# Privacy by Diversity in Sequential Releases of Databases

Erez Shmueli[a], Tamir Tassa[b]

[a]*Department of Industrial Engineering, Tel Aviv University, Tel Aviv, Israel*
[b]*Department of Mathematics and Computer Science, The Open University, Ra'anana, Israel*

**Abstract**

We study the problem of privacy preservation in sequential releases of databases. In that scenario, several releases of the same table are published over a period of time, where each release contains a different set of the table attributes, as dictated by the purposes of the release. The goal is to protect the private information from adversaries who examine the entire sequential release. That scenario was studied in [32] and was further investigated in [28]. We revisit their privacy definitions, and suggest a significantly stronger adversarial assumption and privacy definition. We then present a sequential anonymization algorithm that achieves $\ell$-diversity. The algorithm exploits the fact that different releases may include different attributes in order to reduce the information loss that the anonymization entails. Unlike the previous algorithms, ours is perfectly scalable as the runtime to compute the anonymization of each release is independent of the number of previous releases. In addition, we consider here the fully dynamic setting in which the different releases differ in the set of attributes as well as in the set of tuples. The advantages of our approach are demonstrated by extensive experimentation.

## 1. Introduction

### *1.1. Overview*

Large organizations regularly collect personal data, such as medical records, marketing information, or census data, in order to perform on it data mining for the purpose of revealing trends and patterns in the general population. However, the use of data containing personal information has to be restricted in order to protect individual privacy. Although identifying attributes like ID numbers and names are never released for data mining purposes, sensitive information might still leak due to linking attacks, whereby an attacker may uncover hidden identities

or sensitive information by joining the released data attributes with other publicly available data. The attributes that can be efficiently used to create such links, such as gender, zipcode, and age, are called quasi-identifiers. The problem of anonymity calls for the modification of those attributes in order to thwart such attacks, while maintaining as much as possible of the utility of the released data.

The first model of privacy-preserving data publication was $k$-anonymity [27, 29]. That model suggests to generalize the values of the quasi-identifiers so that each of the released records becomes indistinguishable from at least $k-1$ other records, when projected on those attributes. As a consequence, each individual may be linked to sets of records of size at least $k$ in the released anonymized table, whence privacy is protected to some extent. While $k$-anonymity aims at preventing identity disclosure, the later models of $\ell$-diversity [20, 34] and $t$-closeness [19] aim at preventing sensitive attribute disclosure by imposing conditions on the distribution of the sensitive values within each subset of records that are indistinguishable with respect to their quasi-identifiers.

In all those models, the values of the database are typically modified via the operation of generalization, while keeping them consistent with the original ones. A cost function is used to measure the amount of information that is lost by the generalization process. The objective is to modify the table entries in order to respect the underlying privacy condition while minimizing the information loss.

Most of the studies thus far concentrated on scenarios of a single release, in which the underlying table is released just once in an anonymized manner. However, there are scenarios in which the same table is released more than once, for example, when new records are added to the table, or when partial views of the data have to be released to different clients. In such scenarios, it is imperative to consider the potential threats that may be caused by joining information from the different views.

**Example 1.1.** Consider the table with two quasi-identifiers, `age` and `gender`, and one sensitive attribute, `disease`, that is given in Table 1, alongside with two releases of it. The first release includes the `age` and `disease` attributes, while the second release includes the `gender` and `disease` attributes. Each of the two releases satisfies 2-diversity, since it can be used to link each individual to two sensitive values with equal probabilities. For instance, if an adversary wishes to find sensitive information about Alice, a female of age 20, then the first release allows him to infer that she has either measles or hepatitis, while the second release reveals that she has either measles or flu. However, the combination of the two releases discloses with certainty that Alice has measles. Therefore, in order to

achieve 2-diversity, one of the two releases would have to be further generalized.

Table 1: A table (left) and two corresponding releases (middle and right)

| name | age | gender | disease |
|---|---|---|---|
| Alice | 20 | female | measles |
| Bob | 20 | male | hepatitis |
| Carol | 30 | female | flu |
| David | 30 | male | angina |

| age | disease |
|---|---|
| 20 | measles |
| 20 | hepatitis |
| 30 | flu |
| 30 | angina |

| gender | disease |
|---|---|
| female | measles |
| male | hepatitis |
| female | flu |
| male | angina |

□

The two main scenarios of multiple releases of a given table are the following (see [9]): (a) The scenario of sequential release publishing [28, 32], where different (vertical) projections of a given table on different subsets of attributes are released in a sequential manner; and (b) Continuous data publishing [1, 3, 10, 25, 28, 35, 37], in which the underlying table changes over time (e.g., tuples are added, removed, or updated), and updated snapshots of the table are released over time.

In both scenarios, several releases of partial views of the same basic table are published in a sequential manner, where already published releases cannot be modified. The goal is to anonymize the next release so that the combination of information from all releases does not lead to a privacy breach. In the scenario which is called above "sequential release publishing", the set of tuples (rows) is fixed, while the set of attributes (columns) changes from one release to another. On the other hand, in the scenario of so called "continuous data publishing", the set of attributes is fixed while the set of tuples is dynamic.

In the lion's part of this paper we study the scenario of sequential release publishing (in which the set of attributes changes between releases, but the set of tuples is fixed). The approach that we propose here differs significantly from the one proposed in [32] and then further developed in [28]. Then, in Section 5, we explain how to extend our approach to handle also the case of dynamically changing tables, where tuples can be added from time to time; in such cases, the set of attributes as well as the set of tuples may change from one release to the next one.

### *1.2. Related work on privacy-preservation in sequential release publishing*

Wang and Fung [32] were the first to study the privacy problem in sequential releases and they developed an algorithm for anonymizing such a release. They

focused on the case in which only one previous release of the underlying table was published, and then the problem is to generalize a second release so that a given privacy goal is met. They (as well as the study [28] that we review below) considered two privacy goals in that context: $\ell$-linkability and $\ell$-diversity[1]. $\ell$-Linkability requires that the sequential release does not enable to link any quasi-identifier tuple to less than $\ell$ distinct values of the sensitive attribute. $\ell$-Diversity demands, in addition, that no sensitive value can be linked to any quasi-identifier tuple with probability greater than $1/\ell$. The algorithm of [32] generalizes the second release so that the required privacy goal is met. They considered a strict model of global recoding, called "cut generalization" (see Section 2.1 for a formal definition of generalization models).

Shmueli et al. [28] extended the framework that was considered in [32] in three ways:

- By considering a more flexible local recoding generalization model (the so called "cell generalization"), what enabled substantially better utility results.
- By proposing privacy definitions that are suitable for any number of releases or generalization model and designing a corresponding anonymization algorithm.
- By proposing solutions to the dynamic setting in which tuples may be added to the underlying table in between releases. (It is the only study so far that considered the combination of the sequential release scenario with continuous data publishing.)

The approach in [28] is based on viewing the sequential release as an $R$-partite graph, where $R$ is the number of releases: The nodes in the $r$th part are the tuples in the $r$th release, $1 \leq r \leq R$, and an edge connects two nodes in two different parts if the corresponding two tuples contain consistent generalized values (namely, they could both be the generalized view of the same original tuple). A full clique in that graph is a collection of $R$ nodes, one from each part, that are all pairwise connected.

Two tuples in the sequential release (or two nodes in the multipartite graph) are called *siblings* if they are the generalized images of the same original tuple. A full clique is called *genuine* if all nodes in it are siblings of each other. Some

[1] We use herein the terms that were suggested in [28]; they differ from the terms used in [32].

of the full cliques in the graph are genuine, and some are fake (full cliques consisting of tuples that happen to be consistent even though they are not siblings). [32] assumed that the adversary considers all cliques as genuine, and based their privacy definitions on that assumption. [28] realized that the adversary may recognize some of the cliques as fake, since the set of all genuine cliques forms a perfect matching in the $R$-partite graph, and then modified the privacy definition to reflect that. According to [32, 28], the sequential release offers $\ell$-diversity if any given quasi-identifier tuple is linked to any given sensitive value in no more than $1/\ell$ of the full cliques [32] or no more than $1/\ell$ of the full cliques that cannot be recognized by the adversary as fake [28].

The above described multipartite graph framework has several significant disadvantages, in terms of privacy guarantee, runtime, and utility:

- In order to compute properly the level of linkability or diversity, it is needed to identify all full cliques that are part of a perfect matching; that problem was shown in [28] to be NP-hard for $R > 2$.[2] It is therefore suggested in [28] to perform relaxed computations in order to circumvent those computational barriers; consequently, their solution does not guarantee compliance with the targeted privacy goals.

- Even if we could identify all full cliques that are part of some perfect matching, it is needed to weigh the cliques according to the number of perfect matchings in which they participate; counting the number of perfect matchings even in a bipartite ($R = 2$) graph is #P-complete[3].

- The above described approach is highly non-scalable, since the anonymization of each release requires to compute cliques in the $R$-partite graph; the cost of such computations depends exponentially on the number $R$ of releases.

- The multipartite graph approach may suffer from a problem of a reducing privacy budget. Assume that the data owner already published several anonymized releases of the table that he owns. It is possible that in order to publish the next release, while still meeting the desired privacy goal, he

[2] When $R = 2$ the problem is in P and can be solved at the same cost of finding one perfect matching in the bipartite graph [30].

[3] #P is the class of counting problems associated with the decision problems in NP.

would need to anonymize that new release aggressively (say, by suppressing many entries), because of the previously published releases. Namely, the level of information loss in each release may depend on its position in the sequence of releases and on the previous releases.

### *1.3. Our contributions*

In this paper we offer a different approach for anonymizing sequential releases towards meeting the $\ell$-diversity privacy constraint. Instead of trying to ensure diversity by examining the collection of perfect matchings in the underlying multipartite graph, we devise a technique for anonymizing each release in its turn, so that the resulting multipartite graph has $\ell$ perfect matchings that link each quasi-identifier tuple with $\ell$ different sensitive values. In order to achieve that, we first generate $\ell - 1$ tables having the same set of quasi-identifier tuples as the original table, as well as the same multiset of sensitive values. We refer to those tables as *possible worlds*. Those tables, together with the original table, are $\ell$-diverse, in the sense that each quasi-identifier tuple is linked in each of them to a different sensitive value. Then, whenever we need to publish an anonymized version of some vertical projection of the table, we generalize the entries of the relevant attributes until the resulting generalized table is consistent with all of those $\ell - 1$ possible worlds. By doing so, we guarantee that no adversary can tell the true world from the other possible worlds, and, as a result, cannot link any quasi-identifier tuple with any sensitive value with probability greater than $1/\ell$ (Theorem 4.1).

The advantages offered by our approach are as follows:

- **Privacy.** As opposed to [28, 32], our approach guarantees that even a computationally-unbounded adversary cannot link any quasi-identifier tuple with any sensitive value with probability greater than $1/\ell$. In addition, our adversarial assumption is much stronger — we assume that the adversary knows the quasi-identifier information of *all* tuples in the table, and not just that of a specific target individual.

- **Scalability.** The runtime of our algorithm in computing an anonymized view of a new release is independent of the number of previous releases, as opposed to that of the algorithm in [28] which depends exponentially on the number of releases.

- **Utility.** The anonymization of each release is independent of its position in the release sequence and of the past (or future) releases. Namely, even if we already published many anonymized releases of the same table, the

anonymization of the next release, and therefore its utility, would be exactly the same as if it was the first release.

- **Dynamics.** Our approach can handle both vertical and horizontal dynamics. Specifically, it can handle cases where new attributes (columns) are added to the table, as well as cases where new records (rows) are added.

### *1.4. Organization of the paper*

Section 2 provides the basic definitions and terminology. In Section 3 we define the adversarial model and the privacy goal, and compare them to those in [28, 32]. In Section 4 we describe our algorithm for anonymizing sequential releases while respecting privacy by providing diversity. Section 5 is devoted to the case of dynamically changing tables. Our experiments are described in Section 6 and we conclude in Section 7. In Section 8.3 in the Appendix we discuss the applicability of differential privacy to the problem of sequential release of databases.

## 2. Preliminaries

In this section we provide the basic definitions and terminology. For convenience, Section 8.1 in the Appendix includes a table that summarizes the main notations that we introduce herein.

Let $A_1, \ldots, A_M$ be $M$ attributes and $T = \{t_1, \ldots, t_N\}$ be a table of $N$ tuples in $A_1 \times \cdots \times A_M$. Here, $A_m$, $m \in [M] := \{1, \ldots, M\}$, denotes the $m$th attribute as well as the domain in which it takes values. We shall assume that the first $Q$ attributes are the quasi-identifiers, attributes $A_{Q+1}, \ldots, A_{M-1}$ are the non-identifiers (namely, attributes that are not quasi-identifiers and do not hold sensitive information), and $A_M$ is the sensitive attribute. (We adopt the usual assumption of a single sensitive attribute.) So, if we let $t_n(m)$ denote the $m$th component of $t_n$ (i.e., $t_n = (t_n(1), \ldots, t_n(M))$), $t_n(m)$ is a quasi-identifier value when $1 \leq m \leq Q$, a non-identifier value when $Q < m < M$, and the sensitive value when $m = M$.

### *2.1. Generalization*

For each $m \in [M]$, let $\overline{A}_m$ be a collection of subsets of $A_m$, which can be used as generalized values in anonymized releases.

**Definition 2.1.** *Let $t$ and $v$ be tuples in $A_1 \times \cdots \times A_M$ and $\overline{A}_1 \times \cdots \times \overline{A}_M$, respectively, and for all $m \in [M]$ let $t(m)$ and $v(m)$ denote their $m$th components. Then $v$ is a generalization of $t$ (or $v$ generalizes $t$) if $t(m) \in v(m)$ for all $m \in [M]$. The table $V = \{v_1, \ldots, v_N\}$ is a generalization of $T = \{t_1, \ldots, t_N\}$ if $v_n$ is a generalization of $t_n$ for all $n \in [N]$.*

Typically, if $A_m$ is a categorical attribute (e.g. `occupation`) $\overline{A}_m$ is a corresponding hierarchical generalization tree (a taxonomy); namely, each node in $\overline{A}_m$ is a subset of $A_m$, the root of $\overline{A}_m$ is the entire set, the leaves are all the singleton subsets, and the children of any non-leaf node form a partition of the parent node. If $A_m$ is a fully-ordered attribute (usually a numerical attribute such as `age`), $\overline{A}_m$ is typically the collection of all intervals in $A_m$; i.e., it consists of all subsets of $A_m$ of the form $[a, b] := \{x \in A_m : a \leq x \leq b\}$.

Another notion which we shall use is that of a closure.

**Definition 2.2.** *The closure of a set of (possibly generalized) tuples $B \subset \overline{A}_1 \times \cdots \times \overline{A}_M$ is the generalized quasi-identifier tuple $t^B \in \overline{A}_1 \times \cdots \times \overline{A}_Q$ where $t^B(m)$, $m \in [Q]$, is the minimal subset in $\overline{A}_m$ that contains $\bigcup_{t \in B} t(m)$.*

A sequential release of $T$ is a sequence $\langle V_1, \ldots, V_R \rangle$ of generalizations of $T$; the tuples in those releases will be denoted by $V_r = \{v_1^r, \ldots, v_N^r\}$, $1 \leq r \leq R$. Typically, each generalization $V_r$ includes only a subset of the $M$ attributes in the original table $T$. In that case, all entries in the missing attributes are suppressed. We shall denote hereinafter by $I_r$ the set of indices of attributes that appear in release $V_r$. Namely, $I_r$ is a subset of $[M] := \{1, \ldots, M\}$ and for all $m \in [M] \setminus I_r$, the $m$th attribute in $V_r$ is suppressed.

As explained earlier, the norm is to apply generalization only to the quasi-identifiers, and not to the non-identifiers nor to the sensitive attribute. Hence, the collection $\overline{A}_m$ for an attribute $A_m$ that is either a non-identifier or the sensitive value ($m > Q$), is $\overline{A}_m = A_m \cup \{A_m\}$. Namely, it includes all singleton subsets of $A_m$, together with the entire set $A_m$. Then, if such an attribute appears in a given release, it will appear in its non-generalized form in all tuples; if it does not appear in another release, it will be suppressed in all of the tuples in that release.

There are two main models of generalizations. In local recoding, each entry in the table's $m$th attribute can be generalized independently to any of the generalized values in $\overline{A}_m$ that includes it. In global recoding, each element in $A_m$ is always generalized in the same way in a given release. Cut generalization is a special case of global recoding which is even stricter. In that model, one selects in $\overline{A}_m$ a sub-collection of subsets that forms a partition of $A_m$ (namely, those

subsets are disjoint and cover the entire set) and then each element in $A_m$ is generalized to the subset in that sub-collection that includes it. In this paper we use the local recoding model. Since the local recoding model is more flexible than the global recoding model, it may allow achieving a certain privacy goal with less information loss than the global recoding model.

