

Maintaining K -Anonymity against Incremental Updates

Jian Pei¹ Jian Xu² Zhibin Wang² Wei Wang² Ke Wang¹
¹ Simon Fraser University, Canada, {jpei, wangk}@cs.sfu.ca
² Fudan University, China, {xujian, 052021125, weiwang1}@fudan.edu.cn

Abstract

K-anonymity is a simple yet practical mechanism to protect privacy against attacks of re-identifying individuals by joining multiple public data sources. All existing methods achieving k-anonymity assume implicitly that the data objects to be anonymized are given once and fixed. However, in many applications, the real world data sources are dynamic.

In this paper, we investigate the problem of maintaining k-anonymity against incremental updates, and propose a simple yet effective solution. We analyze how inferences from multiple releases may temper the k-anonymity of data, and propose the monotonic incremental anonymization property. The general idea is to progressively and consistently reduce the generalization granularity as incremental updates arrive. Our new approach guarantees the k-anonymity on each release, and also on the inferred table using multiple releases. At the same time, our new approach utilizes the more and more accumulated data to reduce the information loss.

1 Introduction

Privacy protection for individuals has become a serious concern in many applications. Particularly, protection against privacy attacks of re-identifying individuals by joining multiple public data sources has been emphasized in extensive practice. For example, according to [17], more than 85% of the population of the United States can be uniquely identified using the combination of their zipcode, gender, and date of birth.

To protect privacy against this type of attacks, the mechanism of k -anonymity was proposed [14, 17]. A data set is k -anonymous ($k \geq 1$) if, on the *quasi-identifier attributes* which are the minimal set of attributes in the table that can be joined with external information to re-identify individual records, each record in the data set is indistinguishable from at least $(k - 1)$ other records within the same data set. The larger the value of k , the better the privacy is protected.

Since the concept of k -anonymity has been proposed, a few k -anonymization algorithms have been developed (please see a brief review in Section 3). Generally, to achieve k -anonymity, those methods generalize or suppress some values in the quasi-identifiers. The major goal of the anonymization methods is to meet the k -anonymity requirement and simultaneously minimize the information loss or maximize the utility of the anonymized data.

Real world data sources are often dynamic. However, all of the existing methods assume that *the set of objects to be anonymized are given once and fixed*. This assumption often heavily constrains the applicability of those anonymization methods in many dynamic applications.

Example 1 (Motivation) Suppose a medical management board maintains a database of cases of certain types of diseases. To protect the privacy, the cases have to be anonymized before being released.

The database is incrementally updated periodically. In order to use the existing methods to publish the updates, two approaches may be developed.

Method 1: Anonymizing incremental updates. Once a batch of new cases are inserted, we may anonymize the batch so that it satisfies the k -anonymity. To elaborate, suppose Table 1 records the cases when the database is created, and the 2-anonymity is required. Table 2 is a 2-anonymization of Table 1.

After an incremental update, two new cases (Eddy and Frank in Table 3, shown in bold) are added. Table 4 shows a 2-anonymization of the new tuples.

Although by this approach the privacy of all patients are protected, the cases in the incremental update are over generalized. In the anonymized update data, the gender information of the new cases has to be removed and the age information has to be generalized to a quite large range.

Table 5 shows a 2-anonymization of the updated database. Interestingly, Table 5 is less generalized than both Tables 2 and 4. Intuitively, with more data but a fixed parameter k , we only need to generalize less to meet the k -anonymity requirement.

Thus, the major drawback of the approach anonymizing incremental updates is twofold: it over generalizes the in-

Name	Zipcode	Gender	Age	Disease
Anna	20433	female	21	bird-flu
Bob	20437	male	48	HIV
Carol	20433	female	26	insomnia
Daisy	20437	female	31	cancer

Table 1. The patient database: version 1.

Name	Zipcode	Gender	Age	Disease
<i>Eddy</i>	<i>20437</i>	<i>male</i>	<i>54</i>	<i>obesity</i>
<i>Frank</i>	<i>20435</i>	<i>female</i>	<i>31</i>	<i>SARS</i>

Table 3. An incremental update.

Case-id	Zipcode	Gender	Age	Disease
1	20433	female	[21-26]	bird-flu
2	20437	male	[48-54]	HIV
3	20433	female	[21-26]	insomnia
4	2043*	female	31	cancer
5	20437	male	[48-54]	obesity
6	2043*	female	31	SARS

Table 5. A 2-anonymization of the updated database.

cremental updates, and cannot use the incremental data to reduce the information loss in the anonymization of the existing data.

Method 2: Publishing anonymizations of current versions. As publishing the anonymization of the whole database may reduce the information loss, one may suggest that, after each incremental update, the current database is anonymized and published. For example, Tables 2 and 5 are published.

Analysis may be conducted on the previous releases of data. In order to make such data analysis on the previous releases still useful, it is important to link the occurrences of the same case in different releases together. Here, a unique case-id is attached to each case in multiple releases.

Unfortunately, this approach cannot preserve privacy sufficiently. By simply comparing the records in Tables 2 and 5 using the case-ids, we can infer the information on the quasi-identifier attributes of the cases, as shown in Table 6, while the details of the inferences will be discussed in Example 2. Table 6 is not 2-anonymous anymore.

Using the information across multiple releases may reflect privacy. Even the case-ids are not present, the privacy leaking may still happen since, for example, the other information like disease in this example may help to infer cases.

The major drawback of the approach publishing anonymizations of current versions is that the k -anonymity requirement may be failed due to the possible inferences using multiple releases. ■

As illustrated, directly using the existing methods cannot deal well with k -anonymization against incremental up-

Case-id	Zipcode	Gender	Age	Disease
1	2043*	NA	[21-48]	bird-flu
2	2043*	NA	[21-48]	HIV
3	2043*	female	[26-31]	insomnia
4	2043*	female	[26-31]	cancer

Table 2. A 2-anonymization of Table 1.

Case-id	Zipcode	Gender	Age	Disease
5	2043*	NA	[31-54]	obesity
6	2043*	NA	[31-54]	SARS

Table 4. A 2-anonymization of the update.

