Journal of Biomedical Informatics 50 (2014) 95–106

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

journal homepage: www.elsevier.com/locate/yjbin

A framework to preserve the privacy of electronic health data streams

Soohyung Kim [a], Min Kyoung Sung [b], Yon Dohn Chung [b,*]

[a] Department of IT Convergence, Korea University, Anam-dong, Seongbuk-gu, Seoul, Republic of Korea
[b] Department of Computer Science and Engineering, Korea University, Anam-dong, Seongbuk-gu, Seoul, Republic of Korea

ARTICLE INFO

Article history:
Received 26 August 2013
Accepted 27 March 2014
Available online 4 April 2014

Keywords:
Health data stream
Privacy
Anonymization

ABSTRACT

The anonymization of health data streams is important to protect these data against potential privacy breaches. A large number of research studies aiming at offering privacy in the context of data streams has been recently conducted. However, the techniques that have been proposed in these studies generate a significant delay during the anonymization process, since they concentrate on applying existing privacy models (e.g., *k*-anonymity and *l*-diversity) to batches of data extracted from data streams in a period of time. In this paper, we present *delay-free anonymization*, a framework for preserving the privacy of electronic health data streams. Unlike existing works, our method does not generate an accumulation delay, since input streams are anonymized immediately with counterfeit values. We further devise *late validation* for increasing the data utility of the anonymization results and managing the counterfeit values. Through experiments, we show the efficiency and effectiveness of the proposed method for the real-time release of data streams.

1. Introduction

Recently, data streams of health records have been widely utilized in many applications, such as real-time diagnosis systems, biomedical data analysis, and health monitoring services [1–4]. Diagnosis systems, for example, diagnose patients' diseases by monitoring the behavior of their real-time health data, and biomedical data analysts study the patterns of diseases by using the data streams from bio-signal sensors. In many cases, the data streams are entrusted to third parties who analyze the data in place of the data holders, because of the effective data utilization.

Meanwhile, health data streams typically contain private attributes, such as age, gender, various bio-signals, and diagnoses. Privacy problems can arise in relation to these types of data during the transmission of the data streams to third parties, which can provide paths for a privacy attack. For example, consider a data stream with the schema *(tupleID, time, age, sex, diagnosis)* released from a hospital for the purpose of a biomedical study. The schema does not contain any direct identifiers, such as SSNs (social security numbers), that can distinguish individuals. However, using some background knowledge, i.e., time, age, and sex, an attacker can identify the diagnosis of the target individual [5,6].

To protect data streams from background knowledge attacks, several privacy-preserving methods have been recently proposed [7–14]. These methods ensure that the released data streams meet the requirements of typical privacy models, such as *k*-anonymity [5] and *l*-diversity [6]. Although these methods differ in the way they transform the data streams to offer privacy protection, they are all based on the *tuple accumulation* strategy. That is, streaming tuples are postponed until they satisfy the given publication condition, i.e., reach a certain privacy level or satisfactory data utility. We call this type of method the *accumulation-based method*.

Fig. 1 shows an example of offering privacy protection in data streams by using the accumulation-based method. Assume an input stream of tuples, where a tuple consists of multiple attributes that include personal attributes, i.e., QI (quasi-identifiers) and SI (sensitive information). In the accumulation-based method, the anonymization system continues to accumulate and cluster the input tuples until the given privacy condition, such as *k*-anonymity and *l*-diversity, is satisfied. In Fig. 1, tuple 1, which has arrived at time T_1 and has been accumulated in Cluster 1, waits for tuple 2. When tuple 2 arrives, it is assigned to Cluster 1, then the cluster becomes 2-diverse. When a certain cluster satisfies the privacy condition, the tuples in the cluster are generalized and released. Then, the QIs of the tuples in the cluster become indistinguishable from each other according to the privacy condition. Consequently, an adversary cannot identify an exact tuple from the anonymized data by using the QIs (i.e., background knowledge). Thus, an

* Corresponding author. Address: Department of Computer Science and Engineering, College of Information and Communications, Korea University, Anam-dong, Seongbuk-gu, Seoul 136-713, Republic of Korea.
E-mail addresses: soohyung@korea.ac.kr (S. Kim), mj999@korea.ac.kr (M.K. Sung), ydchung@korea.ac.kr (Y.D. Chung).

http://dx.doi.org/10.1016/j.jbi.2014.03.015
1532-0464/

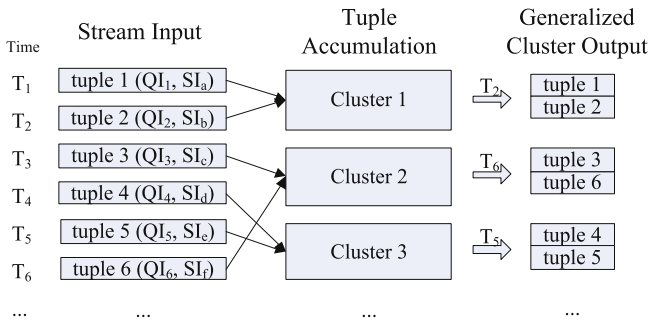

Fig. 1. Privacy protection of a data stream using the accumulation-based method (an example of offering 2-diversity).

adversary cannot be certain about the target's SI, although he or she can infer the list of possible SIs.

In addition to employing accumulation to satisfy the required privacy level, most existing methods aim at accumulating more tuples than necessary, in order to reduce the information loss that is caused by the applied data generalization. For example, tuple 4 in Fig. 1 is accumulated in Cluster 3 instead of Cluster 2, because if it was accumulated in Cluster 2 this would lead to increased information loss.

There are two inevitable delay problems in the accumulation-based methods: *accumulation* and *computation delay*. The former is directly relevant to the accumulation of the input tuples. For example, in Fig. 1, the release of tuple 3 is delayed until tuple 6 has arrived. The delay is due to the time taken to calculate the information loss. To assign an input tuple to an appropriate cluster that generates the minimum information loss, the information loss of every cluster must be calculated. Computing the information loss of all clusters takes a significant amount of time, which prohibits the release of large data streams in real time.

To solve these problems, in this paper, we propose a DF (delay-free anonymization) framework to preserve the privacy of electronic health data streams in real time. We consider two types of attributes of an input tuple: the *quasi-identifier* (QI) and the *sensitive information* (SI). In the DF framework, QIs are released with artificially generated *l*-diverse SI values. For example, in Fig. 2, tuple 1 is anonymized with ST_1, which is the *l*-diverse set of the SI. We organize the set with the original SI from the input tuple and the artificially generated SI. Then, QI_1 has multiple SIs (SI_a and SI_d), which satisfies 2-diversity. Thus, attackers cannot identify

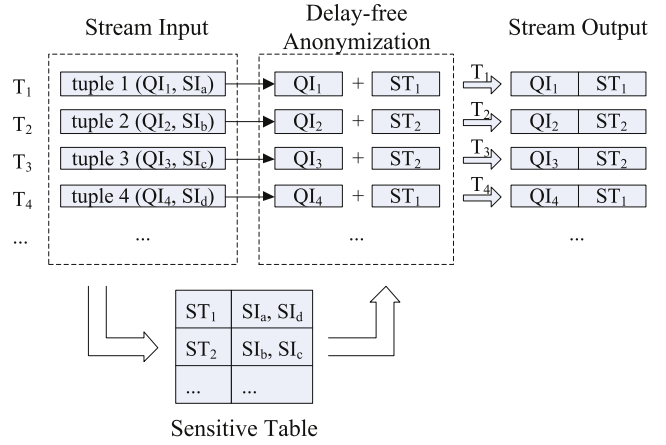

Fig. 2. Privacy protection of a data stream using the delay-free anonymization approach (an example of offering 2-diversity).

the exact SI of the target QI. In this manner, our anonymization process preserves the privacy of the health data streams.

We would like to note that there is no accumulation delay in DF because it immediately anonymizes the input tuples and releases them. This is possible because there is no tuple accumulation in our anonymization process. Furthermore, in comparison with the accumulation-based methods, DF generates remarkably little computation delay, because DF executes only simple operations (i.e., referring to the sensitive table and managing it) instead of complicated operations (e.g., calculating the information loss).