### *2.2. Measures of information loss*

Many measures were suggested in the literature for the cost of generalization, or information loss. Among the commonly used ones is the Loss Metric (LM) measure [15] and the Entropy Measure (EM) [13]. In this study we concentrate on these two measures; however, our entire discussion is independent of the choice of measure of information loss and equally applies to any other measure.

The LM measure associates with a given generalized quasi-identifier tuple $v \in \overline{A}_1 \times \cdots \times \overline{A}_Q$ an information loss as follows:

$$IL(v) := \sum_{m=1}^{Q} \frac{|v(m)| - 1}{|A_m| - 1} \,. \tag{1}$$

Namely, in entries that were not generalized at all ($v(m)$ is a singleton) the penalty is 0, in entries that were totally suppressed ($v(m) = A_m$) the penalty is 1, and in intermediate generalizations the penalty is proportional to the size of the generalizing subset.

The EM measure assumes that the distribution of each of the attributes $A_m$, $m \in [M]$, in $T$ is made known. Let $V = \{v_1, \ldots, v_N\}$ be a generalization of $T$. Then, if the original value $t_n(m)$ was generalized to the subset of values $v_n(m) \subseteq A_m$, the uncertainty regarding the exact original value $t_n(m) \in v_n(m)$ is given by the corresponding conditional entropy. Specifically, if $v_n(m) = \{b_1, \ldots, b_z\}$ and $p_i$ is the relative frequency of $b_i$ in $T$'s $m$th attribute, $1 \leq i \leq z$, then the corresponding conditional entropy is

$$H(v_n(m)) := -\sum_{i=1}^{z} q_i \log_2 q_i \,, \qquad \text{where } \; q_i = \frac{p_i}{p_1 + \cdots + p_z} \,, \;\; 1 \leq i \leq z \,.$$

The overall EM information loss in $V$ is given by the sum of $H(v_n(m))$ for all $n \in [N]$ and $m \in [M]$.

Usually, the purposes of the data release are unknown at the stage when the data is anonymized. Hence, it is customary to use general purpose information loss measures, such as the LM or EM, as an indicator for the utility of the

anonymized data. The question of how to use the anonymized data in order to, say, learn a classifier or mine association rules, and whether those general purpose information loss measures are good indicators for the accuracy of data mining on the anonymized tables, was considered by very few studies so far, e.g. [11, 15, 24]. The recent study [17] is devoted to that topic. It is shown there that reduced information loss, as measured by either the LM or the EM, translates also to enhanced accuracy when using the anonymized tables to learn classification models.

### *2.3. Anonymity and diversity in a single release*

One of the main threats in publishing tables that include data on individuals is that of linking attacks. An adversary who knows the values of the quasi-identifiers of some target individual may look for tuples in the released table that could be linked to that individual, in the sense that their quasi-identifiers are consistent with the known values. Then, the adversary may infer that the sensitive value of his target individual is one of the sensitive values in those tuples. In order to thwart such linking attacks, it is customary to generalize the table before publication.

Let $V$ be a generalization of $T$ and consider the following equivalence relation on $V$'s tuples:

$$v_n \sim v_{n'} \quad \text{iff} \quad v_n(m) = v_{n'}(m) \quad \forall m \in [Q]\,. \tag{2}$$

$V$ is called an anonymization of $T$ if all equivalence classes in $V/\!\sim$ satisfy some privacy criterion. The $k$-anonymity criterion requires that all equivalence classes are of size at least $k$. The $\ell$-diversity criterion, on the other hand, requires that the diversity of each equivalence class is at least $\ell$. A common definition of diversity is the following:

**Definition 2.3.** *The diversity of a set of tuples $B \subset A_1 \times \cdots \times A_M$ is defined as the inverse of the relative frequency of the most frequent sensitive value in $B$'s tuples, i.e.,*

$$\mathit{div}(B) := \min_{s \in B(M)} \frac{|B|}{|\{t \in B : t(M) = s\}|}\,,$$

*where $B(M) = \{t(M) : t \in B\}$.*

## 3. Privacy by diversity in a sequential release

The privacy goal in the sequential release setting is similar to that in a single release setting (see Section 2.3). It is needed to anonymize the sequential release

in a manner that would limit the ability of an adversary to deduce links between individuals and sensitive values. However, the sequential release setting has two major differences compared with the single release setting:

- **Privacy.** Ensuring $\ell$-diversity in each release separately is insufficient, since information from different releases may be combined to infer links between quasi-identifiers and sensitive values with probability greater than $1/\ell$, as exemplified by Example 1.1. Hence, one must take into account the global view of the entire sequence of releases.

- **Information Loss.** The sequential release setting breaks up tuples. As a consequence, the adversary faces a puzzle-like problem of finding the true links between the different pieces of tuples as they appear in the different releases. Because of that, it is possible to achieve in a sequential release a similar level of diversity with much less information loss, in comparison to a scenario in which the table is published at once with all of its attributes.

Therefore, the sequential release setting calls for a new definition of diversity and algorithms that are specially designed for it.

As a first step, we must define the adversarial model. Wang and Fung [32] and Shmueli et al. [28] assumed that the adversary knows the quasi-identifier values of his target individual, but not those of other individuals in the table. We make here a much stronger adversarial assumption (which was also assumed in e.g. [12, 31, 36, 37]): the adversary knows the exact set of individuals that are represented in $T$ and their quasi-identifier values. In other words, such an adversary knows the projection of the table $T$ onto the subset of quasi-identifiers, $A_1 \times \cdots \times A_Q$, and their corresponding identifiers. Hereinafter, if $I \subset [M]$ is a subset of indices, $T|_I$ will denote the table $T$ where all attributes with indices in $[M] \setminus I$ are suppressed. Therefore, the above adversarial assumption means that the adversary knows $ID \parallel \left(T|_{[Q]}\right)$, where $ID$ stands for the attribute of corresponding identifiers. For instance, for Table 1 (see Example 1.1), the assumed background knowledge of the adversary is as shown in Table 2.
Hence, in order to incorporate that adversarial assumption into the release model, we shall always assume that there exists an additional release of $T$ that equals $V_0 = ID \parallel \left(T|_{[Q]}\right)$; i.e., $V_0 = \{v_1^0, \ldots, v_N^0\}$, where

$$v_n^0 = (id_n, t_n(1), \ldots, t_n(Q), *, \ldots, *)\,, \quad n \in [N]\,, \tag{3}$$

and $id_n$ is the identity of the $n$th tuple in $T$.

Next, we define the notion of possible worlds.

Table 2: Adversarial background knowledge for Table 1.

| name | age | gender | disease |
|---|---|---|---|
| Alice | 20 | female | * |
| Bob | 20 | male | * |
| Carol | 30 | female | * |
| David | 30 | male | * |

**Definition 3.1.** *Let $\mathcal{V} := \langle V_0, V_1, \ldots, V_R \rangle$ be a sequential release of a table $T$. Let $W = \{w_1, \ldots, w_N\}$ be a set of $N$ tuples in $ID \times A_1 \times \cdots \times A_M$. Assume that for each $0 \leq r \leq R$ there exists a bijection $f_r : [N] \to [N]$ such that for each $n \in [N]$, the tuple $v^r_{f_r(n)} \in V_r$ generalizes $w_n$. Then $W$ is called a possible world for $\mathcal{V}$. The set of all possible worlds for the sequential release $\mathcal{V}$ is denoted $\mathsf{PW}(\mathcal{V})$.*

Stated differently, a specific table $W$ is a possible world for the sequential release $\mathcal{V}$ if it could be the original table $T$: it agrees with the background knowledge $V_0$ when projected onto $ID \times A_1 \times \cdots \times A_Q$, and in addition, each release is consistent with $W$.

Clearly, $T$ is a possible world for $\mathcal{V}$. Indeed, let us make hereinafter the ordering assumption that for all $0 \leq r \leq R$ and $n \in [N]$, $v^r_n$ is the generalized image of $t_n \in T$ in the $r$th release $V_r$. (That assumption is made only for simplifying our discussion; the tuples in the actual release will be randomly shuffled in order to obfuscate the linkage between them.) Then by taking $f_r$ to be the identity bijection from $[N]$ to $[N]$, for all $0 \leq r \leq R$, we see that $T \in \mathsf{PW}(\mathcal{V})$.

We proceed to define $\ell$-diversity for a set of possible worlds and for a sequential release.

**Definition 3.2.** *Let $E \subseteq \mathsf{PW}(\mathcal{V})$ be a set of possible worlds. For any selection of $v^0_n$ as one of the tuples in $V_0$ (see Eq. (3)) and $s \in A_M$ as one of the sensitive values, we let $E[v^0_n, s]$ denote the subset of possible worlds in $E$ that contain a tuple $w$ such that $w(m) = v^0_n(m)$ for all $m \in [Q]$ and $w(M) = s$. Then $E$ is $\ell$-diverse if*

$$\frac{|E[v^0_n, s]|}{|E|} \leq \frac{1}{\ell} \quad \forall v^0_n \in V_0\,,\ s \in A_M\,. \tag{4}$$

**Definition 3.3.** *The sequential release $\mathcal{V}$ respects $\ell$-diversity if the set $E = \mathsf{PW}(\mathcal{V})$ of all possible worlds that it induces is $\ell$-diverse.*

Namely, a sequential release satisfies $\ell$-diversity if the fraction of possible worlds that link any tuple in the background knowledge table $V_0$ with any sensitive value does not exceed $1/\ell$.

Later on we shall also use the term $\ell$-linkability. The sequential release $\mathcal{V}$ respects $\ell$-linkability if for every $v_n^0 \in V_0$ there exist $\ell$ distinct sensitive values $s \in S$ such that $|E[v_n^0, s]| > 0$, for $E = \mathsf{PW}(\mathcal{V})$; namely, like $\ell$-diversity it requires every quasi-identifier tuple to be linked to at least $\ell$ distinct sensitive values, but it ignores the linkage frequencies.

Any possible world in $\mathsf{PW}(\mathcal{V})$ constitutes a possible linkage of tuples in $V_0$, the background knowledge of the adversary, to sensitive values. Under our adversarial assumption, the adversary cannot distinguish between those possible worlds, whence each of them is equally likely to be the true one. Therefore, if the sequential release respects $\ell$-diversity, the adversary may not infer the sensitive value of a given target individual with probability greater than $1/\ell$.

**Example 3.1.** This toy example illustrates the concepts of possible worlds and $\ell$-diversity. Consider the table $T$ with a single quasi-identifier, $A_1$, and a sensitive attribute, $A_2$, in Table 3(a). Table 3(b) shows a corresponding sequential releases: $V_0$ represents the assumed adversarial background knowledge, and includes just the identifier and quasi-identifier attributes; and $V_1$ is a release in which the quasi-identifier $A_1$ was totally suppressed.

It is easy to see that in this example there are three possible worlds: $T$, and the two additional worlds that are shown in Table 3(c). This sequential release respects $\ell$-diversity with $\ell = \frac{3}{2}$ since every individual is linked with every sensitive value in no more than $\frac{2}{3}$ of the possible worlds. For example, the individual a is linked to the sensitive value 1 by the first two possible worlds, and to the sensitive value 2 by the third possible world.

Table 3: (a) A table $T$; (b) Two releases $V_0$, $V_1$; (c) Additional possible worlds.

| $ID$ | $A_1$ | $A_2$ |
|---|---|---|
| a | x | 1 |
| b | y | 1 |
| c | z | 2 |

| $ID$ | $A_1$ |
|---|---|
| a | x |
| b | y |
| c | z |

| $\overline{A_1}$ | $A_2$ |
|---|---|
| * | 1 |
| * | 1 |
| * | 2 |

| $ID$ | $A_1$ | $A_2$ |
|---|---|---|
| a | x | 1 |
| b | y | 2 |
| c | z | 1 |

| $ID$ | $A_1$ | $A_2$ |
|---|---|---|
| a | x | 2 |
| b | y | 1 |
| c | z | 1 |

□

**The $\ell$-diversity problem.** Assume that the data owner has already published $R-1$ releases, $V_1, \ldots, V_{R-1}$, of the table $T$, and that the sequential release up

to that point, $\langle V_0, V_1, \ldots, V_{R-1} \rangle$, respects $\ell$-diversity. It is needed now to publish another release of $T$ that includes some subset of the attributes, $I_R$. Then it is desired to find a generalization $V_R$ of $T|_{I_R}$ with minimal information loss that maintains $\ell$-diversity.

**Comment.** We assumed that the adversary knows the exact set of individuals that are represented in $T$ and their corresponding quasi-identifiers. Such an assumption corresponds to adversarial environments that are more threatening than what can be expected in practice, but by making it we achieve a better privacy guarantee. To illustrate the importance of basing our privacy analysis on such strong adversaries, assume that we considered only potential leakage of information from $\mathsf{PW}(\mathcal{V}_{[1,R]})$, where $\mathcal{V}_{[1,R]} = \langle V_1, \ldots, V_R \rangle$, and made sure that the sequential release $\mathcal{V}_{[1,R]}$ would not enable to link any identifier in $V_0$ with any sensitive value with probability greater than $1/\ell$. Such a guarantee does not take into account the possibility that the adversary may use his background knowledge $V_0$ in order to rule out some of the possible worlds in $\mathsf{PW}(\mathcal{V}_{[1,R]})$ and, by thus, achieve linkage probabilities greater than the intended bound of $1/\ell$.

**Example 3.2.** Consider the table $T$ with three tuples having two quasi-identifiers and one sensitive value, and the corresponding releases $V_0, V_1, V_2$, as shown in Table 4. (Attributes that were totally suppressed in those releases are not shown.) An adversary who knows only the quasi-identifier values of the individual who is labeled "a", may deduce from $V_1$ and $V_2$ that there are three possible worlds, as shown in Table 5. Hence, such an adversary can only infer that the sensitive value of "a" is either 1 (in probability 2/3) or 2 (in probability 1/3). On the other hand, the adversary which we assume knows also $V_0$. Such an adversary can rule out the two possible worlds that are inconsistent with $V_0$ (the second and third tables in Table 5) and deduce with certainty that the sensitive value of "a" is 1.

Table 4: A table $T$ and three releases $V_0$, $V_1$ and $V_2$

| $ID$ | $A_1$ | $A_2$ | $A_3$ |
|---|---|---|---|
| a | r | x | 1 |
| b | s | x | 2 |
| c | r | y | 2 |

| $ID$ | $A_1$ | $A_2$ |
|---|---|---|
| a | r | x |
| b | s | x |
| c | r | y |

| $\overline{A_2}$ | $A_3$ |
|---|---|
| x | 1 |
| x | 2 |
| y | 2 |

| $\overline{A_1}$ | $A_3$ |
|---|---|
| r | 1 |
| s | 2 |
| r | 2 |

□

Table 5: The three possible worlds that are implied by $V_1$ and $V_2$ and the quasi-identifier values of "a"

| $ID$ | $A_1$ | $A_2$ | $A_3$ |
|---|---|---|---|
| a | r | x | 1 |
| ? | s | x | 2 |
| ? | r | y | 2 |

| $ID$ | $A_1$ | $A_2$ | $A_3$ |
|---|---|---|---|
| a | r | x | 1 |
| ? | r | x | 2 |
| ? | s | y | 2 |

| $ID$ | $A_1$ | $A_2$ | $A_3$ |
|---|---|---|---|
| a | r | x | 2 |
| ? | r | x | 1 |
| ? | s | y | 2 |

Before concluding this section we note that both previous studies that dealt with privacy-preservation in sequential releases, i.e. [28, 32], used weaker privacy definitions. The reader is referred to [28] for a thorough discussion of those definitions and their weakness with respect to the $\ell$-diversity definition (Definition 3.3).

## 4. $\ell$-Diverse anonymizations of sequential releases

There are two approaches towards achieving $\ell$-diversity in sequential releases. In the first approach, one produces a single anonymization $V$ of the entire $T$ (including all its $M$ attributes) that respects $\ell$-diversity, according to one of the acceptable definitions of $\ell$-diversity for a single release (e.g. [20, 31, 34, 36]). Then, whenever it is needed to publish a new release that includes some subset of the attributes, say $I_r \subset [M]$, one releases the projection of $V$ on the $I_r$-attributes. The second approach exploits the sequential nature of the release, by which not all attributes in $T$ are released at once.

The first approach is simplistic and leads to excessive generalization, since by considering the entire table as a whole, we assume that the adversary sees at once the entire tuples, with all of their $M$ attributes. However, in the sequential release setting, the adversary sees each time only part of the tuples and, hence, he does not know how to correctly join the different "pieces" of each tuple. Therefore, by exploiting the sequential nature of the release, it is possible to achieve the same privacy goal with less information loss, as we exemplify later on. Another advantage of the second approach is that it can cope with situations in which new attributes, which were not known upfront, are added to the table.

Our sequential anonymization algorithm takes the second approach. Most of this section is devoted to describing and discussing that algorithm (Sections 4.1–4.4). We conclude in Section 4.5 with a description of an algorithm that follows the first approach (the non-sequential one) which will be used later in our experiments.

