, or t_i is irrelevant to R. We will compute the contribution of blanking out the entry $t_{i,j}$ in each case as follows.

First, if tuple t_i is in $pub(R)$, i.e., $X \cup \{y\}$ publicly appears in t_i, then blanking out $t_{i,j}$ will reduce either the public confidence of R (when $t_{i,j} \in Y$) or the size of the public set of R (when $t_{i,j}$ matches an item in X) by $1/|pub(R)|$. This is the contribution of blanking $t_{i,j}$ to the invalidation of R.

Second, if tuple t_i is in $hid(R)$, i.e., X publicly appears in t_i but $t_i.Y \in P$ (t_i has a blanked value on the attribute that y may appear), then blanking out $t_{i,j}$ marginalizes R in one instance (i.e., this tuple). In order to fully marginalize all instances of R, we need to blank out a total of $|hid(R)|$ entries. Thus, the contribution of blanking out $t_{i,j}$ to the marginalization of R is $1/|hid(R)|$.

Last, if t_i is not in $pub(R)$ or $hid(R)$, then blanking out $t_{i,j}$ does not make any contribution to the invalidation or marginalization of R.

Therefore, the weight $c_{i,j}$ can be calculated as the sum of contributions of blanking out $t_{i,j}$ to all adversarial rules. The entries with the highest weights should be blanked out. Once an entry is blanked out, the weights of other entries should be adjusted. The blanking process can be conducted iteratively.

The process of blanking the entries can turn out to be cumbersome if the weights need to be re-computed at each step. Therefore, we batch the process of blanking out the entries after computing the corresponding weights. The process of computing weights of the different entries is done using a single pass in which all rules are iterated over the entries of a given record in order to determine the corresponding weights. The weights are then stored, and then the batch process of blanking out is started. In the process of batching the blanking of entries, we blank out k sensitive entries (i.e., the entries with $c_{i,j} > 0$) in each pass. This ensures that a maximum of at most m/k passes need to be performed on the data, where m is the total number of sensitive entries. We note that the process of batching leads to some reduction in accuracy, but this is a natural tradeoff with the efficiency of the entire process. The parameter k is called the *blanking factor*, and can correspondingly be tuned in order to achieve the desired tradeoff between accuracy and efficiency. The algorithm *GraDeS* (for Generating Derived Set) is shown in Figure 2.

6. EMPIRICAL EVALUATION

All algorithms were implemented in Microsoft Visual C++ V6.0. All experiments were run on an IBM ThinkPad T42 laptop computer which has one Intel Pentium M 1.5 GHz processor and 768 M main memory, and runs Microsoft Windows XP operating system. We used both synthetic and real data sets in our experiments.

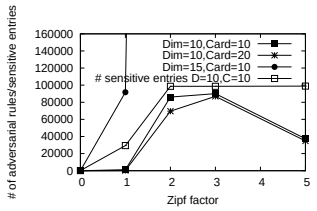

Figure 3: Zipf factor vs. number of adversarial rules.

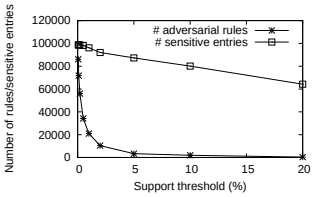

Figure 4: # of adversarial rules/ sensitive entries vs. support.

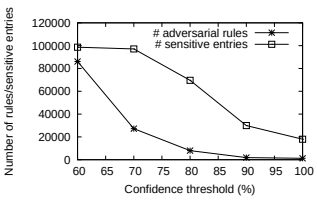

Figure 5: # of adversarial rules/ sensitive entries vs. confidence.

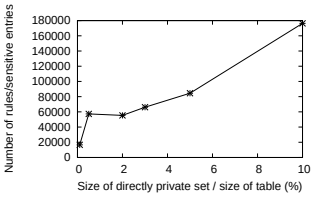

Figure 6: # of adversarial rules vs. directly private set size.

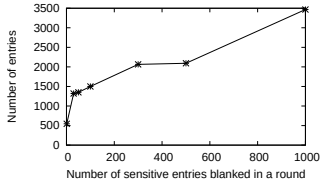

Figure 7: Derived private set size vs. blanking factor.