Case-id	Zipcode	Gender	Age	Disease
1	20433	female	[21-26]	bird-flu
2	20437	male	48	HIV
3	20433	female	26	insomnia
4	2043*	female	31	cancer
5	20437	male	[48-54]	obesity
6	2043*	female	31	SARS

Table 6. Inferring information on the quasi-identifier attributes from Tables 2 and 5.

dates. We need to understand the potential attacks due to inferences using multiple releases and the opportunities to reduce information loss due to the enlarging available data.

In this paper, we investigate the problem of maintaining k -anonymity for incremental updates, and propose a simple yet effective solution. We analyze how inferences from multiple releases may temper the k -anonymity of data, and propose the monotonic incremental anonymization property. The general idea is to progressively and consistently reduce the generalization granularity as incremental updates arrive. Our new approach guarantees the k -anonymity on each release, and also on the inferred table using multiple releases. At the same time, our new approach utilizes the more and more accumulated data to reduce the information loss. We report a systematic empirical study using both synthetic data sets and real data sets.

The rest of the paper is organized as follows. In Section 2, we present the problem definition. We review the related work in Section 3. In Section 4, we investigate in what situations k -anonymity may be broken by inferences using multiple releases of data, and propose the monotonic incremental anonymization property. Our new approach is developed in Section 5. we report a systematic empirical study in Section 6. The paper is concluded in Section 7.

2 Problem Definition

Generally, we consider anonymizing a table $T(A_1, \dots, A_m, B_1, \dots, B_l)$, where A_1, \dots, A_m are the *quasi-identifier* specified by the application (administrator), and B_1, \dots, B_l are the *sensitive attributes*.

An *anonymization function* \mathcal{F} maps a tuple $t = (a_1, \dots, a_m, b_1, \dots, b_l)$ to a tuple $\mathcal{F}(t) = (a'_1, \dots, a'_m, b_1, \dots, b_l)$. The *anonymization* of table T by function \mathcal{F} is $\mathcal{F}(T) = \{\mathcal{F}(t) | t \in T\}$. Given a positive integer k , $\mathcal{F}(T)$ is *k-anonymous* if for any tuple $t \in T$, there exist at least $(k - 1)$ other tuples $s \in T$ such that $\mathcal{F}(t)_{A_1, \dots, A_m} = \mathcal{F}(s)_{A_1, \dots, A_m}$.

Since an anonymization function does not change any values on the sensitive attributes B_1, \dots, B_l , hereafter we only consider the quasi-identifier attributes and omit the sensitive ones. The table T can be written as $T(A_1, \dots, A_m, \dots)$ for short.

In this paper, we consider a series tables $T, \Delta T_1, \Delta T_2, \dots$, where T is a table as it is created, and $\Delta T_1, \Delta T_2, \dots$ are the incremental updates (insertions only) to the table. We assume that the incremental updates share the same schema as table T .

After each update, a new release of the table is published. In order words, releases $\mathcal{F}(T), \mathcal{F}_1(T \cup \Delta T_1), \mathcal{F}_2(T \cup \Delta T_1 \cup \Delta T_2), \dots$ are released in sequence. For each object in the data set, an object-id is used to identify the occurrences of the object in different releases so that the data analysis at the user side based on the previous releases can be reused.

For the sake of simplicity, $\mathcal{F}_i(T \cup_{j=1}^i \Delta T_j)$, the release after the i -th incremental update ($i \geq 1$), is also denoted by $\mathcal{F}_i(T_i)$.

Importantly, we assume the following.

Assumption 1 (Anonymizations of updates) *When computing $\mathcal{F}_i(T_i)$, we only know $T, \mathcal{F}(T), \Delta T_1, \mathcal{F}_1(T_1), \dots, \Delta T_{i-1}, \mathcal{F}_{i-1}(T_{i-1})$, and ΔT_i . ΔT_{i+1} and later incremental updates, and their anonymizations, are unknown.* ■

Using multiple releases and the object-ids, inferences can be made to narrow down the quasi-identifier attributes of some objects, as elaborated in Example 1 (Table 6).

Example 2 (Inferences) Let us examine how Table 6 can be derived from Tables 2 and 5.

For case 1, the information on every quasi-identifier attribute in Table 5 is more detailed than that in Table 2. Thus, the record of case 1 in Table 5 is the more specific information about the case that can be inferred from the two tables.

For case 2, the record in Table 5 is more specific than the record in Table 2 on attributes Zipcode and gender. On attribute age, Table 2 gives range [21-48] and Table 5 gives range [48-54]. The intersection of the two ranges gives the most specific information. Thus, we infer from the two tables that case 2 has the quasi-identifier attribute values Zipcode 20437, male, and age 48.

Similarly we can infer the other cases. ■

Generally, an anonymization may generalize an attribute value to a range. For an object o appearing in multiple

releases, let R_1, \dots, R_n be the ranges of the generalized values of o on a quasi-identifier attribute A in those releases. Then, we infer that the value of o is in range $\cap_{i=1}^n R_i$. Clearly, $\cap_{i=1}^n R_i$ is the minimum range on the attribute that can be inferred from the multiple releases of o .

We can conduct inferences on objects one by one from all the releases that an object appears, and attribute by attribute on all the quasi-identifier attributes. We denote the most specific information inferred from multiple releases $\mathcal{F}_{i_1}(T_{i_1}), \dots, \mathcal{F}_{i_n}(T_{i_n})$ as table $\mathcal{I}(\mathcal{F}_{i_1}(T_{i_1}), \dots, \mathcal{F}_{i_n}(T_{i_n}))$. It is called the *inference table* from those releases.

To protect privacy, we require that the k -anonymity is maintained against incremental updates. Let k be a positive integer. Releases $\mathcal{F}(T), \mathcal{F}_1(T \cup \Delta T_1), \mathcal{F}_2(T \cup \Delta T_1 \cup \Delta T_2), \dots$ are *k-anonymous against incremental updates* if the following three conditions are always satisfied. (1) $\mathcal{F}(T)$ is k -anonymous; (2) for each $i \geq 1$, $\mathcal{F}_i(T_i)$, the release after the i -th incremental update, is k -anonymous; and (3) for each nonempty set of positive integers $\{i_1, \dots, i_n\}$, $\mathcal{I}(\mathcal{F}_{i_1}(T_{i_1}), \dots, \mathcal{F}_{i_n}(T_{i_n}))$, the inference table using the releases, is still k -anonymous.