*4.1. A bird's-eye view of the sequential anonymization algorithm*

In Section 3 we defined the privacy notion of $\ell$-diversity, based on the collection of possible worlds. Verifying satisfiability of that definition may be infeasible, due to the exponential cardinality of $\mathsf{PW}(\mathcal{V})$. Our proposed strategy achieves the same privacy goal, in polynomial time, as we proceed to explain.

Let $T = \{t_1, \ldots, t_N\}$ (the real world) be as given on the left of Table 6; it is described in a schematic manner, where $q_n$ represents the quasi-identifier part of $t_n$ (attributes 1 to $Q$), $r_n$ is the non-identifier part (attributes $Q+1$ to $M-1$), and $s_n$ is the sensitive attribute (attribute $M$), $n \in [N]$. We randomly create $\ell - 1$ fake worlds by permuting the non-identifier and sensitive attributes, using the same permutation $\pi_i$ for all those columns in the $i$th possible world, as shown on the right of Table 6. Letting $E$ be the set of all $\ell$ possible worlds (the true one and the $\ell - 1$ fake ones), as shown in Table 6, we make sure that:

1. All fake worlds in $E$ are generated so that they are semantically close to $T$ in the following sense: in each of those worlds, every tuple is generated as a concatenation of the quasi-identifier part of some tuple $t \in T$ with the non-identifier and sensitive values of another tuple $t' \in T$, such that $t$ and $t'$ have similar quasi-identifiers. In other words, for each $n \in [N]$ and $1 \leq i \leq \ell - 1$, we would like the tuple $(q_n,\, r_{\pi_i(n)},\, s_{\pi_i(n)})$ of the $i$th fake world to be semantically close to its matching tuple in $T$, $(q_{\pi_i(n)},\, r_{\pi_i(n)},\, s_{\pi_i(n)})$. Since the non-identifier and sensitive values of the two tuples are identical, we only need to make sure that the quasi-identifier values of the two tuples are similar.

2. $E$ is $\ell$-diverse; i.e., for each $n \in [N]$, the $\ell$ sensitive values $s_{\pi_i(n)}$, $1 \leq i \leq \ell$, (where $\pi_\ell$ denotes the identity permutation which corresponds to the real world $T$) are distinct.

Table 6: $\ell$-diverse possible worlds

| qid | n.i. | sens. |
|---|---|---|
| $q_1$ | $r_1$ | $s_1$ |
| $q_2$ | $r_2$ | $s_2$ |
| $\vdots$ | $\vdots$ | $\vdots$ |
| $q_N$ | $r_N$ | $s_N$ |

;

| qid | n.i. | sens. |
|---|---|---|
| $q_1$ | $r_{\pi_i(1)}$ | $s_{\pi_i(1)}$ |
| $q_2$ | $r_{\pi_i(2)}$ | $s_{\pi_i(2)}$ |
| $\vdots$ | $\vdots$ | $\vdots$ |
| $q_N$ | $r_{\pi_i(N)}$ | $s_{\pi_i(N)}$ |

$1 \leq i \leq \ell - 1$

After this preliminary computation, we generalize each release in its turn until it becomes consistent with each of those possible worlds. In doing so, we consider all of those possible worlds equivalent (see Section 4.3). Namely, we generalize the next release so that if $\mathcal{V}$ is the sequential release up to that release (including), then each $W \in E$ is a possible world in $\mathsf{PW}(\mathcal{V})$.

The above described method achieves effectively the same privacy goal as that which underlies Definition 3.3:

**Theorem 4.1.** *Let $\mathcal{V}$ be a sequential release of $T$ that was generated by the above described method. Then even an adversary who knows the anonymization algorithm and is computationally-unbounded cannot infer from $\mathcal{V}$ links between quasi-identifier tuples and sensitive values in probability greater than $1/\ell$.*

**Proof.** Given $\mathcal{V}$, the adversary, who is computationally-unbounded, may enumerate the set $\mathsf{PW}(\mathcal{V})$ of all possible worlds that are consistent with $\mathcal{V}$. The adversary, who knows the anonymization algorithm, may then infer that the true world $T$ has in $\mathsf{PW}(\mathcal{V})$ $\ell - 1$ possible worlds $T_1, \ldots, T_{\ell-1}$, such that the set $\{T_1, \ldots, T_{\ell-1}, T_\ell := T\}$ is $\ell$-diverse. Hence, he may look in $\mathsf{PW}(\mathcal{V})$ for all subsets $E \subset \mathsf{PW}(\mathcal{V})$ such that $|E| = \ell$ and $E$ is $\ell$-diverse. Let us denote by $\mathcal{E} = \{E_1, \ldots, E_z\}$ the collection of all such subsets. Then the adversary knows that the true subset $E$ that was used in generating $\mathcal{V}$ is one of the subsets in $\mathcal{E}$. Assume further that the adversary has a probability belief distribution over $\mathcal{E}$, say $\Pr(E_i)$, $1 \leq i \leq z$. (For example, as the adversary knows that the data holder attempted to find a subset $E$ where the $\ell$ possible worlds are semantically close, he may base his belief probabilities on that knowledge.)

Let $v_n^0 \in V_0$ be a quasi-identifier tuple, $n \in [N]$, and $s \in A_M$ be a sensitive value. Assume that the adversary is told that the subset of $\ell$-diverse possible worlds that was used by the data owner when he computed the sequential release was $E_i$, for some $1 \leq i \leq z$. Then, given that information, the adversary's linkage belief probability between $v_n^0$ and $s$ would equal $1/\ell$, if $E_i$ contains a possible world in which $v_n^0$ is linked with the sensitive value $s$ (because all $\ell$ possible worlds in $E_i$ are equally likely and only in one of them $v_n^0$ is linked with $s$), and zero otherwise. Therefore, as the adversary has a belief probability $\Pr(\cdot)$ over $\mathcal{E} = \{E_1, \ldots, E_z\}$, where $\Pr(E_i)$ is his belief that $E_i$ was the actual subset that was used, the resulting linkage belief probability between $v_n^0$ and $s$ is $P(v_n^0, s) := \frac{1}{\ell} \cdot \sum_i \Pr(E_i)$, where the sum is taken over all subsets $E_i$ which contain a possible world that links $v_n^0$ and $s$. Therefore, $P(v_n^0, s) \leq \frac{1}{\ell} \cdot \sum_{1 \leq i \leq z} \Pr(E_i) = \frac{1}{\ell}$. Hence, even such a computationally un-

bounded adversary cannot infer from $\mathcal{V}$ links between quasi-identifier tuples and sensitive values in probability greater than $1/\ell$. □

The case of $\ell$-linkability is easier: clearly, the sequential release as generated by the above described approach links every quasi-identifier tuple to at least $\ell$ different sensitive values. Recall that the previous algorithms in [32] and [28] do not guarantee even the relaxed notion of $\ell$-linkability, as discussed in Section 1.2.

We proceed to describe the algorithm in detail. In Section 4.2 we describe the preliminary computation, and in Section 4.3 we describe the anonymization of each release in the sequence of releases.

### *4.2. Generating the fake possible worlds*

Our goal is to create $\ell - 1$ fake worlds as described above. Bearing in mind that each of the future releases will have to be generalized in a manner that will make it consistent with each of those fake worlds, we have to generate fake worlds that will entail low information losses. Algorithm 1, that we proceed to describe, does that; it is inspired by the algorithm for non-homogeneous anonymization of [36].

The algorithm starts with $\ell - 1$ empty fake possible worlds (Step 1). Then, it divides $T$'s tuples into buckets of diversity at least $\ell$ such that the overall generalization cost of the buckets' closures (see Definition 2.2) is small (Step 2). To that end, we may adopt any algorithm for $\ell$-diverse anonymization. (In our experiments we adopted the Mondrian algorithm [18].)

The rest of the work proceeds within each bucket $B$ independently (Steps 3-8). For each bucket $B$, the algorithm orders the sensitive values in it in a cycle where every $\ell$ consecutive values are distinct (Step 4). As each bucket has diversity at least $\ell$, such an ordering exists. (We provide a constructive proof of that in Section 8.2 in the Appendix.)

After doing so, it finds an ordering of $B$'s tuples that agrees with the ordering of the sensitive values (Step 5). Assuming that $B$'s tuples have $S$ distinct sensitive values, say $a_1, \ldots, a_S$, and $B$ has $n_s$ tuples with the sensitive value $a_s$, $1 \leq s \leq S$, there are $\prod_{s=1}^{S}(n_s!)$ ways of ordering $B$'s tuples in accord with a given ordering of the sensitive values. We select one of these orderings in a greedy manner in order to minimize future generalization costs. We defer the description of this greedy subroutine to a later stage where it will be more easily understood.

Next (Step 6), we randomly select $\ell - 1$ permutations, $\sigma_1, \ldots, \sigma_{\ell-1}$, over $B = \{t_0^B, \ldots, t_{|B|-1}^B\}$ that satisfy the following conditions (indices of $B$'s tuples are modulo $|B|$):

1. Range limitation: Every permutation satisfies $\sigma_i(t_h^B) \in \{t_{h+1}^B, \ldots, t_{h+\ell-1}^B\}$ for all $0 \leq h \leq |B| - 1$.

2. Diversity: If $1 \leq i \neq i' \leq \ell$ then $\sigma_i(t_h) \neq \sigma_{i'}(t_h)$ for all $0 \leq h \leq |B| - 1$, where $\sigma_\ell$ is the identity permutation.

The problem of randomly generating such permutations is equivalent to the problem of finding so called "match different" perfect matchings in bipartite graphs, that was discussed in [36]. A simple algorithm that generates such perfect matchings based on random walks and backtracking is described in [36, Section 5.2]. Its basic time complexity is $O(\ell|B|^2)$, but Wong et al. describe heuristics to reduce the runtime of that procedure. (The reader is referred to [36, Section 5.2] for further details.)

Then (Step 7), we add to the fake world $T_i$ the tuples $\sigma_i(B)$, where $\sigma_i(B)$ holds the tuples in $B$ after permuting their suffixes (the part that consists of the non-identifiers and the sensitive attribute) according to $\sigma_i$ :

$$\sigma_i(B) := \{(t(ID), t(1), \ldots, t(Q), \sigma_i(t)(Q+1), \ldots, \sigma_i(t)(M)) : t \in B\}\,. \quad (5)$$

As $B$'s tuples were ordered so that every cyclically consecutive $\ell$ tuples have different sensitive values, and since the permutations were selected so that they satisfy the above specified range limitation and diversity constraints, we infer that the resulting possible worlds provide the sought-after diversity: each identifier is linked through them to $\ell$ different sensitive values.

We now return to the greedy ordering of the tuples (Step 5), in accord with the ordering of the sensitive values. The fake worlds associate with the quasi-identifiers of $t_h^B$, the suffixes of the tuples $t_{h+1}^B, \ldots, t_{h+\ell-1}^B$. Hence, in order to make each of those fake worlds a possible world in the future releases, we shall have to generalize the tuple $t_h^B$ so that it would be consistent with $t_{h-1}^B, \ldots, t_{h-\ell+1}^B$. The resulting overall generalization cost is then

$$\sum_{h=0}^{|B|-1} IL(\text{closure}(t_h^B, t_{h-1}^B, \ldots, t_{h-\ell+1}^B))\,. \quad (6)$$

Therefore, we order $B$'s tuples greedily in order to minimize the above sum. Specifically, if $|B| = n$ and the ordering of the sensitive values is $z_0, \ldots, z_{n-1}$, we generate an ordering $t_0^B, \ldots, t_{n-1}^B$ of $B$'s tuples so that $t_h^B(M) = z_h$ for all $0 \leq h \leq n-1$, and $t_h^B$ is the tuple in $B \setminus \{t_0^B, \ldots, t_{h-1}^B\}$ (i.e., a tuple that was not

selected yet) that minimizes

$$\sum_{j=1}^{\min\{h,\ell-1\}} IL(\text{closure}\{t_h^B, t_{h-1}^B, \ldots, t_{h-j}^B\}), \tag{7}$$

where ties are broken evenly. (Note that the $j$th term in the sum in Eq. (7) is the contribution of the candidate tuple $t_h^B$ to the generalization cost in (6), given that $t_0^B, \ldots, t_{h-1}^B$ are the $h$ first tuples that were already placed in the ordering.)

**Algorithm 1** Preprocessing the table

**Input:** A table $T = \{t_1, \ldots, t_N\}$; a parameter $\ell \leq \text{div}(T)$.

**Output:** (a) A partition of $T$'s tuples into $\ell$-diverse buckets;
(b) $\ell - 1$ fake worlds, $T_1, \ldots, T_{\ell-1}$.

1: $T_i = \emptyset$, $1 \leq i \leq \ell - 1$.
2: Divide $T$'s tuples into disjoint low cost buckets, where each bucket has diversity at least $\ell$.
3: **for** each bucket $B$ **do**
4: Order the sensitive values in $B$'s tuples in a cycle where every $\ell$ consecutive values are distinct.
5: Order $B$'s tuples in accord with the above ordering of the sensitive values.
6: Randomly select permutations $\sigma_1, \ldots, \sigma_{\ell-1}$ over $B$, that satisfy the range limitation and diversity conditions.
7: For all $1 \leq i \leq \ell - 1$, add to $T_i$ the $|B|$ tuples that are generated by $\sigma_i$ according to Eq. (5).
8: **end for**

**Example 4.1.** To illustrate the operation of Algorithm 1, assume that $\ell = 4$ and that one of the low cost $\ell$-diverse buckets that were generated in Step 2 are as shown on the top of Table 7; we show the bucket tuples in a schematic manner, illustrating their quasi-identifier part (attributes 1 to $Q$), non-identifier part (attributes $Q + 1$ to $M - 1$), and the sensitive attribute (attribute $M$). As the bucket is 4-diverse, the 5 sensitive values, $s_0, \ldots, s_4$, are different. (That implies that the bucket is even 5-diverse.) When processing that bucket (Steps 4-7), we first order the tuples so that every $\ell$ consecutive tuples have distinct sensitive values; in the present example, every ordering will do, and we shall assume that the selected ordering is the one shown on the top of Table 7. Next (Step 6), we randomly select $\ell - 1 = 3$ permutations over $\{0, 1, 2, 3, 4\}$ such that, together with the identity

permutation, form a set of $\ell = 4$ permutations that satisfy the range limitation and diversity conditions. Table 8 shows one possible selection of such a quadruple of permutations. Indeed, each column in Table 8 is a permutation over $\{0, 1, 2, 3, 4\}$; the fourth one is the identity permutation; the range limitation conditions are met (i.e., each value $h$ is mapped, by the non-identity permutations, to one of the values $h + 1, h + 2, h + 3$ mod 5); and all four permutations map each of the values $h$ to four distinct images (e.g., the value 3 is mapped to the four distinct values 3,4,0,1). Finally, the three fake possible worlds that are induced by the non-identity permutations are shown on the bottom of Table 7.

Table 7: A 4-diverse bucket of tuples (top) and three corresponding possible worlds (bottom)

| qid | n.i. | sens. |
|---|---|---|
| $q_0$ | $r_0$ | $s_0$ |
| $q_1$ | $r_1$ | $s_1$ |
| $q_2$ | $r_2$ | $s_2$ |
| $q_3$ | $r_3$ | $s_3$ |
| $q_4$ | $r_4$ | $s_4$ |

| qid | n.i. | sens. |
|---|---|---|
| $q_0$ | $r_1$ | $s_1$ |
| $q_1$ | $r_3$ | $s_3$ |
| $q_2$ | $r_4$ | $s_4$ |
| $q_3$ | $r_0$ | $s_0$ |
| $q_4$ | $r_2$ | $s_2$ |

| qid | n.i. | sens. |
|---|---|---|
| $q_0$ | $r_3$ | $s_3$ |
| $q_1$ | $r_2$ | $s_2$ |
| $q_2$ | $r_0$ | $s_0$ |
| $q_3$ | $r_4$ | $s_4$ |
| $q_4$ | $r_1$ | $s_1$ |

| qid | n.i. | sens. |
|---|---|---|
| $q_0$ | $r_2$ | $s_2$ |
| $q_1$ | $r_4$ | $s_4$ |
| $q_2$ | $r_3$ | $s_3$ |
| $q_3$ | $r_1$ | $s_1$ |
| $q_4$ | $r_0$ | $s_0$ |

Table 8: Four permutations over five elements that satisfy the range limitation and diversity conditions

| $\sigma_1$ | $\sigma_2$ | $\sigma_3$ | $\sigma_4 = id$ |
|---|---|---|---|
| 1 | 3 | 2 | 0 |
| 3 | 2 | 4 | 1 |
| 4 | 0 | 3 | 2 |
| 0 | 4 | 1 | 3 |
| 2 | 1 | 0 | 4 |

□

**Theorem 4.2.** *The set $E := \{T_1, \ldots, T_{\ell-1}, T_\ell := T\}$ of $\ell$ possible worlds that Algorithm 1 generates is $\ell$-diverse (Definition 3.2).*

**Proof.** As $|E| = \ell$, we only need to show that for any $v_n^0 \in V_0$, each of the possible worlds in $E$ associates a different sensitive value. This is indeed the case since the generated permutations $\sigma_1, \ldots, \sigma_\ell$ satisfy the range limitation and diversity conditions (see Step 6 in the algorithm), and, in addition, the tuples within each bucket are arranged so that every $\ell$ cyclically consecutive tuples have distinct sensitive values (Step 4). □

*4.3. Computing a new release that maintains $\ell$-diversity*

Let $I_r$, $r \geq 1$, be the subset of attributes that should be included in the release $V_r$. $I_r$ is typically a subset of $[M]$, which is the set of attributes that was known upfront during the preprocessing stage in which the $\ell$-diverse possible worlds were created. But the algorithm that we present here applies also to cases in which the data owner acquired new type of data and then he augments the table with new attributes that were not known upfront and need to be included in the new release $V_r$. In both of those cases, $V_r$ will be a generalization of $T|_{I_r}$. Our goal is to generalize each release, with minimal information loss, so that it becomes consistent with the table $T$ and the $\ell - 1$ fake worlds that were generated by Algorithm 1. Algorithm 2 does that.

