Privacy-Preserving Incremental Data Dissemination

Ji-Won Byun, Tiancheng Li,
Elisa Bertino, and Ninghui Li
Computer Science
Purdue University
West Lafayette, IN 47906
{byunj,li83,bertino,ninghui}@cs.purdue.edu

Yonglak Sohn
.
Computer Engineering
Seokyeong University
Seoul, Korea
syl@skuniv.ac.kr

ABSTRACT

Although the k-anonymity and ℓ-diversity models have led to a number of valuable privacy-protecting techniques and algorithms, the existing solutions are currently limited to static data release. That is, it is assumed that a complete dataset is available at the time of data release. This assumption implies a significant shortcoming, as in many applications data collection is rather a continual process. Moreover, the assumption entails "one-time" data dissemination; thus, it does not adequately address today's strong demand for immediate and up-to-date information. In this paper, we consider incremental data dissemination, where a dataset is continuously incremented with new data. The key issue here is that the same data may be anonymized and published multiple times, each of the time in a different form. Thus, static anonymization (i.e., anonymization which does not consider previously released data) may enable various types of inference. In this paper, we identify such inference issues and discuss some prevention methods.

1. INTRODUCTION

When person-specific data is published, protecting individual respondents' privacy is a top priority. Among various approaches addressing this issue, the *k-anonymity* model [12, 10] and the *ℓ-diversity* model [8] have recently drawn significant attention in the research community. In the k-anonymity model, privacy protection is achieved by ensuring that every record in a released dataset is indistinguishable from at least $(k-1)$ other records within the dataset. Thus, every respondent included in the dataset corresponds to at least k records in a k-anonymous dataset, and the risk of record identification (i.e., the probability of associating a particular individual with a released record) is guaranteed to be at most $1/k$. While the k-anonymity model primarily focuses on the problem of record identification, the ℓ-diversity model, which is built upon the k-anonymity model, addresses the risk of attribute disclosure (i.e., the probability of associating a particular individual with a sensitive attribute value). As an attribute disclosure may occur without records being identified (e.g., due to lack of diversity in a sensitive attribute), the ℓ-diversity model, in its simplest form,[*] additionally requires that every group of indistinguishable records contain at least ℓ distinct sensitive attribute values; thereby the risk of attribute disclosure is bound to at most $1/\ell$.

[*]We discuss more robust ℓ-diversity requirements in Section 2.

Although these models have yielded a number of valuable privacy-protecting techniques [1, 4, 5, 6, 7, 11], existing approaches only deal with static data release. That is, all these approaches assume that a complete dataset is available at the time of data release. This assumption implies a significant shortcoming, as in many applications data collection is rather a continuous process. Moreover, the assumption entails "one-time" data dissemination. Obviously, this does not address today's strong demand for immediate and up-to-date information, as the data cannot be released before the data collection is considered complete.

As a simple example, suppose that a hospital is required to share its patient records with a disease control agency. In order to protect patients' privacy, the hospital anonymizes all the records prior to sharing. At first glance, the task seems reasonably straightforward, as existing anonymization techniques can efficiently anonymize the records. The challenge is, however, that new records are continuously collected by the hospital (e.g., whenever new patients are admitted), and it is critical for the agency to receive up-to-date data in timely manner.

One possible approach is to provide the agency with datasets containing only the new records, which are independently anonymized, on a regular basis. Then the agency can either study each dataset independently or merge multiple datasets together for more comprehensive analysis. Although straightforward, this approach may suffer from severely low data quality. The key problem is that relatively small sets of records are anonymized independently so that the records may have to be modified much more than when they are anonymized together with previous records [3]. Moreover, a recoding scheme applied to each dataset may make the datasets inconsistent with each other; thus, collective analysis on multiple datasets may require additional data modification. Therefore, in terms of data quality, this approach is highly undesirable. One may believe that data quality can be assured by waiting for new data to be accumulated sufficiently large. However, this approach may not be acceptable in many applications as new data cannot be released in a timely manner.

A better approach is to anonymize and provide the entire dataset whenever it is augmented with new records (possibly along with another dataset containing only new records). In this way, the agency can be provided with up-to-date, quality-preserving and "more complete" datasets each time. Although this approach can also be easily implemented by using existing techniques (i.e., anonymizing the entire dataset every time), it has a significant drawback.

That is, even though each released dataset, when observed independently, is guaranteed to be anonymous, the combination of several released datasets may be vulnerable to various inferences. We illustrate these inferences through some examples in Section 3.1. As such inferences are typically made by comparing or linking records across different tables (or versions), we refer to them as *cross-version inferences* to differentiate them from inferences that may occur within a single table.

Our goal in this paper is to identify and prevent cross-version inferences so that an increasing dataset can be incrementally disseminated without compromising the imposed privacy requirement. In order to achieve this, we first define the privacy requirement for incremental data dissemination. We then discuss three types of cross-version inference that an attacker may exploit by observing multiple anonymized datasets. We also present our anonymization method where the degree of generalization is determined based on the previously released datasets to prevent any cross-version inference. The basic idea is to obscure linking between records across different datasets. We develop our technique in two different types of recoding approaches; namely, full-domain generalization [6] and multidimensional anonymization [7]. One of the key differences between these two approaches is that the former generalizes a given dataset according to predefined generalization hierarchies, while the latter does not. Based on our experimental result, we compare these two approaches with respect to data quality and vulnerability to cross-table inference. Another issue we address is that as a dataset is released multiple times, one may need to keep the history of previously released datasets. We thus discuss how to maintain such history in a compact form to reduce unnecessary overheads.

The remainder of this paper is organized as follows. In Section 2, we review the basic concepts of the k-anonymity and ℓ-diversity models and provide an overview of related techniques. In Section 3, we formulate the privacy requirement for incremental data dissemination. Then in Section 4, we describe three types of inference attacks based on our assumption of potential attackers. We present our approach to preventing these inferences in Section 5 and evaluate our technique in Section 6. We review some related work in Section 7 and conclude our discussion in Section 8.

2. PRELIMINARIES

In this section, we discuss the key concepts of the k-anonymity and ℓ-diversity models and briefly review related techniques.

2.1 Anonymity Models

The k-anonymity model assumes that data are stored in a table (or a relation) of columns (or attributes) and rows (or records). It also assumes that the target table contains person-specific information and that each record in the table corresponds to a unique real-world individual. The process of anonymizing such a table starts with removing all the explicit identifiers, such as name and SSN, from the table. However, even though a table is free of explicit identifiers, some of the remaining attributes in combination could be specific enough to identify individuals. For example, it has been shown that 87% of individuals in the United States can be uniquely identified by a set of attributes such as {ZIP, gender, date of birth} [12]. This implies that each attribute alone may not be specific enough to identify individuals, but a particular group of attributes together may identify a particular individuals [10, 12].

The main objective of the k-anonymity model is thus to transform a table so that no one can make high-probability associations between records in the table and the corresponding individuals by using such group of attributes, called *quasi-identifier*. In order to achieve this goal, the k-anonymity model requires that any record in a table be indistinguishable from at least $(k-1)$ other records with respect to the quasi-identifier. A set of records that are indistinguishable from each other is often referred to as an *equivalence class*. Thus, a k-anonymous table can be viewed as a set of equivalence classes, each of which contains at least k records. The enforcement of k-anonymity guarantees that even though an adversary knows the quasi-identifier value of an individual and is sure that a k-anonymous table T contains the record of the individual, he cannot determine which record in T corresponds to the individual with a probability greater than $1/k$.

