CERIAS Tech Report 2008-29

Injector: Mining Background Knowledge for Data Anonymization

by Tiancheng Li; Ninghui Li
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

Injector: Mining Background Knowledge for Data Anonymization

Tiancheng Li, Ninghui Li
Department of Computer Science, Purdue University
305 N. University Street, West Lafayette, IN 47907, USA
{li83,ninghui}@cs.purdue.edu

***Abstract*— Existing work on privacy-preserving data publishing cannot satisfactorily prevent an adversary with background knowledge from learning important sensitive information. The main challenge lies in modeling the adversary's background knowledge. We propose a novel approach to deal with such attacks. In this approach, one first mines knowledge from the data to be released and then uses the mining results as the background knowledge when anonymizing the data. The rationale of our approach is that if certain facts or background knowledge exist, they should manifest themselves in the data and we should be able to find them using data mining techniques. One intriguing aspect of our approach is that one can argue that it improves both privacy and utility at the same time, as it both protects against background knowledge attacks and better preserves the features in the data. We then present the Injector framework for data anonymization. Injector mines negative association rules from the data to be released and uses them in the anonymization process. We also develop an efficient anonymization algorithm to compute the injected tables that incorporates background knowledge. Experimental results show that Injector reduces privacy risks against background knowledge attacks while improving data utility.**

I. Introduction

Agencies and other organizations often need to publish microdata, e.g., medical data or census data, for research and other purposes. Typically, such data is stored in a table, and each record (row) corresponds to one individual. Each record has a number of attributes, which can be divided into the following three categories. (1) Attributes that clearly identify individuals. These are known as *explicit identifiers* and include *Social Security Number*, *Address*, *Name*, and so on. (2) Attributes whose values can be known from other sources, e.g., publicly-available databases such as a voter registration list. These attributes when taken together can potentially identify an individual; thus they are known as *quasi-identifiers*. These may include, e.g., *Zip-code*, *Birth-date*, and *Gender*. (3) Attributes that are considered sensitive, such as *Disease* and *Salary*.

While releasing microdata gives useful information to researchers, it presents disclosure risk to the individuals whose data are in the table. Two types of information disclosure have been identified in the literature [1], [2]: *identity disclosure* and *attribute disclosure*. Identity disclosure occurs when an individual is linked to a particular record in the released table. Attribute disclosure occurs when new information about some individuals is revealed, i.e., the released data makes it possible to infer the characteristics of an individual more accurately than it would be possible before the data release. Identity disclosure often leads to attribute disclosure. Once there is identity disclosure, an individual is re-identified and the corresponding sensitive values are revealed. Attribute disclosure can occur with or without identity disclosure.

To limit disclosure risk, Samarati and Sweeney [3], [4], [5] introduced the *k-anonymity* privacy requirement, which requires each record in an anonymized table to be indistinguishable with at least *k-1* other records within the dataset, with respect to a set of quasi-identifiers. We say that these records form an *equivalence class*. To achieve k-anonymity, Samarati and Sweeney used both generalization and tuple suppression for data anonymization [3], [4]. *Generalization* replaces a value with a "less-specific but semantically consistent" value and *tuple suppression* removes an entire record from the table.

The k-anonymity property aims at protecting against identity disclosure, and does not provide sufficient protection against attribute disclosure. This has been recognized by several authors, e.g., [6], [7], [8]. Machanavajjhala et al. [6] identified two attacks on k-anonymity. In the *homogeneity attack*, while an equivalence class may contain at least k records, it is possible that all records have the same sensitive attribute value, enabling the adversary to infer the sensitive attribute value of individuals in the equivalence class. Even if not all records have the same value, probabilistic inference is still possible. In the *background knowledge attack*, the adversary may possess background information that enables her to eliminate some values from the set of sensitive attribute values in an equivalence class and then infer the sensitive value with high precision. The background knowledge that an adversary has may be some known facts, e.g., a male patient cannot have *ovarian cancer*, or some public demographical information about a specific group, e.g., it is unlikely that a young patient of certain ethnic groups has *heart disease*. A powerful adversary with these additional information can make more precise inference on the sensitive values of individuals.

To address the limitations of k-anonymity, Machanavajjhala et al. [6] introduced a new notion of privacy, called ℓ-diversity, which requires that each equivalence class has at least ℓ "well-represented" values. To achieve ℓ-diversity, anonymization approaches other than generalization have been proposed. Recently Xiao and Tao [9] introduced the *Anatomy* technique to achieve ℓ-diversity.

	ZIP Code	Age	Sex	Disease
1	47677	29	F	Ovarian Cancer
2	47602	22	F	Ovarian Cancer
3	47678	27	M	Prostate Cancer
4	47905	43	M	Flu
5	47909	52	F	Heart Disease
6	47906	47	M	Heart Disease
7	47605	30	M	Heart Disease
8	47673	36	M	Flu
9	47607	32	M	Flu

TABLE I
ORIGINAL PATIENTS TABLE

While ℓ-diversity directly prevents the homogeneity attack, it does not satisfactorily deal with the background knowledge attack. We illustrate background knowledge attack in Section II and show that the anatomizing algorithm proposed in [9] is especially vulnerable to this attack. We also empirically evaluate the vulnerability of the anatomizing algorithm in Section VI.

The main challenge of dealing with background-knowledge-based attack lies in how to model the adversary's background knowledge. We believe that it is unreasonable to require manual specification of the background knowledge an adversary may have. In this paper, we propose a novel approach to model the adversary's background knowledge. Our approach is to generate such knowledge by mining the data to be released. The rationale of our approach is that if certain facts or knowledge exist, they should manifest themselves in the whole table and we should be able to find them using data mining techniques. We then use the mined knowledge in the data anonymization process. One intriguing aspect about our approach is that one can argue that it improves both privacy and utility at the same time, as it both protects against background knowledge attacks and better preserves features in the data.

Based on this rationale, we propose the *Injector* framework for data anonymization. *Injector* first mines negative association rules from the data using data mining techniques and then uses them in the data anonymization process. A negative association rule is an implication saying that some combination of the quasi-identifier values cannot entail some sensitive attribute values. We also develop an efficient anonymization algorithm to incorporate these negative association rules. Finally, we show the effectiveness of our approach in both protecting privacy and improving data utility through experiments on a real dataset.

While *Injector* uses only negative association rules as the adversary's background knowledge, using other types of background knowledge is possible but is beyond the scope of this paper. We provide more discussion of using other types of background knowledge in Section VIII.

The rest of this paper is organized as follows. We illustrate the background knowledge attack against *Anatomy* in Section II and present the *Injector* methodology in Section III. We discuss negative association rule mining in Section IV and show how to use negative association rules in data anonymization in Section V. Experimental results are presented in Section VI and related work is discussed in Section VII. In Section VIII, we discuss limitations of our approach and avenues for future research.