Before explaining that algorithm, we discuss the case where the new release $V_R$ includes some new attributes for the first time. As those attributes were not known when the possible worlds were generated, they will be suppressed in the possible worlds. Recall that the possible world $T_i$ was induced by some permutation $\sigma_i$ of the tuples of the database. For example, if $\sigma_i$ matched $t_1$ with $t_2$, then the tuple that $T_i$ holds, in lieu of the true $t_1$, would be

$$I_i(t_1) := (t_1(ID), t_1(1), \ldots, t_1(Q), t_2(Q+1), \ldots, t_2(M))\,; \tag{8}$$

namely, it ties together the identifier and quasi-identifiers of $t_1$ with the non-identifiers and sensitive value of $t_2$ (see Eq. (5)). Assume that the quasi-identifier $A_Q$ and the non-identifier $A_{Q+1}$ are revealed for the first time in $V_R$. Then we update their values in $T_i$ (that were suppressed so far) as implied by Eq. (8); i.e., the $Q$th entry in $I_i(t_1)$ will be set to its newly-revealed value in $t_1$ while the $(Q+1)$th entry in $I_i(t_1)$ will be set to its newly-revealed value in $t_2$.

Algorithm 2 receives as an input the set of possible worlds, $T_i$, $1 \leq i \leq \ell$, of which $T_\ell$ is the true world $T$. In addition, it receives the subset of attribute indices $I_R$ to be included in the $R$th release. It produces a generalized table $V_R$ which is consistent with $(T_i)|_{I_R}$ for all $1 \leq i \leq \ell$, while keeping the information loss as small as possible. It achieves that goal by considering the possible worlds, each

one in its turn, according to a randomly generated order (Step 1). $V_R$ is initialized to be the projection of the first possible world in the selected order on the $I_R$ attributes (Step 2). It then starts a loop over the remaining $\ell - 1$ possible worlds (Steps 3-9). In each iteration, it examines the current $V_R$ versus the current possible world, denoted by $W$. It then generalizes the tuples of $V_R$, while minimizing the entailed information loss, until $W$ becomes a possible world for $V_R$. (We explain below in greater detail how this is achieved.) Thus, after the completion of the loop, $V_R$ is a generalized table for which all $T_i$, $1 \leq i \leq \ell$, are possible worlds.

**Algorithm 2** Computing an $\ell$-diverse release

**Input:** Possible worlds $T_i = \{t_1^i, \ldots, t_N^i\}$, $1 \leq i \leq \ell$, where $T_\ell = T$.
A subset of attribute indices $I_R \subset [M]$.

**Output:** A generalization of $T|_{I_R}$ that is consistent with the given possible worlds.

1: Generate a random permutation $\psi$ on $[\ell] = \{1, \ldots, \ell\}$.
2: Initialize $V_R = \{v_1^R, \ldots, v_N^R\}$ to be the projection $T_{\psi(1)}|_{I_R}$.
3: **for** $2 \leq i \leq \ell$ **do**
4: Set $W = T_{\psi(i)}$ and denote by $\{w_1, \ldots, w_N\}$ the tuples in $W$.
5: Find a minimal cost perfect matching $\pi$ between the tuples of $W$ and the tuples of $V_R$.
6: **for** $1 \leq n \leq N$ **do**
7: If $\pi$ matches $v_n^R \in V_R$ with $w_{j_n} \in W$, generalize the quasi-identifiers of $v_n^R$ until they are consistent with those of $w_{j_n}$.
8: **end for**
9: **end for**

In order to make a given table $W$ a possible world for $V_R$, we consider the bipartite graph $G$ on $W$ and $V_R$, where an edge connects $v_n^R \in V_R$ and $w_{n'} \in W$ if and only if those two tuples satisfy the condition

$$v_n^R(m) = w_{n'}(m) \quad \forall m \in I_R \,, m \geq Q + 1 \,. \tag{9}$$

Namely, $G$ connects tuples that agree in the attributes which are not subjected to generalization (i.e., the non-identifier and sensitive attributes, $m \geq Q + 1$) and are included in $I_R$. If condition (9) is verified, it implies that $v_n^R$ could be made consistent with $w_{n'}$ by generalizing the quasi-identifiers of the former.

Next, we define edge weights: The weight of the edge connecting $v_n^R$ and $w_{n'}$ is the distance between them in terms of information loss,

$$\text{dist}(v_n^R, w_{n'}) = IL(\text{closure}\{v_n^R, w_{n'}\}) - IL(v_n^R) \,.$$

Namely, it equals the addition to the overall information loss in case we generalize the quasi-identifiers of $v_n^R$ to be consistent also with those of $w_{n'}$ (see Definition 2.2 and Eq. (1)). Our task now is to find a minimal weight perfect matching $\pi$ in $G$ (Step 5). This problem can be solved optimally by The Kuhn-Munkers algorithm, a.k.a The Hungarian algorithm [16, 23]. Once we find $\pi$, we generalize the tuples of $V_R$ to be consistent with their $\pi$-matched tuples in $W$ (Steps 6-8).

**The following example illustrates the operation of Algorithm 1 and Algorithm 2.**

**Example 4.2.** Consider the table $T$ with two quasi-identifiers, $A_1, A_2$, and a sensitive attribute, $A_3$, in Table 9(a). Assume that we wish to publish two releases of $T$ — one with $I_1 = \{1, 3\}$ and another with $I_2 = \{2, 3\}$, while respecting 2-diversity. **We achieve that by applying Algorithm 1 and Algorithm 2 consecutively as we proceed to explain.**

First, we need to generate a possible world $T_1$ that differs from the real possible world $T$. **This is done by applying Algorithm 1. That algorithm starts by dividing $T$s tuples into disjoint low cost buckets, where each bucket has diversity at least $\ell$ (Step 2). For the sake of simplicity, $T$ in our example consists of just one bucket with 5 tuples, and they are already sorted in a way that every two cyclically consecutive sensitive values are different (whence the bucket is 2-diverse). Hence, Steps 4-5 in Algorithm 1 are already accomplished. Then, we need to generate $\ell - 1$ permutations that, together with the identity permutation, satisfy the range limitation and diversity conditions (Step 6). In our case, the permutation $\sigma_1(t_h) = t_{(h+1) \bmod 4}$, $0 \leq h \leq 4$, satisfies the range limitation and diversity constraints. The fake world that it induces through Eq. (5) (see Step 7 in Algorithm 1), is given in Table 9(b).**

In order to compute $V_1$ as a generalization of $T|_{I_1}$, **we apply Algorithm 2.** We put side by side $T|_{I_1} = T|_{\{1,3\}}$ and the possible world $T_1$, see Table 10(a), and then we look for a perfect matching between those two tables that connects tuples that agree in their sensitive value, and achieves a minimal cost **(Step 5 in Algorithm 2). After doing so, we generalize the records in the table to be released in accord with the found matching (Steps 6-8).**

The sensitive values induce a partition of the 5 tuples on each side to 3 buckets: those with the sensitive value 1 (there is only one such tuple), those with the sensitive value 2 (there are two such tuples), and those with the sensitive value 3 (two tuples).

The tuple (r,1) in $T|_{I_1}$ can go only with (e,t,z,1) in $T_1$; hence, we generalize the tuple (r,1) to (rt,1), as can be seen in the final release $V_1$, which is shown in Table

10(b). The tuples (r,2) and (s,2) on the left ($T|_{I_1}$) can be matched with (a,r,x,2) and (c,s,y,2) on the right ($T_1$) without generalization. Therefore, they appear in their original form in the final release $V_1$. Finally, the tuples (s,3) and (t,3) on the left can be matched with (b,r,y,3) and (d,s,x,3) on the right. If the distance function that is induced by the information loss measure is a metric (in the sense that it satisfies the triangle inequality), the optimal matching is to match (s,3) with (d,s,x,3) (no generalization is needed) and (t,3) with (b,r,y,3) (this requires to generalize "t" to "rt"). Hence, $V_1$ includes the tuples (s,3) and (rt,3).

The second release is computed similarly, see Table 11. Table 12 shows the two releases $V_1$ and $V_2$ together with the background knowledge table $V_0$.

Table 9: (a) A table $T$ and (b) A fake world $T_1$.

| $ID$ | $A_1$ | $A_2$ | $A_3$ |
|---|---|---|---|
| a | r | x | 1 |
| b | r | y | 2 |
| c | s | y | 3 |
| d | s | x | 2 |
| e | t | z | 3 |

| $ID$ | $A_1$ | $A_2$ | $A_3$ |
|---|---|---|---|
| a | r | x | 2 |
| b | r | y | 3 |
| c | s | y | 2 |
| d | s | x | 3 |
| e | t | z | 1 |

Table 10: (a) $T|_{I_1}$ versus $T_1$ (b) $V_1$.

| $A_1$ | $A_3$ |
|---|---|
| r | 1 |
| r | 2 |
| s | 3 |
| s | 2 |
| t | 3 |

| $ID$ | $A_1$ | $A_2$ | $A_3$ |
|---|---|---|---|
| a | r | x | 2 |
| b | r | y | 3 |
| c | s | y | 2 |
| d | s | x | 3 |
| e | t | z | 1 |

| $\overline{A_1}$ | $A_3$ |
|---|---|
| rt | 1 |
| r | 2 |
| s | 3 |
| s | 2 |
| rt | 3 |

Table 11: (a) $T|_{I_2}$ versus $T_1$ (b) $V_2$.

| $A_2$ | $A_3$ |
|---|---|
| x | 1 |
| y | 2 |
| y | 3 |
| x | 2 |
| z | 3 |

| $ID$ | $A_1$ | $A_2$ | $A_3$ |
|---|---|---|---|
| a | r | x | 2 |
| b | r | y | 3 |
| c | s | y | 2 |
| d | s | x | 3 |
| e | t | z | 1 |

| $\overline{A_2}$ | $A_3$ |
|---|---|
| xz | 1 |
| y | 2 |
| y | 3 |
| x | 2 |
| xz | 3 |

□

Table 12: $V_0$, $V_1$ and $V_2$.

| $ID$ | $A_1$ | $A_2$ |
|---|---|---|
| a | r | x |
| b | r | y |
| c | s | y |
| d | s | x |
| e | t | z |

| $\overline{A_1}$ | $A_3$ |
|---|---|
| rt | 1 |
| r | 2 |
| s | 3 |
| s | 2 |
| rt | 3 |

| $\overline{A_2}$ | $A_3$ |
|---|---|
| xz | 1 |
| y | 2 |
| y | 3 |
| x | 2 |
| xz | 3 |

**Theorem 4.3.** *The generalization $V_R$ of $T|_{I_R}$ that Algorithm 2 generates is consistent with each of the possible worlds $T_1, \ldots, T_\ell$.*

**Proof.** Let $\psi$ be the random permutation on $[\ell]$ that was generated in Step 1 of the algorithm. After Step 2, $V_R$ is consistent with $T_{\psi(1)}$, since $V_R$ is initialized to $T_{\psi(1)}|_{I_R}$. Then, after the $i$th iteration of the loop in Steps 3-8, $V_R$ is further generalized to be consistent also with $T_{\psi(i)}$, $2 \leq i \leq \ell$. Hence, after completing the loop, $V_R$ is consistent with all of the $\ell$ possible worlds $T_1, \ldots, T_\ell$. □

**Theorem 4.4.** *The generalization $V_R$ of $T|_{I_R}$ that Algorithm 2 generates, as well as the time to compute it, are indifferent to the index $R$ of the release and to the previous releases in the sequence.*

**Proof.** Let us denote by $\tau = 0$ the time of the preprocessing stage, by $\tau = R$ the time of release $R$, and by $\tau = \infty$ any time in the future until which all columns of the table are exposed. For any $\tau \in \{0, R, \infty\}$ we let $T^\tau$ denote the table that is known at time $\tau$, and $T_1^\tau, \ldots, T_{\ell-1}^\tau$ be the corresponding fake worlds at that time.

Algorithm 2 computes $V_R$, the generalization of of $T|_{I_R}$, based only on $I_R$ and the possible worlds $T_1^R, \ldots, T_{\ell-1}^R, T^R$. Consider a clairvoyant Algorithm 2C that operates exactly like Algorithm 2, but instead of $T_1^R, \ldots, T_{\ell-1}^R, T^R$ it uses $T_1^\infty, \ldots, T_{\ell-1}^\infty, T^\infty$. We claim as follows:

CLAIM 1. For each $1 \leq j \leq \ell$, $T_j^R|_{I_R} = T_j^\infty|_{I_R}$ (the case $j = \ell$ corresponds to the true world). Indeed, $T_j^\infty$ differs from $T_j^R$ only in the columns that will be exposed after time $R$. (Each of those columns is suppressed in $T_j^R$ but has specific values in $T_j^\infty$, $1 \leq j \leq \ell$.) Hence, the two tables coincide on the $I_R$-columns.

CLAIM 2. Algorithms 2 and 2C will compute the same $V_R$. Indeed, as both algorithms concentrate only on the $I_R$-columns of the possible worlds, and those columns are the same in $T_j^R$ and $T_j^\infty$ (Claim 1), the equality of the corresponding outputs follows.

CLAIM 3. The fake worlds $T_1^\infty, \ldots, T_{\ell-1}^\infty$ can be computed at time $\tau = 0$, given $T^0$ and $T^\infty$. Indeed, after computing the permutations $\sigma_1, \ldots, \sigma_{\ell-1}$ and the corresponding fake worlds $T_1^0, \ldots, T_{\ell-1}^0$, the fake worlds $T_1^\infty, \ldots, T_{\ell-1}^\infty$ are derived through Eq. (8) in Section 4.3.

Finally, Claims 2 and 3 above imply that the output $V_R$ of Algorithm 2 depends only on $I_R$, $T^0$ and $T^\infty$. In particular, it does not depend neither on the index $R$ of the release nor on the previous releases $V_1, \ldots, V_R$. Arguing along the same lines we infer that also the runtime for computing $V_R$ has no dependence on neither $R$ nor $V_1, \ldots, V_R$. □

A direct consequence of Theorem 4.4 is that our algorithm can be run in parallel in order to compute simultaneously all releases (as opposed to the algorithms in [28, 32] that need to compute all previous releases before starting to compute the next release in the sequence). Specifically, if the subsets of attributes $I_1, I_2, \ldots, I_R$ are known in advance, they can be computed in parallel on $R$ different machines.

Before concluding this section we note the following. Most anonymization algorithms aim at minimizing the information loss in the anonymized tables that they output. More specifically, such algorithms will not generalize the data more than necessary for achieving the underlying privacy requirement. However, striving for minimal information loss may allow so-called "minimality attacks" [33]. In our context, a possible world $W \in \mathsf{PW}(\mathcal{V})$ can be ruled out if an execution of the anonymization algorithm on $W$ cannot output the published release sequence $\mathcal{V}$, because another release sequence that respects $\ell$-diversity exists, and it has a smaller total information loss. As a consequence of such a potential ruling out of possible worlds, some individuals may be linked to some sensitive values with probabilities greater than $1/\ell$. As discussed in [5], one of the countermeasures against minimality attacks is the use of randomization. The sequential algorithm uses randomization twice: Once in Step 6 of Algorithm 1, and once in Step 1 of Algorithm 2. Therefore, minimality attacks are practically ineffective against it. In addition, randomness may be introduced also in Step 2 of Algorithm 1, if using a randomized clustering algorithm (e.g., [14]).

### *4.4. Efficiency and scalability*

The most time consuming procedure in the sequential algorithm is the application of the Hungarian algorithm. The Hungarian algorithm is highly inefficient, as its runtime is cubic in the number of nodes.

The edge structure in the bipartite graph $G$ is defined by condition (9); that edge structure implies that $G$ is a disjoint union of connected components, each

of which is a *complete* bipartite graph[4]. Hence, since there are no edges in $G$ that connect nodes in different connected components, we may apply the Hungarian algorithm separately in each of the connected components. However, even those components may be too large for applying on them the Hungarian algorithm directly. To cope with that problem, we split each of the graph components to sub-components of sizes no greater than some threshold $p$, and then find a perfect matching within each such sub-component. Clearly, smaller values of $p$ will entail less good generalizations, but they will allow faster runtimes.