We also considered the data utility of the anonymization results. Our anonymization method, generating *l*-diverse SI values, generates counterfeit values. For example, in Fig. 2, SI_d in ST_1 is a counterfeit at T_1. Counterfeit values function as a protection mechanism to prevent the disclosure of the correct SI values; however, they also decrease data utility. To ameliorate this issue, we propose *late validation*, which utilizes the SI values of the newly entered tuples to validate the existing counterfeits. In Fig. 2, SI_d in ST_1 is *late validated* at T_4. After tuple 4, whose SI is SI_d, is anonymized using ST_1, SI_d in ST_1 is considered a real value. To measure the data utility, including the utility loss from the counterfeits, we further devise *sensitive attribute uncertainty* as the quality metric for DF.

The contributions of our delay-free anonymization framework are summarized as follows:

- **Immediate release:** DF facilitates tuple-by-tuple release with the guarantee of *l*-diversity. DF is characterized by no accumulation delay and a low computation cost, since it immediately releases the data streams and requires only simple operations. The immediate release is a big advantage for applications that utilize real-time data streams.
- **High-level data utility:** To anonymize data streams, DF artificially generates *l*-diverse SI values instead of generalizing QIs. Thus, DF does not generate the typical information loss with respect to QIs. In addition, the counterfeits in the sensitive table are also minimized by *late validation*.

The rest of this paper is organized as follows. In Section 2, we present the related work in this area. In Section 3, we define the data model and propose DF. In Section 4, we present a late validation scheme for effective anonymization, and the development of the DF algorithm. In Section 5, we discuss data utility issues in the context of our proposed framework. Section 6 presents our experimental evaluation and Section 7 concludes this work.

2. Related work

A large number of research studies in the area of data privacy has been conducted in the last decade. In what follows, we first present methods that have been proposed to offer privacy in the context of static relational data. Following that, we discuss approaches that have been proposed for offering dynamic data privacy and data stream privacy. The latter are directly relevant to our approach.

2.1. Privacy for static, relational data

A significant amount of work has been conducted in the area of privacy for relational data. Several studies (e.g., [5,15,16]) proposed methods that employ *k*-anonymity to block re-identification attacks, in which individuals in the released microdata are identified based on QIs. To solve this problem, *k*-anonymity makes *k* tuples indistinguishable by generalizing the QIs. *l*-diversity is a model that solves the lack of diversity in *k*-anonymity [6]. It makes SIs *l*-diverse with indistinguishable QIs. Anatomy [17] is a technique

that also uses the *l*-diverse tuples similarly to the method proposed in [6]; however, it does not generalize QIs. Thus, although an attacker knows the exact target QI is included in the dataset, she or he cannot distinguish which SI belongs to the target.

The authors of *t*-closeness [18] proposed a method that addresses certain limitations of *l*-diversity. Two possible attacks can be made against *l*-diversity: a *skewness attack* and *similarity attack*. In order to prevent these attacks, *t*-closeness requires that the distribution of the sensitive attribute values meets a certain condition.

The above-mentioned studies deal only with relational data and do not consider any insertion, deletion or update operations applied on these data. As a result, these methods are not applicable in the context of data streams.

2.2. Dynamic data privacy

Insertions, deletions, and updates frequently occur in real datasets. Several methods have been proposed to protect dynamically evolving datasets, ensuring that the latest version of the data can be safely published. It is important to note that previous privacy-protection methods, involving static data, are not applicable in the context of dynamic datasets because they assume only a single data release. In dynamic datasets, a new type of privacy problem can arise because of the potential linkage between different data releases. Therefore, several privacy preservation techniques [19–21] protect the privacy of dynamic releases by considering their context or maintaining extra information.

Byun et al. presented an anonymization method for dynamic data; however, this work considered only insertion operations [19]. Xiao et al. in [20] proposed *m*-invariance, a privacy method which considers both insertion and deletion operations. Bu et al. addressed another privacy problem that occurs by the permanent sensitive values in the dynamic datasets [21].

The methods proposed in these studies deal with multiple releases, which are analogous to data streams. However, they cannot be employed to anonymize data streams since they deal with the anonymization of the entire table and cannot be applied on data that becomes available in real time.

2.3. Data stream privacy

Recently, there have been several studies on preserving the privacy of data streams. Data streams consist of sequences of tuples transmitted in real time, and hence, the assumption concerning the requirements of the privacy preservation technique is different from that in the previous privacy protection methods. To preserve the privacy of data streams, the unit of anonymization must be a tuple, and the anonymization should be performed in real time.

To the best of our knowledge, the first study that dealt with the privacy preservation of data streams is [9], in which the specialization tree for accumulating data streams was introduced. Nodes in the tree are classified into candidate nodes and work nodes, and data streams are generalized by the node structure.

The CASTLE scheme generates clusters in which the tuples of data streams are accumulated [7]. Then, it releases the clusters when the size of the accumulation reaches the delay-constraint, δ. Basically, CASTLE guarantees *k*-anonymity, and also provides an algorithm for *l*-diversity.

The method presented in [8] uses a probability function to determine the release of data streams; it also accumulates data from data streams and decides on the release time according to this function. The function promotes releases when the accumulated data experience less information loss but a longer delay. For the sake of data utility, the method proposed in this study also makes a prediction on the data that may appear in the streams by considering the data distribution.

The authors of SABRE [14] presented an algorithm for anonymizing microdata to satisfy *t*-closeness, and also extended the algorithm to anonymize data streams. The SABRE framework maintains sliding windows that function as buffers of the input tuples. When tuples in a certain window are expired due to newly inserted tuples, old tuples should be released. These tuples are anonymized by SABRE and released to an output stream.

The authors of B-CASTLE [10] presented an improved algorithm of CASTLE [7]. The algorithm of B-CASTLE considers the distribution of data in data streams when the input data is clustered, in an effort to improve data utility. The method presented in [12] also considers the density distribution to improve the utility of the data in the anonymized stream. The method constructs a TDS-tree (top-down specialization tree) of QIs, and considers the density distribution of tuples in each leaf of the tree. The authors of SANATOMY [11] employed the bucketization approach (i.e. Anatomy [17]) to anonymize data streams, instead of the generalization approach. SANATOMY preserves the data correlation, due to the bucketization approach. The authors of [13] pointed out that the previous studies had been based on time-consuming algorithms due to the *k*-anonymity model. The algorithm presented in [13] is an efficient anonymization algorithm, which is cluster-based and supports numerical data streams.

All the studies on data stream privacy preservation thus far have been based on the tuple accumulation strategy. Consequently, the existing methods suffer from several limitations in terms of the real-time release of data and their applicability in a distributed setting, as also mentioned in Section 1.

3. The delay-free anonymization framework

In this section, we present the delay-free anonymization (DF) framework for protecting the privacy of the data in health data streams. First, we define the data model of data streams. Then, we describe our proposed method in detail.

3.1. Data model

We assume real-time health data streams containing personal information. Among the various attributes in a data stream, we focus on only three main attributes in order to simplify the problem: tupleID, QI, and SI. According to the attributes, a data stream can be described as

$$(\text{tupleID},\ \text{QIs},\ \text{SI})$$

TupleID indicates the unique number of a tuple. It is usually removed before releasing the data stream, because it can be an identifier of the tuple. The *QI* is an attribute of an individual in the tuple, such as age, sex, nationality, and zipcode. The SI is the private attribute, which should not be directly related to the individual, such as bio-signals and diagnosis. In the following definition of the data stream, we omit the tupleID, since it is consequentially removed in the anonymization process.

Definition 1 (*Data stream*). Let *QI* be the quasi-identifier attributes of a tuple, where $QI = \{qi_1, qi_2, \ldots, qi_i\}$, and let *si* be the sensitive information of a tuple. We define a data stream *S* as a set of tuples (QI, si).