	ZIP Code	Age	Sex	Group-ID
1	47677	29	F	1
2	47602	22	F	1
3	47678	27	M	1
4	47905	43	M	2
5	47909	52	F	2
6	47906	47	M	2
7	47605	30	M	3
8	47673	36	M	3
9	47607	32	M	3

(a) The quasi-identifier table (QIT)

Group-ID	Disease	Count
1	Ovarian Cancer	2
1	Prostate Cancer	1
2	Flu	1
2	Heart Disease	2
3	Heart Disease	1
3	Flu	2

(b) The sensitive table (ST)

TABLE II
THE ANATOMIZED TABLE

II. BACKGROUND KNOWLEDGE ATTACK AGAINST ANATOMY

Recently, Xiao and Tao [9] introduced *Anatomy* as an alternative anonymization technique to generalization. *Anatomy* releases all the quasi-identifier and sensitive data directly into two separate tables. For example, the original table shown in Table I is decomposed into two tables, the quasi-identifier table (QIT) in Table II(a) and the sensitive table (ST) in Table II(b). The QIT table and the ST table are then released.

The authors also proposed an anatomizing algorithm to compute the anatomized tables. The algorithm first hashes the records into buckets based on the sensitive attribute, i.e., records with the same sensitive values are in the same bucket. Then the algorithm iteratively obtains the ℓ buckets that currently have the largest number of records and selects one record from each of the ℓ buckets to form a group. Each remaining record is then assigned to an existing group.

We show background knowledge attack on the anatomized tables. Suppose Alice knows that Bob's record belongs to the first group in Table II where the two sensitive values are "prostate cancer" and "ovary cancer", then Alice immediately knows that Bob has "prostate cancer". The apparent diversity does not help provide any privacy, because certain values can be easily eliminated. This problem is particularly acute in the *Anatomy* approach. The anatomizing algorithm randomly picks records and groups them together (rather than grouping records with similar quasi-id values together). Therefore, it is likely that one may be grouping records with incompatible sensitive attribute values together. We will empirically evaluate the effect of background knowledge attack on the anatomizing algorithm in Section VI.

III. INJECTOR: MINING BACKGROUND KNOWLEDGE FROM THE DATA

In this section, we propose a novel approach to deal with background knowledge attack. We first identify the possible sources where an adversary may obtain additional background knowledge. Then we explain the rationale of our approach of mining background knowledge from the data and describe the *Injector* framework.

A. Background Knowledge

Since the background knowledge attack is due to additional information that the adversary has, it is helpful to examine how the adversary may obtain this additional knowledge.

In traditional settings of data anonymization, the adversary is assumed to know certain knowledge besides the released data, e.g., the quasi-identifier values of individuals in the data and the knowledge of whether some individuals are in the data. In the following, we identify a list of additional knowledge that an adversary may have.

First, the adversary may know some absolute facts. For example, a male can never have *ovarian cancer*.

Second, the adversary may have partial knowledge of the demographic information of some specific groups. For example, the adversary may know that the probability that young females of certain ethnic groups have *heart disease* is very low. This knowledge can be represented as patterns or association rules that exist in the data.

Third, the adversary may have some adversary-specific knowledge, which is available to the adversary for some reason. For example, an adversary may know some targeted victim in person and have partial knowledge on the sensitive values of that individual (e.g., Alice may know that his friend Bob does not have *short breath problem* since she knows that Bob runs for two hours every day). An adversary may get additional information from other sources (e.g., Bob's son told Alice that Bob does not have *heart disease*). This type of knowledge is associated with specific adversaries and the channel through which an adversary obtains this type of knowledge can be varied among different adversaries.

While adversary-specific knowledge is hard to predict, it is possible to discover the other two types of background knowledge. Now we describe our proposed approach.

B. Our Approach

The main problem of dealing with background knowledge attacks is that we are unaware of the exact knowledge that an adversary may have and we believe that requiring the background knowledge as an input parameter is not feasible as it places too much a burden on the user. In this paper, we propose a novel approach to model the adversary's background knowledge. Our approach is to extract background information from the data to be released. For example, the fact that male can never have *ovarian cancer* should manifest itself in the data to be released, and thus it should be possible for us to discover the fact from the data. Also, it is often the case that an adversary may have access to similar data, in which case patterns or association rules mined from one data can be an important source of the adversary's background knowledge on the other data. We are aware that we do not consider adversary-specific knowledge. The specific knowledge that an adversary may have is hard to predict. Also, since the adversary cannot systematically obtain such knowledge, it is unlikely that the adversary knows specific knowledge about a large number of individuals.

With this background knowledge extracted from the data, we are able to anonymize the data in such a way that inference attacks using this background knowledge can be effectively prevented. For example, if one is grouping records together for privacy purposes, one should avoid grouping a male patient with another record that has *ovarian cancer* (or at least recognize that doing so does not help meet attribute disclosure privacy requirements).

One may argue that such an approach over-estimates an adversary's background knowledge, as the adversary may not possess all knowledge extracted from the data. We justify our approach through the following arguments. First, as it is difficult for us to bound exactly what the adversary knows and what she doesn't know, a conservative approach of utilizing all extracted knowledge of a certain kind is appropriate. Second, it is often the case that the adversary has access to similar data and knowledge extracted from the data can be the adversary's background knowledge on the other data. Finally, utilizing such extracted knowledge in the anonymization process typically results in (at least partial) preservation of such knowledge; this increases the data utility. Note that privacy guarantees are still met.

One intriguing aspect about our approach is that one can argue that it improves both privacy and data utility at the same time. Grouping a male patient with another record that has *ovarian cancer* is bad for privacy because it offers a false sense of protection; it is also bad for data utility, as it contaminates the data. By not doing that, one avoids introducing false associations and improves data utility. This is intriguing because, in the literature, privacy and utility have been viewed as two opposing properties. Increasing one leads to reducing the other.

C. The Injector Framework

We now present the *Injector* framework for data anonymization. *Injector* focuses on one type of background knowledge, i.e., a certain combination of quasi-identifier values cannot entail certain sensitive values. This type of background knowledge can be represented as negative association rules of the form *"Sex=M⇒ ¬ Disease=ovarian cancer"* and we are able to discover them from the data using data mining techniques. Mining other types of knowledge from the data and using them in data anonymization is an interesting direction for future work.

Injector uses permutation-based bucketization as the method of constructing the published data from the original data, which is similar to the *Anatomy* technique [9] and the

permutation-based anonymization approach [10]. The bucketization method first partitions tuples in the table into buckets and then separates the quasi-identifiers with the sensitive attribute by randomly permuting the sensitive attribute values in each bucket. The anonymized data consists of a set of buckets with permuted sensitive attribute values.