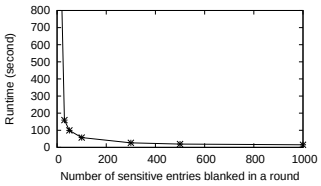

Figure 8: Runtime vs. blanking factor.

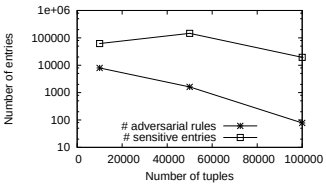

Figure 9: # of adversarial rules/ sensitive entries vs. db size.

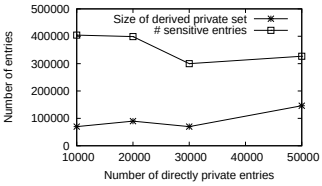

Figure 10: Derived private set and sensitive entries (Adult)

6.1 Results on Synthetic Data Sets

We generated synthetic data sets using the Zipf distribution to determine the tuple values on each dimension. The data generator uses the following parameters (default values in brackets): (1) dimensionality (10); (2) cardinality (10); (3) Zipf factor (2.0); and (4) number of tuples in the table ($10,000$). We generated the directly private sets so that a $p\%$ of randomly selected entries are hidden. The parameter p is chosen to be 1% by default. This means that a table containing $10,000$ tuples and 10 dimensions will contain $1,000$ entries in the directly private set.

Figure 3 shows the variation in the number of adversarial rules with varying Zipf factor for different dimensionalities and cardinalities of the data set. The number of tuples was set at $10,000$, the confidence threshold was set at 60%, the minimum support threshold was set at 0.05%, and the directly private set randomly hid 1% of entries in the table. We tested different cases of data set dimensionality and cardinality. The data becomes much more correlated when the Zipf factor increases. When the Zipf factor is in the range of 2 to 3, the number of adversarial rules is large. When the Zipf factor keeps increasing to 5, the number of adversarial rules decreases. The reason is that when Zipf factor is large, there are some strong rules with high support across many tuples. However, there are not many such rules. In the same figure, we also show the number of sensitive entries (the entries not in directly private set, but which are relevant to at least one adversarial rule). It goes up as the Zipf factor increases, but is bounded by the total number of non-blanked entries in the table. Interestingly, the number of sensitive entries does not drop with high Zipf factor, whereas the number of adversarial rules decreases. In such situations, the support of rules increases and the number of tuples affected by the adversarial rules remains stable. This helps in containing the final number of values in the derived private set.

The major observation is that the number of rules increases with dimensionality, but reduces with increasing cardinality on each categorical attribute. This is because increasing cardinality makes the data set more diverse whereas increasing dimensionality increases the number of possibilities for finding adversarial rules.

In Figure 4, we tested the effect of support threshold on the number of adversarial rules and the number of sensitive entries. The parameters of the data set were set to default, and the confidence threshold was set to 60%. As can be seen, the number of adversarial rules increases exponentially as the support threshold goes down, which is similar to the well-known effect in frequent pattern mining. However, the number of sensitive entries changes linearly, since the Zipf distribution embeds some correlations with high support, which affects many tuples. The number of entries sensitive to adversarial rules with low support is limited. This is an encouraging observation since a reduced number of sensitive entries would indicate a modest size of the derived private set. We will examine this issue in more detail in the next section.

The effect of confidence threshold on the number of adversarial rules and the number of sensitive entries was tested using the same synthetic data set, as shown Figure 5. The support threshold was set to 0.05%. As expected, a lower confidence threshold introduces more adversarial rules and sensitive entries. An interesting observation is that the data set following a Zipf distribution has a non-trivial number of adversarial rules of 100% confidence, which affect about 20% of the tuples in the table. We note that the corresponding entries are very valuable from the point of view of an adversary.

Using the same synthetic data set, we tested the effect of size of directly private set on the number of adversarial rules, as shown in Figure 6. The support threshold and confidence threshold were fixed to 0.05% and 60%, respectively. When the directly private set is small, the number of adversarial rules is also small since the rules must be associated with some directly private entries. When the directly private set becomes large, the number of adversarial rules increases linearly. The relatively modest increase in the number of adversarial rules with increasing direct private set size is an encouraging observation, since it tends to indicate that the derived private set is also likely to increase modestly.