3.2. Delay-free anonymization

The goal of the delay-free anonymization scheme is to organize *l*-diverse data streams in real time, and guarantee that the probability of guessing the individual's sensitive information should be equal to or less than $1/l$. In order to anonymize data

streams, we adopt the ideas behind the Anatomy technique [17], which preserves privacy by dividing the QI and SI. Then, we inflate each *si* of the input tuples into *l*-diverse values. Assume that a tuple $t(QI_t, si_t)$ has arrived. DF generates a *QIT* (quasi-identifier tuple) and an *ST* (inflated sensitive tuple) and releases them immediately. The *QIT* is generated from QI_t in the tuple *t* and the *ST* is an *l*-diverse set of the artificial *si* that contains si_t. The *QIT* and *ST* can be joined by the join key *groupID*.

Definition 2 (*Delay-free anonymization*). Given a tuple $t(QI_t, si_t)$ from a data stream and a domain of the sensitive information D_{SI}, DF generates the *QIT* and *l*-diverse *ST*. The schema of the *QIT* is

$$(groupID, QI),$$

where *groupID* is a positive integer and *QI* is from QI_t. For a groupID j, QIT_j has the schema (j, QI).

The schema of the *ST* is

$$(groupID, si, count),$$

where *groupID* and *count* are positive constants, and *si* is an element of D_{SI}. For a groupID j, ST_j has the schema $(j, si, count(si))$ where $count(si)$ denotes the number of *si*.

Example 1. [Delay-free anonymization] Let us assume that a data stream $((age, sex), diagnosis)$ is generated by the hospital and the anonymization result of the data stream must satisfy 2-diversity. Given that a tuple $t_1((24, male), Diag.A)$ has arrived, DF generates the *QIT* $(1, (24, male))$ where 1 is a randomly generated *groupID*. The *Diag.A*, an SI of t_1, is inflated into an l-diverse *ST* $(1, Diag.A, 1)$ and $(1, Diag.B, 1)$ with *groupID* 1 and *count* 1.

The QI of t_1 has 2-diverse SIs as a result of applying DF. Thus, an adversary cannot find the exact sensitive value of $(24, male)$ with a probability greater than 1/2. Therefore, the result of the anonymization satisfies 2-diversity.

It is important to mention that *Diag.B*, the inflated SI, is not a real value. It is counterfeit and invalid, because it is arbitrarily generated from the domain of the sensitive information to preserve privacy. The counterfeits can decrease the utility of anonymized results; thus, in Section 4, we present an effective mechanism for the generation and management of counterfeits, which aims to maintain high data utility.

3.3. Privacy preservation

In order to protect privacy of data streams, we aim at guaranteeing *l*-diversity on DF, which allows adversaries to guess the individual's sensitive information with a probability equal to or less than $1/l$. Since the aim of *l*-diversity on DF is slightly different from the traditional *l*-diversity, in this section, we discuss the privacy preservation achieved by DF and define *l*-diversity in the context of DF.

First, in order to guarantee the $1/l$ probability, it is essential that the number of distinct *si* values in an arbitrary group are equal to or more than *l*. Consider an arbitrary group *j* and its sensitive tuples ST_j. The first condition for *l*-diversity on ST_j is

$$l \leqslant |ST_j| \tag{1}$$

It should be noted that $|ST_j|$, which denotes the size of ST_j, also represents the number of distinct values in ST_j, because there is no duplicated *si* in ST_j (we use the *count* value to describe repeated sensitive values).

Second, the frequency of a sensitive value should be considered. Let us focus on the count attribute in the *ST*. In Example 1, the count value of each *si* is 1. Thus, the ratio of the *Diag.A* and the *Diag.B* is 1:1 for the *QI* of t_1. It is 2-diverse, because the attacker who has background knowledge (*QI* of t_1) cannot identify an exact sensitive value with more than 1/2 probability. Meanwhile, let us assume the count value of *Diag.A* = 2 and *Diag.B* = 1. Then, the ratio becomes 2:1 and the probability that the *QI* of t_1 corresponds to *Diag.A* becomes 2/3, which is greater than 1/2 probability. We can derive a conclusion that the count values in the *ST* should be generated such that they guarantee the $1/l$ probability.

Consider an arbitrary group *j*, and its sensitive tuples ST_j. The probability that the arbitrary si_i is a sensitive value of a given *qi* is $P(SI_{qi} = si_i) = count(si_i)/||ST_j||$, where $||ST_j||$ is the sum of all counts in group *j*. It should be noted that the probability denotes the proportion of the arbitrary sensitive value si_i among all sensitive values in ST_j. In order to protect privacy, the probability of every *si* in ST_j must be equal to or less than $1/l$. Now, we can determine the count value.

$$P(SI_{qi} = si_i) \leq 1/l$$

$$count(si_i)/||ST_j|| \leq 1/l$$

$$count(si_i) \leqslant ||ST_j||/l \tag{2}$$

Finally, we define *l*-diversity on DF that formalizes the two conditions above.

Definition 3 (*l-diversity on DF*). Let us assume that an anonymized result by the DF framework consists of *QIT* $(groupID, QI)$, and *ST* $(groupID, si, count)$. Consider that ST_j $(j, si, count(si))$ is an arbitrary subset of the *ST*, whose *groupID* is *j*. The anonymized result satisfies *l*-diversity on DF if and only if $l \leqslant |ST_j|$ and $\forall\ st_j[count(si)] \leqslant ||ST_j||/l$, where $st_j \in ST_j$. $|ST_j|$ is the number of distinct *si* values in group *j*, and $||ST_j||$ is the sum of all counts in group *j*.

4. Effective counterfeit management

At least *l*-1 counterfeit values are generated when anonymizing a tuple by *l*-diversity. In this section, we propose the *late validation* method, which reduces the generation of counterfeits by utilizing newly input tuples. Then, we show the development of an algorithm for DF with *late validation*.

4.1. Sensitive group management

Late validation is a process that replaces an existing counterfeit with a valid sensitive value using an input tuple that will be anonymized. For example, let us assume an input tuple t_1 has been anonymized as Example 1. The anonymized results *QIT* and *ST* might be interpreted by applications as

The sensitive value of $(24, M)$ is related to ST_1''

There is only one tuple in QIT_1, whereas the sum of the count in ST_1 $||ST_1||$ is 2, which means there is one counterfeit sensitive value (*Diag.B*) in addition to the real value (*Diag.A*) in ST_1. Then, assume an input tuple t_2 has arrived and its sensitive value is a *Diag.B*. If t_2 is anonymized according to DF as described in Section 3.2, QIT_2 and ST_2, which includes *Diag.B* as a sensitive value, are generated. Now, we need to focus on the counterfeit value of ST_1, *Diag.B*. Using this value, instead of anonymizing the input tuple, we can relate the *Diag.B* to ST_1 as

The sensitive value of $(32, F)$ is related to ST_1''

This means that t_2 has been included in group 1 and there are two tuples in QIT_1. Since $||ST_1||$ is 2, applications might interpret the result as that both *Diag.A* and *Diag.B* are valid values. In this manner, the generation of counterfeits can be reduced and existing counterfeits can be "late validated" with real values.

Meanwhile, to achieve correct validation, the limit of the count value must be considered while late validating. For example, assume that the sensitive value of t_2 is *Diag.A*, not *Diag.B*. Since the count of *Diag.A* in ST_1 is 1, t_1 occupies the count. Hence, t_2 must not be released with QIT_1 as in the above case. Thus, the number of tuples that can be late validated must be restricted to the count of the sensitive value. To make this possible, the metadata of released tuples should be maintained in the anonymization system. We define this information as *released tuples RT*.

Definition 4 (*Released tuples*). Given *groupID* and *si*, the schema of an *RT* is

$$(groupID, si, count_r),$$

where $count_r(si)$ is the number of the sensitive information elements of the tuple that have already been released.
RT_j, which denotes the *RT* with the groupID *j*, is generated with ST_j. Thus, always $|RT_j| = |ST_j|$. The *RT* is the information that is utilized only in the anonymizing process, but is never released. Now, we can define *late validation* as follows.

Definition 5 (*Late validation*). Given QIT_j, ST_j, RT_j, and an input tuple $t(tupleID_t, QI_t, si_t), t$ can be released as (j, QI_t), if and only if $count(si_t)$ and $count_r(si_t)$ exist, $count_r(si_t) < count(si_t)$, and $QI_t \notin QIT_j$.