The *Injector* framework consists of two components: (1) mining negative association rules from the table and (2) using these rules in data anonymization. We discuss these two components in the following two sections respectively.

IV. Mining Negative Association Rules

In this section, we study the problem of mining negative association rules from the data. We first formalize our problem and introduce the *expectation* measure in Section IV-A. We present techniques for dealing with quantitative attributes in Section IV-B and describe the algorithm in Section IV-C.

A. Problem Formulation

Let T be a table which has m quasi-identifier attributes $A_j (1 \leq j \leq m)$, each with an attribute domain $\mathcal{D}_j$, and a sensitive attribute A_{m+1} with a domain $\mathcal{D}_{m+1}$. We define a value generalization hierarchy (VGH) for each quasi-identifier attribute where leaf nodes correspond to actual attribute values, and internal nodes represent less-specific values. We denote $t_i[j]$ as the j-th attribute value of tuple t_i.

Our objective is to discover interesting negative association rules [11]. In our setting, a negative association rule is an implication saying that some combination of quasi-identifier values cannot entail certain sensitive values. Specifically, a negative association rule is an implication of the form $X \Rightarrow \neg Y$, where X is a predicate involving only the quasi-identifiers and Y is a predicate involving only the sensitive attribute. The intuitive meaning of such a rule is that tuples that satisfy X do not satisfy Y with a high confidence. Usually, Y is a predicate of the form $A_{m+1} = s$ with $s \in D_{m+1}$ and X is a conjunction of predicates each of which is of the form $A_i = v_i$ $(1 \leq i \leq m)$ with $v_i \in D_i$.

In the rules defined above, only values at the leaf level of the VGHs are involved in the predicate X. To allow rules to take values from any level of the VGH, we define extended attribute domains $\mathcal{D}'_j = \mathcal{D}_j \cup E_j$, where E_j is the set of internal nodes of the VGH for the j-th attribute for $1 \leq j \leq m$, and $\mathcal{D}'_{m+1} = \mathcal{D}_{m+1}$. A *generalized negative association rule* is an implication of the form $X \Rightarrow \neg Y$, where Y is a predicate of the form $A_{m+1} = s$ with $s \in D'_{m+1}$ and X is a conjunction of predicates each of which is of the form $A_i = v_i$ $(1 \leq i \leq m)$ with $v_i \in D'_i$.

We now define "interestingness" of a negative association rule. Some traditional interestingness measures are based on *support* and *confidence*. Specifically, a rule is interesting if its support is at least $minSup$ and its confidence is at least $minConf$ where $minSup$ and $minConf$ are user-defined parameters. The rule $X \Rightarrow \neg Y$ has support $s\%$ if $s\%$ of tuples in T satisfy both X and $\neg Y$. The rule $X \Rightarrow \neg Y$ holds with confidence $c\%$ if $c\%$ of tuples which satisfy X in T also satisfy $\neg Y$. If we denote the fraction of tuples that satisfy predicate Z as $P(Z)$, then $s\% = P(X \cup \neg Y)$ and $c\% = P(X \cup \neg Y)/P(X)$.

We observe that setting a single $minSup$ value for different sensitive values would be inappropriate for our purpose. A frequent sensitive value is expected to occur with a high probability even in a small number of tuples; when this probability turns out to be small, it is an interesting rule. On the other hand, an infrequent sensitive value is expected to occur with a low probability even in a large number of tuples; even when this probability is small, the rule may not be interesting. Intuitively, we should set a larger $minSup$ value for a negative association rule involving a frequent sensitive attribute value.

Based on this observation, we propose to use *expectation* instead of *support* as the measure of the strength of a negative association rule. Given a negative association rule $X \Rightarrow \neg Y$, the number of tuples satisfying X is $n * P(X)$ where n is the total number of tuples in T. Among these tuples, the probability that the sensitive value of Y occurs at least once is $1 - (1 - P(Y))^{n*P(X)}$. We define this probability as the *expectation* of the rule. The rule $X \Rightarrow \neg Y$ is interesting if it has *expectation* at least $minExp$, i.e., $1-(1-P(Y))^{n*P(X)} \geq minExp$, which is equivalent to $P(X) \geq \frac{1}{n} \log_{1-P(Y)}(1 - minExp)$.

We now define our objective as finding all generalized negative association rules $X \Rightarrow \neg Y$ that satisfy the following two requirements where $minExp$ is a user-defined parameter and $minConf$ is fixed to be 1.

- Minimum *expectation* requirement: $P(X) \geq Sup_Y$, where $Sup_Y = \frac{1}{n} \log_{1-P(Y)}(1 - minExp)$.
- Minimum *confidence* requirement: $P(X \cup \neg Y)/P(X) \geq minConf$.

Note that in *Injector*, $minConf$ is fixed to be 1. A more general approach would allow us to probabilistically model the adversary's knowledge. We provide more discussion on this in Section VIII.

B. Dealing with Quantitative Attributes

The above definition does not consider the semantics of quantitative attributes. Consider the rule $\{Age = 21\} \Rightarrow \neg\{Salary = 50K\}$. Suppose few records with age 21 in the table have a salary close to $50K$. However, the rule $\{Age = 21\} \Rightarrow \neg\{Salary = 50K\}$ may not hold if a large number of records with age close to 21 in the table have a salary of $50K$. This suggests that while tuples with age exactly 21 directly support the rule, tuples with age close to 21 have partial support for this rule.

To consider partial support of quantitative attributes, we interpret nodes in the VGH of a quantitative attribute as a fuzzy set [12]. A value can belong to the node with set membership between $[0, 1]$. We denote the membership of value $t[a_i]$ in $Z[i]$ as $Mem(Z[i], t[a_i])$. There are two ways to define the support of Z from t (denoted as $P(Z, t)$): (1) the product of the membership of each attribute value, i.e., $P(Z, t) = \Pi_{1 \leq i \leq q} Mem(Z[i], t[a_i])$ and (2) the minimum

of the membership of each attribute value, i.e., $P(Z,t) = \min_{1\leq i\leq q} Mem(Z[i], t[a_i])$. We adopt the first method to compute $P(Z,t)$. Again, $P(Z) = \sum_{t\in T} P(Z,t)$. We are then able to use this support function to define the interestingness measures.

C. The Algorithm

As we have discussed in Section IV-A, the *expectation* requirement is equivalent to $P(X) \geq Sup_Y$. We define $minSup = \min_{s\in D_{m+1}} Sup_Y$ where Y is the predicate $A_{m+1} = s$. Then the problem of discovering interesting negative association rules can be decomposed into two subproblems: (1) Discovering all itemsets that involve only quasi-identifiers and have support at least $minSup$; (2) Finding all negative association rules satisfying the expectation and confidence requirements. We study the two problems in the rest of this section.