We define the problem of *maintaining k-anonymity for incremental updates* as, given a positive integer k , a table T and a series of updates $\Delta T_1, \Delta T_2, \dots$, compute a series of anonymizations $\mathcal{F}(T), \mathcal{F}_1(T \cup \Delta T_1), \mathcal{F}_2(T \cup \Delta T_1 \cup \Delta T_2), \dots$, one anonymization after each update, such that they are k -anonymous against incremental updates.

Please note that the setting defined here is just one among the many possible application scenarios of incremental updates. Here we assume that the adversary does not have any temporal background knowledge (e.g., Eddy was sent to hospital in week 2). Generally, with different scenarios, we may need different strategies to maintain k -anonymity.

3 Related Work

Samarati and Sweeney proposed K-anonymization [13, 15, 17, 16]. Generally, data items are recoded to achieve anonymization. Suppression is a specific form of recoding that recodes a data item to null value (i.e., unknown).

In [13, 15], the *full-domain generalization* was developed, which maps the whole domain of each quasi-identifier attribute to a more general domain in the domain generalization hierarchy. Full-domain generalization guarantees that all values of a particular attribute still belong to the same domain after generalization.

To achieve full-domain generalization, two types of partitioning can be applied. First, the single-dimensional partitioning [4, 7] divides an attribute into a set of non-overlapping intervals, and each interval is replaced by a summary value (e.g., the mean, the median, or the range). On the other hand, the (strict) multidimensional partition-

ing [10] divides the domain into a set of non-overlapping multidimensional regions, and each region is generalized into a summary tuple.

Generally, anonymization is accompanied by information loss. Various models have been proposed to measure the information loss. For example, the *discernability model* [4] assigns to each tuple t a penalty based on the size of the group that t is generalized, i.e., the number of tuples equivalent to t on the quasi-identifier. That is, $C_{DM} = \sum_{E \in \text{group-bys on quasi-identifier}} |E|^2$.

Alternatively, the *normalized average equivalence class size metric* was given in [10]. The intuition of the metric is to measure how well the partitioning approaches the best case where each tuple is generalized in a group of k indistinguishable tuples. That is, $C_{AVG} = \text{number of tuples in the table} / (\text{number of group-bys on quasi-identifier} \times k)$.

The quality of anonymization can also be evaluated by its usefulness in data analysis applications, such as classification [6, 19]. Recently, [8, 11, 21] investigated the utility of anonymized data.

The ideal anonymization should minimize the information loss or maximize the utility. However, the theoretical analysis [2, 12, 10, 3, 1] indicates that the optimal anonymization under many non-trivial quality models is NP-hard. A few approximation methods were developed [3], such as datafly [16], annealing [20], and Mondrian multidimensional k -anonymity [10]. Interestingly, some optimal methods [4, 9] with exponential cost in the worst case were proposed. The experimental results in those studies show that they are feasible and can achieve good performance in practice.

Most of the previous studies considered only one-time anonymization. The incremental updates are not addressed. To the best of our knowledge, [18] is the only existing study addressing the incremental update issue. However, [18] is essentially different from our study here. In [18], updates are “horizontal”, i.e., the set of objects are fixed and new attributes are added over time. In our study here, updates are “vertical”, i.e., new objects are added over time and the attributes are fixed. Thus, the two studies are orthogonal.

4 Anonymization for Incremental Updates

In this section, we first investigate in what situations the k -anonymity in an inference table using two releases may be broken. Then, we propose the monotonic incremental anonymization property which can always maintain k -anonymity. Hereafter, we assume that the anonymizations use *multidimensional partitioning* [10]. That is, for two tuples $t_1 = t_2$, any anonymization maps them identically. This scheme has been used extensively in previous methods, such as [13, 15]. Single-dimensional partitioning [4, 7] can be viewed as a special case of multidimensional parti-

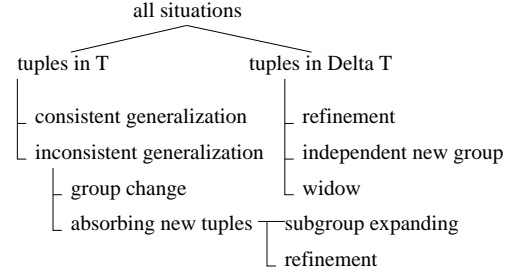


Figure 1. The situations discussed in Section 4.1.

tioning. We use the terms “tuple” and “object” interchangeably.

4.1 K -Anonymity Breaking Inferences

Consider a table T and an incremental update ΔT . Let $\mathcal{F}(T)$ and $\mathcal{F}_1(T \cup \Delta T)$ be the k -anonymous releases of T and $T \cup \Delta T$, respectively. For each tuple $t \in (T \cup \Delta T)$, we analyze in what situations t is not k -anonymous in $\mathcal{I}(\mathcal{F}(T), \mathcal{F}_1(T \cup \Delta T))$, the inference table using $\mathcal{F}(T)$ and $\mathcal{F}_1(T \cup \Delta T)$. Our discussion on the various situations follows a divide-and-conquer hierarchy in Figure 1.

Clearly, we can divide the tuples in $(T \cup \Delta T)$ into two excluding groups: the ones in T and the ones in ΔT .

4.1.1 Inferences on Tuples in T

For a tuple $t \in T$, the (*generalization*) *group of t in \mathcal{F}* , denoted by $G_{\mathcal{F}(T)}(t) = \{s \in T \mid \mathcal{F}(t) = \mathcal{F}(s)\}$, is the set of tuples that are generalized to the same tuple as t by \mathcal{F} .

Consider $G_{\mathcal{F}(T)}(t)$ and $G_{\mathcal{F}_1(T \cup \Delta T)}(t)$, the generalization groups of t in \mathcal{F} and \mathcal{F}_1 , respectively. One of the following situations happens.