We explain briefly how to perform such a partition to sub-components of size no larger than $p$. Assume that the entire graph $G$ consists of $K$ connected components of sizes $N_1, \ldots, N_K$, where $\sum_{k=1}^{K} N_k = N$. Consider the $k$th connected component in $G$ that includes $N_k$ tuples from $V_R$ and $N_k$ tuples from the current possible world $W$; we shall denote those two sets of $N_k$ tuples by $V_R^{(k)}$ and $W^{(k)}$ respectively. We first apply any clustering algorithm on the $N_k$ tuples of $V_R^{(k)}$ in order to split them to groups of size no more than $p$, where all tuples within each group are semantically close to each other. Assume that we got $J$ such groups with $n_j$ tuples in the $j$th group, and that (a) $\sum_{j=1}^{J} n_j = N_k$, and (b) $p \geq n_1 \geq \cdots \geq n_J$. Let $u_j$ be the closure (Definition 2.2) of the $n_j$ tuples in the $j$th group, $1 \leq j \leq J$. There is a natural distance between the tuples $w \in W^{(k)}$ and the generalized tuples $u_1, \ldots, u_J$,

$$\text{dist}(u_j, w) = IL(\text{closure}\{u_j, w\}) - IL(u_j)\,. \tag{10}$$

Namely, the distance equals the added information loss in case $u_j$ is further generalized to be consistent with $w$. Next, we compute a corresponding partition of the $N_k$ tuples in $W^{(k)}$. First, we select the $n_1$ tuples that are closest to $u_1$; then, we select from among the remaining $n - n_1$ tuples the $n_2$ ones that are closest to $u_2$, and so forth. This way, we create a partition of the $k$th connected component in $G$ to smaller sub-components (where the partition is geared towards reducing the information loss) and we may proceed to apply the Hungarian algorithm in each of the $J$ sub-components separately.

The runtime of the above described partitioning procedure is analyzed as follows, for every single connected component of size $N_k$ ($1 \leq k \leq K$). The cost of the initial partitioning of $V_R^{(k)}$'s tuples to $J$ sub-components depends on the chosen clustering algorithm. Let us denote that cost by $F_p(N_k, |I_R|)$; the cost depends on $p$, which is the required upper bound on the size of the sub-components, and

[4] A complete bipartite graph is a bipartite graph in which each node in one part is connected by an edge to each of the nodes in the other part.

of course on the number of tuples $N_k$ and the number of attributes $|I_R|$ . Next, the computation of the closures $u_1, \ldots, u_J$ entails a runtime of $O(N_k|I_R|)$. (For the sake of the complexity analysis, we consider the sizes of the attribute domains $A_m$ and the collections of generalized subsets $\overline{A}_m$, $m \in [M]$, as constant.) Next, we turn to estimate the runtime of computing the corresponding partition of the tuples in $W^{(k)}$. To find the tuples in $W^{(k)}$ that are closest to $u_1$ (in terms of the distance in Eq. (10)), we have to compute $N_k$ distances (where each distance computation depends linearly on $|I_R|$) and then find the $n_1$ closest tuples, what requires an additional runtime of $O(n_1 N_k)$. In the next stage, we need to compute $N_k - n_1$ distances and then find the $n_2$ tuples that are closest to $u_2$, in additional cost of $O(n_2(N_k - n_1))$. That procedure has an overall cost of

$$\left(\sum_{j=1}^{J-1}\left(N_k - \sum_{i<j} n_i\right)\right) \cdot |I_R| + \sum_{j=1}^{J-1} n_j(N_k - \sum_{i<j} n_i)\,. \tag{11}$$

For the sake of estimating that runtime, we may assume that $n_j = \Theta(p)$ for all $1 \leq j \leq p$. In that case, the first addend in Eq. (11) is $O(N_k^2/p) \cdot |I_R|$, while the second addend is $O(N_k^2)$. Hence, the overall runtime of that partitioning procedure is

$$\sum_{k=1}^{K} F_p(N_k, |I_R|) + \left(1 + \frac{|I_R|}{p}\right) \sum_{k=1}^{K} O(N_k^2)\,. \tag{12}$$

Now, we turn to analyze the runtime of Algorithm 2:

1. The runtimes of Steps 1 and 2 are $O(\ell)$ and $O(N|I_R|)$, respectively.

2. Then, we start a loop over $\ell - 1$ possible worlds (Steps 3-8). For each possible world we have the following runtimes:

   (a) (Step 5) Create a partition of each connected component to sub-components of size $p$ at most. The runtime of that procedure is given in Eq. (12).

   (b) (Step 5) For each of the sub-components we first compute the $O(p^2)$ distances between the nodes in the corresponding bipartite graph, where the cost of each single distance computation depends linearly on $|I_R|$. Then we apply the Hungarian algorithm, that runs in time $O(p^3)$. Hence, the overall cost in each sub-component is $O(p^2)|I_R| + O(p^3)$. Since there are $O(N/p)$ sub-components altogether, the overall cost of that stage is $O(Np|I_R|) + O(Np^2)$.

(c) (Steps 6-8) After finding the perfect matching in the entire graph $G$, the generalization stage takes $O(N|I_R|)$.

Hence, the overall runtime of Algorithm 2 for computing the $R$th release is

$$O(\ell)+O(N|I_R|)+(\ell-1)\cdot\left(\sum_{k=1}^{K}\left(F_p(N_k,|I_R|)+\left(1+\frac{|I_R|}{p}\right)O(N_k^2)\right)+O(Np|I_R|)+O(Np^2)\right).$$

As can be seen from the above expression, the runtime decreases with $p$ up to some value $p_0$ and then it starts to increase; our experimental evaluation (see Section 6) validates that behavior.

In particular, as implied by Theorem 4.4, Algorithm 2's runtime for computing the anonymization of the $R$th release is independent of $R$, since that computation is indifferent to the history of releases so far. Namely, the time to compute the one hundredth release is on par with the time to compute the first release; the runtime for the $R$th release depends only on the number $|I_R|$ of attributes that are included in it, and on the sizes of $A_m$ and $\overline{A}_m$ for all $m \in I_R$.

The situation with the anonymization algorithm of [28] is different. There, it is necessary to keep all previous releases and to find all full cliques in the growing multipartite graph. As the number of such full cliques may grow exponentially with $R$, it is highly non-scalable. Therefore, on top of the significant advantage in terms of privacy, our algorithm offers also a significant advantage in terms of scalability.

### *4.5. The non-sequential approach*

As noted in the beginning of the section, a more simplistic approach would be to produce a single anonymization $V$ of the entire $T$ that respects $\ell$-diversity and then publish the needed projections of $V$. If the entire table $V$ does not enable to link any quasi-identifier tuple to any sensitive value in probability greater than $1/\ell$, then neither does the combination of all of its projections, since it does not contain additional information. However, as explained earlier, such an approach leads to excessive generalization, as we proceed to illustrate.

Consider the basic table $T$ in Example 4.2 (see Table 9(a)). Its optimal 2-diverse anonymization is as shown on the left of Table 13. That anonymization is a homogeneous one, in the sense that the tuples in it may be clustered into buckets, where within each bucket all tuples have the same generalized quasi-identifiers, and each bucket is $\ell$-diverse (Definition 2.3). However, as shown in [36, 31], $\ell$-diversity may be also obtained through so-called non-homogeneous

anonymizations. The model of non-homogeneous anonymization was introduced in [12] and then further explored in [36, 31]. That model extends the model of homogeneous anonymization and it allows achieving the same privacy goal (be it $k$-anonymity or $\ell$-diversity) with smaller information loss. The anonymization on the right of Table 13 is an optimal non-homogeneous anonymization of the same basic table $T$ that respects 2-diversity. (The exact meaning of 2-diversity here is that even an adversary who has the background knowledge of all quasi-identifier tuples in the table cannot link any quasi-identifier tuple to any sensitive value in probability greater than $1/2$.) As can be seen, that anonymization is better than the homogeneous one, since it has 7 generalized entries, as opposed to 10.

Table 13: 2-Diverse anonymizations of $T$ in Table 9(a) — homogeneous and non-homogeneous

| $\overline{A}_1$ | $\overline{A}_2$ | $A_3$ |
|---|---|---|
| rs | xy | 1 |
| rs | xy | 2 |
| rs | xy | 3 |
| st | xz | 2 |
| st | xz | 3 |

| $\overline{A}_1$ | $\overline{A}_2$ | $A_3$ |
|---|---|---|
| r | xy | 2 |
| rs | y | 3 |
| s | xy | 2 |
| st | xz | 3 |
| rt | xz | 1 |

In view of the above, we base our non-sequential approach on the non-homogeneous anonymization algorithm of [36]. Using the non-homogeneous anonymization in Table 13 for producing the anonymized sequential release in the simplistic approach, we would get the sequential release shown in Table 14. Comparing that sequential release to the one in Table 12, we see that the latter one, which was obtained by our sequential algorithm, is better since it has 4 generalized entries rather than 7.

Table 14: The sequential release that is produced by the non-sequential algorithm.

| $\overline{A}_1$ | $A_3$ |
|---|---|
| r | 2 |
| rs | 3 |
| s | 2 |
| st | 3 |
| rt | 1 |

| $\overline{A}_2$ | $A_3$ |
|---|---|
| xy | 2 |
| y | 3 |
| xy | 2 |
| xz | 3 |
| xz | 1 |

In Section 6 we exemplify the advantages of the sequential algorithm over the non-sequential one through experimentation on real datasets.

## 5. Dynamically changing tables

### *5.1. Related work on privacy-preservation in multiple releases of dynamically changing tables*

Anonymization of dynamically changing tables was discussed in several studies so far. The first study was that of Byun et al. [3]. It dealt with the case of tuple insertions only (i.e., no tuple removals or updates). It considered anonymizations that respect $\ell$-linkability; it did not offer anonymizations that achieve $\ell$-diversity. The case of tuple insertions only was studied also in [10, 25]; the privacy measure that was applied there was that of $k$-anonymity rather than $\ell$-linkability or $\ell$-diversity; Pei et al. [25] assumed that each release includes a key attribute that enables to link tuples that correspond to the same individual in different releases. Xiao and Tao [37] were the first to include in their discussion the case of tuple removals; that scenario in which tuples may be removed presents some difficulties which we describe later on. The subsequent studies [1, 35] considered the case where tuples can be updated between releases.

We note that all of those studies utilized methods of homogeneous anonymization; i.e., the anonymization of each release consists of blocks of tuples which are identical when projected onto the quasi-identifier attributes. In addition, they concentrated on the case in which all releases include the exact same set of attributes. Those two features imply excessive information losses, as we proceed to explain by focusing on the $m$-invariance framework [37]. In that framework, the multiple release has to comply with two conditions: (a) each block of indistinguishable tuples in each release has diversity at least $m$ (see Definition 2.3); and (b) for any tuple in $T$, the multiset of sensitive values in the block to which it belongs in any of the releases remains the same. Namely, an adversary who will try to collect information on Alice's disease will always be able to link Alice to the same multiset of disease values (no matter which of the releases he examines), and that multiset will not include any specific value in relative frequency greater than $1/m$. Therefore, using the $m$-invariance method for anonymizing a sequential release (where each release may have a different set of attributes) is equivalent to applying our non-sequential algorithm, when using homogeneous anonymization. In other words, it reduces to computing a single homogeneous anonymization of $T$ and then publishing the required vertical projections. As discussed and exemplified in Section 4.5, such a simplistic approach is significantly inferior to our sequential algorithm (which exploits the vertical dynamics), even when it uses the more powerful non-homogeneous anonymization model. (See also Section 6 for an experimental validation of that claim.)

The recent study [28] differs from all previously mentioned studies in two aspects: it does not restrict itself to homogeneous anonymizations, and it is the only study so far that handles both vertical and horizontal dynamics (focusing on tuple insertions only). In this section we describe how to extend our framework to include the case of tuple insertions. Hence, both [28] and the current study share the above two features which distinguish them from the other literature. However, they achieve that goal by taking different privacy and anonymization approaches, which translate to different privacy guarantees and scalability (see the Introduction and Section 3).

### *5.2. Extending the sequential algorithm to support addition of tuples*

To cope with the scenario of tuple insertions, let $V_0 = \{v_1^0, \ldots, v_N^0\}$ denote the set of all background knowledge tuples in the table $T$ during its entire life time (see Eq. (3)), and let $f_n \geq 1$ be the index of the release in which $v_n^0$ was inserted to the table $T$. Hence, $v_n^0$ will have a generalized image $v_n^r$ only in releases $V_r$ with $r \geq f_n$. As in [37], we assume that the adversary knows $V_0$ as well as the insertion times $f_n$ of $v_n^0$, for all $n \in [N]$.

Just like [3, 37], we assume that when a new group of tuples is added to the table $T$, that group of tuples is $\ell$-diverse (see Definition 2.3). Such an assumption is inevitable. Indeed, an adversary who targets an individual that was part of that group can compare the sensitive values in releases before and after that group entered $T$ and then extract the subset of sensitive values of the newly added tuples; if the corresponding diversity is smaller than $\ell$, he may link some sensitive values to those tuples with probability greater than $1/\ell$.

Hence, under these assumptions, we may create separate $\ell$-diverse possible worlds for the group of newly added tuples, say $T_1^{\text{new}}, \ldots, T_\ell^{\text{new}}$. Then, if $T_1^{\text{old}}, \ldots, T_\ell^{\text{old}}$ were the selected $\ell$-diverse possible worlds on the "old" population, we augment each of them in the following manner to arrive at an updated list of $\ell$-diverse possible worlds, $T_i^{\text{all}} = T_i^{\text{old}} \cup T_i^{\text{new}}$, $1 \leq i \leq \ell$. (The order in which we pair the "old" and "new" worlds is insignificant, as long as we pair the true "old" world with the true "new" world.) Then, Algorithm 2, which computes the generalization of the next release, is applied as is, without making any further distinction between the old and new tuples.

[28] suggested two approaches for dealing with dynamically changing tables. The so-called *separative* approach treats the new set of tuples as a first release of a new database that is anonymized independently from the old database (the database that includes only the old tuples). The other *unifying* approach considers

the whole set of tuples, old and new, together. The approach that we take here may be seen as a combination of those two approaches. In the preprocessing stage in which we generate the $\ell$ possible worlds we take a separative approach, as we augment the existing set of possible worlds by a new set of possible worlds that are computed only based on the new tuples. But then, when computing the anonymized view of each new release, we unify the two parts of the possible worlds and no longer separate between old and new tuples, as such a unifying approach may achieve lower information losses.

### *5.3. A preliminary discussion of the case of tuple removals*

The case where tuples can be also removed from $T$ (or tuples may be updated, an action that can be seen as a tuple removal followed by a tuple addition) is harder, due to a phenomenon that was called in [37] *critical absence*. For example, if the table $T$ had only one individual with some sensitive value, and that tuple was removed at some point, subsequent releases of $T$ would not include that sensitive value. If the adversary knows when that individual was removed from the table (since he knows, for example, when that individual was discharged from the hospital), he may be able to infer his sensitive value.

That problem was addressed in [37] by introducing counterfeit tuples so that the multiset of sensitive values in the dynamic table will never narrow down. That solution is designed for settings where the table has only three types of attributes — identifiers, quasi-identifiers, and sensitive attributes. However, as discussed in the introduction, there usually exist also attributes of a fourth type (see [2]), which we refer to as non-identifiers. Those attributes, on one hand, are not sensitive, and on the other hand, represent data that an adversary is unlikely to get hold of. Examples for such attributes in health information may include lab results, dietary information, or information on allergies. Such non-identifiers are insignificant, from privacy preservation perspective, in the case of a single release. However, in the case of multiple releases they empower the adversary by allowing him to create links between tuples in different releases.

**Example 5.1.** Consider a table that has two quasi-identifiers, `age` and `zipcode`, one non-identifier, `blood type`, and the sensitive attribute `disease`. Assume that the table is released in two times. The snapshot of the table at the time of the first release is shown in Table 15, alongside its anonymized view. The snapshot of the table at the time of the second release is shown in Table 16, alongside its anonymized view. As can be seen, between the two releases Alice was removed from the data and Janet was added. The shown sequence of two releases satisfies

2-invariance, since for each of the three individuals (Bob, David and Helen), their candidate sensitive set in the first release (namely, the set of diseases that can be linked to them through that anonymization) equals their candidate sensitive set in the second release, and, moreover, that candidate sensitive set has diversity 2. Alas, the existence of the `blood type` attribute changes everything. An adversary who wishes to infer Bob's disease, based on the knowledge of his `age` and `zipcode` and the fact that he appears in both releases, may conclude that Bob suffers from measles, since that is the disease which is attached to blood type O+, which is the only blood type that appears in both blocks in which Bob can be traced.