Although the k-anonymity model does not consider sensitive attributes, a private dataset typically contains some sensitive attributes that are not part of the quasi-identifier. For instance, in patient table, *Diagnosis* is considered a sensitive attribute. For such datasets, the key consideration of anonymization is the protection of individuals' sensitive attributes. However, the k-anonymity model does not provide sufficient protection in this setting, as it is possible to infer certain individuals' attributes without precisely re-identifying their records. For instance, consider a k-anonymized table where all records in an equivalence class have the same sensitive attribute value. Although none of these records can be uniquely matched with the corresponding individuals, their sensitive attribute value can be inferred with probability 1. Recently, Machanavajjhala et al. [8] pointed out such inference issues in the k-anonymity model and proposed the notion of ℓ-diversity. Several formulations of ℓ-diversity are introduced in [8]. In its simplest form, the ℓ-diversity model requires that records in each equivalence class have at least ℓ distinct sensitive attribute values. As this requirement ensures that every equivalence class contains at least ℓ distinct sensitive attribute values, the risk of attribute disclosure is kept under $1/\ell$. Note that in this case, the ℓ-diversity requirement also ensures ℓ-anonymity, as the size of every equivalence class must be greater than or equal to ℓ. Although simple and intuitive, modified datasets based on this requirement could still be vulnerable to probabilistic inferences. For example, consider that among the ℓ distinct values in an equivalence class, one particular value appears much more frequently than the others. In such a case, an adversary may conclude that the individuals contained in the equivalence class are very likely to have that specific value. A more robust diversity is achieved by enforcing entropy ℓ-diversity [8], which requires every equivalence class to satisfy the following condition.

$$-\sum_{s \in S} p(e,s)\ log\ p(e,s)\ >\ log\ \ell$$

where S is the domain of the sensitive attribute and $p(e,s)$ represents the fraction of records in e that have sensitive value s. Although entropy ℓ-diversity does provide stronger privacy, the requirement may sometimes be too restrictive. For instance, as pointed out in [8], in order for entropy ℓ-

diversity to be achievable, the entropy of the entire table must also be greater than or equal to $log\ \ell$.

2.2 Anonymization Techniques

The k-anonymity (and ℓ-diversity) requirement is typically enforced through *generalization*, where real values are replaced with "less specific but semantically consistent values" [12]. Given a domain, there are various ways to generalize the values in the domain. Intuitively, numeric values can be generalized into intervals (e.g., $[11-20]$), and categorical values can be generalized into a set of possible values (e.g., {USA, Canada, Mexico}) or a single value that represents such a set (e.g., North-America). As generalization makes data more vague, the utility of the data is inevitably downgraded. The key challenge of anonymization is thus to minimize the amount of ambiguity introduced by generalization while enforcing anonymity requirement.

Various generalization strategies have been developed. In the *hierarchy-based generalization* schemes, a non-overlapping generalization-hierarchy is first defined for each attribute of quasi-identifier. Then an algorithm in this category tries to find an optimal (or good) solution which is allowed by such generalization hierarchies. Here an optimal solution is a solution that satisfies the privacy requirement and at the same time minimizes a desired cost metric. Based on the use of generalization hierarchies, the algorithms in this category can be further classified into two subclasses. In the *single-level generalization* schemes [6, 10, 11], all the values in a domain are generalized into a single level in the corresponding hierarchy. This restriction could be a significant drawback in that it may lead to relatively high data distortion due to unnecessary generalization. The *multi-level generalization* [4, 5] schemes, on the other hand, allows values in a domain to be generalized into different levels in the hierarchy. Although this leads to much more flexible generalization, possible generalizations are still limited by the imposed generalization hierarchies.

Recently, *hierarchy-free generalization* schemes [1, 2, 7] have been proposed, which do not rely on the notion of pre-defined generalization hierarchies. In [1], Bayardo et al. propose an algorithm based on a powerset search problem, where the space of anonymizations (formulated as the powerset of totally ordered values in a dataset) is explored using a tree-search strategies. In [7], LeFevre et al. transform the k-anonymity problem into a partitioning problem and proposes a greedy approach that recursively splits a partition at the median value until no more split is allowed with respect to the k-anonymity requirement. [2], on the other hand, introduces a flexible k-anonymization approach which uses the idea of clustering to minimize information loss and thus ensures good data quality.

3. PROBLEM FORMULATION

In this section, we start with an example to illustrate the problem of inference. We then describe our notion of incremental dissemination and formally define a privacy requirement for it.

3.1 Motivating Examples

Let us revisit our previous scenario where a hospital needs to provide the anonymized version of its patient records with a disease control agency. As previously discussed, to assure data quality, the hospital anonymizes the patient table whenever it is augmented with new records. To make our example more concrete, suppose that the hospital relies on a model where both the k-anonymity and ℓ-diversity are considered; therefore, a '(k, ℓ)-anonymous' dataset is a dataset that satisfies both the k-anonymity and ℓ-diversity requirements. The hospital initially has a table like the one in Figure 1 and reports to the agency its $(2, 2)$-anonymous table shown in Figure 2. As shown, the probability of identity disclosure (i.e., the association between individual and record) and attribute disclosure (i.e., the association between individual and diagnosis) are kept under $1/2$ in the dataset, respectively. For example, even if an attacker knows that the record of Tom, who is a 21-year-old male, is in the released table, he cannot learn about Tom's disease with a probability greater than $1/2$ (although he learns that Tom has either asthma or flu). At a later time, three more patient records (shown in *Italic*) are inserted into the dataset, resulting the table in Figure 3. The hospital then releases a new $(2, 2)$-anonymous table as depicted in Figure 4. Observe that Tom's privacy is still protected in the newly released dataset. However, not every patient's privacy is protected from the attacker.

NAME	AGE	Gender	Diagnosis
Tom	21	Male	Asthma
Mike	23	Male	Flu
Bob	52	Male	Alzheimer
Eve	57	Female	Diabetes

Figure 1: Patient table

AGE	Gender	Diagnosis
$[21-25]$	Male	Asthma
$[21-25]$	Male	Flu
$[50-60]$	Person	Alzheimer
$[50-60]$	Person	Diabetes

Figure 2: Anonymous patient table

Example 1. "Alice has cancer!" Suppose the attacker knows that Alice, who is in her late twenties, has recently been admitted to the hospital. Thus, he knows that Alice's record is not in the old dataset in Figure 2, but in the new dataset in Figure 4. From the new dataset, he learns only that Alice has one of {Asthma, Flu, Cancer}. However, by consulting the previous dataset, he can easily deduce that Alice has neither asthma nor flu (as they must belong to patients other than Alice). He now infers that Alice has cancer.

Example 2. "Bob has alzheimer!" The attacker knows that Bob is 52 years old and has long been treated in the hospital. Thus, he is sure that Bob's record is in both datasets in Figures 2 and 4. First, by studying the old dataset, he learns that Bob suffers from either alzheimer or diabetes. Now the attacker checks the new dataset and learns that Bob has either alzheimer or heart disease. He can thus conclude that Bob suffers from alzheimer. Note that three other records in the new dataset are also vulnerable to similar inferences.

As shown in the examples above, anonymizing a dataset without considering previously released information may enable various inferences.

NAME	AGE	Gender	Diagnosis
Tom	21	Male	Asthma
Mike	23	Male	Flu
Bob	52	Male	Alzheimer
Eve	57	Female	Diabetes
Alice	*27*	*Female*	*Cancer*
Hank	*53*	*Male*	*Hepatitis*
Sal	*59*	*Female*	*Flu*

Figure 3: Updated patient table

AGE	Gender	Diagnosis
$[21-30]$	Person	Asthma
$[21-30]$	Person	Flu
$[21-30]$	Person	Cancer
$[51-55]$	Male	Alzheimer
$[51-55]$	Male	Hepatitis
$[56-60]$	Female	Flu
$[56-60]$	Female	Diabetes

Figure 4: Updated anonymous patient table

3.2 Incremental data dissemination and privacy requirement

Let T be a private table with a set of quasi-identifier attributes Q and a sensitive attribute S. We assume that T consists of person-specific records, each of which corresponds to a unique real-world individual. We also assume that T continuously grows with new records and denote the state of T at time i as T_i. For the privacy of individuals, each T_i must be "properly" anonymized before being released to public. Our goal is to address both identity disclosure and attribute disclosure, and we adopt an anonymity model[†]that combines the requirements of k-anonymity and ℓ-diversity as follows.

Definition 1. (**(k, c)-Anonymity**) Let table T be with a set of quasi-identifier attributes Q and a sensitive attribute S. With respect to Q, T consists of a set of non-empty equivalence classes, where $\forall\ e \in T$, record $r \in e \Rightarrow r[Q] = e[Q]$. We say that T is (k, c)*-anonymous* with respect to Q if the following conditions are satisfied.

1. $\forall\ e \in T, |e| \geq k,$ where $k > 0$.
2. $\forall\ e \in T, \frac{|\{r | r \in e \wedge r[S]=s\}|}{|e|} \leq c,$ where $0 < c \leq 1$.

The first condition ensures the k-anonymity requirement, and the second condition enforces the diversity requirement in the sensitive attribute. In its essence, the second condition dictates that the maximum confidence of association between any quasi-identifier value and a particular sensitive attribute value in T must not exceed a threshold c.