If $G_{\mathcal{F}(T)}(t) = G_{\mathcal{F}_1(T \cup \Delta T)}(t)$, i.e., the same group of tuples in T are generalized together in both anonymizations, then clearly the k -anonymity holds in the inference table for t . An object t is called being *consistently generalized* in two anonymizations \mathcal{F} and \mathcal{F}_1 if $G_{\mathcal{F}(T)}(t) = G_{\mathcal{F}_1(T \cup \Delta T)}(t)$.

Example 3 (Consistent generalization) We can always produce a consistent generalization for every tuple. In Example 1, the method 1 (anonymizing incremental updates) conducts a consistent generalization for every tuple. When the incremental update in Table 3 arrives, the new tuples are anonymized (as shown in Table 4). The union of Tables 2 and 4 is the anonymization of the table after the update.

As we analyzed before, a naïve consistent generalization may over-generalize the incremental updates and cannot use the incremental data to reduce the information loss in the anonymization. ■

If t is not consistently generalized, that is, t is generalized with different other tuples in different releases, which is called *inconsistent generalization*, then two sub-situations may happen.

A tuple t is generalized in two anonymizations \mathcal{F} and \mathcal{F}_1 with *group change* if there exists a tuple $t' \in G_{\mathcal{F}(T)}(t)$ but $t' \notin G_{\mathcal{F}_1(T \cup \Delta T)}(t)$, and there is no tuple $s \in \Delta T$ such that $s \in G_{\mathcal{F}_1(T \cup \Delta T)}(t)$.

Example 4 (Group change generalization) Case 1 is generalized with case 2 together in Table 2, and is generalized with case 3 together in Table 5. This results in that case 1 is different from cases 2 and 3 in the inference table using the two releases (Table 6), and thus case 1 is not 2-anonymous in the inference table.

Similarly, case 3 is not 2-anonymous in the inference table due to the change of its groupmate in the two anonymizations. ■

Lemma 1 (Group change generalization) *A tuple o is not k -anonymous in the inference table using two anonymizations $\mathcal{F}(T)$ and $\mathcal{F}_1(T \cup \Delta T)$ if t is generalized in the two anonymizations with group change, and $|G_{\mathcal{F}(T)}(t)| = k$.* ■

Lemma 1 identifies one situation where the k -anonymity is broken. Please note that the lemma holds only the multi-dimensional partitioning is adopted.

As the second sub-situation of inconsistent generalization, t is generalized together with some tuples in ΔT in anonymization $\mathcal{F}_1(T \cup \Delta T)$. We say t *absorbs new tuples*.

Example 5 (Absorbing new tuples) Consider case 2 in Example 1. It is generalized with case 1 as a group in Table 2, and with case 5, a new case in the incremental update, in Table 5. The 2-anonymity of case 2 is broken in the inference table since case 5, the new case, is not right within the ranges where case 2 is generalized in Table 2. In other words, case 2 is in two different generalization groups in the two tables, and thus the intersection which appears in the inference table covers less than enough cases.

Case 4 is also generalized together with a new case in Table 5. However, the situation is different. The new case here, case 6, is in the range where case 4 is generalized in Table 2. Intuitively, the new group in Table 5 can be regarded as a subgroup in Table 2. With case 5 in the incremental update, the subgroup has enough tuples to satisfy the 2-anonymity. In the inference table, although the information about case 4 is refined, but the 2-anonymity is kept. Such a situation where information loss is reduced and k -anonymity is maintained is highly desirable. ■

We have two situations for absorbing new tuples.

A tuple t is generalized with *subgroup expanding* in anonymizations $\mathcal{F}(T)$ and $\mathcal{F}_1(T \cup \Delta T)$ if (1) for every object $t' \in G_{\mathcal{F}_1(T \cup \Delta T)}(t)$, either $t' \in G_{\mathcal{F}(T)}(t)$ or $t' \in \Delta T$; and (2) $\mathcal{F}_1(t)$ is not a subrange of $\mathcal{F}(t)$.

Lemma 2 (Subgroup expanding) *If a tuple t is generalized with subgroup expanding in anonymizations $\mathcal{F}(T)$ and $\mathcal{F}_1(T \cup \Delta T)$, and among the tuples in $G_{\mathcal{F}_1(T \cup \Delta T)}(t)$, there are less than k tuples in the range of $\mathcal{F}(t)$, then the k -anonymity of t is broken in the inference table using the two anonymizations.* ■

A tuple t is generalized with (*subgroup*) *refinement* in anonymizations $\mathcal{F}(T)$ and $\mathcal{F}_1(T \cup \Delta T)$ if for any $t' \in G_{\mathcal{F}_1(T \cup \Delta T)}(t)$, either $t' \in G_{\mathcal{F}(T)}(t)$ or t' is in the range of $\mathcal{F}(t)$. It can be shown that t is k -anonymous in the inference table using the two anonymizations.

4.1.2 Inferences on Tuples in ΔT

When a tuple t in the incremental update ΔT is generalized, whether t is k -anonymous in the inference table depends on the other tuples in the generalization group of t (i.e., $G_{\mathcal{F}_1(T \cup \Delta T)}(t)$). Three situations may happen.

First, t is generalized only with some other new tuples in the incremental update, such as Table 4 in our running example. That is, $G_{\mathcal{F}_1(T \cup \Delta T)}(t) \subseteq \Delta T$. This is a consistent generalization (e.g., case 6 in Example 3).

Second, t is generalized with some existing tuples, but the generalization is a subgroup refinement. Then, the k -anonymity also holds for t in the inference table.

Last, t is generalized with some existing tuples, but the generalization is not a subgroup refinement. The k -anonymity of t may not hold in the inference table since the k -anonymity of the existing tuples cannot be maintained in the inference table.