In the anonymized counterpart of a data stream, an individual can appear multiple times, unlike in those of relational data. However, when multiple QIs related to an individual appear in the same *group*, the *l*-diversity requirement cannot be guaranteed since the adversaries can infer the sensitive information with a probability greater than $1/l$. In order to avoid the above problem, the *late validation* method prevents multiple, repeated deployment of the same QIs in a group by checking that $QI_t \notin QIT_j$.

4.2. Sensitive value generation

In Section 3.2, the *l*-diverse sensitive values are arbitrarily generated when *ST* is organized. The arbitrary generation of sensitive values may have a significant impact on data utility and privacy protection.

The generated values significantly affect the data utility resulting from the process of *late validation*, because *late validation* is possible when the generated sensitive value matches the value of the input tuple. For the sake of data utility, it is important to generate sensitive values that are likely to appear in the input stream. A good prediction of the input data would be the best way to increase *late validations*. To achieve this, in this work we assume that the future is reflected by the past. More specifically, we utilize the past data for generating the sensitive values. First, we prepare a pool of sensitive values from the past data streams. An element of the pool consists of $(si, count(si))$, which is analogous to the element of *ST*. Then, by randomly choosing elements from the pool, *l*-diverse *ST* is organized.

It is known that the *l*-diversity model suffers from two attacks (*skewness attack* and *similarity attack*) in terms of the privacy preservation. Consider a group that has two values: *flu* and *cough*. Although it satisfies 2-diversity, the patient's privacy may not be preserved since the values are semantically similar. As another example, consider 3-diverse values of blood pressure (150, 151, and 153). The patient's privacy may not be preserved since the range of the values is too dense. The privacy of *l*-diverse data is not properly preserved when the sensitive values are similar. In order to block such attacks, Ninghui Li et al. proposed the concept of *t*-closeness [18], which considers the distribution of the sensitive values. However, this concept is beyond the scope of this paper, since we focus primarily on the *l*-diversity model.

4.3. The algorithm

In this section we propose an algorithm for delay-free anonymization. The algorithm is separated into two parts: *late validation* (lines 9–12) and *counterfeit generation* (lines 14–28). When a tuple has arrived, the algorithm finds an existing counterfeit value that can be validated by the input tuple according to Definition 5 (line 7). If there is a suitable counterfeit, the input tuple is released in the form of a *QIT* with the *groupID* of the counterfeit (line 9). Then, the *RT* is updated with an increased count value (line 12). If there are no counterfeits that can be validated, the algorithm generates *l*-diverse counterfeit sensitive values. First, the QIs of the tuple are released in the form of a *QIT* with an assigned *groupID* (line 14). Then, the function *l-diverse_ST_Generation* generates *l*-diverse sensitive values including the sensitive value of the input tuple (line 17). Since the set already satisfies *l*-diversity under Definition 3, the pairs of the set can be released in the form of an *ST* with the assigned *groupID* (line 19). Then, in order to maintain metadata for *late validation*, the *RT* is stored in the anonymization system (lines 21–24). The count value of the counterfeit sensitive value is initialized as 0, while the count value of the input tuple's sensitive value is initialized as 1.

Algorithm 1. DELAY-FREE ANONYMIZATION(*T*,*l*)

1: $QIT = \emptyset$; $ST = \emptyset$; $RT = \emptyset$
2: $gcnt = 0$; $qcnt = 0$; $scnt = 0$
3: let qit, st, and rt be elements of QIT, ST and RT, respectively
4: **while** $t \in T$ is not empty **do**
5: let QI_t and si_t be $t[QI]$ and $t[si]$, respectively
6: let QIS_{ID} be a set of all $qit[QI]$ where $qit[groupID] = ID$
7: let st_i be a randomly chosen st where $st[si] = si_t$, $rt_i[count] < st_i[count]$ and $QI_t \notin QIS_{st[groupID]}$
8: **if** st_i is not *null* **then**
9: release $qit_{qcnt}(st[groupID], QI_t)$
10: insert qit_{qcnt} into QIT
11: $qcnt = qcnt + 1$
12: update rt_i into $(rt_i[groupID], rt_i[si], rt_i[count] + 1)$
13: **else**
14: release $qit_{qcnt}(gcnt, QI_t)$
15: insert qit_{qcnt} into QIT
16: $qcnt = qcnt + 1$
17: let LST be the *l*-diverse set of pairs $(si, count)$ returned by *l-diverse_ST_Generation* (si_t, l)
18: **for all** $lst \in LST$ **do**
19: release $st_{scnt}(gcnt, lst[si], lst[count])$
20: insert st_{scnt} into ST
21: **if** $lst[si] = si_t$ **then**
22: insert $rt_{scnt}(gcnt, lst[si], 1)$ into RT
23: **else**
24: insert $rt_{scnt}(gcnt, lst[si], 0)$ into RT
25: **end if**
26: $scnt = scnt + 1$
27: **end for**
28: $gcnt = gcnt + 1$
29: **end if**
30: **end while**

Function l-DIVERSE_ST_GENERATION(si_t, l)
1: let SV be the domain of sensitive values
2: let LST be a set of $(si, count)$ pairs
3: insert $(si_t, 1)$ into LST
4: **while** $|LST| < l$ or $||LST||/l < \exists lst[count]$ where $lst \in LST$ **do**
5: let sv be a randomly chosen element from SV
6: let lst_i be an element of LST where $lst_i[si] = sv$
7: **if** lst_i is *null* **then**
8: insert $(sv, 1)$ into LST
9: **else**
10: update lst_i into $(lst_i[si], lst_i[count] + 1)$
11: **end if**
12: **end while**
13: return LST

l-diverse_ST_Generation is a function that generates an LST (l-diverse ST) according to Definition 3. First, it inserts si_t into the LST (line 3) because the input tuple's sensitive value should be included. Then, counterfeit sensitive values are inserted into the LST (lines 5–10), until it satisfies the conditions of l-diversity for the LST (line 4). The counterfeits are randomly chosen from the SV, which is the domain of sensitive values. In this process, if the chosen sensitive value already exists in LST, the count of the value is increased instead of the sensitive value being inserted into the LST (line 10). We utilize the sampled past dataset for the SV, as we mentioned in Section 4.2.

5. Data utility considerations

In this section, we discuss data utility considerations in the context of DF. In Section 5.1, we compare the DF method and accumulation-based methods with respect to information loss. In Section 5.2, we discuss *sensitive attribute uncertainty* generated by counterfeit sensitive values and devise a measure to evaluate the uncertainty.

5.1. Discussion of information loss

First, in Section 5.1.1, we present the existing metrics for measuring information loss. Then, in Section 5.1.2, we extend the metric to measure the information loss of the sensitive attribute.

5.1.1. Information loss

Existing information loss metrics are designed to measure the generalization quality of QI attributes [22]. QI attributes are classified into continuous and categorical attributes. They are represented as an interval and a taxonomy tree, respectively. The information loss of a continuous and a categorical attribute is computed as follows.

$$IL_a(a_{cat}) = \frac{|M_P| - 1}{|M| - 1} \tag{3}$$

$$IL_a(a_{con}) = \frac{U_i - L_i}{U - L} \tag{4}$$

M denotes the set of leaf nodes in the tree and M_P denotes the set of leaf nodes of the generalized node. U_i and L_i denote the upper bound and the lower bound of the generalized interval, respectively. U and L denote the maximum and the minimum value of the whole domain, respectively.

The information loss of a tuple is defined as the average value of each attribute's information loss. The information loss is

$$IL = \frac{1}{n}\sum_{i=1}^{n} IL_a(a_i), \tag{5}$$

where n is the number of QI attributes.

5.1.2. Information loss considering sensitive attribute

DF preserves the QI values. Hence, the above information loss metric, which considers only the distortion of QI attributes, is not appropriate for measuring the quality of DF. To verify this, we would like to join QIT and ST, QIT and RT by a join key *groupID*. The schema of $QIT \bowtie ST$ with a *groupID* j is

$$T_j^A(QI, si, count(si))$$

Let us consider the information loss incurred when input tuple t is anonymized into T_j^A by DF. There might be multiple elements in T_j^A, with l-diverse sensitive values. Let t_j^A be an arbitrary element among T_j^A; then, $IL(t_j^A)$ is always zero because DF does not distort QI attributes.