Discovering Frequent Itemsets: We can efficiently solve this problem by modifying existing frequent itemset generation algorithm Apriori [13] or the FP-tree algorithm [14]. For each frequent itemset X, we also record a count C_X indicating the support for X and an array of counts $C_X[s]$ for each sensitive value s indicating the number of tuples that support X and have sensitive value s (Note that $C_X = \sum_s C_X[s]$). These counts are used in solving the second subproblem.

Finding Negative Association Rules: The second subproblem is to generate negative association rules. For each frequent itemset X and for each sensitive value Y, we check if the following two conditions hold:

- $\frac{C_X}{n} \geq Sup_Y$
- $\frac{C_X[Y]}{C_X} \leq 1 - minConf$

If both conditions are satisfied, $X \Rightarrow \neg Y$ is identified as a negative association rule. The first condition ensures that the negative association rule has sufficient expectation (Note that $\frac{C_X}{n} = P(X)$). The second condition ensures that the negative association rule has sufficient confidence (Note that $1 - \frac{C_X[Y]}{C_X} = P(X \cup \neg Y)/P(X)$).

V. Using Negative Association Rules in Data Anonymization

In this section, we study the problem of using negative association rules in data anonymization. We define the privacy requirement for data anonymization in the new framework and develop an anonymization algorithm to achieve the privacy requirement.

A. Privacy Requirement

Let g be a group of tuples $\{t_1, ..., t_p\}$. We say a tuple t *cannot take* a sensitive attribute value s if there exists a negative association rule $X \Rightarrow \neg s$ and t satisfies X.

A simple privacy requirement would require that each group g satisfies the condition that, for every tuple t_i in g, g contains at least ℓ sensitive attribute values that t_i can take. This simple privacy requirement is, however, insufficient to prevent background knowledge attack. Suppose that a group contains 1 female record and ℓ male records and the values of the sensitive attribute include 1 *ovarian cancer* and ℓ other diseases common to both male and female. This satisfies the simple privacy requirement. The female record is compatible with all $\ell+1$ sensitive attribute values while each male record is compatible with ℓ sensitive attribute values. However, when one considers the fact that there must be a one-to-one mapping between the records and the values, one can infer that the only female record must have *ovarian cancer*, since no other record can be mapped to this value. This suggests that a stronger privacy requirement is needed to bound the privacy risk caused by background knowledge attack.

Definition 1 (The Matching ℓ-Diversity Requirement): Given a group of tuples, we say a sensitive value is *valid* for a tuple if there exists an assignment for the remaining tuples in the group. A group of tuples satisfy the matching ℓ-diversity requirement if every tuple in the group has at least ℓ valid sensitive values.

The matching ℓ-diversity requirement guarantees that the adversary with background knowledge cannot learn the sensitive value of an individual from a set of at least ℓ values.

B. Checking Privacy Breaches

We have defined our privacy requirement, now we study the checking problem: given a group of tuples, are there at least ℓ valid sensitive values for every tuple in the group? To check whether a sensitive value s_j is valid for a tuple t_i, we assigns s_j to t_i and then check if there is an assignment for the group of remaining tuples g'.

Checking the existence of an assignment for a group of tuples g' can be reduced to the maximum bipartite matching problem [15] where all q_i's in g' form one set of nodes, all s_i's in g' form the other set of nodes, and an edge between q_i and s_j represents that t_i can take value s_j. Specifically, there is an assignment for g' if and only if there is a matching of size $|g'|$ in the corresponding bipartite graph.

There are polynomial time algorithms (e.g., path augmentation [15]) for the maximum bipartite matching (MBM) problem. We denote $MBM(g)$ as the procedure for solving the MBM problem given the bipartite graph constructed from g. Our algorithm iteratively invokes MBM procedure.

The algorithm is given in Figure 1. The input to the algorithm is a group of tuples g. The algorithm checks if there are at least ℓ valid sensitive values for every tuple in g. The $MBM(g)$ procedure takes time $O(|g|^2 p)$ where p is the number of edges in the constructed bipartite graph for g. The checking algorithm invokes the $MBM(g)$ procedure at

```
1. for every t_i ∈ g
2.     Set = ∅
3.     for every t_j ∈ g
4.         construct a new tuple t' = (q_j, s_i)
5.         if MBM(g − {t_i, t_j} ∪ {t'})==|g| − 1
6.             Set = Set ∪ {s_j}
7.     if |Set| < ℓ return false
8. return true
```

Fig. 1. The checking algorithm

/* *Line 1 computes* $\mathcal{N}[s][O]$ *and* $\mathcal{N}[s][S']$ */
1. **for** $\forall t_i \in T$, increment $\mathcal{N}[s_i][O]$ and $\mathcal{N}[s_i][IVS[t_i]]$
/* *Lines 2-17 groups tuples into buckets* L */
2. **while** $|T| \geq \ell$
3. pick a tuple $t_i \in T$ that maximizes $NIT[\{t_i\}]$
4. $g = \{t_i\}$, $T = T - \{t_i\}$
5. decrement $\mathcal{N}[s_i][O]$ and $\mathcal{NS}[s_i][IVS[t_i]]$
6. **while** $|g| < \ell$
7. **if** no $t_j \in T$ is compatible with g
8. **for** every $t_j \in g$
9. increment $\mathcal{N}[s_j][O]$ and $\mathcal{N}[s_j][IVS[t_j]]$
10. insert all tuples in $g - \{t_i\}$ into T
11. insert t_i into T_r, go to line 2
12. **else** select $t_j \in T$ that is compatible with g and
13. minimizes $NIT[g \cup \{t_j\}]$
14. $g = g \cup \{t_j\}$, $T = T - \{t_j\}$
15. decrement $\mathcal{N}[s_j][O]$ and $\mathcal{N}[s_j][IVS[t_j]]$
16. insert g into L
17. insert all tuples in T into T_r
/* *Lines 18-23 add the remaining tuples* T_r *to groups* */
18. **for** every t_j in T_r
19. select $g \in L$ having the smallest number of tuples
20. that are incompatible with t_j, set $g = g \cup \{t_j\}$
21. **while** t_j has less than ℓ valid sensitive values in g
22. select $g' \in L$ maximizing $|SEN(g') - SEN(g)|$
23. $g = g \cup g'$, remove g' from L

Fig. 2. The bucketization algorithm

most $|g|^2$ times. It follows that our checking algorithm takes time $O(|g|^4 p)$.

C. A Bucketization Algorithm

We present the bucketization algorithm. To describe the algorithm, we introduce the following notations. Let g be a group of tuples $\{t_1, ..., t_p\}$ such that $t_i = \langle q_i, s_i \rangle$ where q_i is the quasi-identifier value of t_i and s_i is the sensitive attribute value of t_i. Let $IVS[t_i]$ denote the set of sensitive attribute values that t_i cannot take. Let $SEN[g]$ denote the set of sensitive attribute values in g.