Example 6 (Widow) In our running example, case 5 is generalized together with case 2 in Table 5. Since case 2 is generalized with subgroup expanding (Example 5), and the 2-anonymity of case 2 is broken in the inference table, the 2-anonymity of case 5 in the inference table is broken, too. ■

In anonymizations $\mathcal{F}(T)$ and $\mathcal{F}_1(T \cup \Delta T)$, a tuple $t \in \Delta T$ is called a *widow* if there exists a tuple $t' \in T$ such that $t' \in G_{\mathcal{F}_1(T \cup \Delta T)}(t)$ and the k -anonymity of t' is broken in the inference table using $\mathcal{F}(T)$ and $\mathcal{F}_1(T \cup \Delta T)$.

Lemma 3 (Widow) *In anonymizations $\mathcal{F}(T)$ and $\mathcal{F}_1(T \cup \Delta T)$, if tuple $t \in \Delta T$ is a widow, and in the set of tuples $G_{\mathcal{F}_1(T \cup \Delta T)}(t)$, the k -anonymity of more than $(|G_{\mathcal{F}_1(T \cup \Delta T)}(t)| - k)$ tuples are broken in the inference table using the two anonymizations, then the k -anonymity of t is also broken in the inference table.* ■

4.2 Monotonic Incremental Anonymization

As analyzed before, the k -anonymity of tuples can be broken in the inference tables of multiple anonymizations

in various ways. However, consistent generalization and subgroup refinement can maintain k -anonymity in inference tables. A feasible idea is that, when subgroup refinement can be conducted, then the anonymization takes the advantages of the incremental updates to reduce information loss; otherwise consistent generalization is used to maintain the k -anonymity.

Formally, for anonymization $\mathcal{F}(T)$ and incremental update ΔT , anonymization $\mathcal{F}_1(T \cup \Delta T)$ is *monotonic* with respect to $\mathcal{F}(T)$ if (1) for any tuple $t \in T$, $\mathcal{F}_1(t) = \mathcal{F}(t)$ or $\mathcal{F}_1(t)$ is a subrange of $\mathcal{F}(t)$ on every quasi-identifier attribute; and (2) for any tuples $t_1, t_2 \in T$, if $\mathcal{F}(t_1) \neq \mathcal{F}(t_2)$ then $\mathcal{F}_1(t_1) \neq \mathcal{F}_1(t_2)$.

Releases $\mathcal{F}(T), \mathcal{F}_1(T \cup \Delta T_1), \mathcal{F}_2(T \cup \Delta T_1 \cup \Delta T_2), \dots$ are *monotonic incremental anonymizations* if $\mathcal{F}_1(T \cup \Delta T_1)$ is monotonic with respect to $\mathcal{F}(T)$, and for any $i \geq 1$, $\mathcal{F}_{i+1}(T \cup (\cup_{j=1}^{i+1} \Delta T_j))$ is monotonic with respect to $\mathcal{F}_i(T \cup (\cup_{j=1}^i \Delta T_j))$.

Monotonic incremental anonymizations have a nice property: the inference table is the same as the last anonymization in a series. Formally, if anonymization $\mathcal{F}_1(T \cup \Delta T)$ is monotonic with respect to $\mathcal{F}(T)$, then $\mathcal{I}(\mathcal{F}(T), \mathcal{F}_1(T \cup \Delta T)) = \mathcal{F}_1(T \cup \Delta T)$.

It is easy to verify the following.

Lemma 4 (Transitivity) *If $\mathcal{F}_1(T \cup \Delta T_1)$ is monotonic with respect to $\mathcal{F}(T)$, and $\mathcal{F}_2(T \cup \Delta T_1 \cup \Delta T_2)$ is monotonic with respect to $\mathcal{F}_1(T \cup \Delta T_1)$, then $\mathcal{F}_2(T \cup \Delta T_1 \cup \Delta T_2)$ is monotonic with respect to $\mathcal{F}(T)$.* ■

Using the properties of consistent generalization and subgroup refinement, as well as the transitivity of monotonic incremental anonymizations, we can show the following desirable property.

Theorem 1 (Monotonic incremental anonymizations) *If $\mathcal{F}(T), \mathcal{F}_1(T \cup \Delta T_1), \mathcal{F}_2(T \cup \Delta T_1 \cup \Delta T_2), \dots$ are monotonic incremental anonymizations, and each of them is k -anonymous, then they are k -anonymous against incremental updates.* ■

5 An Incremental Anonymization Algorithm

In this section, we present a simple yet efficient algorithm to anonymize a data set with a series of incremental updates. The algorithm satisfies the monotonic incremental anonymization requirement discussed in Section 4.2.

We describe our method in two steps. First, we discuss how to anonymize the initial data set. Then, we address how to anonymize incremental updates.

5.1 Anonymizing the Initial Data Set

In principle, we can use any global recoding or constrained local recoding anonymization algorithms to

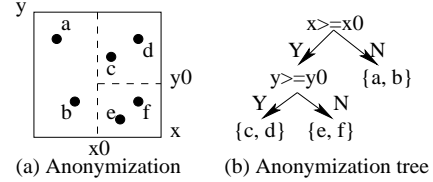


Figure 2. Anonymizing the initial data set.

anonymize the initial data set, as long as the multidimensional partitioning requirement is maintained.

For example, we can use the top-down greedy partitioning method Mondrian [10]. The method recursively partitions the quasi-identifier space into disjoint regions with the constraint that each region contains at least k tuples. The partitioning procedure resembles the kd -tree [5] construction at large.

Example 7 (Initial data set) Figure 2(a) shows partitioning 6 points in a 2-d space into 3 regions such that each region contains 2 points.

Once the partitioning is done, the points at the same region are generalized into a group. This partitioning achieves a 2-anonymization of the data set.

According to the partitioning, we can maintain the groups and the tuples in a binary *anonymization tree*, as shown in Figure 2(b). ■

Generally, after the recursive partitioning, each region contains at least k tuples (and at most $(2k - 1)$ distinct tuples). We can maintain the groups and the tuples in the groups using a binary tree according to the partitioning. The complexity of the method is $O(|T| \log |T|)$ for a table T .

5.2 Anonymizing Incremental Updates

After a batch of incremental updates, we compute a new anonymization when it is needed.

Example 8 (Updates) Continued with Example 7, suppose an incremental update consisting of objects g, h, i, j, k, l arrives as shown in Figure 3(a). In order to generate a new anonymization, we first insert the new objects into the anonymization tree in Figure 2(b). The updated tree is shown in Figure 3(b).