Instead of distorting QI values, DF inflates sensitive values into an l-diverse ST. In other words, as $t(QI, si)$ has been anonymized, the original QI gains multiple sensitive values. This means that the correlation between QI and si diminishes, because counterfeit values are added to the original sensitive value. Therefore, to measure the quality of DF, the metric considers the inflation of sensitive values. We measure the information loss based on the size of counterfeit values over the size of all sensitive values including counterfeits and an original value in the group. Each size corresponds to the number of distinct values in a given set. It should be noted that the size of the counterfeit sensitive values is always $|ST_j| - 1$, since there is always one original value among all the sensitive values in a group. According to this metric, we define the information loss of the sensitive attribute as

$$IL_a(a_{sens}) = \frac{|ST_j - 1|}{|ST_j|} \tag{6}$$

Then, to measure the information loss of the tuple considering the sensitive attribute, we apply Eq. (6) to the existing information loss metric (i.e., Eq. (5)):

$$IL(t) = \frac{1}{n+1}\left[\sum_{i=1}^{n} IL_a(a_i) + IL_a(a_{sens})\right] \tag{7}$$

As compared with Eq. (5), Eq. (7) considers one more attribute (i.e., the sensitive attribute).

Example 2 (*Information loss considering sensitive attribute*). Let us revisit Example 1. There is one QI(24, male) that consists of two attributes (i.e., age and sex). There are two sensitive values (Diag.A and Diag.B) in group 1. According to Eq. (6), the information loss of the sensitive attribute is 1/2. Thus, according to Eq. (7), the information loss of the tuple is computed as $\frac{1}{2+1}[(0+0)+\frac{1}{2}] = \frac{1}{6}$.

5.2. Discussion of sensitive attribute uncertainty

As a result of the DF anonymization, counterfeit sensitive values are generated. These counterfeit sensitive values affect data utility since they lower the statistical value of the anonymized data. For example, consider a query "count the records whose blood pressure is higher than 200 for the past one hour". The query result might have some uncertainty due to the counterfeit blood pressure values. In order to evaluate the uncertainty of the sensitive values, we need a new measure. We define the uncertainty as the proportion of counterfeit values among all generated values.

Meanwhile, the objective of *late validation* is to reduce the number of counterfeits, thus lower the uncertainty. Since *late validation* can occur when a matched input tuple has arrived, the uncertainty varies according to the arrival time of the matched input tuple. Thus, the measure of the uncertainty is dependent on time.

Table 1
Attributes of *Adult* dataset.

Attribute	Type	No. of distinct values
Age	Continuous	74
Education	Continuous	16
Workclass	Categorical	8
Marital	Categorical	7
Race	Categorical	5
Sex	Categorical	2
Native-country	Categorical	40
(Salary–occupation)	Sensitive	30

Table 2
Attributes of *NPS* dataset.

Attribute	Type	No. of distinct values
Age	Continuous	107
Length of stay	Continuous	53
Sex	Categorical	2
Location of the hospital	Categorical	16
Admission route	Categorical	6
Disease	Sensitive	4848

In what follows, we define *Sensitive Attribute Uncertainty* (SAU), which is the uncertainty of sensitive values at a given time interval T_0 to T, where T_0 denotes the time of the initial state of the anonymization process and T denotes the current time. SAU is computed as (the number of counterfeit sensitive values from T_0 to T)/(the number of all generated sensitive values from T_0 to T). The equation can be represented by the count values of RT and ST as follows, where $st[count](T)$ and $rt[count](T)$ denote the count value of st from T_0 to T and the count value of rt from T_0 to T, respectively.

$$SAU(T) = \frac{\sum_{st \in ST} st[count](T) - \sum_{rt \in RT} rt[count](T)}{\sum_{st \in ST} st[count](T)} \quad (8)$$

6. Experiments

We evaluated the effectiveness and the efficiency of DF through a set of experiments using two datasets: the Adult dataset from the UCI repository, and the NPS dataset from HIRA (Health Insurance Review and Assessment service in Korea).

The Adult dataset consists of 32,561 tuples and has 14 attributes of which we utilized the following 9 attributes: *age, education, workclass, marital, race, sex, native-country, salary, and occupation*. We classified these attributes into continuous, categorical, and sensitive attributes. We considered the first seven attributes to be QIs and the remaining two attributes to be sensitive attributes. In the experiments, we merged two sensitive attributes, salary and occupation, into a single attribute *salary–occupation* to enlarge the domain size of the sensitive attribute, so that the value of the attribute is represented as {*salary, occupation*}. (We do not consider multiple sensitive values in this paper.) The domain size of the sensitive attribute becomes 30.

The NPS (National Patients Sample) dataset consists of electronic medical records of Korean people sampled with 3% rate. We analyzed 117,047 tuples, which are records of a day (14th, September, 2011). We utilized the following 6 attributes: *age, sex, length of stay, location of the hospital, admission route, and disease*. We considered the first five attributes to be QIs and the remaining *disease* attribute to be a sensitive attribute.

Each of Adult-d and NPS-d denotes a subset of the dataset where the first d attributes among the original attributes are considered as QIs. Table 1 and Table 2 show the type and the number of the distinct values of the attributes. The experiments were conducted on a computer equipped with an Intel i3 3.07 GHz CPU and 8 GB RAM.

6.1. Comparative study

We measured the *average processing time per tuple* (APTT) and information loss in order to compare the DF method with accumulation-based methods. The average processing time per tuple is the average elapsed time from the input of the tuples to their release. We implemented an accumulation-based method based on CASTLE [7], which is a typical accumulation-based method. CASTLE has an important parameter δ, which represents a delay constraint, i.e., the number of tuples that would be accumulated for the anonymization. The value of δ varies from 10 to 10,000, and we represent it as an alphanumeric ABM-n, which signifies an accumulation-based method with $\delta = 10^n$. For example, ABM-2 indicates the accumulated-based method with δ of 10^2. For reference, the default setting in CASTLE is ABM-4, i.e., $\delta = 10^4$. The value of l is fixed to 10.

Fig. 3 illustrates the average processing time per tuple of DF and the accumulation-based methods. In the case of accumulation-based methods, the average processing time per tuple increases as the value of δ increases, since the accumulation continues until the number of input tuples reaches δ. This proves that the accumulation generates the delay. In the result of the Adult dataset, DF takes about 0.037 ms to anonymize a tuple, regardless of the dimensionality, which is less time than ABM-1 takes (~0.18 ms) that accumulates only 10 tuples for offering 10-diversity. In the case of the NPS dataset, the average processing time per tuple of DF is about 0.09 ms, which is similar to that of ABM-1 (~0.8 ms).

Fig. 4 presents the information loss of each method. DF shows significantly low values compared to the accumulation-based methods. The information loss of DF decreases as the number of

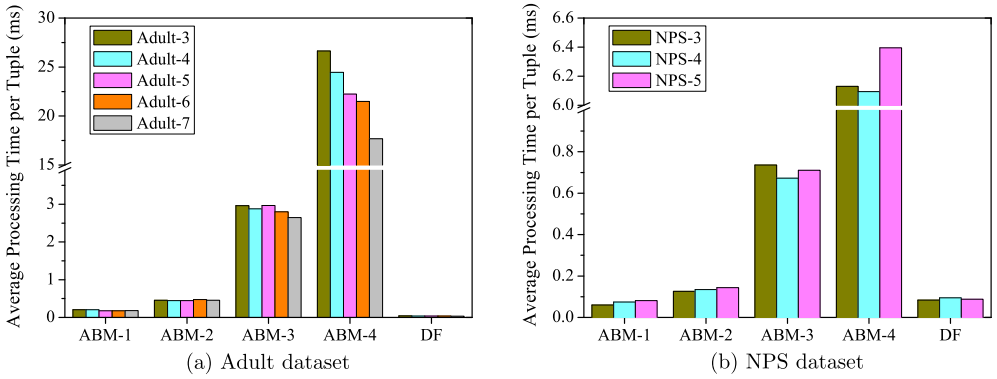

Fig. 3. Average processing time per tuple.