The bucketization algorithm, when given a set of tuples T and an IVS set for each tuple, outputs a number of groups of tuples for publication. We first give the following definition.

Definition 2: A tuple t_j is incompatible with a tuple t_i if at least one of the following three conditions holds: (1) $s_j = s_i$, (2) t_i cannot take the value s_j, and (3) t_j cannot take the value s_i. A tuple t_j is incompatible with a group of tuples g if t_j is incompatible with at least one tuple in g.

Our bucketization algorithm includes three phases. The first phase is the initialization phase, which initializes the data structures. The second phase is the grouping phase where groups are formed. To form a group, the algorithm first chooses a tuple t_i that has the largest number of incompatible tuples. The group g initially contains only t_i. Then additional tuples are added to the group iteratively. Each time, a tuple t_j is selected such that t_j is compatible with g and the new group (formed by adding t_j to g) has the smallest number of incompatible tuples. If no tuples are compatible with g, we put t_i in the set of remaining tuples and consider the next tuple. The third phase is the group assignment phase where each of the remaining tuples t_j is assigned to a group. Initially, t_j is added to the group g which has the smallest number of tuples that are incompatible with t_j (i.e., $g = g \cup \{t_j\}$). We iteratively merge g with the group which has the largest number of sensitive values that are different from g until t_j has at least ℓ valid sensitive values in g (the checking algorithm is invoked to count the number of valid sensitive values for t_j). One nice property about the matching ℓ-diversity requirement is that if a group of tuples satisfy the requirement, they still satisfy the requirement when additional tuples are added. We thus only need to consider the remaining tuples in this phase.

	Attribute	Type	# of values	Height
1	Age	Numeric	74	5
2	Workclass	Categorical	8	3
3	Education	Categorical	16	4
4	Marital_Status	Categorical	7	3
5	Race	Categorical	5	3
6	Gender	Categorical	2	2
7	Occupation	Sensitive	14	N/A

TABLE III
DESCRIPTION OF THE **ADULT** DATASET

The key component of the algorithm is to compute the number of tuples that are incompatible with g (denoted as $NIT[g]$). To efficiently compute $NIT[g]$, we maintain a compact data structure. For each sensitive value s, we maintain a list of counts $\mathcal{N}[s][S']$ which denotes the number of tuples whose sensitive value is s and whose IVS set is S'. Note that we only maintain positive counts. Let $\mathcal{N}[s][O]$ denote the number of tuples whose sensitive value is s, i.e., $\mathcal{N}[s][O] = \sum_{S'} \mathcal{N}[s][S']$, and O is only a special symbol.

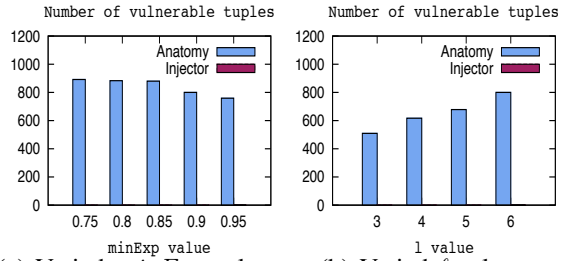
We denote $IVS[g]$ as the set of sensitive values that are incompatible with g. We have $IVS[g] = \cup_{t \in g} IVS[t]$, i.e., a sensitive value is incompatible with g if it is incompatible with at least one tuple in g. We denote $IS[g] = SEN[g] \cup IVS[g]$. We can then partition the set of tuples that are incompatible with g into two groups: (1) $\{t_j | s_j \in IS[g]\}$ and (2) $\{t_j | (s_j \notin IS[g]) \wedge (SEN[g] \cap IVS[t_j] \neq \emptyset)\}$. Then, $NIT[g]$ can be computed as follows.

$$NIT[g] = \sum_{s \in IS[g]} \mathcal{N}[s][O] + \sum_{(s \notin IS[g])} \sum_{(S' \cap SEN[g] \neq \emptyset)} \mathcal{N}[s][S']$$

The algorithm is given in Figure 2. We now analyze its complexity. Let $|T| = n$ and assume $n \gg |S|$ and $n \gg \ell$. The initialization phase scans the data once and thus takes $\mathcal{O}(n)$ time. The grouping phase takes at most n rounds. In each round, the algorithm scans the data once and the computation of $NIT[g]$ takes time $\mathcal{O}(q)$ where q is the number of positive $N[s][S']$ entries (note that $q \leq n$). The grouping phase thus takes $\mathcal{O}(qn^2)$ time. The group assignment phase takes $|T_r| \leq n$ rounds and at most n merges, each takes $\mathcal{O}(n)$ time. Thus, the total time complexity is in $\mathcal{O}(qn^2)$.

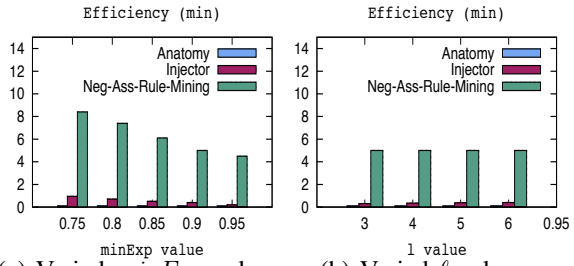
$minExp$	$\|R\|$	N_0	N_1	N_2	N_3	N_4	$N_{\geq 5}$
0.75	45	0	80.4%	15.9%	2.3%	1.2%	0.2%
0.80	39	0	84.6%	12.2%	2.1%	1.0%	0.1%
0.85	32	0	87.5%	9.2%	2.0%	1.0%	0
0.90	22	0	87.9%	9.2%	2.5%	0.4%	0
0.95	15	0	96.2%	1.8%	2.0%	0	0

Fig. 3. Number of discovered rules and percentage of tuples with incompatible sensitive values



(a) Varied $minExp$ values (b) Varied ℓ values

Fig. 4. Background knowledge attack



(a) Varied $minExp$ values (b) Varied ℓ values

Fig. 5. Efficiency

(a) Varied $minExp$ values (b) Varied ℓ values

Fig. 6. Confidence error

VI. Experiments

The main goals of the experiments are to study the effect of background knowledge attack on *Anatomy* and to investigate the effectiveness of *Injector* in both privacy protection and data utility preservation. For privacy protection, we compare Anatomy and Injector in terms of the number of tuples that are vulnerable to background knowledge attack. For data utility preservation, we compare Anatomy and Injector using two utility metrics: (1) error in association rule mining and (2) accuracy in aggregate query answering. We also evaluate the efficiency of the Injector approach.