At a given time i, only an (k, c)-anonymous version of T_i, denoted as $\widehat{T_i}$, is released to public. Thus, users, including potential attackers, may have access to a series of (k, c)-anonymous tables, $\widehat{T}_1, \widehat{T}_2, \ldots$, where $|\widehat{T_i}| \leq |\widehat{T_j}|$ for $i < j$. As every released table is (k, c)-anonymous, by observing each table independently, one cannot associate a record with a particular individual with probability higher than $1/k$ or infer any individual's sensitive attribute with confidence higher than c. However, as shown in Section 3.1, it is possible that one can increase the confidence of such undesirable inferences by observing difference between the released tables. For instance, if an observer can be sure that two (anonymized) records in two different versions indeed correspond to the same individual, then he may be able to use this knowledge to infer more information than what is allowed by the (k, c)-anonymity protection. If such a case occurs, we say that there is an inference channel between the two versions.

[†]A similar model is also introduced in [14].

Definition 2. (**Cross-version inference channel**) Let $\Theta = \{\widehat{T}_1, \ldots, \widehat{T}_n\}$ be the set of all released tables for private table T, where $\widehat{T_i}$ is an (k, c)-anonymous version released at time i, $1 \leq i \leq n$. Let $\theta \subseteq \Theta$ and $\widehat{T_i} \in \Theta$. We say that there exists *cross-version inference channel* from θ to $\widehat{T_i}$, denoted as $\theta \rightarrowtail \widehat{T_i}$, if observing tables in θ and $\widehat{T_i}$ collectively increases the risk of either identity disclosure or attribute disclosure in $\widehat{T_i}$ higher than $1/k$ or c, respectively.

When data are disseminated incrementally, it is critical to ensure that there is no cross-version inference channel among the released tables. In other words, the data provider must make sure not only that each released table is free of undesirable inferences, but also that no released table creates cross-version inference channels with respect to the previously released tables. We formally define this requirement as follows.

Definition 3. (**Privacy-preserving incremental data dissemination**) Let $\Theta = \{\widehat{T}_0, \ldots, \widehat{T}_n\}$ be the set of all released tables of private table T, where $\widehat{T_i}$ is an (k, c)-anonymous version of T released at time $i, 0 \leq i \leq n$. Θ is said to be *privacy-preserving* if and only if $\nexists\ (\theta,\ \widehat{T_i})$ such that $\theta \subseteq \Theta$, $\widehat{T_i} \in \Theta$, and $\theta \rightarrowtail \widehat{T_i}$.

4. CROSS-VERSION INFERENCES

We first describe potential attackers and their knowledge that we assume in this paper. Then based on the attack scenario, we identify three types of cross-version inference attacks in this section.

4.1 Attack scenario

We assume that the attacker has been keeping track of all the released tables; he thus possesses a set of released tables $\{\widehat{T}_0, \ldots, \widehat{T}_n\}$, where $\widehat{T_i}$ is a table released at time i. We also assume that the attacker has the knowledge of who is and who is not contained in each table; that is, for each anonymized table $\widehat{T_i}$, the attacker also possesses a population table U_i which contains the explicit identifiers and the quasi-identifiers of the individuals in $\widehat{T_i}$. This may seem to be too farfetched at first glance; however, we assume the worst case, as we cannot rely on attacker's lack of knowledge. Also, such knowledge is not always difficult to acquire for a dedicated attacker. For instance, consider medical records released by a hospital. Although the attacker may not be aware of all the patients, he may know when target individuals in whom he is interested (e.g., local celebrities) are admitted to the hospital. Based on this knowledge, the attacker can easily deduce which tables may include such individuals and which tables may not. Another, perhaps the worst, possibility is that the attacker may collude with an insider who has access to detailed information about the patients; e.g., the attacker could obtains a list of patients from a registration staff. Thus, it is reasonable to assume that the attacker's knowledge includes the list of individuals contained in each table as well as their quasi-identifier values. However, as all the released tables are (k, c)-anonymous, the attacker cannot infer the individuals' sensitive attribute values with a significant probability, even utilizing such knowledge. That is, in each released table, the probability that an individual with a certain quasi-identifier has a particular sensitive attribute is still bound to c. Therefore, the goal of the attacker is to increase his/her confidence of attribute disclosure (i.e., above c) by comparing the released tables all together. In the remainder of this section, we describe three types of cross-version inferences that the attacker may exploit in order to achieve this goal.

4.2 Notations

We first introduce some notations we use in our discussion. Let T be a table with a set of quasi-identifer attributes Q and a sensitive attribute S. Let A be a set of attributes, where $A \subseteq (Q \cup S)$. Then $T[A]$ denotes the duplicate-eliminating projection of T onto the attributes A. Let $e_i = \{r_0, \ldots, r_m\}$ be an equivalence class in T, where $m > 0$. By definition, the records in e_i all share the same quasi-identifier value, and $e_i[Q]$ represents the common quasi-identifier value of e_i. We also use similar notations for individual records; that is, for record $r \in T$, $r[Q]$ represents the quasi-identifier value of r and $r[S]$ the sensitive attribute value of r. In addition, we use $T\langle A\rangle$ to denote the duplicate-preserving projection of T. For instance, $e_i\langle S\rangle$ represents the multiset of all the sensitive attribute values in e_i. We also use $|\mathcal{N}|$ to denote the cardinalities of set $\mathcal{N}$.

Regardless of recoding schemes, we consider a generalized value as a set of possible values. Suppose that v is a value from domain $\mathcal{D}$ and $\widehat{v}$ a generalized value of v. Then we denote this relation as $v \preceq \widehat{v}$, and interpret $\widehat{v}$ as a set of values where $(v \in \widehat{v}) \wedge (\forall v_i \in \widehat{v},\ v_i \in \mathcal{D})$. Overloading this notation, we say that $\widehat{r}$ is a generalized version of record r, denoted as $r \preceq \widehat{r}$, if $(\forall q_i \in Q,\ r[q_i] \preceq \widehat{r}[q_i]) \wedge (r[S] = \widehat{r}[S])$. Moreover, we say that two generalized values $\widehat{v_1}$ and $\widehat{v_2}$ are *compatible*, denoted as $\widehat{v_1} \bowtie \widehat{v_2}$, if $\widehat{v_1} \cap \widehat{v_2} \neq \emptyset$. Similarly, two generalized records $\widehat{r_1}$ and $\widehat{r_2}$ are compatible (i.e., $\widehat{r_1} \bowtie \widehat{r_2}$) if $\forall q_i \in Q,\ \widehat{r_i}[q_i] \cap \widehat{r_j}[q_i] \neq \emptyset$. We also say that two equivalence classes e_1 and e_2 are compatible if $\forall q_i \in Q,\ e_1[q_i] \cap e_2[q_i] \neq \emptyset$

4.3 Difference attack

```
**Check_Difference_Attack**
**Input:** Two $(k,c)$-anonymous tables $\widehat{T}_i$ and $\widehat{T}_j$
**Output:** *true* if the two tables are vulnerable to difference attack and *false* otherwise

if (DirectedCheck($\widehat{T}_i, \widehat{T}_j$) = true) return true
else return DirectedCheck($\widehat{T}_j, \widehat{T}_i$)
end_if
```

Figure 5: Algorithm for checking if given two tables are vulnerable to difference attack

Let $\widehat{T}_i = \{e_{0,1}, \ldots, e_{0,n}\}$ and $\widehat{T}_j = \{e_{1,1}, \ldots, e_{1,m}\}$ be two (k, c)-anonymous tables that are released at time i and j $(i \neq j)$, respectively. As previously discussed in Section 4.1, we assume that an attacker knows who is and who is not in each released table. Also knowing the quasi-identifier values of the individuals in $\widehat{T}_i$ and $\widehat{T}_j$, for any equivalence class e in either $\widehat{T}_i$ or $\widehat{T}_j$, the attacker knows the individuals whose records are contained in e. Let $I(e)$ represent the set of individuals in e. With this information, the attacker can now perform difference attack as follows. Let E_i and E_j be two sets of equivalence classes, where $E_i \subseteq \widehat{T}_i$ and $E_j \subseteq \widehat{T}_j$. If $\cup_{e \in E_i} I(e) \subseteq \cup_{e \in E_j} I(e)$, then set $D = \cup_{e \in E_j} I(e) - \cup_{e \in E_i} I(e)$ represents the set of individuals whose records are in E_i, but not in E_j. Furthermore, set $S_D = \cup_{e \in E_j} e\langle S\rangle - \cup_{e \in E_j} e\langle S\rangle$ indicates the set of sensitive attribute values that belong to those individuals in D. Therefore, if D contains less than k records, or if the most frequent sensitive value in S_D appears with a probability larger than c, the (k, c)-anonymity requirement is violated.

