Journal of Biomedical Informatics 50 (2014) 46–61

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

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

# Disassociation for electronic health record privacy

Grigorios Loukides [a], John Liagouris [b], Aris Gkoulalas-Divanis [c,*], Manolis Terrovitis [d]

[a] School of Computer Science and Informatics, Cardiff University, United Kingdom
[b] Department of Electrical and Computer Engineering, NTUA, Greece
[c] IBM Research – Ireland, Dublin, Ireland
[d] IMIS, Research Center Athena, Greece

ARTICLE INFO

*Article history:*
Received 23 August 2013
Accepted 16 May 2014
Available online 28 May 2014

*Keywords:*
Privacy
Electronic health records
Disassociation
Diagnosis codes

ABSTRACT

The dissemination of Electronic Health Record (EHR) data, beyond the originating healthcare institutions, can enable large-scale, low-cost medical studies that have the potential to improve public health. Thus, funding bodies, such as the National Institutes of Health (NIH) in the U.S., encourage or require the dissemination of EHR data, and a growing number of innovative medical investigations are being performed using such data. However, simply disseminating EHR data, after removing identifying information, may risk privacy, as patients can still be linked with their record, based on diagnosis codes. This paper proposes the first approach that prevents this type of data linkage using *disassociation*, an operation that transforms records by splitting them into carefully selected subsets. Our approach preserves privacy with significantly lower data utility loss than existing methods and does not require data owners to specify diagnosis codes that may lead to identity disclosure, as these methods do. Consequently, it can be employed when data need to be shared broadly and be used in studies, beyond the intended ones. Through extensive experiments using EHR data, we demonstrate that our method can construct data that are highly useful for supporting various types of clinical case count studies and general medical analysis tasks.



## 1. Introduction

Healthcare data are increasingly collected in various forms, including Electronic Health Records (EHR), medical imaging databases, disease registries, and clinical trials. Disseminating these data has the potential of offering better healthcare quality at lower costs, while improving public health. For instance, large amounts of healthcare data are becoming publicly accessible at no cost, through *open data* platforms [4], in an attempt to promote accountability, entrepreneurship, and economic growth ($100 billion are estimated to be generated annually across the US health-care system [11]). At the same time, sharing EHR data can greatly reduce research costs (e.g., there is no need for recruiting patients) and allow large-scale, complex medical studies. Thus, the National Institutes of Health (NIH) calls for increasing the reuse of EHR data [7], and several medical analytic tasks, ranging from building predictive data mining models [8] to genomic studies [14], are being performed using such data.

Sharing EHR data is highly beneficial but must be performed in a way that preserves patient and institutional privacy. In fact, there are several data sharing policies and regulations that govern the sharing of patient-specific data, such as the HIPAA privacy rule [47], in the U.S., the Anonymization Code [6], in the U.K., and the Data Protection Directive [3], in the European Union. In addition, funding bodies emphasize the need for privacy-preserving healthcare data sharing. For instance, the NIH requires data involved in all NIH-funded Genome-Wide Association Studies (GWAS) to be deposited into a biorepository, for broad dissemination [44], while safeguarding privacy [1]. Alarmingly, however, a large number of privacy breaches, related to healthcare data, still occur. For example, 627 privacy breaches, which affect more than 500 and up to 4.9 M individuals each, are reported from 2010 to July 2013 by the U.S. Department of Health & Human Services [15].

One of the main privacy threats when sharing EHR data is *identity disclosure* (also referred to as *re-identification*), which involves the association of an identified patient with their record in the published data. Identity disclosure may occur even when data are *de-identified* (i.e., they are devoid of identifying information). This is because publicly available datasets, such as voter registration lists, contain identifying information and can be linked to published datasets, based on potentially identifying information,

* Corresponding author. Tel.: +353 18103096.
*E-mail addresses:* g.loukides@cs.cf.ac.uk (G. Loukides), liagos@dblab.ece.ntua.gr (J. Liagouris), arisdiva@ie.ibm.com (A. Gkoulalas-Divanis), mter@imis.athena-innovation.gr (M. Terrovitis).

http://dx.doi.org/10.1016/j.jbi.2014.05.009
1532-0464/

such as demographics [52], diagnosis codes [34], and lab results [9]. The focus of our work is on diagnosis codes, because: (i) they pose a high level of re-identification risk [34], and (ii) ensuring that diagnosis codes are shared in a privacy-preserving way, is challenging, due to their characteristics [55,25,28]. For example, more than 96% of 2700 patient records that are involved in an NIH-funded GWAS were shown to be uniquely re-identifiable, based on diagnosis codes, and, applying popular privacy-preserving methods, distorts the published data to the point that they lose their clinical utility [34].