Table 15: A table at the 1st release (left) and its anonymization (right)

| name | age | zip | b.t. | disease |
|---|---|---|---|---|
| Bob | 21 | 65K | O+ | measles |
| Alice | 23 | 58K | A+ | hepatitis |
| David | 44 | 12K | B− | flu |
| Helen | 48 | 23K | A+ | angina |

| age | zip | b.t. | disease |
|---|---|---|---|
| $[21, 23]$ | [58-65K] | O+ | measles |
| $[21, 23]$ | [58-65K] | A+ | hepatitis |
| $[44, 48]$ | [12-23K] | B− | flu |
| $[44, 48]$ | [12-23K] | A+ | angina |

Table 16: A table at the 2nd release (left) and its anonymization (right)

| name | age | zip | b.t. | disease |
|---|---|---|---|---|
| Bob | 21 | 65K | O+ | measles |
| Janet | 30 | 61K | B+ | hepatitis |
| David | 44 | 12K | B− | flu |
| Helen | 48 | 23K | A+ | angina |

| age | zip | b.t. | disease |
|---|---|---|---|
| $[21, 30]$ | [61-65K] | O+ | measles |
| $[21, 30]$ | [61-65K] | B+ | hepatitis |
| $[44, 48]$ | [12-23K] | B− | flu |
| $[44, 48]$ | [12-23K] | A+ | angina |

□

There are two naïve solutions to that problem. One is to consider the non-identifiers as sensitive too; e.g., in Example 5.1 it entails taking the coupling of `blood type` and `disease` as the sensitive value. However, such a strategy increases significantly the domain of sensitive values (bearing in mind that there could be several non-identifiers); as a consequence, it would be necessary to introduce more counterfeit tuples in lieu of removed tuples, because the probability of finding a newly added tuple with the same sensitive value as a tuple that was removed could be very small. (In other words, it is more probable to find among the newly added individuals one that suffers from a specific disease, than to find one

that suffers from a specific disease *and* has a certain blood type.) The result would be, effectively, the replacement of almost all deleted tuples by counterfeit tuples. That result is of-course not recommended, since in that case it would be better to retain all tuples, instead of replacing them with counterfeit ones that could damage the utility of the table for data mining. A second naïve solution would be to consider those attributes as quasi-identifiers and, therefore, to generalize them so that they too would have the same generalized value in all tuples in the same block. However, such an approach would significantly damage utility.

Hence, the case of tuple removals or updates is genuinely harder. The extension of our algorithms to that case is left for future research.

## 6. Experimental evaluation

We conducted most of our experiments on the CENSUS dataset [26]. It has 500000 tuples with 7 quasi-identifiers ($A_1$ =`age`, $A_2$ =`gender`, $A_3$ =`education level`, $A_4$ =`marital status`, $A_5$ =`race`, $A_6$ =`work class`, $A_7$ =`country`) and one sensitive attribute, $A_8$. We applied on it generalization using taxonomies of heights between 2 and 4, which were adopted from [28]. The diversity of the sensitive attribute in the entire database is 13.425.

Another dataset that we used was ADULT, from the UCI Machine Learning Repository [8]. That dataset was extracted from the US Census Bureau Data Extraction System. It holds demographic information of a small sample of US population with 14 quasi-identifiers such as `age`, `education level`, `marital status`, and `native country` and contains 32,561 tuples. We adopted the taxonomies in [28, 32] for this dataset. Since its sensitive attribute is too narrow (binary), we used one of the quasi-identifiers, `occupation`, as the sensitive attribute in our experiments. It has 15 distinct values and its overall diversity is 7.86.

All experiments were conducted on an Intel Core i7 CPU 2.67 GHz personal computer with 4 GB of RAM, running Windows 7 Enterprise.

### *6.1. Comparing the possible worlds and multipartite consistency graph approaches*

We compared the sequential algorithm to the algorithms proposed in [28] and [32] in three settings that were considered in [28]:

1. two releases, where $I_1 = \{1, 3\}$ and $I_2 = \{1, 8\}$;
2. three releases, where $I_1 = \{1, 2\}$, $I_2 = \{2, 8\}$ and $I_3 = \{1, 8\}$;

3. four releases, where $I_1 = \{1, 2\}$, $I_2 = \{1, 8\}$, $I_3 = \{1, 2, 6\}$ and $I_4 = \{2, 8\}$.

We did not conduct experiments with more than four releases since the algorithms in [28] and [32] are highly non-scalable. (Note that the algorithm in [32] was designed for $R = 2$ only; we used here an extension of it for $R > 2$.)

In the first set of experiments, $N$ was set to $100000$ and the underlying privacy requirement was $5$-linkability. Figures 1, 2 and 3 show the average information losses that were obtained by the three algorithms in the three settings, alongside the total runtime. The displayed information loss is the average over all entries in all releases, as given by the LM measure (see Section 2.2). (If release $r$ includes the attributes $I_r$, then the suppressed entries in that release outside $I_r$ are not included in the average computation.) The total runtime is the sum of the computation time of all releases (including preprocessing time for the sequential algorithm).

In terms of information loss, the sequential algorithm achieves much lower information losses than the algorithm in [32], in all three settings, due to its utilization of the flexible local recoding generalization model. In addition, we see that the algorithm of [28], which also uses the local recoding model, achieves even lower information losses than the sequential algorithm. As for runtime, in the case of two releases (Figure 1) the sequential algorithm is slower than the algorithms in [28] and [32], because of the preprocessing stage. However, in the case of three releases (Figure 2) and four releases (Figure 3), the sequential algorithm is significantly faster than the other two algorithms since the time it takes to anonymize the $R$th release is independent of $R$, while in the other algorithms the dependence on $R$ is exponential.

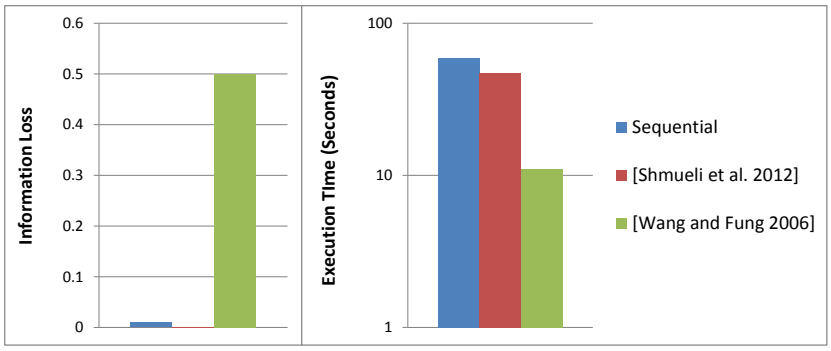


Figure 1: Two releases ($5$-linkability)

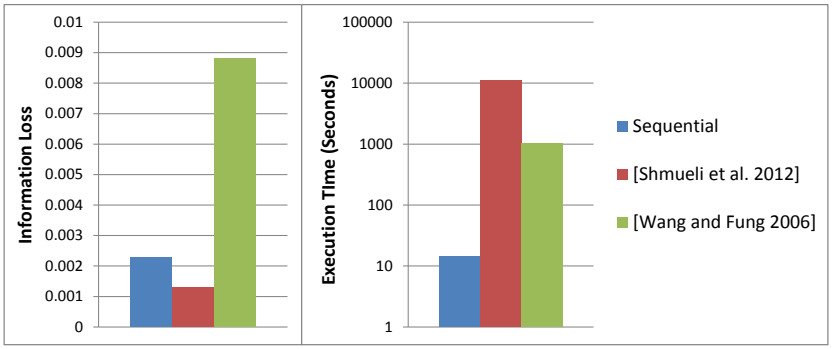


Figure 2: Three releases (5-linkability)

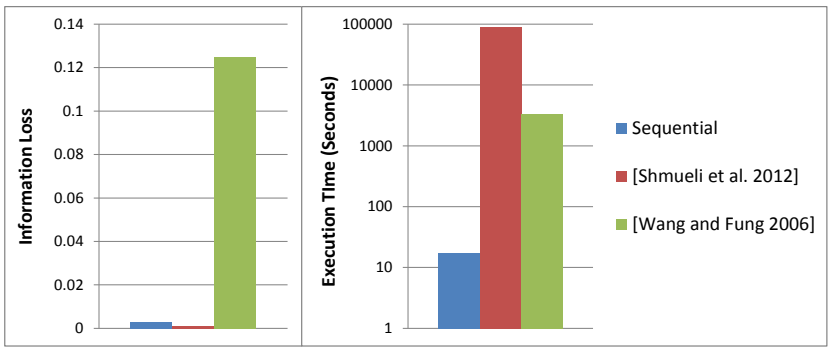


Figure 3: Four releases (5-linkability)

We repeated the above experiments with 3-diversity as the underlying privacy requirement. The results were similar to those reported in the previous experiments. We include here only one figure, Figure 4, that shows the average information loss and total runtime for the three algorithms in the second setting. Note that changing the underlying privacy requirement from 5-linkability to 3-diversity increased the information loss that was obtained by the algorithms of [32] and [28], but decreased the information loss that was obtained by the sequential algorithm. The reason for the latter effect is that the sequential algorithm is designed to achieve the stronger measure of $\ell$-diversity, even when required to satisfy the weaker measure of $\ell$-linkability. In other words, for the sequential algorithm, the underlying privacy requirement was changed from 5-diversity to 3-diversity; that is the reason why the information loss in its output was reduced.

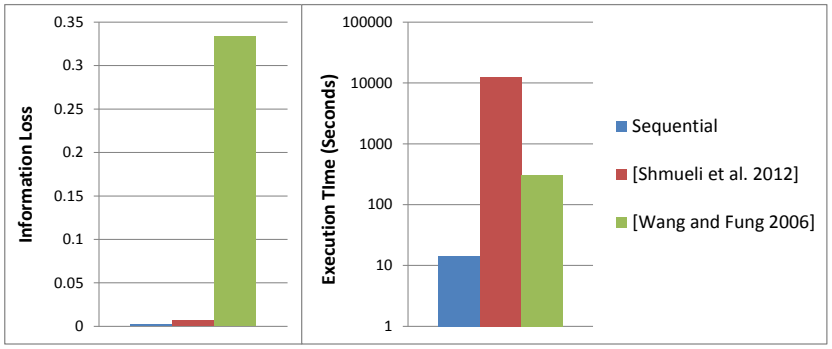


Figure 4: Three releases (3-diversity)

We repeated our experiments with another dataset and also with another measure of information loss. To that end, we considered the following two settings:

1. The CENSUS dataset with $N = 100000$ tuples, where the two releases included attributes $\{1, 5\}$ and $\{1, 5, 8\}$, respectively.

2. The ADULT dataset with $N = 32561$ tuples, where the two releases included attributes $\{\texttt{work class}, \texttt{native country}\}$ and $\{\texttt{work class}, \texttt{native country}, \texttt{occupation}\}$, respectively.

Each of these settings was evaluated twice: once with the LM measure and once with the EM measure. The underlying privacy requirement was 5-diversity. The information losses are reported in Figure 5. The results are consistent with those of the previous experiments.

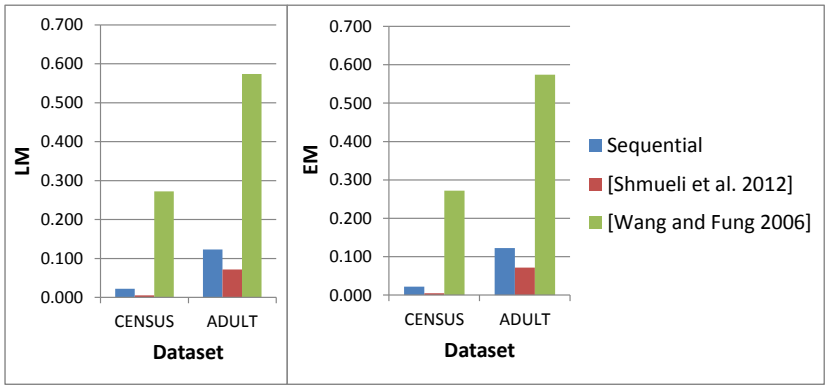


Figure 5: Additional settings of two releases (5-diversity)

To summarize, the sequential algorithm offers staggering improvements in terms of runtime; while the other two algorithms become impractical already for very low values of $R$, our algorithm remains efficient for any value of $R$. (Recall that in our algorithm, the anonymization of each release is independent of the previous releases.) In terms of information loss, the sequential algorithm is substantially better than that of [32], and comparable to the algorithm of [28]. Moreover, we expect the sequential algorithm to obtain lower information loss than the algorithm of [28] for a large number of releases, as it does not suffer from the problem of a reducing privacy budget. However, we were not able to conduct such experiments since the algorithm of [28] is highly non-scalable. In terms of privacy, while the algorithms in [28] and [32] do not guarantee the required linkability or diversity level for $R > 2$, the sequential algorithm does, for all $R$.

### *6.2. Comparing the sequential and non-sequential algorithms*

We tested the performance of the non-sequential and sequential algorithms in terms of information loss and runtime. To that end, we created ten releases, each containing a random subset of the quasi-identifier attributes, and the sensitive attribute: $I_1 = \{2, 6, 7, 8\}$, $I_2 = \{4, 5, 6, 7, 8\}$, $I_3 = \{3, 5, 8\}$, $I_4 = \{1, 8\}$, $I_5 = \{1, 2, 6, 7, 8\}$, $I_6 = \{1, 2, 3, 4, 8\}$, $I_7 = \{2, 3, 5, 8\}$, $I_8 = \{2, 5, 6, 8\}$, $I_9 = \{4, 8\}$, and $I_{10} = \{2, 3, 8\}$.

Figure 6 shows the average LM-information losses in each of the ten releases, as obtained by the non-sequential and sequential algorithm, over $N = 100000$ tuples, when the privacy requirement was 5-diversity. The value of $p$ (see Section 4.4) was set to $p = 100$. Those results, like the example in Section 4.5, show that the sequential algorithm is capable of achieving the same level of diversity with significantly smaller information losses, as it exploits the fact that each release includes only a subset of the attributes.

Recall that the anonymization of each release depends only on the attributes that were selected to be included in that release; it does not depend on the position of that release in the sequence of releases, nor it depends on the preceding releases in the sequence. Therefore, if we had shuffled the order of the ten releases in this example, the average information losses in each of those releases would remain the same as those shown in Figure 6.

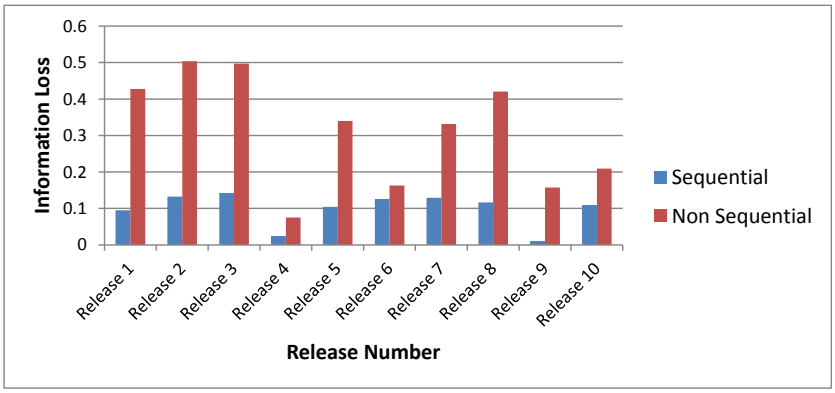


Figure 6: Information loss (LM) in ten releases

Next, we examined the influence of the target diversity parameter. We ran both algorithms on the dataset with $N = 100000$ for five values of $\ell$ and computed the average information loss in all ten releases. As can be seen from Figure 7, the information loss increases with $\ell$, since larger values of $\ell$ require to generalize the releases to be consistent with more possible worlds.

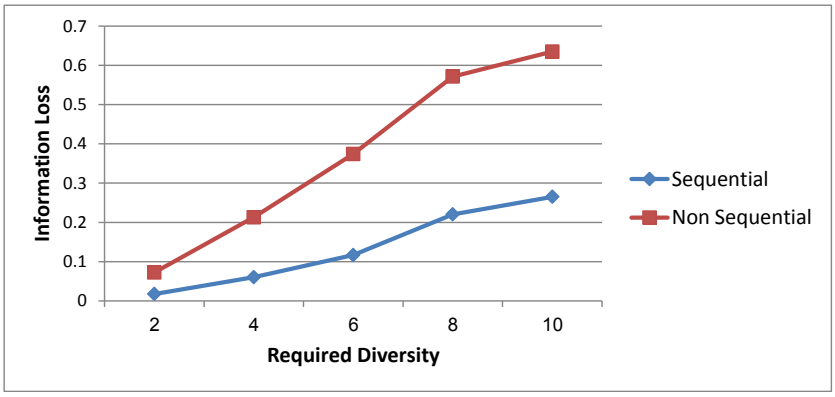


Figure 7: Information loss (LM), different $\ell$

We then examined the influence of $N$ (with $\ell = 5$) and report the average information loss in all ten releases for the two algorithms for various values of $N$ (Figure 8). The difference between the performance of the two algorithms in terms of information loss remains more or less the same.