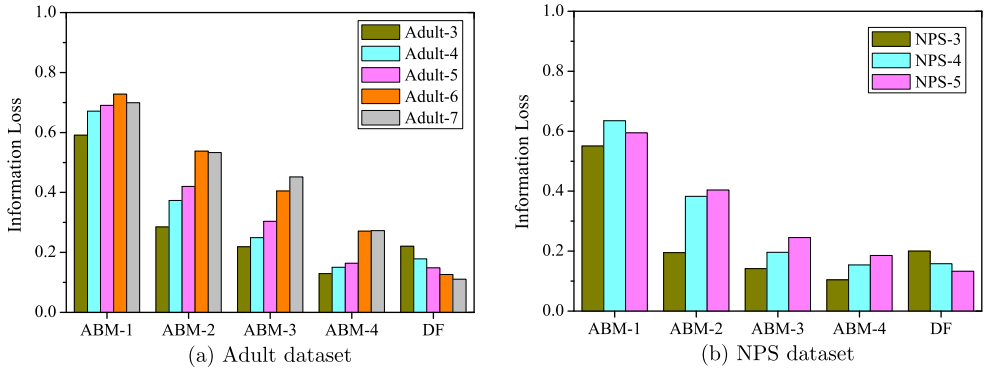

Fig. 4. Information loss.

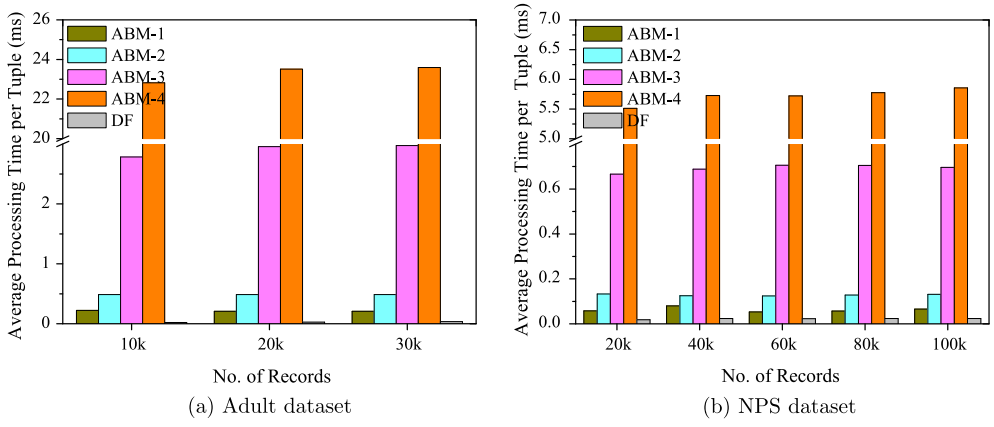

Fig. 5. Average processing time per tuple by No. of records.

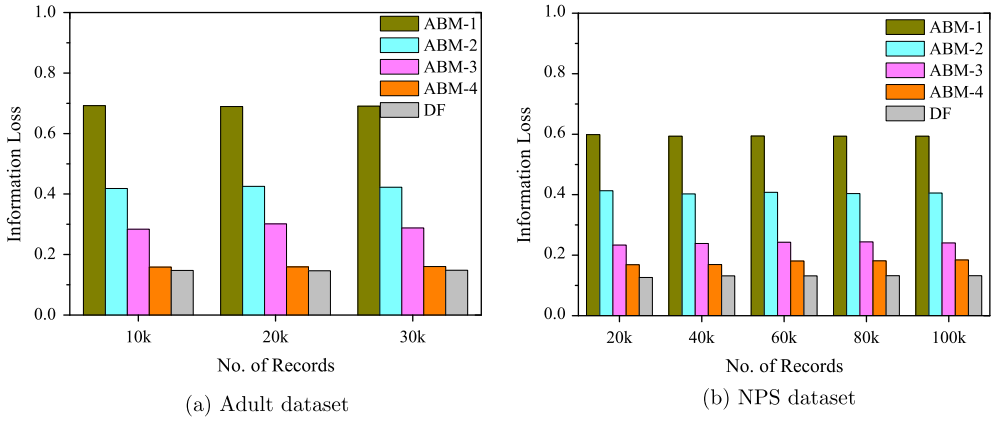

Fig. 6. Information loss by No. of records.

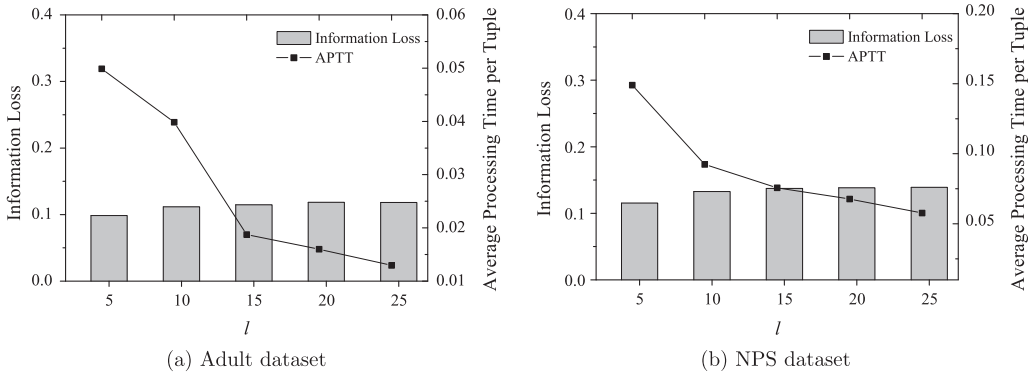

Fig. 7. Effect of *l*.

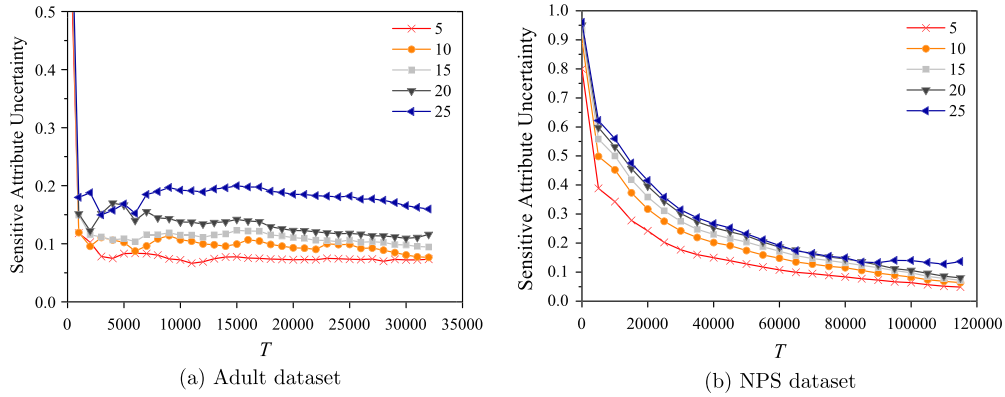

Fig. 8. Sensitive attribute uncertainty.