The pseudo-code in Figures 5, 6, and 7 gives an algorithm for checking whether two (k, c)-anonymous tables are vulnerable to the difference attack. The *Check_Difference_Attack* procedure in Figure 5 is the main procedure that checks for the vulnerability between two tables bidirectionally.

```
**DirectedCheck**
**Input:** Two $(k,c)$-anonymous tables $\widehat{T}_i$ and $\widehat{T}_j$, where
$\widehat{T}_i = \{e_{i,1}, \ldots, e_{i,m}\}$ and $\widehat{T}_j = \{e_{j,1}, \ldots, e_{j,n}\}$
**Output:** *true* if $\widehat{T}_i$ is vulnerable to difference attack
with respect to $\widehat{T}_j$ and *false* otherwise

$Q = \{(\emptyset, 0)\}$
while $Q \neq \emptyset$
  Remove the first element $p = (E, index)$ from Q
  if $E \neq \emptyset$
    $E' \leftarrow GetMinSet(E, \widehat{T}_j)$
    if $|E' - E| < k$
      return true
    else if $(E'\langle S\rangle - E\langle S\rangle)$ does not satisfy $c$-diversity
      return true
    end_if
  end_if
  for each $\ell \in \{index + 1, \ldots, m\}$ // generates subsets
with size $|E| + 1$
    insert $(E \cup e_{i,\ell},\ \ell)$ into Q
  end_for
end_while
return false
```

Figure 6: Algorithm for checking if one table is vulnerable to difference attack with respect to another table

The *DirectedCheck* procedure in Figure 6 checks if the first table of the input is vulnerable to difference attack with respect to the second table. This procedure enumerates all the subset equivalence class sets of $\widehat{T}_i$, and for each set E, the procedure calls *GetMinSets* procedure in Figure 7 to get the minimum set E' of equivalence classes in $\widehat{T}_j$ that contains all the records in E. We call such E' the *minimum covering set* of E. The procedure then checks whether there is vulnerability between the two equivalence classes E and E'. As the algorithm checks the all the subsets of $\widehat{T}_i$, the time complexity is apparently exponential (i.e, it is $O(2^n)$, where n is the number of equivalence classes in $\widehat{T}_i$). As such, for large tables with many equivalence classes, this brute-force check is clearly unacceptable. In what follows, we discuss a

```
GetMinSet
Input: An equivalence class set $E$ and a table $\widehat{T}$
Output: An equivalence class set $E' \subseteq \widehat{T}$ that is
minimal and contains all records in $E$

$E' = \emptyset$
while $E \not\subseteq E'$
  choose a tuple $t$ in $E$ that is not in $E'$
  find the equivalence class $e \in \widehat{T}$ that contains $t$
  $E' = E' \cup \{e\}$
end_while
return $E$
```

Figure 7: Algorithm for obtaining a minimum equivalence class set that contains all the records in the given equivalence class set

few observations that result in effective heuristics to reduce the space of the problem in most cases.

Observation 1. Let E_1 and E_2 be two sets of equivalence classes in $\widehat{T}_i$, and E'_1 and E'_2 be their minimal covering sets in $\widehat{T}_j$, respectively. If E_1 and E_2 are not vulnerable to difference attack, and $E'_1 \cap E'_2 = \emptyset$, then we do not need to consider any subset of $\widehat{T}_i$ which contains $E_1 \cup E_2$.

That E_1 and E_2 are not vulnerable to difference attack means that sets $(E'_1 - E_1)$ and $(E'_2 - E_2)$ are both (k, c)-anonymous. As the minimum covering set of $E_1 \cup E_2$ is $E'_1 \cup E'_2$, and E'_1 and E'_2 are disjoint, $(E'_1 \cup E'_2) - (E_1 \cup E_2)$ is also (k, c)-anonymous. This also implies that if each E_1 and E_2 is not vulnerable to difference attack, then neither is any set containing $E_1 \cup E_2$. Based on this observation, we can modify the method in Figure 6 as follows. In each time we union one more element to a subset to create larger subsets, we check if their minimum covering sets are disjoint. If they are, we do not insert the unioned subset to the queue. Note that this also prevents all the sets containing the unioned set from being generated.

Observation 2. Let E_1 and E_2 be two sets of equivalence classes in $\widehat{T}_i$, and E'_1 and E'_2 be their minimal covering sets in $\widehat{T}_j$, respectively. If $E'_1 = E'_2$, then we only need to check if $E_1 \cup E_2$ is vulnerable to difference attack.

In other words, we can skip checking if each of E_1 and E_2 is vulnerable to difference attack. This is because unless $E_1 \cup E_2$ is vulnerable to difference attack, E_1 and E_2 must not be vulnerable. Thus, we can save our computational effort as follows. When we insert a new subset to the queue, we check if there exists another set with the same minimum covering set. If such a set is found, we simply merge the new subset to the found set.

Observation 3. Consider the method in Figure 6. Suppose that $\widehat{T}_i$ was released after $\widehat{T}_j$; that is, $\widehat{T}_i$ contains some records that are not in $\widehat{T}_j$. If equivalence class $e \in \widehat{T}_j$ contains such records, then we do not need to consider that equivalence class for difference attack.

It is easy to see that if $e \in \widehat{T}_i$ contains some record(s) that $\widehat{T}_j$ do not, the minimum covering set of e is an empty-set. Since e itself must be (k, c)-anonymous, e is safe from difference attack. Based on this observation, we can purge all such equivalence classes from the initial problem set. As the method in Figure 5 shows, our algorithm checks two tables in both directions. While it may seem that this doubles the already-heavy computation, this observation relieves such concern.

4.4 Intersection attack

The key idea of k-anonymity is to introduce sufficient ambiguity into the association between quasi-identifier values and sensitive attribute values. However, this ambiguity may be reduced to an undesirable level if the structure of equivalence classes are varied in different releases. For instance, suppose that the attacker wants to know the sensitive attribute of Alice, whose quasi-identifier value is q_A. Then the attacker can select a set of tables, θ^+_A, that all contain Alice's record. As the attacker knows the quasi-identifier of Alice, he does not need to examine all the records; he just needs to consider the records that may possibly correspond to Alice. That is, in each $\widehat{T}_i \in \theta^+_A$, the attacker only need to consider an equivalence class $e_i \subseteq \widehat{T}_i$, where $q_A \preceq e_i[Q]$. Let $E_A = \{e_0, \ldots, e_n\}$ be the set of all equivalence classes identified from θ^+_A such that $q_A \preceq e_i[Q]$, $0 \le i \le n$. As every e_i is (k, c)-anonymous, the attacker cannot infer Alice's sensitive attribute value with confidence higher than c by examining each e_i independently. However, as every equivalence class in E_A contains Alice's record, the attacker knows that Alice's sensitive attribute value, s_A, must be present in every equivalence class in E_A; i.e., $\forall e_i \in E_A, s_A \in e_i\langle S\rangle$. This implies that s_A must be found in set $SI_A = \bigcap_{e_i \in E_A} e_i\langle S\rangle$. Therefore, if the most frequent value in SI_A appears with a probability greater than c, then the sensitive attribute value of Alice can be inferred with confidence greater than c.

```
Check_Intersection_Attack
Input: Two $(k, c)$-anonymous tables $\widehat{T}_0$ and $\widehat{T}_1$
Output: true if the two tables are vulnerable to
intersection attack and false otherwise

for each equivalence class $e_{0,i}$ in $\widehat{T}_0$
    for each $e_{1,j} \in \widehat{T}_1$ that contains any record in $e_{0,i}$
      if $(e_{0,i}\langle S\rangle \cap e_{1,j}\langle S\rangle)$ does not satisfy $c$-diversity
        return true
      end_if
    end_for
end_for
return false
```

Figure 8: Algorithm for checking if given two tables are vulnerable to intersection attack

The pseudo-code in Figure 8 provides an algorithm for checking the vulnerability to the intersection attack for given two (k, c)-anonymous tables, $\widehat{T}_0$ and $\widehat{T}_1$. The basic idea is to check every pair of equivalence classes $e_i \in \widehat{T}_0$ and $e_j \in \widehat{T}_1$ that contain the same record(s).