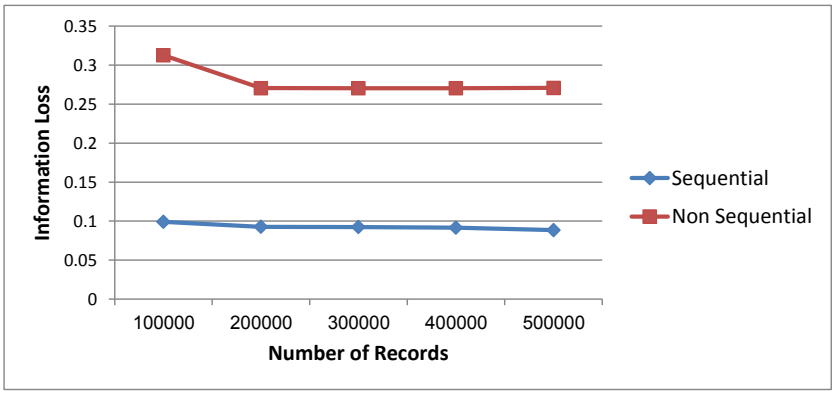


Figure 8: Information loss (LM), different $N$

We also examined the influence of $p$ (with $N = 100000$ and $\ell = 5$) and report the average information loss in each of the ten releases for the sequential algorithm for various values of $p$ (Figure 9). As expected, the information loss decreases with $p$, but the rate of decrease is rather slow.

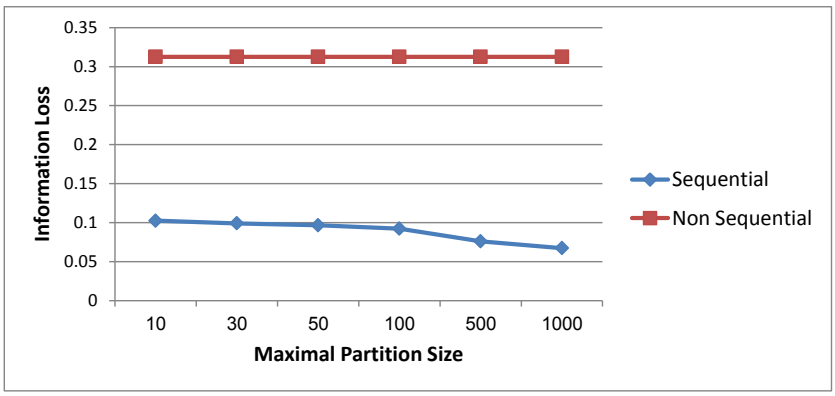


Figure 9: Information loss, different $p$

Figures 10, 11 and 12 report the runtime of the sequential algorithm, as a function of $\ell$ (with $N = 100000$, $p = 100$), and $N$ (with $\ell = 5$, $p = 100$) and $p$ (with $N = 100000$ and $\ell = 5$), respectively. The figures show the runtime of the preprocessing stage and the runtime for the computation of each of the ten releases. (See Section 4.4 for an analysis of the runtime for computing each release; as discussed there, the runtime for a given release depends on the attributes that it includes, but is independent on its position within the sequence of releases.) The time which is

shown for the preprocessing stage of the sequential algorithm reflects also the running time of the non-sequential algorithm. Indeed, the non-sequential algorithm performs the same preprocessing stage, and then only projects the generalized table onto the required subsets of attributes; the latter operation entails a negligible addition to the running time. (In some of the figures, e.g., Figure 12, the preprocessing time is not visible since it is much smaller than the computation times of the following releases.) As expected, the dependence of the runtime on either $\ell$ or $N$ is roughly linear, while the dependence on $p$ is consistent with the analysis in Section 4.4; indeed, the runtime decreases with $p$ up to $p \approx 100$ and then it starts to increase.

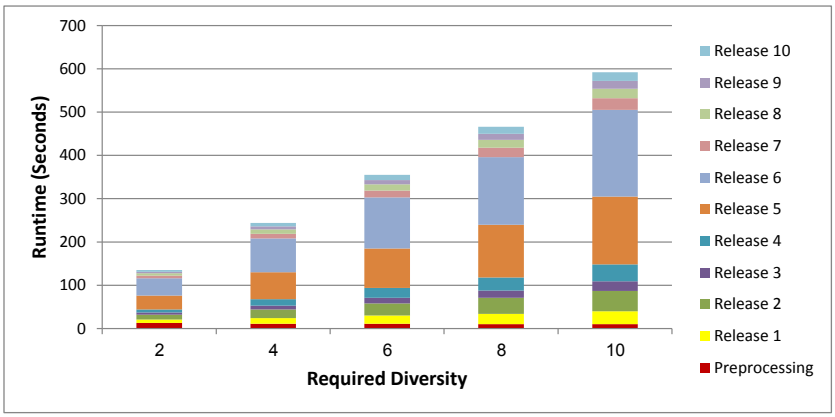


Figure 10: Runtime, different $\ell$

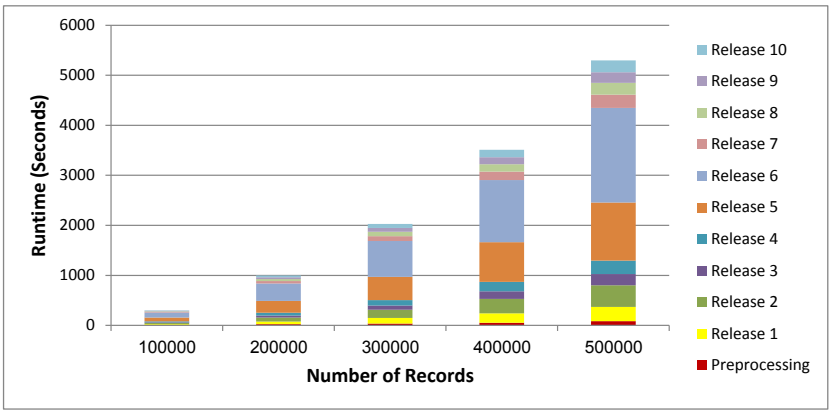


Figure 11: Runtime, different $N$

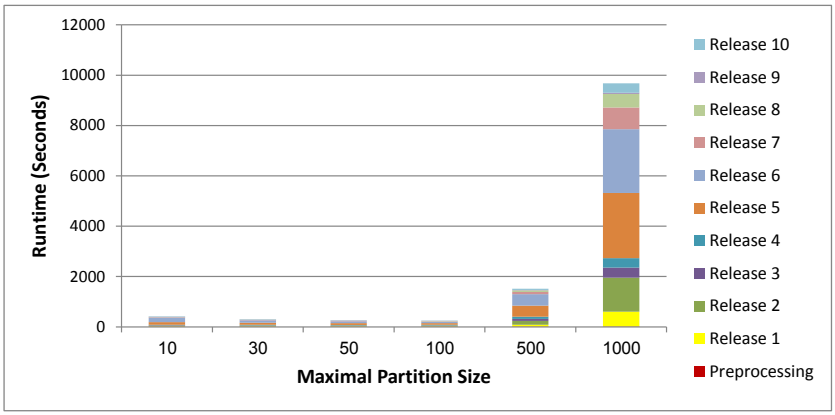


Figure 12: Runtime, different $p$

We conducted all of the above experiments also with the entropy measure instead of LM as in the above experiments. The findings were quite consistent with the ones reported in Figures 6-11. We include herein only one figure, Figure 13, that shows the average entropy information loss in all ten releases, for $N = 100000$ and five values of $\ell$.

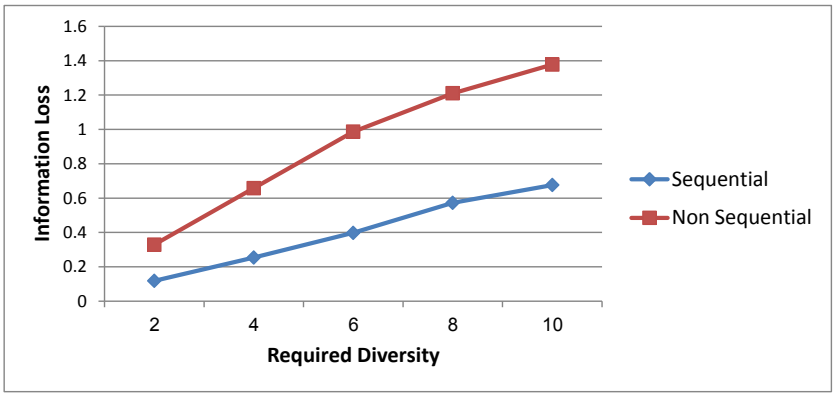


Figure 13: Information loss (EM), different $\ell$

We do not report herein experiments that we performed with additional releases beyond the reported ten releases, since those experiments produced similar results, in the sense that the sequential algorithm constantly yields significantly reduced information losses, and the fact that the runtime for each new release is independent on the number of previous releases.

We repeated all of our experiments also with the ADULT dataset. The results were consistent with those that we obtained with the CENSUS dataset. We include herein only one figure, Figure 14, that shows the average information loss in ten, randomly selected, releases: {age, gen, education, marital status, race, occupation}, {education, marital status, work class, native country, occupation}, {age, gen, marital status, race, occupation}, {age, race, native country, occupation}, {education, marital status, work class, native country, occupation}, {marital status, race, native country, occupation}, {age, gen, marital status, race, native country, occupation}, {age, native country, occupation}, {marital status, occupation}, {education, race, work class, occupation}, for $N = 32561$ and five values of $\ell$.

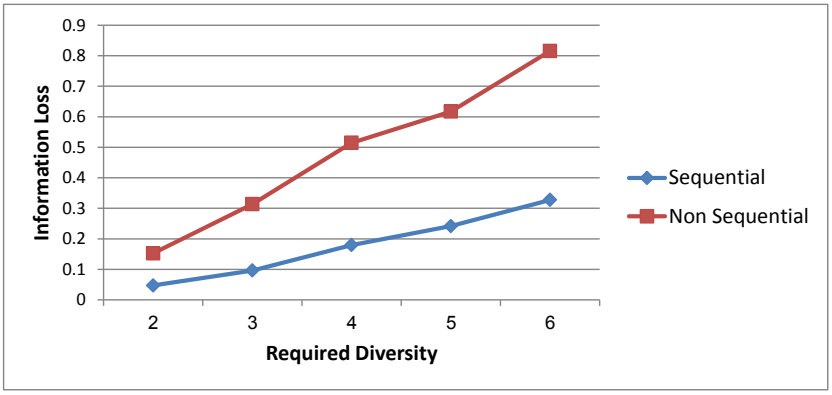


Figure 14: Information loss (Adult), different $\ell$

### *6.3. The dynamic setting*

Here we report results of experiments that we conducted with vertical and horizontal dynamics. To test the performance of the sequential algorithm in the vertically dynamic setting, we created the following four releases: $I_1 = \{2, 5, 8\}$ (namely, $V_1$ includes gender, race and the sensitive attribute $A_8$), $I_2 = \{1, 7, 8\}$, $I_3 = \{1, 3, 4, 8\}$, and $I_4 = \{6, 7, 8\}$. However, this time, the sequential algorithm was initially oblivious of the attributes $A_7$ =country (which is revealed for the first time only in the second release) and $A_6$ =work class (which is revealed only in the fourth release). Figure 15 shows the average information losses in the four releases in that setting, compared to the information losses that were obtained by a "prophetic" sequential algorithm that knows upfront all 8 attributes. The former "adaptive" algorithm achieves slightly better information losses in the first

and third releases, since those consist only of attributes that it knew upfront, and, consequently, the possible worlds that it created are better suited for such releases. The advantages of the prophetic algorithm are expressed in releases 2 and 4, since those releases include attributes of which that algorithm was aware upfront, while the other adaptive algorithm was not.

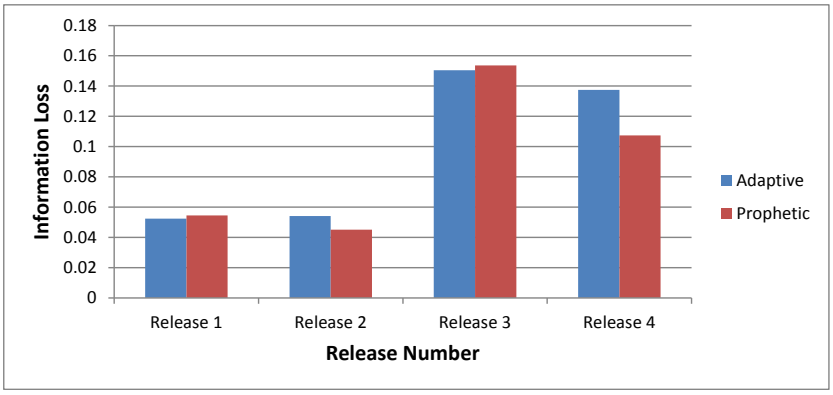


Figure 15: A vertically dynamic setting

To test the performance of the sequential algorithm in the horizontally dynamic setting, we considered the case where all releases have all seven quasi-identifiers, but they differ from each other in the number of tuples: The first release includes $N = 100000$ tuples, the second one has additional $50000$ tuples, the third one has additional $30000$ tuples, and the fourth one has additional $20000$ tuples (so it holds $200000$ tuples altogether). Here too we compared an adaptive sequential algorithm (that learns of tuples only when they are introduced) to a prophetic algorithm that knows all $200000$ tuples upfront. The average information losses in this setting are shown in Figure 16.

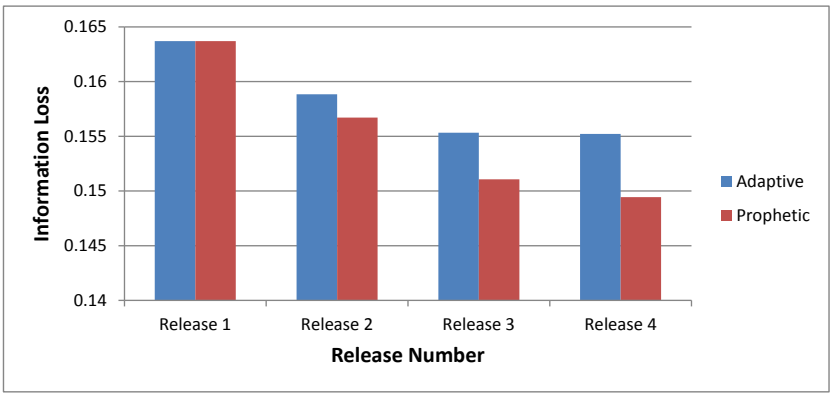


Figure 16: A horizontally dynamic setting

Figure 17 compares the adaptive and prophetic algorithms in the combination of the above described vertical and horizontal settings. Namely, the set of attributes changes from one release to another, as in the experiment reported in Figure 15, and the number of tuples changes, as in the experiment reported in Figure 16. In addition, the adaptive algorithm knows upfront only the tuples and attributes that are included in the first release, and it learns of each tuple or attribute only when it appears first; in contrast, the prophetic algorithm has all information upfront.

It is interesting to see that the adaptive algorithm maintains its advantage over the prophetic algorithm in releases 1 and 3. In release 3, the prophetic algorithm has only "horizontal advantage" over the adaptive one (namely, it knows upfront the new tuples that appear for the first time in that release), but no "vertical advantage", since releases 1 and 3 consist of attributes that were all known from the start. However, the adaptive algorithm has the advantage of constructing possible worlds that are better suitable for minimizing information loss over the attributes in those two releases. The advantage of focusing on the relevant attributes has in this case a greater impact than the advantage of knowing all tuples upfront. As for releases 2 and 4, the prophetic algorithm has advantage over the adaptive one regarding both vertical and horizontal information.

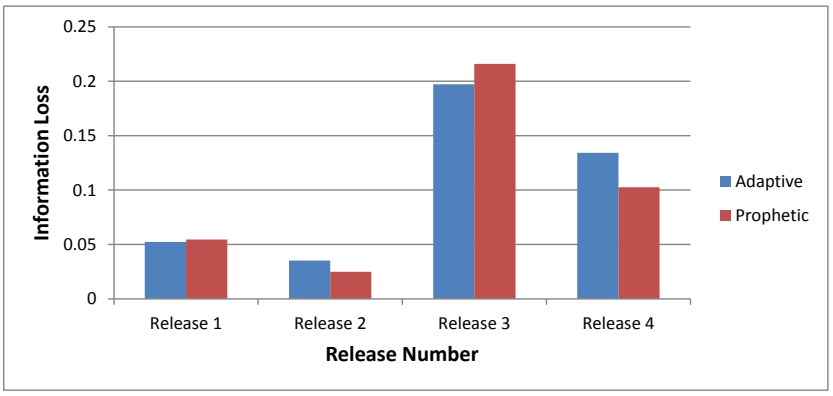


Figure 17: A fully dynamic setting

## 7. Conclusions and future work

In this study we revisited the problem of anonymizing sequential releases. We presented a general framework for dealing with such scenarios. We defined an enhanced notion of privacy that takes into account *all* information available to the adversary. We considered a strong adversary who knows all quasi-identifiers of all tuples in the underlying table. Our algorithm achieves $\ell$-diversity — a notion of privacy that bounds the probabilities of linking sensitive values to quasi-identifier tuples. Our algorithm, as opposed to previous ones, is highly scalable with respect to the number of releases, as the runtime for anonymizing any given release is independent of the number of previous releases. Finally, we showed how to extend our methods to the fully dynamic setting, in which the set of attributes may change and tuples may be added from time to time. Our experiments showed the importance of exploiting the vertical dynamics (i.e., the fact that different releases may include different attributes) for significantly reducing the information loss.