The cost of our approach consists of two parts: mining the adversarial rules and computing the derived private set. The first step is quite efficient. In our experiments, it takes less than 5 seconds for data sets with $10,000$ tuples and is linearly scalable with respect to database size. This is only a very minor component of the total cost. Limited by space, we omit these details and concentrate on the more computationally challenging issue of finding the derived private set from the rules. In our heuristic approach (Section 5), the blanking factor can be leveraged as a useful parameter to control the tradeoff between the size of derived private set and efficiency.

Figure 7 plots the size of derived private set with respect to the blanking factor. As a reference, the total number of sensitive entries in this test is $61,964$. The support threshold and the confidence threshold were set to 0.1% and 80%, respectively. The corresponding running time is shown in Figure 8. We note that the running time is quite modest for most practical settings.

From Figures 7 and 8, we observe the following. First, the privacy can be preserved by blanking out only a very small subset of sensitive entries. In the setting of this experiment, if we blank one

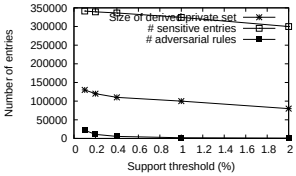

Figure 11: # of sensitive entries/adverserial rules/derived private set size vs. support (Adult).

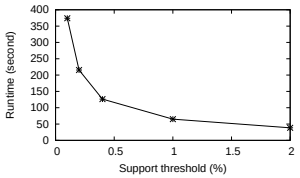

Figure 12: Runtime vs. support (Adult).

entry at a time, we only need to blank 545 entries in the derived private set, which is less than 1% of the set of all sensitive entries. Blanking out less than 600 entries to preserve the privacy of $1,000$ directly private entries allows the approach to remain practical from the point of view of information preservation.

In general, the less entries we blank out in a round, the smaller the derived private set we get. On the other hand, it increases the running time. We show that the tradeoff is such that a minor increase of the derived private set can lead to a substantial gain in efficiency. For example, by increasing the blanking factor from 1 to 30 the runtime was drastically reduced from $1,988$ seconds to 157 seconds, while the derived private set size only modestly increased from 546 entries to $1,320$ entries. The iterative use of a blanking factor helps in substantially improving the efficiency of the algorithm with only a modest loss in the data entries.

We tested the scalability of privacy preservation against adversarial data mining with respect to the number of tuples in the table. The results are shown in Figure 9. The data sets took the default parameters except that the number of tuples ranged from $10,000$ to $100,000$. The directly private set randomly hid 1% entries in the table. We fixed the confidence and support threshold to 80% and 0.05%, respectively. Interestingly, the number of adversarial rules decreases as the size of table increases. In a data set following the Zipf distribution, as the database size goes up, the rules about the bias values in dimensions gain support much more than the other rules. Since we kept the support threshold constant in percentage, those rules whose support do not grow in the same rate became infrequent. This resulted in fewer rules in large table of high support. This phenomenon matches the scenarios in real applications. In small tables, we can often observe many occasional correlations. However, in large tables, only the strong correlations become statistically meaningful and affect the privacy preservation against adversarial data mining.

The number of sensitive entries relies on two factors: the number of adversarial rules and the size of directly private set. We observed from our experiments that the runtime of our method is mainly proportional to the number of sensitive entries, since the determination of derived private set dominates the cost. Since the number of sensitive entries show more modest scalability behavior, this also helps to contain the running time of our method.

6.2 Experiments on Real Data Set (Adult)

To examine the effectiveness of our approach in real applications, here we report the experimental results on real data set *Adult* from the UCI Machine Learning Repository (http://www.ics.uci.edu/∼mlearn/). It was extracted from the census bureau database in year 1994. It has $48,842$ tuples and 14 dimensions, 6 of which are continuous and 8 are nominal. We removed tuples containing missing values. After removal, the data set still has $45,222$ tuples. We also removed 4 attributes in which most tuples have the same value. We discretized continuous attributes *ages*, *fnlwgt* and *education-num*. After discretization, those attributes have cardinality 10, 150 and 17, respectively.