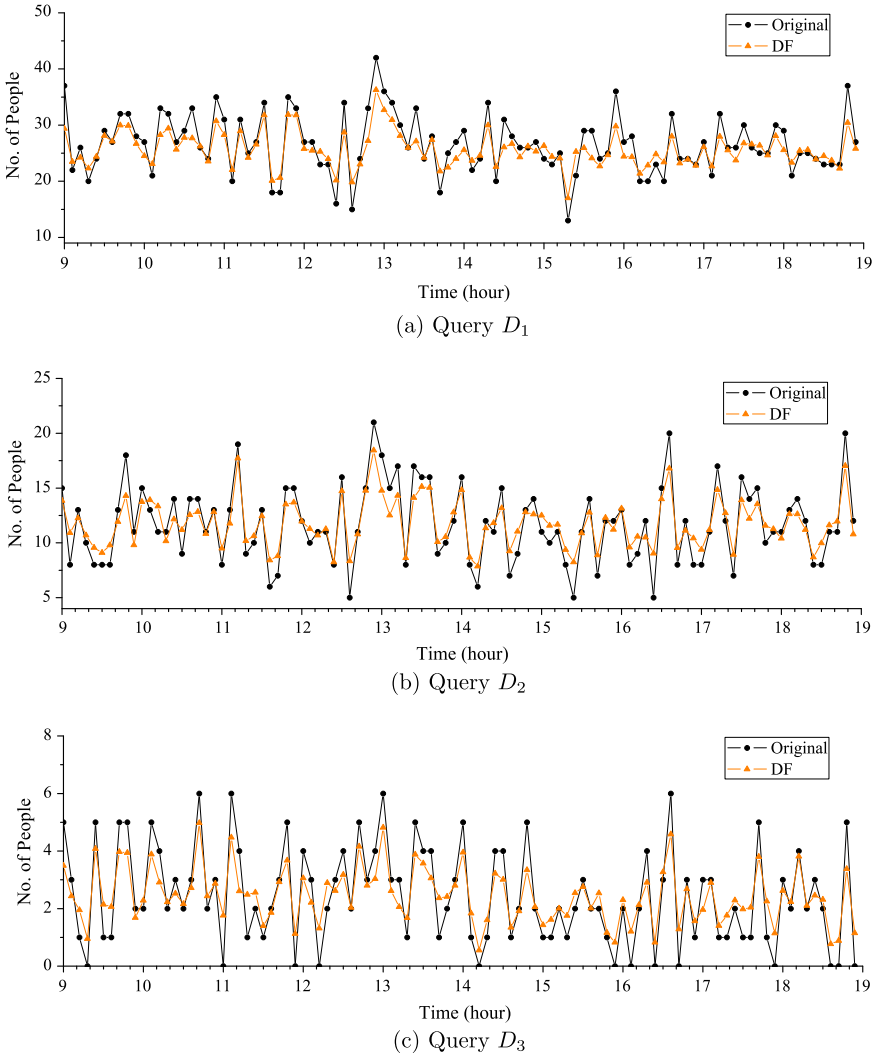

Fig. 9. Results of the analysis queries on the demographic data streams.

QIs grows. The reason is that the loss is affected only by the sensitive attributes, and their portion decreases as the dimensionality grows. In contrast, the information loss of accumulation-based methods tends to increase as the dimensionality grows, because the loss is affected by QIs as well as by sensitive attributes.

Fig. 5 illustrates the average processing time per tuple of each method for increasingly larger subsets of the datasets. In Fig. 5a and b, the experiments are conducted on Adult-5 and NPS-4, respectively. Both figures show that the processing time is almost stable though the number of records grows.

Fig. 6 presents the information loss of each method for increasingly larger subsets of the datasets. The information loss is almost stable despite the variance of the number of records, because the information loss is already stabilized in the initial stage of the anonymization.

6.2. Effects of l

Fig. 7 shows how the average processing time per tuple and the information loss change according to the value of l. We chose 5, 10, 15, 20, and 25 for the various l. The average processing time per tuple decreases as l grows, because the computation time is reduced as a larger l generates a smaller number of groups. In contrast, information loss slightly increases as l grows, because the proportion of counterfeits among all sensitive values increases. It is also explained by Eq. (6).

6.3. Sensitive attribute uncertainty

Fig. 8 illustrates *sensitive attribute uncertainty* (SAU). The x-axis represents the time, which corresponds to the order of the tuple input. As the l value increases, SAU tends to increase. The reason is that the larger l value leads to the generation of more counterfeit values in the anonymization process. In Fig. 8a, SAU is unstable before T reaches about 7500. This means that sufficient sensitive values are prepared for the late validation method after T reaches about 7500. In Fig. 8b, SAU is stabilized more gradually than that of Fig. 8a. The reason is that late validation occurs more frequently where the distinct number of sensitive values is smaller. We would like to note that the NPS dataset has 4848 distinct sensitive values, while the adult dataset has 30.

Without the late validation method, $rt[count](T)$ is always 1. Then, we can easily predict that SAU would be nearly 1, because Eq. (8) would be changed as

$$\frac{\sum_{st\in ST} st[count](T) - \sum_{rt\in RT} 1}{\sum_{st\in ST} st[count](T)}$$

However, our experiments show SAU values between 0.05 and 0.2 according to the l value (after the values become stable). This means that the number of counterfeit sensitive values is significantly reduced by the late validation method.

6.4. Example scenarios

In order to illustrate the usefulness of anonymized data streams, which are produced by DF, let us present a scenario of data analysis. In the following scenarios, we assume two types of data streams (i.e., demographic data stream and medical data stream), which continue for 10 h (9:00–19:00). Each data stream was generated from the Adult dataset and the NPS dataset, respectively. We anonymized the data streams using the proposed meth-

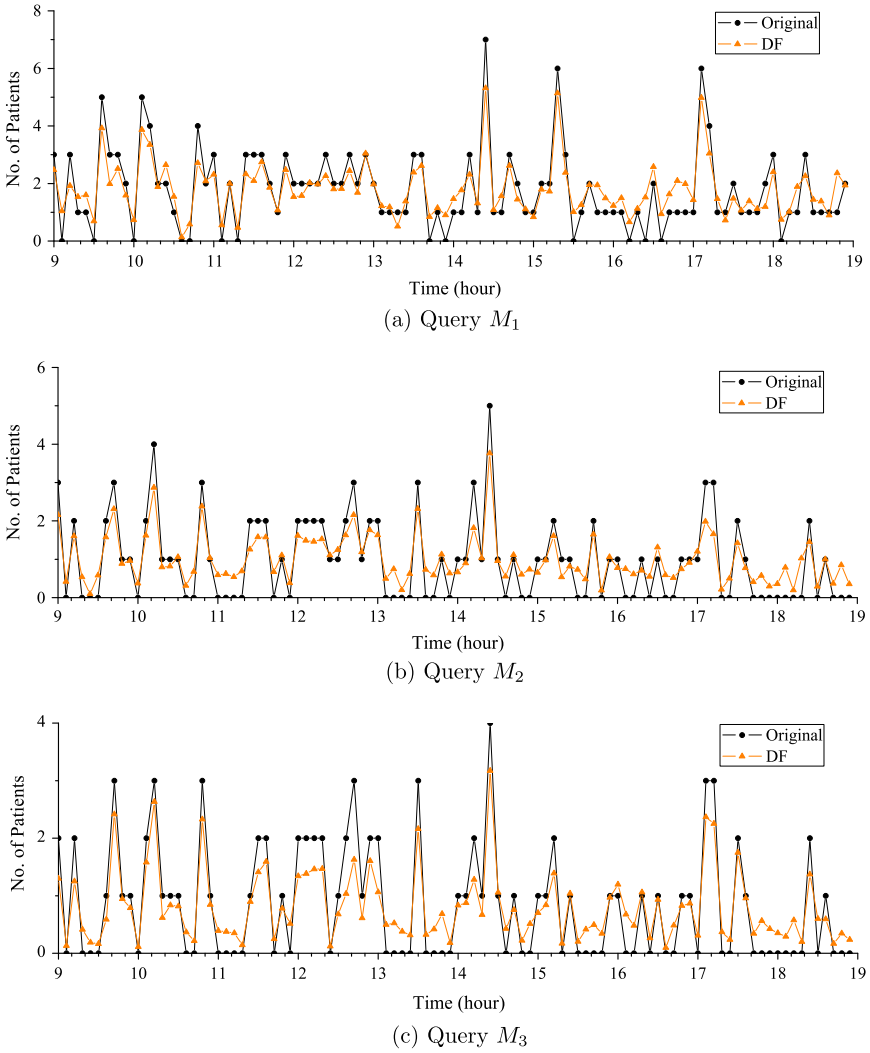

Fig. 10. Results of the analysis queries on the medical data streams.

od ($l = 10$), then analyzed the anonymized data streams through monitoring queries. The queries were executed every 6 min, thus a total of one hundred monitoring queries were conducted for 10 h. We further compare the monitoring results of the anonymized data streams with those of the original data streams.