If we generalize the objects at the same node as a group in the adjusted anonymization tree, the k -anonymity against incremental updates is satisfied. As 2-anonymity is required, however, we notice that some nodes may have 4 or more objects. Those nodes are over-generalized and can be further partitioned.

Therefore, we can further partition the nodes of more than 2 objects, as shown in Figure 3(c). The method to partition one node is the same as the one anonymizing the initial data set. This partitioning is a subgroup refinement. The k -anonymity against incremental update is maintained.

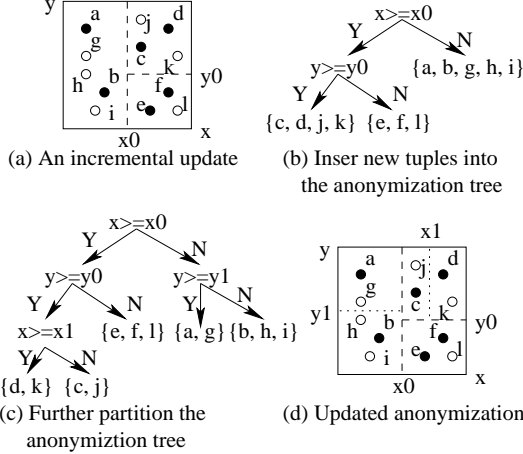


Figure 3. Anonymizing an incremental update.

In the resulted anonymization tree, each node contains up to 3 objects, which cannot be further partitioned. Then, we generalize each node and output the anonymization as shown in Figure 3(d). ■

As elaborated in Example 8, to anonymize the incremental updates, we conduct the following steps.

First, we insert the new tuples in the incremental update into the groups formed in the previous anonymization. Since the groups and the tuples in the previous anonymization are organized in a binary anonymization tree, the cost of is $O(|\Delta T| \log |T|)$, where T and ΔT are the original data set and the incremental update set, respectively.

Second, for each group, we check whether the group has more than $(2k-1)$ tuples. If so, then we apply the Mondrian partitioning method recursively to refine the group, i.e., partitioning into subgroups such that each subgroup contains at least k and cannot be further partitioned.

The refinement step can be regarded as growing the binary tree of groups and tuples in groups. If a leaf node has more than $(2k-1)$ tuples, it may be further partitioned. The result is an extended binary anonymization tree of new groups on all tuples (including the ones in the incremental update).

All tuples in a new group are generalized to the same in the new anonymization. The complexity of the incremental update anonymization step is $O((|\Delta T| + 2k - 1) \log(|\Delta T| + 2k - 1))$.

It is easy to see that by our method produces only monotonic incremental anonymizations.

6 Empirical Study

In this section, we report a systematic empirical study on the effectiveness and the efficiency of our proposed method

on anonymizing incrementally updated data.

We compare two methods: our method as described in Section 5 and the state-of-the-art top-down multidimensional partitioning method Mondrian [10]. Both methods were implemented using Microsoft Visual C++ .NET 2003. All the experiments were conducted on a PC with a Pentium PM 1.5 GHz CPU and 512 MB main memory, running the Microsoft Windows XP operating system.

6.1 Results on Real Data Set Adults

In our experiments, we use the Adults census data set from the UC Irvine machine learning repository which has become a de facto benchmark for k -anonymization. The data set was configured as described in [4]. The salary class attribute was dropped, and the tuples with missing values were removed. The resulted data set contains 30, 162 tuples.

6.1.1 Privacy Leakages

To examine the privacy leakage when releases are independently anonymized, in Figure 4, we randomly chose 12, 000 tuples as the initial data set T , and chose another 6, 000 tuples as the incremental update set ΔT . We anonymized both T and $(T \cup \Delta T)$ using Mondrian. Then, for each tuple in $(T \cup \Delta T)$, we checked whether it is k -anonymous. If so, then the tuple is called *safe*. Otherwise, it is *unsafe*. We ran the experiment 10 times. The figure reports the average ratio of the safe tuples to all the tuples with respect to parameter k for k -anonymity. The larger the ratio, the better the privacy is preserved.

When k is very small (2 or 5 in this experiment), the chance that a generalization group is broken is small, and thus the ratio of safe tuples is high. However, when k becomes large, the ratio decreases because many generalization groups are broken. When k is 10 or over, over 40% tuples are unsafe.

The ratio also depends on the relative size of the incremental update against the initial data set. In Figure 5, We varied the incremental update size from 10% to 90% of the initial data set size (i.e., from 1, 200 tuples to 10, 800 tuples) where k is set to 10. As the incremental update size becomes large, more tuples become unsafe. When the incremental update size is large, many tuples in the update may form generalization groups by themselves without involving any tuples in the initial data set. Thus, the ratio of safe tuples does not decrease dramatically.

Figure 6 shows the ratio of safe tuples with respect to the size of the initial data set, where the incremental update size is kept as 50% of the initial data set. The ratio is stable (about 60-70%). That shows the privacy leakage is persistent in independent anonymized data releases.

As guaranteed by the monotonic incremental anonymization property, our method always maintains

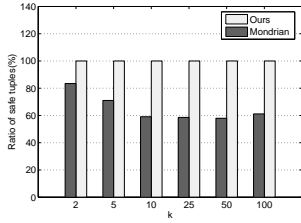


Figure 4. Privacy leakage with respect to k .

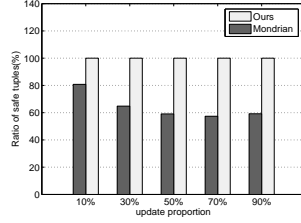


Figure 5. Privacy leakage with respect to incremental update size.

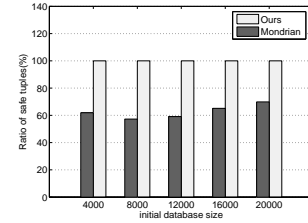


Figure 6. Privacy leakage with respect to initial database size.