To perform identity disclosure, an attacker must possess three types of knowledge: (i) a patient's identity, (ii) a set of diagnosis codes, and (iii) whether a patient is included in the published dataset (research sample) [36]. Knowledge of the first two types can come in the form of background knowledge [36] or may be solicited by exploiting external data sources.[1] At the same time, knowledge of the third type is obtainable through interaction with data subjects [19], or it can be inferred by applying the procedure used to create the research sample from a larger patient population, which is often described in the literature [36]. To see how identity disclosure may occur, consider the de-identified data in Fig. 1. In these data, each record corresponds to a distinct patient and contains the set of diagnosis codes that this patient is associated with. The description of the diagnosis codes in Fig. 1 is shown in Fig. 2. An attacker, who knows that a patient is diagnosed with *Bipolar I disorder, single manic episode, mild* (denoted with the code 296.01) and *Closed dislocation of finger, unspecified part* (denoted with 834.0), can associate an identified patient with the first record, denoted with $r_1$, in the data of Fig. 1, as the set of codes $\{296.01, 834.0\}$ appears in no other record. Note that the attacker cannot perform this association, based on knowledge of either 296.01 or 834.0, but can associate the identified patient with $r_1$, if they know any other code or codes, in addition to the set of codes $\{296.01, 834.0\}$. Notice also that, in this work, we consider ICD-9 codes,[2] following [36,35]. However, our approach can be applied to other standardized codes, such as Common Procedure Terminology (CPT) codes.

### 1.1. Motivation

Preventing identity disclosure based on diagnosis codes is possible by applying the methods proposed in [36,35]. Both methods transform diagnosis codes to ensure that the probability of performing identity disclosure, based on *specified* sets of diagnosis codes, will not exceed a data-owner specified parameter $k$. Data transformation is performed using *generalization* (i.e., by replacing diagnosis codes with more general, but semantically consistent, ones) and *suppression* (i.e., by deleting diagnosis codes). Furthermore, both methods aim at transforming data in a way that does not affect the findings of biomedical analysis tasks that the data are intended for. These tasks are specified by data owners and used to control the potential ways diagnosis codes are generalized and/or suppressed. For example, applying the *Clustering-Based Anonymizer* (CBA) algorithm [35], which outperforms the method in [36] in terms of preserving data utility, to the data in Fig. 1, produces the data in Fig. 3(a). In this example, CBA was applied using $k = 3$ and with the goal of (i) thwarting identity disclosure, based on all sets of 2 diagnosis codes, and (ii) preserving the findings of studies $u_1$ to $u_5$ in Fig. 3(b), which require counting the number of patients diagnosed with any combination of codes in them. Observe that the codes 294.10, 295.04, and 296.00 to 296.03 are generalized to $(294.10, 295.04, 296.00, 296.01, 296.02, 296.03)$,

| ID | Records |
|---|---|
| $r_1$ | {296.00, 296.01, 296.02, 834.0, 944.01} |
| $r_2$ | {296.00, 296.02, 296.01, 401.0, 944.01, 692.71, 695.10} |
| $r_3$ | {296.00, 296.02, 692.71, 834.0, 695.10} |
| $r_4$ | {296.00, 296.01, 692.71, 401.0} |
| $r_5$ | {296.00, 296.01, 296.02, 692.71, 695.10} |
| $r_6$ | {296.03, 295.04, 404.00, 480.1} |
| $r_7$ | {294.10, 296.03, 834.0, 944.01} |
| $r_8$ | {294.10, 295.04, 296.03, 480.1} |
| $r_9$ | {294.10, 295.04, 404.00} |
| $r_{10}$ | {294.10, 295.04, 296.03, 834.0, 944.01} |

**Fig. 1.** Original dataset $D$.

which is interpreted as any non-empty subset of these codes, and that 7 out of 13 distinct codes are suppressed. The result of CBA thwarts identity disclosure (i.e., all combinations of 2 diagnosis codes appear at least 3 times in Fig. 3) and allows performing $u_1$ and $u_3$ accurately. To see why this is the case, consider $u_3$, for example. Note that 4 patients are associated with a combination of the codes $\{401.0, 404.00\}$ in $u_3$, in both Fig. 1 and in Fig. 3(a). However, the studies $u_2, u_4$, and $u_5$ can no longer be performed accurately, as some of their associated diagnosis codes have been suppressed.

In fact, the methods in [36,35] assume a setting in which data owners possess domain expertise that allows them to specify: (i) sets of diagnosis codes that lead to identity disclosure, and (ii) sets of diagnosis codes that model analytic tasks that the published data are intended for. The ability of the published data to support these tasks is a strong requirement, and suppression is used when this requirement cannot be satisfied.[3] As can be seen in Fig. 3(a), the fact that $u_2 = \{692.71, 695.10\}$ was not satisfied led CBA to suppress both 692.71 and 695.10. The setting considered in [36,35] can model some real data sharing scenarios, such as the sharing of data between collaborating researchers, who perform specific analytic tasks [36].