4.5 Record-tracing attack

Unlike the previous attacks, the attacker may be interested in knowing who may be associated with a particular attribute value. In other words, instead of wanting to know what sensitive attribute value a particular individual has, the attacker now wants to know which individuals possess a specific sensitive attribute value; e.g., the individuals who

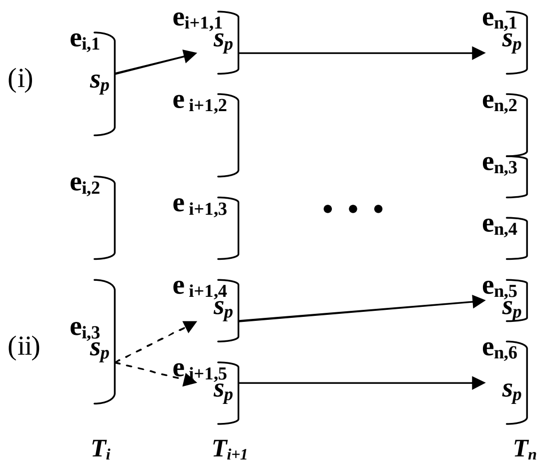

Figure 9: Record-tracing attacks

suffer from 'HIV+'. Let s_p be the sensitive attribute value in which the attacker is interested and $\widehat{T}_i \in \Theta$ be the table in which (at least) one record with sensitive value s_p appears. Although $\widehat{T}$ may contain more than one record with s_p, suppose, for simplicity, that the attacker is interested in a particular record r_p such that $(r_p[S] = s_p) \wedge (r_p \in e_i)$. As $\widehat{T}$ is (k, c)-anonymous, when the attacker queries the population table U_i with $r_p[Q]$, he obtains at least k individuals who may correspond to r_p. Let I_p be the set of such individuals. Suppose that the attacker also possesses a subsequently released table $\widehat{T}_j$ $(i < j)$ which includes r_p. Note that in each of these tables the quasi-identifier of r_p may be generalized differently. This means that if the attacker can identify from $\widehat{T}_j$ the record corresponding to r_p, then he may be able to learn additional information about the quasi-identifier of the individual corresponding to r_p and possibly reduce the size of I_p. There are many cases where the attacker can identify r_p in $\widehat{T}_j$. However, in order to illustrate our point clearly, we show some simple cases in the following example.

Example 3. The attacker knows that r_p must be contained in the equivalence class of $\widehat{T}_j$ that is compatible with $r_p[Q]$. Suppose that there is only one compatible equivalence class, e_{i+1} in $\widehat{T}_j$ (see Figure 9 (i)). Then the attacker can confidently combine his knowledge on the quasi-identifier of r_p; i.e., $r_p[Q] \leftarrow r_p[Q] \cap e_{i+1}[Q]$. Suppose now that there are more than one compatible equivalence classes in $\widehat{T}_{i+1}$, say e_{i+1} and e'_{i+1}. If $s_p \in e_{i+1}[S]$ and $s_p \notin e'_{i+1}[S]$, then the attacker can be sure that $r_p \in e_{i+1}$ and updates his knowledge of $r_p[Q]$ as $r_p[Q] \cap e_{i+1}[Q]$. However, if $s_p \in e_{i+1}[S]$ and $s_p \in e'_{i+1}[S]$, then r_p could be in either e_{i+1} and e'_{i+1} (see Figure 9 (ii)). Although the attacker may or may not determine which equivalence class contains r_p, he is sure that $r_p \in e_{i+1} \cup e'_{i+1}$; therefore, $r_p[Q] \leftarrow r_p[Q] \cap (e_{i+1}[Q] \cup e'_{i+1}[Q])$.

After updating $r_p[Q]$ with $\widehat{T}_j$, the attacker can reexamine I_p and eliminate individuals whose quasi-identifiers are no longer compatible with the updated $r_p[Q]$. When the size of I_p becomes less than k, the attacker can infer the association between the individuals in I_p and r_p with a probability higher than $1/k$.

In the above example, when there are more than one compatible equivalence classes $\{e_{i+1,1}, ..., e_{i+1,r}\}$ in $\widehat{T}_{i+1}$, we say that the attacker updates $r_p[Q]$ as $r_p[Q] \cap (\cup_{1 \leq j \leq r} e_{i+1,j})$. While correct, this is not a sufficient description of what the attacker can do, as there are cases where the attacker can notice that some equivalence classes in $\widehat{T}_{i+1}$ cannot contain r_p. For example, let $r_1 \in e_{i,1}$ and $r_2 \in e_{i,2}$ be two records in $\widehat{T}_i$, both taking s_r as the sensitive value (see Figure 10(i)). Suppose that $\widehat{T}_{i+1}$ contains a single equivalence class $e_{i+1,1}$ that is compatible to r_1 and two compatible equivalence classes $e_{i+1,1}$ and $e_{i+1,2}$ that are compatible to r_2. Although r_2 has two compatible equivalence classes, the attacker can be sure that r_2 is included in $e_{i+1,2}$, as the record with s_r in $e_{i+1,1}$ must correspond to r_1. Figure 10(ii) illustrates another case of which the attacker can take advantage. As shown, there are two records in $e_{i,1}$ that take s_r as the sensitive value. Although the attacker cannot be sure that each of these records is contained in $e_{i+1,1}$ or $e_{i+1,2}$, he is sure that one record is in $e_{i+1,1}$ and the other in $e_{i+1,2}$. Thus, he can make an arbitrary choice and update his knowledge about the quasi-identifiers of the two records accordingly. Using such techniques, the attacker can make more precise inference by eliminating equivalence classes in $\widehat{T}_{i+1}$ that are impossible to contain r_p.

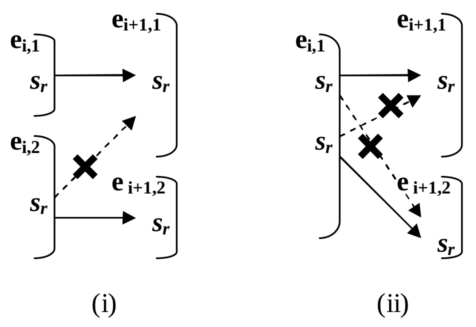

Figure 10: More inference in record-tracing

We now describe a more thorough algorithm that checks two (k, c)-anonymous tables for the vulnerability to the record-tracing attack. First, we construct a bipartite graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$ and each vertex in $\mathcal{V}_1$ represents a record in $\widehat{T}_i$, and each vertex in $\mathcal{V}_2$ represents a record in $\widehat{T}_{i+1}$ that is compatible with at least one record in $\widehat{T}_i$. We define $\mathcal{E}$ as the set of edges from vertices in $\mathcal{V}_1$ to vertices in $\mathcal{V}_2$, which represents possible matching relationships. That is, if there is an edge from $r_i \in \mathcal{V}_1$ to $r_j \in \mathcal{V}_2$, this means that records r_i and r_j may both correspond to the same record although they are generalized into different forms. We create such edges between $\mathcal{V}_1$ and $\mathcal{V}_2$ as follows. For each vertex $r \in \mathcal{V}_1$, we find from $\mathcal{V}_2$ the set of records R where $\forall r_i \in R, (r[Q] \bowtie r_i[Q]) \wedge (r[S] = r_i[S])$. If $|R| = 1$ and $r' \in E$, then we create an edge from r to r' and mark it with $\langle d \rangle$, which indicates that r *definitely* corresponds to r'. If $|R| > 1$, then we create an edge from r and every $r'_i \in R$ and mark it with $\langle p \rangle$ to indicate that r *plausibly* corresponds to r'_i. Now given the constructed bipartite graph, the pseudo-code in Figure 11 removes plausible edges that are not feasible and discovers more definite edges by scanning through the edges.