6.4.1. Monitoring occupational information over anonymized demographic data streams

Let us assume that there are researchers who study occupational information from demographic data streams. The researchers execute monitoring queries to analyze the data streams. The SI (i.e., *salary–occupation*) and two QIs (i.e., *sex* and *marital*) are used in the monitoring queries. The queries are described as follows:

- **Query D_1**: SELECT count (∗) FROM demographic data stream WHERE salary–occupation = '⩽50 K, sales' every 6 min.
- **Query D_2**: SELECT count (∗) FROM demographic data stream WHERE salary–occupation = '⩽50 K, sales' sex = 'female' every 6 min.
- **Query D_3**: SELECT count (∗) FROM demographic data stream WHERE salary–occupation ='⩽50 K, sales' sex = 'female' and marital = 'divorced' every 6 min.

Fig. 9 shows results of the analysis queries based on the original data stream and the anonymized data stream by DF. In each figure, the result of the anonymized data stream shows similar phases to that of the original data stream, although there are trivial errors between them. In Fig. 9a–c, the average errors are 2.59, 1.47, and 0.77, respectively.

6.4.2. Monitoring epidemic diseases over anonymized medical data streams

Hospitals generate medical records whenever the patients receive medical treatments. The sequence of medical records composes an electronic health data stream. In this environment, let us assume a feasible scenario. For research purposes, the Ministry of Health and Welfare in Korea collects the data streams from hospitals in the entire country. Researchers of the government analyze the data streams to monitor the occurrence of epidemic diseases in real time. The SI (i.e., *disease*) and two QIs (i.e., *location of the hospital* and *sex*) are used in the monitoring queries. The queries for the analysis are described as follows ('nncd' is the name set of diseases legally designated as epidemics):

- **Query M_1**: SELECT count (∗) FROM medical data stream WHERE disease in 'nncd' every 6 min.
- **Query M_2**: SELECT count (∗) FROM medical data stream WHERE disease in 'nncd' and location = 'Seoul' every 6 min.
- **Query M_3**: SELECT count (∗) FROM medical data stream WHERE disease in 'nncd' and location = 'Seoul' and sex = 'Male' every 6 min.

Fig. 10 shows results of the analysis queries based on the original data stream and the anonymized data stream by DF. In each figure, the result of the anonymized data stream shows similar phases to that of the original data stream, although there are trivial errors between them. In Fig. 10a–c, the average errors are 0.58, 0.47, and 0.42, respectively.

7. Conclusion

In this paper, we presented a delay-free anonymization method for preserving the privacy of electronic health data streams. The proposed method does not generate a significant delay during the anonymization process and release of data streams. Since each QI has l-diverse sensitive values, the privacy of the data is preserved with a probability of $1/l$. Late validation significantly decreases the number of counterfeit values, and hence, the anonymization result has high data utility. Our method overcomes the drawbacks of existing accumulation-based methods in real-time data stream anonymization.

Several future studies can be considered as an extension of the study reported in this paper. First, we will devise a method for generating counterfeit and count values. If the result of statistical analysis or past data are considered for generating the values, more reliable anonymization results can be realized. Second, it would be meaningful to investigate the timing of the late validation. In this paper, we focus on matching input tuples and counterfeits for the late validation. The effectiveness can be improved if the elapsed time of the counterfeits is considered. Additionally, we intend to employ the Slicing [23] instead of the Anatomy [17] technique for DF anonymization. We predict that the correlation between the QIs and the sensitive attributes will be preserved by the Slicing technique. Finally, we would like to design and study a distributed version of our framework. As the volume of electronic health data streams is rapidly growing, data streams increasingly need to be processed in a distributed environment.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2013-022884).

References

[1] Babcock B, Babu S, Datar M, Motwani R, Widom J. Models and issues in data stream systems. In: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems. ACM; 2002 . p. 1–16.
[2] Abadi DJ, Carney D, Çetintemel U, Cherniack M, Convey C, Lee S, et al. Aurora: a new model and architecture for data stream management. Int J Very Large Data Bases 2003;12(2):120–39.
[3] Gaber MM, Zaslavsky A, Krishnaswamy S. Mining data streams: a review. ACM Sigmod Record 2005;34(2):18–26.
[4] Bifet A, Holmes G, Pfahringer B, Kirkby R, Gavaldà R. New ensemble methods for evolving data streams. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining. ACM; 2009. p. 139–48.
[5] Sweeney L. k-anonymity: a model for protecting privacy. Int J Uncertain, Fuzz Knowl-Based Syst 2002;10(05):557–70.
[6] Machanavajjhala A, Kifer D, Gehrke J, Venkitasubramaniam M. l-diversity: privacy beyond k-anonymity. ACM Trans Knowl Discov Data (TKDD) 2007;1(1):3.
[7] Cao J, Carminati B, Ferrari E, Tan K-L. Castle: continuously anonymizing data streams. IEEE Trans Dependable Secure Comput 2011;8(3):337–52.
[8] Zhou B, Han Y, Pei J, Jiang B, Tao Y, Jia Y. Continuous privacy preserving publishing of data streams. In: Proceedings of the 12th international conference on extending database technology: advances in database technology. ACM; 2009. p. 648–59.
[9] Li J, Ooi BC, Wang W. Anonymizing streaming data for privacy protection. In: Proceedings of IEEE 24th international conference on data engineering. IEEE; 2008. p. 1367–9.
[10] Wang P, Lu J, Zhao L, Yang J. B-castle: an efficient publishing algorithm for k-anonymizing data streams. In: Proceedings of 2010 second WRI global congress on intelligent systems (GCIS), vol. 2; 2010. p. 132–6.
[11] Wang P, Zhao L, Lu J, Yang J. Sanatomy: privacy preserving publishing of data streams via anatomy. In: Proceedings of the 2010 third international symposium on information processing (ISIP). IEEE; 2010. p. 54–7.
[12] Zhang J, Yang J, Zhang J, Yuan Y. Kids: K-anonymization data stream based on sliding window. In: Proceedings of the 2010 2nd international conference on future computer and communication (ICFCC), vol. 2; 2010. p. V2-311–6.
[13] Zakerzadeh H, Osborn SL. Faanst: fast anonymizing algorithm for numerical streaming data. In: Data privacy management and autonomous spontaneous security. Springer; 2011. p. 36–50.
[14] Cao J, Karras P, Kalnis P, Tan K-L. Sabre: a sensitive attribute bucketization and redistribution framework for t-closeness. VLDB J 2011;20(1):59–81.

[15] LeFevre K, DeWitt DJ, Ramakrishnan R. Incognito: efficient full-domain k-anonymity. In: Proceedings of the 2005 ACM SIGMOD international conference on management of data. ACM; 2005. p. 49–60.
[16] Samarati P. Protecting respondents identities in microdata release. IEEE Trans Knowl Data Eng 2001;13(6):1010–27.
[17] Xiao X, Tao Y. Anatomy: simple and effective privacy preservation. In: Proceedings of the 32nd international conference on very large data bases, VLDB endowment; 2006. p. 139–50.
[18] Li N, Li T, Venkatasubramanian S. t-closeness: privacy beyond k-anonymity and l-diversity. In: Proceedings of IEEE international conference on data engineering; 2007. p. 106–15.
[19] Byun J-W, Sohn Y, Bertino E, Li N. Secure anonymization for incremental datasets. In: Secure data management. Springer; 2006. p. 48–63.
[20] Xiao X, Tao Y. M-invariance: towards privacy preserving re-publication of dynamic datasets. In: Proceedings of the 2007 ACM SIGMOD international conference on management of data. ACM; 2007. p. 689–700.
[21] Bu Y, Fu AWC, Wong RCW, Chen L, Li J. Privacy preserving serial data publishing by role composition. Proceedings of the VLDB endowment 2008;1(1):845–56.
[22] Iyengar VS. Transforming data to satisfy privacy constraints. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining. ACM; 2002. p. 279–88.
[23] Li T, Li N, Zhang J, Molloy I. Slicing: a new approach for privacy preserving data publishing. IEEE Trans Knowl Data Eng 2012;24(3):561–74.