Figure 10 shows the number of sensitive entries and size of derived private set as the size of directly private set changes from $10,000$ to $50,000$. The support threshold and the confidence threshold were set to 2% and 80%, respectively. The blanking factor was fixed to $10,000$. The number of sensitive entries is not very sensitive to the change of private set, since it is bounded by the total number of non-blanked entries. On the other hand, the number of derived private entries increases with the size of directly private set. If we want to hide more entries, we likely have to blank out more entries as well. However, when we hide more entries in the directly private set, some rules may also be hidden. Consequently, one interesting observation from Figure 10 is that the number of derived private entries increases quite slowly with increasing number of directly private entries. This tends to indicate that the (proportionate) loss in entries with increasing level of incompleteness of the data set is likely to be lower. The running time with increasing size of the directly private set roughly follows a similar trend in Figure 10. Limited by space, we omit the curves.

Figure 11 shows the numbers of adversarial rules and sensitive entries, as well as the size of derived private set on the support threshold. The directly private set has $30,000$ entries and the confidence threshold was set to 80%. The blanking factor was set to $10,000$. All three measures go up as the support threshold goes down. They follow similar trends. The running time is shown in Figure 12. Again, we observe that the size of the derived private set is within a small factor of the directly private set over all ranges of the support parameter. This tends to indicate only a modest level of information loss. In fact, these results show that the derived private set does not change very much for different values of user-specified support.

In summary, the empirical study on both synthetic and real data sets strongly suggests that privacy preservation against adversarial data mining is practical from an information-loss perspective. This is because the results seem to indicate that the derived private set does not increase as fast as the directly private set, and tends to be quite stable over a wide range of user parameters. At the same time, our heuristic approach is also efficient from a computational perspective and requires a few seconds over many practical settings on data sets containing thousands of records. In addition, the scheme scales modestly over a wide range of user-specified parameters. This contributes to the practicality and usability of the approach.

7. REFERENCES

[1] Aggarwal C. C. and Yu P. S. A Condensation Based Approach to Privacy Preserving Data Mining. *EDBT Conference*, 2004.

[2] Agrawal R. and Srikant R. Privacy Preserving Data Mining. *Proceedings of the ACM SIGMOD Conference*, 2000.

[3] Agrawal D. and Aggarwal C. C. On the Design and Quantification of Privacy Preserving Data Mining Algorithms. *ACM PODS Conference*, 2002.

[4] Agrawal R. and Bayardo R. J. Data Privacy through Optimal k-anonymization. *ICDE Conference*, 2005.

[5] Aggarwal C. and Parthasarathy S. Mining Massively Incomplete Data Sets by Conceptual Reconstruction. *ACM KDD Conference*, 2001.

[6] Clifton C. and Marks D. Security and Privacy Implications of Data Mining. *ACM SIGMOD DMKD Workshop*, 1996.

[7] Dalvi N. et al. Adversarial classification. KDD Conference, pp. 99-108 , 2004

[8] Evfimievski A. et al. Privacy Preserving Mining Of Association Rules. *ACM KDD Conference*, 2002.

[9] Liew C. K. et al. A data distortion by probability distribution. *ACM TODS*, 10(3):395-411, 1985.

[10] Rizvi S. and Haritsa J. Maintaining Data Privacy in Association Rule Mining. *VLDB Conference*, 2002.

[11] Rymon R. Search through systematic set enumeration. In *Proceedings of KR'92*, 1992.

[12] Samarati P. and Sweeney L. Protecting Privacy when Disclosing Information: *k*-Anonymity and its Enforcement Through Generalization and Suppression. *Proc. of the IEEE Symposium on Research in Security and Privacy*, May 1998.

[13] Vaidya J. and Clifton C. Privacy Preserving Association Rule Mining in Vertically Partitioned Data. *ACM KDD Conference*, 2002.

[14] Verykios V. S. et al. Association Rule Hiding, *IEEE TKDE*, 16(4), 2004.

[15] Xiong H. et al. Privacy Leakage in Multi-relational Databases via Pattern based Semi-Supervised Learning. *Univ. of Minnesotta, Technical Report 04-23*, 2004.