The dataset used in the experiments is the adult dataset from the UC Irvine machine learning repository, which is comprised of data collected from the US census. The description of the dataset is given in Table III.

Given the dataset, we compute both the corresponding anatomized tables and the injected tables. The anatomized tables are computed using the anatomizing algorithm described in [9]. To compute the injected tables, we first find negative association rules in the original dataset using different $minExp$ values. In all our experiments, $minConf$ is fixed to be 1.

Figure 3 shows the results of negative association rule mining on the original data. $|R|$ indicates the number of discovered negative association rules. N_0, N_1, N_2, N_3, and N_4 indicate the percentage of tuples that have 0, 1, 2, 3, and 4 incompatible sensitive values, respectively. $N_{\geq 5}$ indicates the percentage of tuples that have at least 5 incompatible sensitive values. The negative association rules discovered from the data include, for example, $\{$*Workclass = Government*$\}\Rightarrow \neg\{$*Occupation = Priv-house-serv*$\}$ and $\{$*Education = Doctorate*$\}\Rightarrow \neg \{$*Occupation = Handlers-cleaners*$\}$. We then compute the injected tables using the bucketization algorithm described in Section V.

We evaluate the performance of *Anatomy* and *Injector* on two parameters: (1) $minExp$ value within the range $[0.75, 0.95]$ (the default value is 0.9); (2) ℓ value which ranges from 3 to 6 (the default value is 6). Since the anatomizing algorithm is a randomized algorithm, for each set of selected parameters, we run the anatomizing algorithm for 10 times and the average value is reported.

A. Background Knowledge Attack

To illustrate the effects of background knowledge attack on *Anatomy* and *Injector*, we count the number of tuples in the anatomized tables and the injected tables that have less than ℓ possible sensitive values using the extracted negative association rules. These tuples are viewed as vulnerable to background knowledge attack.

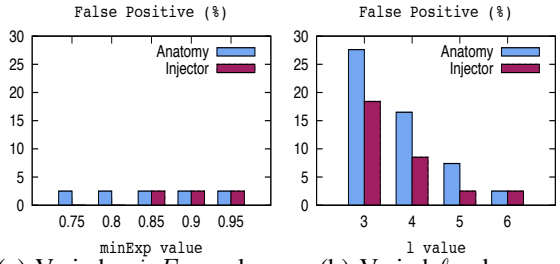
The experimental results are shown in Figure 4. In all experiments, *Injector* has no vulnerable tuples, indicating that *Injector* better protects the data against background knowledge attacks.

B. Efficiency

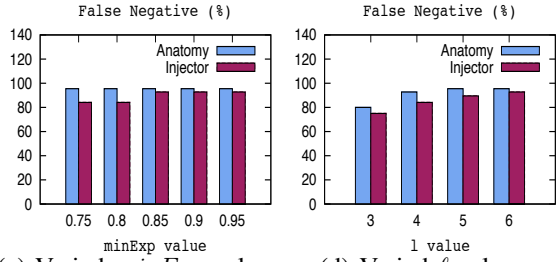
We compare the efficiency of computing the anatomized tables and the injected tables. The time for computing the injected tables consists of two parts: (1) the time for computing the negative association rules and (2) the time for computing the injected tables using the bucketization algorithm.

Experimental results are shown in Figure 5. The time to compute the injected tables using the bucketization algorithm is roughly the same as the time to compute the anatomized tables using the anatomizing algorithm, usually within seconds. The main efficiency issue of computing the injected tables lies in computing the negative association rules. However, computing the negative association rules using a variation of the FP-tree algorithm is fast enough for large datasets.

In the rest of this section, we compare *Injector* against *Anatomy* and show that *Injector* also improves data utility.

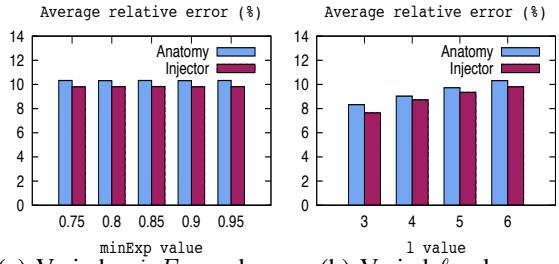


(a) Varied $minExp$ values (b) Varied ℓ values

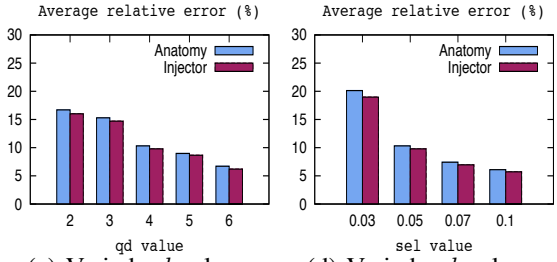


(c) Varied $minExp$ values (d) Varied ℓ values

Fig. 7. Identification error



(a) Varied $minExp$ values (b) Varied ℓ values



(c) Varied qd values (d) Varied sel values

Fig. 8. Aggregate query answering error

C. Association Rule Mining

The first set of experiments on data utility evaluates the errors in association rule mining. We mine positive association rules from the original data and the anonymized data. We use *conviction* [16] as the interestingness metric. Specifically, the *conviction* of $X \Rightarrow Y$ is defined as $P(X)P(\neg Y)/P(X \cup \neg Y)$. Both *Apriori* [13] and FP-tree [14] can be used to find all positive association rules that satisfy both minimum support and minimum conviction requirements. Two error metrics are evaluated in the experiments: confidence error and identification error.

To evaluate confidence error, we find the set of association rules R with sufficient support and conviction in the original table. For each $r \in R$ (the confidence of r in the original table is denoted as org_conf_r), we compute the corresponding confidence of r in the anonymized table (through *Anatomy* or *Injector*) which is denoted as the reconstructed confidence rec_conf_r. The *Confidence Error* is computed as:

$$\rho = \frac{1}{|R|} \sum_{r \in R} \frac{|rec\_conf_r - org\_conf_r|}{org\_conf_r} * 100$$

The confidence error metric reflects the preservation of the association between the quasi-identifiers and the sensitive attribute. Experimental results are shown in Figure 6. In all experiments, *Injector* has smaller confidence errors, indicating that *Injector* captures a larger amount of association between the quasi-identifiers and the sensitive attribute.

Errors in confidence estimation can have more pernicious effects. They can result in errors in the identification of interesting association rules, i.e., association rules that exist in the original data may be hidden in the anonymized data and association rules that do not exist in the original data may be erroneously identified in the anonymized data. This error is called identification error.