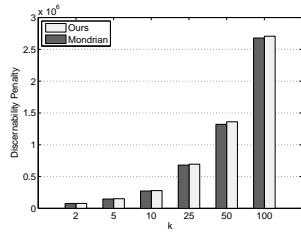


Figure 7. Anonymization quality with respect to k .

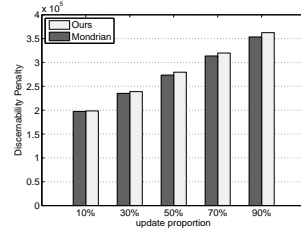


Figure 8. Anonymization quality with respect to incremental update size.

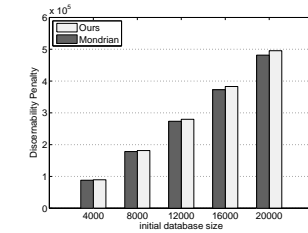


Figure 9. Anonymization quality with respect to initial database size.

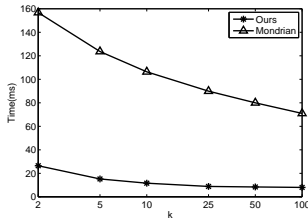


Figure 10. Runtime with respect to k .

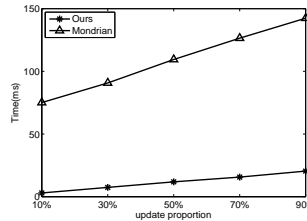


Figure 11. Runtime with respect to incremental update size.

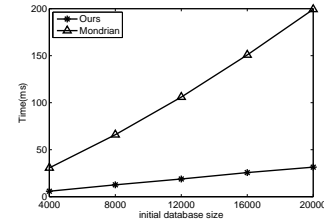


Figure 12. Runtime with respect to initial database size.

the k -anonymity against incremental updates. The ratio of safe tuples in our method is always 100%. This set of experiments clearly show that privacy leakage is severe if we simply anonymize releases of data independently.

This observation strongly justifies the necessity of developing effective methods to maintain k -anonymity against incremental updates.

6.1.2 Anonymization Quality

We use the *discernability penalty score* [4] as the measure of the quality of anonymization.

Figures 7, 8, and 9 report the discernability penalty score of the anonymizations with respect to parameter k , the relative size of incremental updates, and the initial data set size, respectively. The data sets used in those experiments are the

same as those generating Figures 4, 5, and 6, respectively.

Our method anonymizes T and $(T \cup \Delta T)$. The discernability penalty score of the anonymization of $(T \cup \Delta T)$ is reported. For comparison, we applied the Mondrian method on $(T \cup \Delta T)$. The discernability penalty scores are reported in the figures. In other words, the anonymization formed by Mondrian here does not guarantee the anonymity against incremental updates.

The two methods have very similar performance, and the independent anonymizations always have a little bit smaller discernability penalty score. In order to maintain the k -anonymity against incremental updates, our method does not have the full freedom to generalize tuples in $(T \cup \Delta T)$. Instead, only consistent generalization and subgroup refinement can be used. The difference of the discernability penalty scores is the tradeoff of the anonymization quality

for the k -anonymity against incremental updates. However, in all cases, the differences in discernability penalty scores are very small. Our method achieves the anonymization quality highly comparable to independently anonymizing the whole table ($T \cup \Delta T$). At the same time, it maintains the k -anonymity against incremental updates.

This set of experiments clearly show that our method is effective in anonymizing incremental updates. The cost in anonymization quality of maintaining k -anonymity against incremental updates is very small.

6.1.3 Anonymization Efficiency

Figures 10, 11, and 12 report the runtime of anonymizing incremental updates using our method. For comparison, we also report the runtime of Mondrian to anonymize the whole data set ($T \cup \Delta T$). The data sets used in those experiments are the same as those used to generate Figures 4, 5, and 6, respectively.

For an incremental update, our method only needs to insert the new tuples into the existing anonymization tree, and further partition those leaf nodes having more than $(2k - 1)$ tuples. Thus, as expected, the runtime of our incremental anonymization method is much faster than anonymizing the whole data set again using the Mondrian method.

In Figure 10, when k is small (e.g., 2), many leaf nodes in the anonymization tree have more than $(2k - 1)$ tuples after the new tuples are inserted. Those nodes need to be further partitioned into smaller groups. Thus, the runtime is long. When k becomes larger, fewer leaf nodes have more than $(2k - 1)$ tuples after the update. Moreover, less partitioning steps are needed to partition those nodes into subgroups having less than $2k$ tuples. Thus, the runtime decreases.

Figure 11 shows that the runtime of our method is linear with respect to the relative size of incremental updates, and is much shorter than anonymizing the whole data set. Importantly, the new tuples in the incremental update are dispatched into leaf nodes in the existing anonymization tree. Thus, our method often does not need to anonymize a large subset of tuples. This explains the big difference in runtime. Similarly, the results in Figure 12 can be explained.

This set of experiments verify that our method is efficient in anonymizing incremental updates. It is much faster than anonymizing the whole data set again after each update.

6.2 Results on Synthetic Data Sets

To further test the performance of our method on data sets with various distributions, we used an extensive set of synthetic data sets. Limited by space, here we only report the results on two kinds of synthetic data sets. The results on other synthetic data sets are consistent.

The first group of synthetic data sets we used are in uniform distribution. The second group of data sets are in

Gaussian distribution ($\sigma = 2.0$). In all those data sets, we used 4 attributes. The domain of each attribute is real numbers in range $[0, 15]$. By default, we generated 10,000 tuples as the initial data set T .

Following the same methodology in Figures 4, Figures 13 and 14 show the ratio of safe tuples in the independent anonymizations of the whole data sets in uniform and Gaussian distributions, respectively. The incremental update contains 5,000 tuples (50% of the size of the initial data sets). Clearly, privacy leakage is non-trivial. Moreover, the more bias the data, the more privacy is leaked. In bias data sets, the new tuples in an incremental update may have better chances to be similar to some existing tuples. Thus, if different releases are anonymized independently, more generalization groups in the previous release may be broken. Thus, bias data sets provide more chances for inferences if the releases are not anonymized carefully.