Our study is the first one that offers an algorithm that achieves provable $\ell$-diversity in the general scenario of sequential release. While $\ell$-diversity significantly enhances the privacy offered by simpler notions like $k$-anonymity or $\ell$-linkability, it may be vulnerable to skewness or correlation attacks. The notion of $t$-closeness [19] is a stronger privacy model which is more immune to such attacks. An interesting research direction is to use the ideas presented here to achieve stronger privacy notions, such as $t$-closeness. That may be achieved if the set of possible worlds that the algorithm generates in the preprocessing stage links every quasi-identifier tuple with a multiset of sensitive values that have a distribution which is close to the general distribution of sensitive values. In doing

so, a careful attention must be given to the resulting utility. Imposing such a rigid condition on the possible worlds, towards enhancing privacy, may increase the information loss to levels which could render the released tables useless.

In the future, we intend to extend our methods to support more intricate scenarios of continuous data publishing, such as scenarios in which tuples can be removed or updated.

## 8. Appendix

### *8.1. Summary of notations*

Table 17: Notations

| Notation | Meaning |
|---|---|
| $[N]$ | The set $\{1, \ldots, N\}$ |
| $A_1, \ldots, A_M$ | The $M$ attributes of the table, as well as the attribute domains |
| $A_1, \ldots, A_Q$ | The quasi-identifier attributes |
| $A_{Q+1}, \ldots, A_{M-1}$ | The non-identifier attributes |
| $A_M$ | The sensitive attribute |
| $\overline{A}_m$ | The collection of subsets of $A_m$ that could be used for generalization, $1 \leq m \leq M$ |
| $T$ | The underlying table |
| $t_n$ | The $n$th tuple in $T$, $1 \leq n \leq N$ |
| $t_n(m)$ | The $m$th entry in $t_n$, $1 \leq m \leq M$, $1 \leq n \leq N$ |
| $id_n$ | The identifier of the $n$th tuple in $T$, $1 \leq n \leq N$ |
| $V$ | A single anonymization of the entire table $T$ |
| $v_n$ | The $n$th tuple in $V$, $1 \leq n \leq N$ |
| $v_n(m)$ | The $m$th entry in $v_n$, $1 \leq m \leq M$, $1 \leq n \leq N$ |
| $\mathcal{V} := \langle V_0, V_1, \ldots, V_R \rangle$ | A sequential release of $T$ |
| $V_0$ | The adversarial background knowledge |
| $V_r$ | The $r$th release of $T$, $0 \leq r \leq R$ |
| $I_r$ | The set of indices of all attributes that are included in $V_r$, $0 \leq r \leq R$ |
| $v_n^r$ | The $n$th tuple in $V_r$, $0 \leq r \leq R$, $1 \leq n \leq N$ |
| $v_n^r(m)$ | The $m$th entry in $v_n^r$, $0 \leq r \leq R$, $1 \leq m \leq M$, $1 \leq n \leq N$ |
| $T\|_{I_r}$ | The table $T$ where all attributes with indices in $[M] \setminus I_r$ are suppressed |
| $W$ | A possible world |
| $E$ | A set of possible worlds |
| $\mathsf{PW}(\mathcal{V})$ | The set of all possible worlds |
| $G_{\mathcal{V}}$ | The multipartite consistency graph of $\mathcal{V}$ |
| $C$ | A full clique in $G_{\mathcal{V}}$ |
| $t^B$ | The closure of the set of tuples $B$ |
| $div(B)$ | The diversity of a set of tuples $B$ |

*8.2. Ordering $\ell$-diverse sets*

In Algorithm 1 we partition $T$'s tuples to buckets that are all $\ell$-diverse (Definition 2.3). Let $B$ be one of those buckets and $\{a_1, \ldots, a_S\}$ be the set of sensitive values in $B$'s tuples. We denote by $n_s$ the number of $B$'s tuples that have the sensitive value $a_s$. Assume further that $n_1 \geq \cdots \geq n_S$ and let $n = |B|$. The $\ell$-diversity of $B$ implies that $\text{div}(B) = n/n_1 \geq \ell$.

Our goal is to order $B$'s tuples in a cycle so that every $\ell$ consecutive tuples in that cycle have distinct sensitive values. Assume that $n = q\ell + r$ where $0 \leq r \leq \ell - 1$. Let $A$ be an array of $q$ rows and $p = \lceil n/q \rceil$ columns, whose rows and columns are indexed starting from zero.

Assume that

$$(z_0, \ldots, z_{n-1}) = \left( \overbrace{a_1, \ldots, a_1}^{n_1}, \cdots, \overbrace{a_S, \ldots, a_S}^{n_S} \right) \tag{13}$$

is the sequence of all sensitive values in $B$, sorted from the most frequent to the least frequent one. We assign those values to $A$'s entries by columns, namely $A(i, j) = z_{i+rj}$ for $0 \leq i \leq q - 1$ and $0 \leq j \leq p - 1$ (when $j = p - 1$ the last $pq - n$ entries remain empty). The following lemma states that the order that is induced by the rows of $A$ satisfies the sought-after property that is described in Step 4 of the algorithm.

**Lemma 8.1.** *The sequence*

$$A(0, 0), \ldots, A(0, p - 1), \ldots, A(q - 1, 0), \cdots, A(q - 1, p - 1)\,, \tag{14}$$

*where empty entries are skipped, has the property that every cyclically consecutive $\ell$ values in it are distinct.*

**Proof.** It is easy to see that if $r = 0$ then $p = \ell$ and all entries in $A$ are filled; otherwise, if $r > 0$, then $p = \ell + 1$ and the last $pq - n$ rows of $A$ have only their first $\ell$ entries filled. Therefore, when we look at the ordering of $A$'s entries, (14), every $\ell$ consecutive elements in it are elements from $\ell$ distinct columns of $A$. It is impossible that such a subsequence will have the same value twice. Assume, towards contradiction, that there exists a subsequence with the same value repeated twice. Then, since the number of columns is at least $\ell$, there exist $1 \leq j_1 < j_2 \leq p$ such that $A(i, j_1) = A(i, j_2)$ or $A(i + 1, j_1) = A(i, j_2)$. But that cannot be since we allocated the values to $A$ by columns, starting from the most frequent one, according to (13), and each of them occurs at most $q$ times (which is the number of rows) since $n_s \leq n_1 \leq \lfloor n/\ell \rfloor = q$ for all $1 \leq s \leq S$. □

*8.3. On the applicability of differential privacy to the problem of sequential release*

Recently, there has been a growing debate over approaches for handling and analyzing private data. Research has identified issues with syntactic approaches such as $k$-anonymity and $\ell$-diversity. Differential privacy, which is based on adding noise to the analysis outcome, has been promoted as *the* answer to privacy-preserving data mining. The recent study [4] looks at the criticisms of both approaches and identifies the main problems with each of them. The conclusion in that paper is that both approaches have their place, and that each approach has issues that call for further research. Here, we focus on the main issues that render differential privacy less applicable to the problem of sequential release. The interested reader is referred to [4] for other issues and a more thorough discussion. Another recent study that reaches similar conclusions by comparative evaluation of the two approaches is [6].

Models based on the release of anonymized data can safely be used for as many distinct uses as desired. Methods that release only query results require tracking the results: early uses of the data can affect the quality of later uses, or even result in a threshold beyond which no new queries can be permitted on the data. While noise can be applied to a data release (e.g., the techniques used in Public Use Microdata Sets [22]), most recent research has concentrated on answering queries against the data. It has long been known that care must be taken to ensure that multiple queries do not violate privacy [7]. Differential privacy does address this, as differentially private answers to queries are composable, with each consuming a portion of the "privacy budget". For example, in order to achieve $\varepsilon$-differential privacy over two queries, the answer to each query can be made noisier so that each complies with $\varepsilon/2$-differential privacy. However, if the query stream is not known in advance, adding too little noise to early queries can prevent reasonable answers to later queries.

There is also an issue of determining how to set a "privacy budget". Assuming public access to a dataset, any privacy budget could quickly be exhausted. An alternative is to assign individual privacy budgets, but this requires ensuring that individuals do not collude, limiting dataset access to individuals who can be authenticated and vetted. This poses interesting policy challenges for use of differential privacy.

As explained in detail herein, our approach does not suffer from the problem of having a limited privacy budget that is consumed by each release. In our approach, it is possible to publish as many releases of the underlying table, without violating the $\ell$-diversity privacy requirement, and without degrading the utility of the later

releases, since the anonymization of each release is independent on the past or future releases.

As a concluding remark we note that Mohammed et al. [21] proposed an algorithm for producing a single release of a given dataset that complies with the differential privacy condition. Their algorithm is a variation of the top-down specialization algorithm of [32], where compliance with differential privacy is achieved by modifying two of its stages: (a) in the course of the top-down specialization loop, instead of choosing the optimal taxonomy node for specialization, the choice is made in a probabilistic manner in accord with the so-called exponential mechanism; and (b) at the end of the specialization loop, instead of publishing for each block of records that have the same generalized quasi-identifiers the exact counts of sensitive values, those counts are being perturbed by Laplacian noise. Clearly, such modifications reduce further the utility of the output, which is limited in the first place due to the usage of the rigid cut generalization model, rather than the more flexible local recoding model (as discussed in detail in [28]). That algorithm may be used in the sequential release scenario by applying to it the non-sequential approach (as described in Section 4.5). However, such an approach yields poor utility of the sequential release (which adds to the poor utility of the basic anonymization algorithm of the entire table), as discussed in Section 4 and exemplified by our experimental evaluation. In addition, such an approach is not suitable to handling dynamics in the table.

## References


[1] BU, Y., FU, A. W.-C., WONG, R. C.-W., CHEN, L., AND LI, J. 2008. Privacy preserving serial data publishing by role composition. *PVLDB 1,* 1, 845–856.

[2] BURNETT, L., BARLOW-STEWART, K., PROOS, A., AND AIZENBERG, H. 2003. The GeneTrustee: a universal identification system that ensures privacy and confidentiality for human genetic databases. *Journal of Law and Medicine 10,* 4, 506–513.

[3] BYUN, J.-W., SOHN, Y., BERTINO, E., AND LI, N. 2006. Secure anonymization for incremental datasets. In *Secure Data Management (SDM)*. 48–63.

[4] CLIFTON, C. AND TASSA, T. 2013. On syntactic anonymity and differential privacy. *Transactions on Data Privacy 6*, 161–183.

[5] CORMODE, G., LI, N., LI, T., AND SRIVASTAVA, D. 2010. Minimizing minimality and maximizing utility: Analyzing method-based attacks on anonymized data. *PVLDB 3*, 1045–1056.

[6] CORMODE, G., PROCOPIUC, C. M., SHEN, E., SRIVASTAVA, D., AND YU, T. 2013. Empirical privacy and empirical utility of anonymized data. In *ICDE Workshops*. 77–82.

[7] DOBKIN, D., JONES, A. K., AND LIPTON, R. J. 1979. Secure databases: Protection against user influence. *ACM Trans. Database Syst. 4,* 1, 97–106.

[8] FRANK, A. AND ASUNCION, A. 2010. UCI machine learning repository [http://archive.ics.uci.edu/ml]. *University of California, Irvine, School of Information and Computer Sciences*.

[9] FUNG, B., WANG, K., CHEN, R., AND YU, P. 2010. Privacy-preserving data publishing: a survey of recent developments. *ACM Computing Surveys (CSUR) 42,* 4, 1–53.

[10] FUNG, B., WANG, K., FU, A., AND PEI, J. 2008. Anonymity for continuous data publishing. In *Proceedings of the international conference on Extending Database Technology: advances in database technology (EDBT)*. 264–275.

[11] FUNG, B., WANG, K., AND YU, P. 2005. Top-down specialization for information and privacy preservation. In *Proceedings of the International Conference on Data Engineering (ICDE)*. 205–216.

[12] GIONIS, A., MAZZA, A., AND TASSA, T. 2008. $k$-Anonymization revisited. In *Proceedings of the International Conference on Data Engineering (ICDE)*. 744–753.

[13] GIONIS, A. AND TASSA, T. 2009. $k$-Anonymization with minimal loss of information. *IEEE Transactions on Knowledge and Data Engineering 21*, 206–219.

[14] GOLDBERGER, J. AND TASSA, T. 2010. Efficient anonymizations with enhanced utility. *Transactions on Data Privacy 3*, 149–175.

[15] IYENGAR, V. 2002. Transforming data to satisfy privacy constraints. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 279–288.

[16] KUHN, H. W. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly 2*, 83–97.

[17] LAST, M., TASSA, T., AND ZHMUDYAK, A. Forthcoming. Improving accuracy of classification models induced from anonymized datasets. *Information Sciences*.

[18] LEFEVRE, K., DEWITT, D. J., AND RAMAKRISHNAN, R. 2006. Mondrian multidimensional $k$-anonymity. In *International Conference on Data Engineering (ICDE)*. 25.

[19] LI, N., LI, T., AND VENKATASUBRAMANIAN, S. 2010. Closeness: A new privacy measure for data publishing. *IEEE Transactions on Knowledge and Data Engineering 22,* 7, 943–956.

[20] MACHANAVAJJHALA, A., GEHRKE, J., KIFER, D., AND VENKITASUBRAMANIAM, M. 2006. $l$-Diversity: privacy beyond $k$-anonymity. In *Proceedings of the International Conference on Data Engineering (ICDE)*. 24.

[21] MOHAMMED, N., CHEN, R., FUNG, B. C. M., AND YU, P. S. 2011. Differentially private data release for data mining. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 493–501.

[22] MOORE, JR., R. A. 1996. Controlled data-swapping techniques for masking public use microdata sets. Statistical Research Division Report Series RR 96-04, U.S. Bureau of the Census, Washington, DC.

[23] MUNKERS, J. 1957. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics 5*, 32–38.

[24] NERGIZ, M. AND CLIFTON, C. 2007. Thoughts on $k$-anonymization. *Data and Knowledge Engineering 63,* 3, 622–645.

[25] PEI, J., XU, J., WANG, Z., 0009, W. W., AND WANG, K. 2007. Maintaining k-anonymity against incremental updates. In *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*. 5.

[26] RUGGLES, S., ALEXANDER, T., GENADEK, K., GOEKEN, R., SCHROEDER, M., AND SOBEK, M. 2010. Integrated public use microdata

series: Version 5.0 [machine-readable database]. *Minneapolis: University of Minnesota*.

[27] SAMARATI, P. 2001. Protecting respondent's privacy in microdata release. *IEEE Transactions on Knowledge and Data Engineering 13*, 1010–1027.

[28] SHMUELI, E., TASSA, T., WASSERSTEIN, R., SHAPIRA, B., AND ROKACH, L. 2012. Limiting disclosure of sensitive data in sequential releases of databases. *Information Sciences 191*, 98–127.

[29] SWEENEY, L. 2002. $k$-Anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10,* 5, 557–570.

[30] TASSA, T. 2012. Finding all maximally-matchable edges in a bipartite graph. *Theoretical Computer Science 423*, 50–58.

[31] TASSA, T., MAZZA, A., AND GIONIS, A. 2012. $k$-Concealment: an alternative model of $k$-type anonymity. *Transactions on Data Privacy 5*, 189–222.

[32] WANG, K. AND FUNG, B. 2006. Anonymizing sequential release. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 414–423.

[33] WONG, R., FU, A., WANG, K., AND PEI, J. 2007. Minimality attack in privacy preserving data publishing. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB)*. 543–554.

[34] WONG, R., LI, J., FU, A., AND WANG, K. 2006. $(\alpha, k)$-anonymity: An enhanced $k$-anonymity model for privacy preserving data publishing. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 754–759.

[35] WONG, R. C.-W., FU, A. W.-C., LIU, J., WANG, K., AND XU, Y. 2010a. Global privacy guarantee in serial data publishing. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 956–959.

[36] WONG, W. K., MAMOULIS, N., AND CHEUNG, D. W.-L. 2010b. Non-homogeneous generalization in privacy preserving data publishing. In *Proceedings of the international conference on Management of Data (SIGMOD)*. 747–758.

[37] XIAO, X. AND TAO, Y. 2007. M-invariance: towards privacy preserving re-publication of dynamic datasets. In *Proceedings of the ACM SIGMOD International Conference on Management of data (SIGMOD)*. 689–700.