To evaluate identification error, we perform association rule mining on all three tables: (1) the original table, (2) the anatomized tables, and (3) the injected tables. We denote the set of association rules discovered from the original table and the anonymized tables (through *Anatomy* or *Injector*) as R_{org} and R_{rec}, respectively. The identification error has two components: (1) false positive σ^+ indicating the percentage of association rules that are not in R_{org} but in R_{rec}, and (2) false negative σ^- indicating the percentage of association rules that are in R_{org} but not in R_{rec}. The two metrics are computed as:

$$\sigma^+ = \frac{|R_{rec} - R_{org}|}{|R_{org}|} * 100 \qquad \sigma^- = \frac{|R_{org} - R_{rec}|}{|R_{org}|} * 100$$

The identification error metric reflects errors in identifying association rules. The experimental results are in Figure 7. In all figures, *Injector* has smaller identification errors, indicating that *Injector* is more effective in identifying association rules.

D. Aggregate Query Answering

The second set of experiments on data utility evaluates the accuracy of aggregate query answering. We only consider the "COUNT" operator where the query predicate involves the sensitive attribute, as in [9]. It is also possible to compute other aggregate query operators such as "MAX" and "AVERAGE" on numerical attributes [10]. Specifically, the queries that we consider are of the form:

```
SELECT COUNT(*) FROM Table
WHERE v_{i_1} ∈ V_{i_1} AND ... v_{i_dim} ∈ V_{i_dim} AND s ∈ V_S
```

where $v_{i_j} (1 \leq j \leq dim)$ is the quasi-identifier value for attribute A_{i_j}, $V_{i_j} \subseteq D_{i_j}$ and D_{i_j} is the domain for attribute A_{i_j}, s is the sensitive attribute value and $V_S \subseteq D_S$ and D_S is the domain for the sensitive attribute S. A query predicate is

characterized by two parameters: (1) the predicate dimension dim, indicating the number of quasi-identifiers involved in the predicate; (2) the query selectivity sel, indicating the number of values in each $V_{i_j}, (1 \leq j \leq dim)$. Specifically, the size of $V_{i_j}, (1 \leq j \leq dim)$ is randomly chosen from $\{0, 1, ..., sel * |D_{i_j}|\}$. For each selected parameter, we generate 1000 queries Q for the experiments.

For each query q, we run the query on the three tables: (1) the original table, (2) the anatomized tables, and (3) the injected table. We denote the count from the original table and the reconstructed count from the anonymized tables (through *Anatomy* or *Injector*) as org_count_q and rec_count_q respectively. Then the average relative error is computed over all queries as:

$$\rho = \frac{1}{|Q|} \sum_{q \in Q} \frac{|rec\_count_q - org\_count_q|}{org\_count_q} * 100$$

Experimental results are shown in Figure 8. In all figures, *Injector* has smaller errors, which indicates that *Injector* permits more accurate data analysis in aggregate query answering than *Anatomy*.

VII. Related Work

Existing work on data anonymization can be classified into two categories. The first category of work aims at devising privacy requirements. Samarati and Sweeney [3], [4], [5] first proposed the k-anonymity model which assumes that assumes that the adversary has access to some publicly-available databases (e.g., a vote registration list) from which she obtains the quasi-identifier values of the individuals. The model also assumes that the adversary knows that some individuals are in the table. Much of the subsequent work on data anonymization assumes to use this adversarial model. In [6], [8], the authors recognized that the adversary also has knowledge of the distribution of the sensitive attribute in each equivalence class and she may be able to infer sensitive values of some individuals using this knowledge. In [6], the authors proposed the ℓ-diversity requirement [6] which, however, does not satisfactorily prevent an adversary with background knowledge from learning important sensitive information. Recently, Li et al. [17] observed that the distribution of the sensitive attribute in the overall table should be public information and the adversary can infer sensitive information with this additional knowledge. They proposed the t-closeness requirement as a stronger notion of privacy against this more-powerful adversary. In [18], Martin et al. proposed a formal language to express background knowledge about the data and a polynomial time algorithm for computing the disclosure risk in the worst case. However, this work does not consider the exact background knowledge that the adversary may have.

The second category of work aims at developing anonymization techniques to achieve the privacy requirements. One popular approach to anonymize the data is generalization and suppression [3], [4]. In generalization, some generalization schemes [3], [19], [20], [21], [22] have the consistency property, which means multiple occurrences of the same value are always generalized in the same way, i.e., they are replaced with the same value. A serious defect of generalization that has been recognized by [23], [24], [8] is that experimental results have shown that many attributes have to be suppressed in order to guarantee privacy. A number of techniques [9], [24] have been proposed to remedy this defect of generalization. On the other hand, some generalization schemes [25] do not have the consistency property. One example is the Mondrian [25] multi-dimensional k-anonymity. Another anonymization technique is clustering [26], which typically groups records based on some distance metric. Recently, Xiao and Tao [9] proposed *Anatomy* as an alternative anonymization technique. Koudas et al. [10] explored the permutation-based anonymization approach and examined the anonymization problem from the perspective of answering downstream aggregate queries.

Several research works also consider background knowledge in other contexts. Yang and Li [27] studied the problem of information disclosure in XML publishing when the adversary has knowledge of functional dependencies about the XML data. In [28], Lakshmanan et al. studied the problem of protecting the true identities of data objects in the context of frequent set mining when an adversary has partial information of the items in the domain. In their framework, the adversary's prior knowledge was modeled as a *belief function* and formulas were derived for computing the number of items whose identities can be "cracked".

The problem of association rule mining was introduced by Agrawal et al. [29] in the context of transactional database. A number of algorithms such as Apriori [13] and FP-Tree [14] have been proposed to discover interesting association rules. The generalization of association rule mining to multiple levels of hierarchies over items is studies in [30], [31]. Association rules involving quantitative attributes are studied in [32], [33], [34], [12]. [32] partitioned the domain of a numerical attribute into equi-depth intervals based on some quality metric. [33] proposed that the interestingness measure should consider the semantics of interval data. [34] studied the problem of mining association rules from data containing both categorical and numerical attributes. And [12] proposed to use fuzzy sets instead of intervals to deal with quantitative attributes. Negative association rule mining is studied in [11], [35]. [11] proposed an algorithm for generating negative association rules based on a complex measure of rule parts. [35] generalized both positive and negative association rules to correlations and proposed the chi-squared test for correlation as an interestingness measure. Finally, [36] studied how to select the right measure for evaluating the interestingness of association rules.

VIII. Conclusions and Future Work

While recent work has shown that background knowledge attack can present disclosure risks to the anonymized data, no work has used background knowledge in data anonymization mainly because of the challenge of modeling the adversary's background knowledge.

In this paper, we explicitly consider and use background knowledge. Our approach first mines knowledge from the data

and then uses the knowledge in data anonymization. One key advantage of our approach is that it protects the data against background knowledge attacks while improving data utility.

We have proposed the *Injector* framework which uses one type of knowledge (negative association rules). Our experimental results show the effectiveness of *Injector* in both privacy protection and utility preservation. Below we discuss some interesting open research issues.