Following the methodologies in Figures 7 and 8, Figures 15, 16, and 17 show the anonymization quality on synthetic data sets. Consistently, our method can achieve the k -anonymity against incremental updates by a minor compensation in discernability penalty score.

Last, following the methodologies in Figures 10, 11, and 12, Figures 18, 19, 20, and 21 show the efficiency of our method on synthetic data sets. The results are consistent with the results on the real data set.

In summary, the extensive performance study clearly shows that privacy leakage is severe if incremental updates are not handled carefully. Moreover, our method is effective and efficient in maintaining k -anonymity against incremental updates.

7 Conclusions

In this paper, we identify and investigate the novel and practical problem of maintaining k -anonymity against incremental updates, and propose a simple yet effective solution. Maintaining k -anonymity against various types of incremental updates is an important and practical problem. Although good progress on some scenarios have been made in [18] and this paper, the problem at large remains open and challenging. For example, if an adversary has some temporal background knowledge (e.g., he knows Eddy was sent to hospital in Week 2 in our running example), even the subgroup refinement may leak privacy. As future work, we will investigate how to preserve privacy against incremental updates for other application scenarios.

References

- [1] C. C. Aggarwal. On k -anonymity and the curse of dimensionality. In *VLDB '05*.
- [2] G. Aggarwal, et al. Anonymizing tables. In *ICDT'05*.

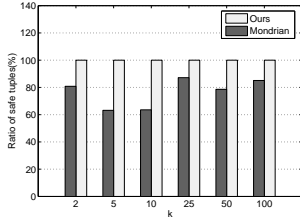


Figure 13. Privacy leakage with respect to k (uniform distribution).

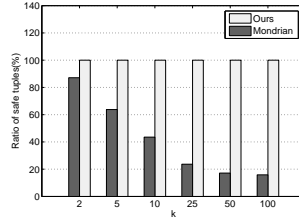


Figure 14. Privacy leakage with respect to k (Gaussian distribution).

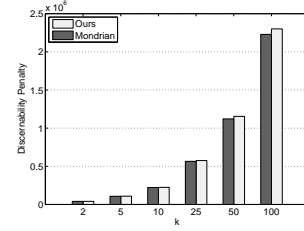


Figure 15. Anonymization quality with respect to k (uniform distribution).

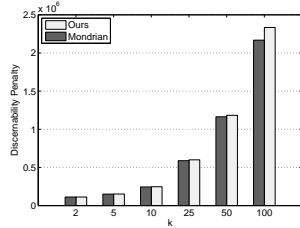


Figure 16. Anonymization quality with respect to k (Gaussian distribution).

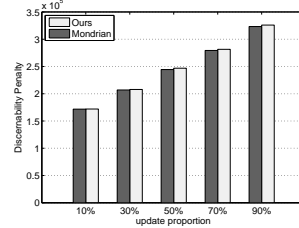


Figure 17. Anonymization quality with respect to incremental update size (Gaussian distribution).

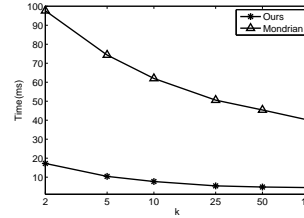


Figure 18. Runtime with respect to k (uniform distribution).

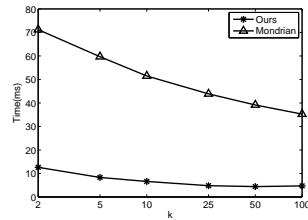


Figure 19. Runtime with respect to k (Gaussian distribution).

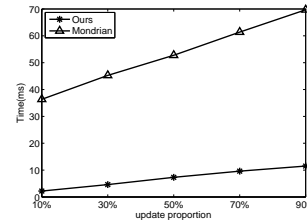


Figure 20. Runtime with respect to incremental update size (Gaussian distribution).

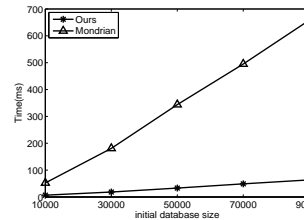


Figure 21. Runtime with respect to initial database size (Gaussian distribution).

- [3] G. Aggarwal, et al. Approximation algorithms for k -anonymity. *Journal of Privacy Technology*, (2005112001).
- [4] R. J. Bayardo and R. Agrawal. Data privacy through optimal k -anonymization. In *ICDE'05*, pages 217–228.
- [5] J. H. Freidman, et al. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, 1977.
- [6] B. C. M. Fung, et al. Top-down specialization for information and privacy preservation. In *ICDE'05*, pages 205–216.
- [7] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *KDD'02*, pages 279–288.
- [8] D. Kifer and J. Gehrke. Injecting utility into anonymized datasets. In *SIGMOD'06*.
- [9] K. LeFevre, et al. Incognito: Efficient full-domain k -anonymity. In *SIGMOD'05*.
- [10] K. LeFevre, et al. Mondrian multidimensional k -anonymity. In *ICDE'06*.
- [11] K. LeFevre, et al. Workload-aware anonymization. In *KDD'06*, pages 277–286.
- [12] A. Meyerson and R. Williams. On the complexity of optimal k -anonymity. In *PODS'04*, pages 223–228.
- [13] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.
- [14] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information. In *PODS'98*.
- [15] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k -anonymity and its enforcement through generalization and suppression. In *Technical Report SRI-CSL-98-04*, 1998.
- [16] L. Sweeney. Achieving k -anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems*, 10(5):571–588, 2002.
- [17] L. Sweeney. K -anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems*, 10(5):571–588, 2002.
- [18] K. Wang and B. C. M. Fung. Anonymizing sequential releases. In *KDD'06*, pages 414–423.
- [19] K. Wang, et al. Bottom-up generalization: A data mining solution to privacy protection. In *ICDM'04*, pages 249–256.
- [20] W. E. Winkler. Using simulated annealing for k -anonymity. *Technical Report Statistics 2002-7, U.S. Census Bureau, Statistical Research Division*, 2002.
- [21] J. Xu, et al. Utility-based anonymization using local recoding. In *KDD'06*, pages 785–790.