Note that the algorithm above does not handle the case illustrated in Figure 10(ii). In order to address such cases, we also performs the following. For each equivalence class

```
Remove_Infeasible_Edges
Input: A bipartite graph $G = (V, E)$ where $V = V_1 \cup V_2$
and $E$ is a set of edges representing possible matching
relationships.
Output: A bipartite graph $G' = (V, E')$ where $E' \subset E$
with infeasible edges removed

$E' = E$
while true
  change1 $\leftarrow$ false
  change2 $\leftarrow$ false
  for each $r_j \in V_2$
    $e \leftarrow$ all the incoming edges of $r_j$
    if $e$ contains both a definite edge and plausible edge(s)
      remove all plausible edges in $e$ from $E'$
      change1 $\leftarrow$ true
    end_if
  end_for
  if change1 = true
    for each $r_i \in V_1$
    $e \leftarrow$ all the outgoing edges of $r_i$
    if $e$ contains only a single plausible edge
      mark the edge in $e$ as definite
      change2 $\leftarrow$ true
    end_if
  end_for
  end_if
  if change2 = false
    break
  end_if
end_while
return $(V, E')$
```

Figure 11: Algorithm for removing infeasible edges from a bipartite graph

$e_{1,i} \in \widehat{T}_i$, we find from $\widehat{T}_j$ the set of equivalence classes E where $\forall e_{2,j} \in E, e_{1,i}[Q] \bowtie e_{2,j}[Q]$. If the same number of records with any sensitive value s appear in both $e_{1,i}$ and E, we remove unnecessary plausible edges such that each of such records in $e_{1,i}$ has a definite edge to a distinct record in E.

After all infeasible edges are removed, each record $r_{1,i} \in \mathcal{V}_1$ is associated with a set of possibly matching records $\{r_{2,j}, \ldots, r_{2,m}\}$ $(j \leq m)$ in $\mathcal{V}_2$. Now we can follow the edges and compute for each record $r_{1,i} \in \widehat{T}_i$ the inferrable quasi-identifier $r'_{1,i}[Q] = r_{1,i}[Q] \cap (\bigcup_{\ell=j,\ldots,m} r_{2,\ell}[Q])$. If any inferred quasi-identifer maps to less than k individuals in the population table U_i, then table $\widehat{T}_i$ is vulnerable to the record-tracing attack with respect to $\widehat{T}_j$.

It is worth noting that the key problem enabling the record-tracing attack arises from the fact that the sensitive attribute value of a record, together with its generalized quasi-identifier, may uniquely identify the record in different anonymous tables. This issue can be especially critical for records with rare sensitive attribute values (e.g., rare diseases) or tables where every individual has a unique sensitive attribute value (e.g., DNA sequence).

5. INFERENCE PREVENTION

In this section, we describe our incremental data anonymization which incorporates the inference detection techniques in the previous section. We first describe our data/history management strategy which aims to reduce the computational overheads. Then, we describe the properties of our checking algorithms which make them suitable for existing data anonymization techniques such as full-domain generalization [6] and multidimensional anonymization [7].

5.1 Data/history management

Consider a newly anonymized table, $\widehat{T}_i$, which is about to be released. In order to check whether $\widehat{T}_i$ is vulnerable to cross-version inferences, it is essential to maintain some form of history about previously released datasets, $\Theta = \{\widehat{T}_0, \ldots, \widehat{T}_{i-1}\}$. However, checking the vulnerability in $\widehat{T}_i$ against each table in Θ can be computationally expensive. To avoid such inefficiency, we maintain a history table, H, which has the following attributes.

- *RID* : is a unique record ID (or the explicit identifier of the corresponding individual). Assuming that each $\widehat{T}_i$ also contains *RID* (which is projected out before being released), *RID* is used to join H_i and $\widehat{T}_i$.
- *TS* (Time Stamp) : represents the time (or the version number) when the record is first released.
- *IS* (Inferable Sensitive values) : stores the set of sensitive attribute values with which the record can be associated. For instance, if record r is released in equivalence class e_i of $\widehat{T}_i$, then $r[IS]_i \leftarrow (r[IS]_{i-1} \cap e_i\langle S \rangle)$. This field is used for checking vulnerability to intersection attack.
- *IQ* (Inferable Quasi-identifier) : keeps track of the quasi-identifiers into which the record has previously been generalized. For instance, for record $r \in \widehat{T}_i$, $r[IQ]_i \leftarrow r[IQ]_{i-1} \cap r[Q]$. This field is used for checking vulnerability to record-tracing attack.

The main idea of H is to keep track of the attacker's accumulated knowledge on each released record. For instance, value $r[IS]$ of record $r \in H_{i-1}$ indicates the set of sensitive attribute values that the attacker may be able to associate with r prior to the release of $\widehat{T}_i$. This is indeed the worst case as we are assuming that the attacker possesses every released table, i.e., Θ. However, as discussed in Section 4.1, we need to be conservative when estimating what the attacker can do. Using H, the cost of checking $\widehat{T}_i$ for vulnerability can be significantly reduced; for intersection and record-tracing attacks, we check $\widehat{T}_i$ against H_{i-1}, instead of every $\widehat{T}_j \in \Theta$.[‡]

5.2 Incorporating inference detection into data anonymization

We now discuss how to incorporate the inference detection algorithms into secure anonymization algorithms. We first consider the full-domain anonymization, where all values of an attribute are consistently generalized to the same level in the predefined generalization hierarchy. In [6], LeFevre et al. propose an algorithm that finds minimal generalizations for a given table. In its essence, the proposed algorithm is an bottom-up search approach in that it starts with ungeneralized data and tries to find minimal generalizations by

[‡]In our current implementation, difference attack is still checked against every previously released table.

increasingly generalizing the target data in each step. The key property on which the algorithm relies is generalization property: given a table T and two generalization strategies G_1, G_2 ($G_1 \preceq G_2$), if $G_1(T)$ is k-anonymous, then $G_2(T)$ is also k-anonymous.[§] Although intuitive, this property is critical as it guarantees the optimality to the discovered solutions; i.e., once the search finds a generalization level that satisfies the k-anonymity requirement, we do not need to search further.

Observation 4. Given a table T and two generalization strategies G_1, G_2 ($G_1 \preceq G_2$), if $G_1(T)$ is not vulnerable to any inference attack, then neither is $G_2(T)$.

The proof is simple. As each equivalence class in $G_2(T)$ is the union of one or more equivalence classes in $G_1(T)$, the information about each record in $G_2(T)$ is more vague than that in $G_1(T)$; thus, G_2 does not create more inference attacks than G_1. Based on this observation, we modify the algorithm in [6] as follows. In each step of generalization, in addition to checking the (k, c)-anonymity requirement, we also checks for the vulnerability to inference. If either check fails, then we need to further generalize the data.

Next, we consider the multidimensional k-anonymity algorithm proposed in [7]. Specifically, the algorithm consists of the following two steps. The first step is to find a partitioning scheme of the d-dimensional space, where d is the number of attributes in the quasi-identifier, such that each partition contains more than k records. In order to find such a partitioning, the algorithm recursively splits a partition at the median value (of a selected dimension) until no more split is allowed with respect to the k-anonymity requirement. Note that contrast to the previous algorithm, this algorithm is a top-down search approach, and the quality of the search relies on the following property:[¶] given a partition p, if p does not satisfy the k-anonymity requirement, then any sub-partition of p does not satisfy the requirement.

Observation 5. Given a partition p of records, if p is vulnerable to any inference attack, then so is any sub-partition of p.

Suppose that we have a partition P_1 of the dataset, in which some records are vulnerable to inference attacks. Then, any further cut of P_1 will lead to a dataset that is also vulnerable to inference attacks. This is based on the fact that any further cut on P_1 leads to de-generalization of the dataset; thus, it reveals more information about each record than P_1. Based on this observation, we modify the algorithm in [7] as follow. In each step of partition, in addition to checking the (k, c)-anonymity requirement, we also checks for the vulnerability to inference. If either check fails, then we do not need to further partition the data.

6. EXPERIMENTS

The main goal of our experiments is to show that our approach effectively prevents the previously discussed inference attacks when data is incrementally disseminated. We also show that our approach produces datasets with good data quality. We first describe our experimental settings and then report our experimental results.

[§]This property is also used in [8] for ℓ-diversity and is thus applicable for (k, c)-anonymity.

[¶]It is easy to see that the property also holds for any diversity requirement.

6.1 Experimental Setup

6.1.1 Experimental Environment

The experiments were performed on a 2.66 GHz Intel *IV* processor machine with 1 GB of RAM. The operating system on the machine was Microsoft Windows XP Professional Edition, and the implementation was built and run in Java 2 Platform, Standard Edition 5.0. For our experiments, we used the Adult dataset from the UC Irvine Machine Learning Repository [9], which is considered a de facto benchmark for evaluating the performance of anonymization algorithms. Before the experiments, the Adult data set was prepared as described in [1, 5, 7]. We removed records with missing values and retained only nine of the original attributes. In our experiments, we considered {*age*, *work class*, *education*, *marital status*, *race*, *gender*, *native country*, *salary*} as the quasi-identifier, and *occupation* attribute as the sensitive attribute.