A. Mining Other Knowledge from the Data

It may be possible for us to discover knowledge from the data other than negative association rules. One type of knowledge is the summary statistics of the data. An example of this might be that the average salary of physical therapists is $70K$. One direction of future work is to study how an adversary might use this additional knowledge to make more precise inferences and how we can use this knowledge in data anonymization to prevent these inference attacks. Also, how this would affect data utility is an interesting research problem. For example, we are able to discover positive association rules in the data. However, using positive association rules in anonymization would hide the association rules.

B. A General Model to Deal with Background Knowledge

The *Injector* framework considers only negative association rules with $minConf = 1$. A general framework would allow us to probabilistically model the adversary's background knowledge. It is an interesting direction to study how to compute the adversary's posterior knowledge in this framework and use these knowledge in data anonymization. In this framework, we can model both the adversary's posterior knowledge and prior knowledge as distributions of the sensitive attribute for all tuples. How to model the adversary's prior knowledge and how to compute the posterior knowledge, e.g., using Bayesian inference techniques, are interesting research problems.

C. Other Uses of Knowledge Mined from the Data

The extracted knowledge from the data might be useful for purposes other than anonymizing the data. For instance, we can use the extracted knowledge to guide the construction of generalization hierarchies. For example, attribute values with very different probabilities on some sensitive value should not be grouped together before they are combined with attribute values that have similar probabilities on the sensitive values. Investigating other uses of the exacted knowledge is an interesting research direction.

REFERENCES

[1] G. T. Duncan and D. Lambert, "Disclosure-limited data dissemination," *J. Am. Stat. Assoc.*, pp. 10–28, 1986.

[2] D. Lambert, "Measures of disclosure risk and harm," *J. Official Stat.*, vol. 9, pp. 313–331, 1993.

[3] P. Samarati and L. Sweeney, "Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression, Tech. Rep. SRI-CSL-98-04, 1998.

[4] L. Sweeney, "Achieving k-anonymity privacy protection using generalization and suppression," *Int. J. Uncertain. Fuzz.*, vol. 10, no. 6, pp. 571–588, 2002.

[5] ——, "k-anonymity: A model for protecting privacy," *Int. J. Uncertain. Fuzz.*, vol. 10, no. 5, pp. 557–570, 2002.

[6] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "ℓ-diversity: Privacy beyond k-anonymity." in *ICDE*, 2006, p. 24.

[7] R. C.-W. Wong, J. Li, A. W.-C. Fu, and K. Wang, "(α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing," in *KDD*, 2006, pp. 754–759.

[8] X. Xiao and Y. Tao, "Personalized privacy preservation," in *SIGMOD*, 2006, pp. 229–240.

[9] ——, "Anatomy: simple and effective privacy preservation," in *VLDB*, 2006, pp. 139–150.

[10] N. Koudas, D. Srivastava, T. Yu, and Q. Zhang, "Aggregate query answering on anonymized tables," in *ICDE*, 2007, pp. 116–125.

[11] A. Savasere, E. Omiecinski, and S. B. Navathe, "Mining for strong negative associations in a large database of customer transactions," in *ICDE*, 1998, pp. 494–502.

[12] C. M. Kuok, A. Fu, and M. H. Wong, "Mining fuzzy association rules in databases," *SIGMOD Record*, pp. 209–215.

[13] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *VLDB*, 1994, pp. 487–499.

[14] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *SIGMOD*, 2000, pp. 1–12.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. The MIT Press, 2001.

[16] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur, "Dynamic itemset counting and implication rules for market basket data," in *SIGMOD*, 1997, pp. 255–264.

[17] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: Privacy beyond k-anonymity and ℓ-diversity," in *ICDE*, 2007, pp. 106–115.

[18] D. J. Martin, D. Kifer, A. Machanavajjhala, J. Gehrke, and J. Y. Halpern, "Worst-case background knowledge for privacy-preserving data publishing," in *ICDE*, 2007, pp. 126–135.

[19] P. Samarati, "Protecting respondent's privacy in microdata release," *TKDE*, vol. 13, no. 6, pp. 1010–1027, 2001.

[20] V. S. Iyengar, "Transforming data to satisfy privacy constraints," in *KDD*, 2002, pp. 279–288.

[21] R. J. Bayardo and R. Agrawal, "Data privacy through optimal k-anonymization," in *ICDE*, 2005, pp. 217–228.

[22] K. LeFevre, D. DeWitt, and R. Ramakrishnan, "Incognito: Efficient full-domain k-anonymity," in *SIGMOD*, 2005, pp. 49–60.

[23] C. Aggarwal, "On k-anonymity and the curse of dimensionality," in *VLDB*, 2005, pp. 901–909.

[24] D. Kifer and J. Gehrke, "Injecting utility into anonymized datasets," in *SIGMOD*, 2006, pp. 217–228.

[25] K. LeFevre, D. DeWitt, and R. Ramakrishnan, "Mondrian multidimensional k-anonymity," in *ICDE*, 2006, p. 25.

[26] G. Aggrawal, T. Feder, K. Kenthapadi, S. Khuller, R. Panigrahy, D. Thomas, and A. Zhu, "Achieving anonymity via clustering," in *PODS*, 2006, pp. 153–162.

[27] X. Yang and C. Li, "Secure xml publishing without information leakage in the presence of data inference," in *VLDB*, 2004, pp. 96–107.

[28] L. V. S. Lakshmanan, R. T. Ng, and G. Ramesh, "To do or not to do: the dilemma of disclosing anonymized data," in *SIGMOD*, 2005, pp. 61–72.

[29] R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," in *SIGMOD*, 1993, pp. 207–216.

[30] R. Srikant and R. Agrawal, "Mining generalized association rules," in *VLDB*, 1995, pp. 407–419.

[31] J. Han and Y. Fu, "Discovery of multiple-level association rules from large databases," in *VLDB*, 1995, pp. 420–431.

[32] R. Srikant and R. Agrawal, "Mining quantitative association rules in large relational tables," in *SIGMOD*, 1996, pp. 1–12.

[33] R. J. Miller and Y. Yang, "Association rules over interval data," in *SIGMOD*, 1997, pp. 452–461.

[34] R. Rastogi and K. Shim, "Mining optimized association rules with categorical and numeric attributes," in *ICDE*, 1998, pp. 503–512.

[35] S. Brin, R. Motwani, and C. Silverstein, "Beyond market baskets: Generalizing association rules to correlations," in *SIGMOD*, 1997, pp. 265–276.

[36] P.-N. Tan, V. Kumar, and J. Srivastava, "Selecting the right objective measure for association analysis," *Inf. Syst.*, vol. 29, no. 4, pp. 293–313, 2004.