6.1.2 Data quality metrics

The quality of generalized data has been measured by various metric. In our experiment, we measure the data quality mainly based on *Average Information Loss* (*AIL*, for short) metric proposed in [2]. The basic idea of *AIL* metric is that the amount of generalization is equivalent to the expansion of each equivalence class (i.e., the geometrical size of each partition). Note that as all the records in an equivalence class are modified to share the same quasi-identifer, each region indeed represents the generalized quasi-identifier of the records contained in it. Thus, data distortion can be measured naturally by the size of the region covered by each equivalence class. Following this idea, IL measures the amount of data distortion in an equivalence class as follows.

Definition 4. (**Information loss**) [2] Let $e{=}\{r_1, \ldots, r_n\}$ be an equivalence class where $Q{=}\{a_1, \ldots, a_m\}$. Then the amount of data distortion occurred by generalizing e, denoted by $AIL(e)$, is defined as:

$$\text{AIL}(e) = |e| \times \textstyle\sum_{j=1,\ldots,m} \frac{|G_j|}{|D_j|}$$

where $|e|$ is the number of records in e, and $|D_j|$ the domain size of attribute a_j. $|G_j|$ represents the amount of generalization in attribute a_j (e.g., the length of the shortest interval which contains all the a_j values existing in e).

Based on IL, the average information loss of a given table $\widehat{T}$ is computed as: $AIL(\widehat{T}) = (\sum_{e\in\widehat{T}} IL(e)) \,/\, |T|$. The key advantage of AIL metric is that it precisely measures the amount of generalization (or vagueness of data), while being independent from the underlying generalization scheme (e.g, anonymization technique used or generalization hierarchies assumed). For the same reason, we also use the Discernibility Metric (DM) [1] as another quality measure in our experiment. Intuitively, DM measures the quality of anonymized data based on the size of the equivalence classes, which indicates how much records are indistinguishable from each other.

6.2 Experimental Results

We first measured how many records were vulnerable in statically anonymized datasets with respect to the inference attacks we discussed. For this, we modified two k-anonymization algorithms, Incognito [6] and Mondrian [7], and used them as our *static* (k, c)-anonymization algorithms.

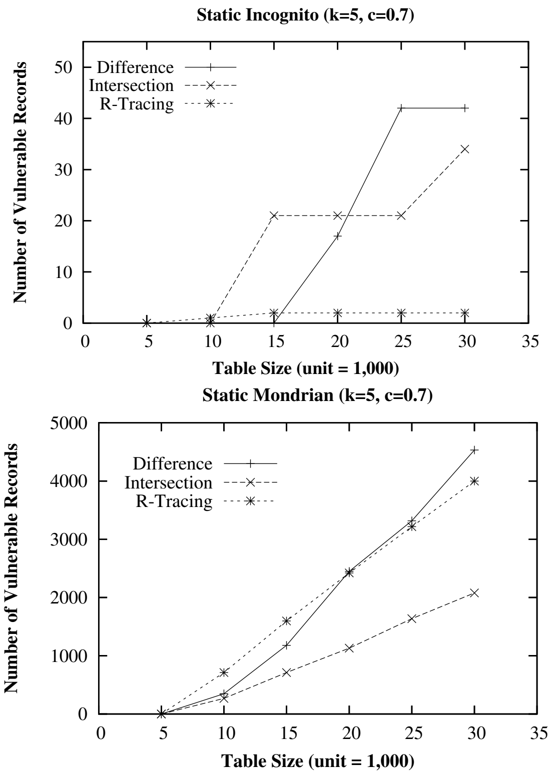

Figure 12: Vulnerability to Inference Attacks

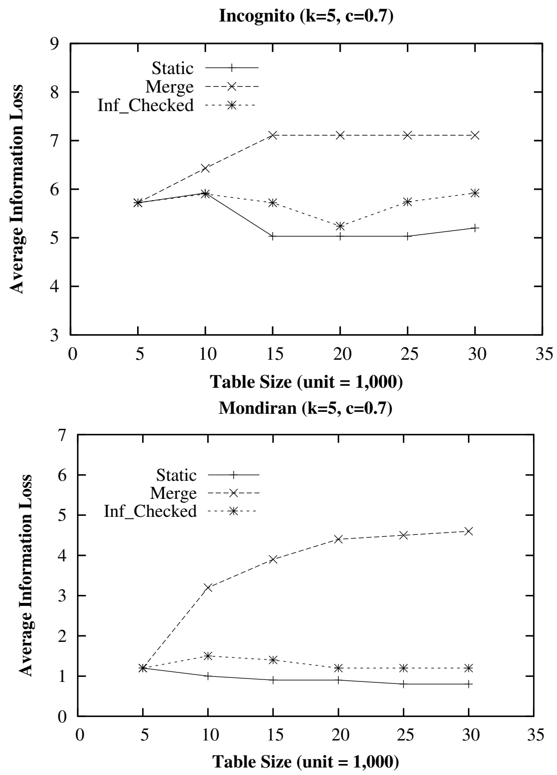

Figure 13: Data Quality: Average Information Loss

Using these algorithms, we first anonymized 5K records and obtained the first "published" datasets. We then generated five more subsequent datasets by adding 5K more records each time. Then we used our vulnerability detection algorithms to count the number of records among these datasets that are vulnerable to each of inference attack. Figure 12 shows the result. As shown, much more records were found to be vulnerable in the datasets anonymized by Mondrian. This is indeed unsurprising, as Mondrian, taking a multidimensional approach, produces datasets with much less generalization. In fact, for Incognito, even the initial dataset was highly generalized. This clearly illustrates the unfortunate reality; that is, the more precise data are, the more vulnerable they are to undesirable inferences.

The next step was to investigate how effectively our approach would work with a real dataset. The main focus was its effect on the data quality. As previously discussed, in order to prevent undesirable inferences, one needs to hide more information. In our case, it means that the given data must be generalized until there is no vulnerability to any type of inference attack. We modified the static (k, c)-anonymization algorithms as discussed in Section 5 and obtained our *inf-checked* (k, c)-anonymization algorithms. Note that although we implemented the full-featured algorithms for difference and intersection attacks, we took a simple approach for record-tracing attack. That is, we considered all the edges without removing infeasible/unnecessary edges as discussed in Section 4.5. We also implemented a *merge* approach where we anonymize each dataset independently and merge it with the previously released dataset. Although this approach is secure from any type of inference attacks, we expected that the data quality would be the worst, as merging would inevitably have a significant effect on generalization (recoding) scheme.

With these algorithms as well as the static anonymization algorithms, we repeated our experiment. As before, we started with 5K records and increased the dataset by 5K each time. We then checked the vulnerability and measured the data quality of such datasets. We measured the data quality both with AIL and DM, and the results are illustrated in Figures 13 and 14, respectively. It is clear that in terms of data quality the inf-checked algorithm is much superior than the merge algorithm. Although the static algorithms produced the best quality datasets, these data are vulnerable to inference attacks as previously shown. The datasets generated by our inf_checked algorithm and the merge algorithm were not vulnerable to any type of inference attack.

We also note that the quality of datasets generated by the inf-checked algorithm is not optimal. This was mainly due to the complexity of checking for difference attack. Even though our heuristics to reduce the size of subsets (see Section 4.3) were highly effective in most cases, there were some cases where the size of subsets grew explosively. As such cases not only caused lengthy execution times, they caused memory blow-ups. In order to avoid such cases, we set an upper limit threshold for the size of subsets in this experiment. For example, while our modified algorithm of Incognito is processing a node in the generalization lattice, if the size of subsets needed to be checked exceeds the threshold, we stop the iteration and consider the node as a vulnerable node. Similarly, when we encounter such a case while considering a split in Mondrian, we stop the check and do not consider the split. Note that this approach does not af-

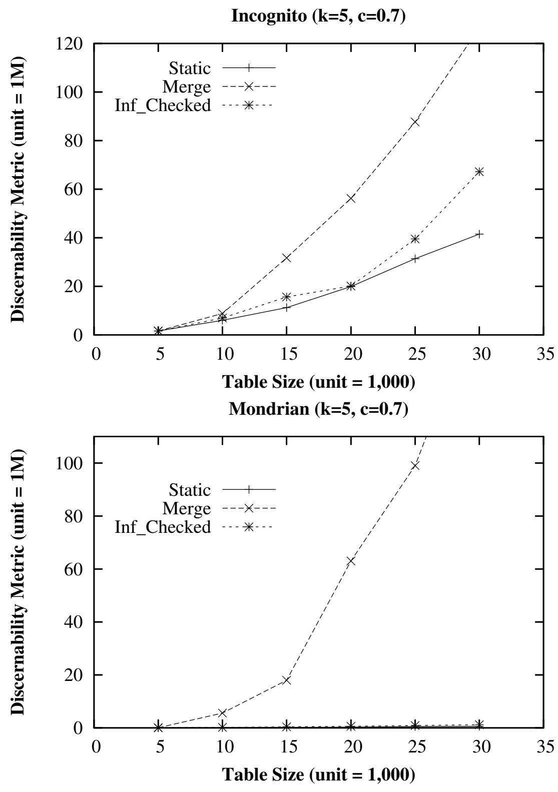

Figure 14: Data Quality: Discernibility Metric

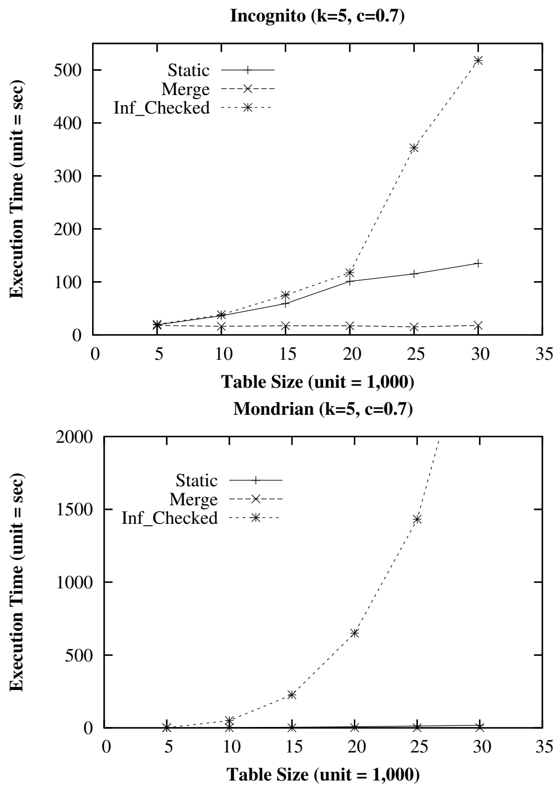

Figure 15: Execution Time

fect the security of data, although it may negatively affect the overall data quality. Even if the optimality cannot be guaranteed, we believe that the data quality seems to be still acceptable, considering the results shown in Figures 13 and 14.

Another important comparison was the computational efficiency of these algorithms. Figure 15 shows our experimental result for each algorithm. The merge algorithm is highly efficient with respect to execution time (although it was very inefficient with respect to data quality). As the merge algorithm anonymizes the same sized dataset each time and merging datasets can be done very quickly, the execution time is closely constant. While equipped the heuristics and the data structure discussed in Sections 4.3 and 5.1, the inf-checked algorithm is still slow. However, considering the previously discussed results, we believe that this is the price you have to pay for better data quality and reliable privacy. Also, when compared to our previous implementation without any heuristics, this is a very promising result.

7. RELATED WORK

While static anonymization has been extensively investigated in the past few years [1, 4, 5, 6, 7, 11], only a few approaches address the problem of anonymization in dynamic environments.

In [12], Sweeney identified possible inferences when new records are inserted and suggested two simple solutions. The first solution is that once records in a dataset are anonymized and released, in any subsequent release of the dataset, the records must be the same or more generalized. As previously mentioned, this approach may suffer from unnecessarily low data quality. Also, this approach cannot protect newly inserted records from difference attack, as discussed in Section 4. The other solution suggested is that once a dataset is released, all released attributes (including sensitive attributes) must be treated as the quasi-identifier in subsequent releases. This approach seems reasonable as it may effectively prevent linking between records. However, this approach has a significant drawback in that every equivalence class will inevitable have a homogeneous sensitive attribute value; thus, this approach cannot adequately control the risk of attribute disclosure.

Yao et al. [15] addressed the inference issue when a single table is released in the form of multiple views. They proposed several methods to check whether or not a given set of views violates the k-anonymity requirement collectively. However, they did not address how to deal with such violations. Recently, Wang and Fung [13] further investigated this issue and proposed a top-down specialization approach to prevent record-linking across multiple anonymous tables. However, their work does not address how to protect records that are newly inserted to the dataset.

Recently, Wang and Fung [13] further investigated this issue and proposed a top-down specialization approach to prevent record-linking across multiple anonymous tables. However, their work focuses on the horizontal growth of databases (i.e., addition of new attributes), and does not address vertically-growing databases where records are inserted.

In [3], we presented a preliminary limited investigation concerning the inference problem of dynamic anonymization with respect to incremental datasets. We identified some inferences and also proposed an approach where new records are directly inserted to the previously anonymized dataset for computational efficiency. However, compared to this current work, our previous work has several limitations.The key differences of this work with respect to [3] are as follows. In [3], we focused only on the *inference enabling sets* that may exist between two tables, while in this work we consider more robust and systematic inference attacks in a collection of released tables. The inference attacks discussed in this work subsume attacks using inference enabling sets and address more sophisticated inferences. For instance, our study of the record-tracing attack is a new contribution in this work. We also provide a detailed descriptions of attacks and algorithms for detecting them. Our previous approach was also limited to the multidimensional generalization. By contrast, our current approach considers and is applicable to both the full-domain and multidimensional approaches; therefore it can combined with a large variety of anonymization algorithms. In this paper we also address the issue of computational costs in detecting possible inferences. We discuss various heuristics to significantly reduce the search space, and also suggest a scheme to store the history (of previously released tables).

8. CONCLUSIONS

In this paper, we discussed inference attacks against the anonymization of incremental data. In particular, we discussed three basic types of cross-version inference attacks and presented algorithms for detecting each attack. We also presented some heuristics to address the efficiency of our algorithms. Based on these ideas, we developed secure anonymization algorithms for incremental datasets using two existing anonymization algorithms. We also empirically evaluated our approach by comparing to other approaches. Our experimental result showed that our approach outperformed other approaches in terms of privacy and data quality.

For the future work, we are working on essential properties (e.g, correctness) of our methods and analysis. Another interesting direction for the further work is to see if there are other types of inferences. For instance, one can devise an attack where more than one type of inference are jointly utilized. We also plan to investigate inference issues in more dynamic environments where deletions and updates of records are allowed.

9. REFERENCES

[1] R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *the 21st International Conference on Data Engineering*, Washington, DC, USA, 2005. IEEE Computer Society.

[2] J. Byun, A. Kamra, E. Bertino, and N. Li. Efficient k-anonymization using clustering techniques. Technical Report 2006-10, Purdue University, 2006.

[3] J. Byun, Y. Sohn, E. Bertino, and N. Li. Secure anonymization for incremental datasets. In *the 3rd VLDB Workshop on Secure Data Management*, San Diego, CA, USA, 2006. Springer-Verlag.

[4] B. C. M. Fung, K. Wang, and P. S. Yu. Top-down specialization for information and privacy preservation. In *the 21st International Conference on Data Engineering*, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[5] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *ACM Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2002. ACM Press.

[6] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *ACM International Conference on Management of Data*, New York, NY, USA, 2005. ACM Press.

[7] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *the 22nd International Conference on Data Engineering*, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[8] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. In *the 22nd International Conference on Data Engineering*, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[9] C. B. S. Hettich and C. Merz. UCI repository of machine learning databases, 1998.

[10] P. Samarati. Protecting respondent's privacy in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

[11] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 5(10):571–588, 2002.

[12] L. Sweeney. K-anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 5(10):557–570, 2002.

[13] K. Wang and B. Fung. Anonymizing sequential releases. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2006. ACM Press.

[14] 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 *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2006. ACM Press.

[15] C. Yao, X. S. Wang, and S. Jajodia. Checking for k-anonymity violation by views. In *the 31st International Conference on Very Large Data Bases*, Saratoga, CA, USA, 2005. VLDB Endowment.