However, it is important to consider a different setting, where data are shared more broadly and may be used for studies beyond those that are specified by data owners. This setting becomes increasingly common, as databanks (e.g., [2,5]) host a wide range of patient-specific data and grow in size and popularity. Addressing this setting calls for developing methods that offer strong privacy and permit the publishing of data that remain useful, for analytic tasks that cannot be predetermined, in addition to any intended ones. In fact, the aforementioned methods [36,35] are not suitable for this setting, because their application would cause excessive loss of data utility, as it will become clear later.

### 1.2. Contributions

In this paper, we propose the first approach for the privacy-preserving sharing of diagnosis codes under this new setting. Our approach allows data owners to share data that prevent identity disclosure, and does not incur excessive information loss or harm the usefulness of data in medical analysis. This work makes the following specific contributions.

First, we develop an effective algorithm that prevents identity disclosure, based on all sets of $m$ or fewer diagnosis codes, by limiting its probability to $\frac{1}{k}$, where $k$ and $m$ are data-owner specified parameters. To achieve this, the algorithm transforms data using *disassociation*, an operation that splits the records into carefully constructed subrecords, containing original (i.e., non-transformed) diagnosis codes. As such, disassociation can "hide" combinations of

[1] These include publicly available voter lists combined with hospital discharge summaries [50], or the identified EHR system available to the primary care environment [34].

[2] http://www.cdc.gov/nchs/data/icd9/icdguide10.pdf.

[3] Due to the computational complexity of the problem, no guarantees that these requirements will be satisfied are provided by the methods in [36,35].

| Diagnosis code | Description |
|---|---|
| 294.10 | Dementia in conditions classified elsewhere without behavioral disturbance |
| 295.04 | Simple type schizophrenia, chronic with acute exacerbation |
| 296.00 | Bipolar I disorder, single manic episode, unspecified |
| 296.01 | Bipolar I disorder, single manic episode, mild |
| 296.02 | Bipolar I disorder, single manic episode, moderate |
| 296.03 | Bipolar I disorder, single manic episode, severe, without mention of psychotic behavior |
| 401.0 | Malignant essential hypertension |
| 404.00 | Hypertensive heart and chronic kidney disease, malignant, without heart failure and with chronic kidney disease stage I through stage IV, or unspecified |
| 480.1 | Pneumonia due to respiratory syncytial virus |
| 692.71 | Sunburn |
| 695.10 | Erythema multiforme, unspecified |
| 834.0 | Closed dislocation of finger, unspecified part |
| 944.01 | Burn of unspecified degree of single digit (finger (nail) other than thumb |

**Fig. 2.** Diagnosis codes in $D$ and their description.

| ID | Records |
|---|---|
| $r_1$ | (294.10, 295.04, 296.00, 296.01, 296.02, 296.03), *834.0, 944.01* |
| $r_2$ | (294.10, 295.04, 296.00, 296.01, 296.02, 296.03), (401.0, 404.00), *944.01, 692.71, 695.10* |
| $r_3$ | (294.10, 295.04, 296.00, 296.01, 296.02, 296.03), *692.71, 834.0, 695.10* |
| $r_4$ | (294.10, 295.04, 296.00, 296.01, 296.02, 296.03), (401.0, 404.00), *692.71* |
| $r_5$ | (294.10, 295.04, 296.00, 296.01, 296.02, 296.03), *692.71, 695.10* |
| $r_6$ | (294.10, 295.04, 296.00, 296.01, 296.02, 296.03), (401.0, 404.00), *480.1* |
| $r_7$ | (294.10, 295.04, 296.00, 296.01, 296.02, 296.03), *834.0, 944.01* |
| $r_8$ | (294.10, 295.04, 296.00, 296.01, 296.02, 296.03), *480.1* |
| $r_9$ | (294.10, 295.04, 296.00, 296.01, 296.02, 296.03), (401.0, 404.00) |
| $r_{10}$ | (294.10, 295.04, 296.00, 296.01, 296.02, 296.03), *834.0, 944.01* |

(a) Anonymized dataset $D^A$ produced by *CBA* (suppressed codes appear in gray).

| ID | Utility constraints |
|---|---|
| $u_1$ | {294.10, 295.04, 296.00, 296.01, 296.02, 296.03} |
| $u_2$ | {692.71, 695.10} |
| $u_3$ | {401.0, 404.00} |
| $u_4$ | {480.1} |
| $u_5$ | {834.0, 944.01} |

(b) Utility constraints.

**Fig. 3.** *CBA* example.

diagnosis codes that appear few times in the original dataset, by scattering them in the subrecords of the published dataset. For instance, consider the record $r_1$ in Fig. 1 and its counterpart, which has produced by applying disassociation with $k = 3$ and $m = 2$, in Fig. 4. Note that the codes in $r_1$ are split into two subrecords in Fig. 4, which contain the sets of codes $\{296.00, 296.01, 296.02\}$ and $\{834.0, 401.0, 944.01\}$, respectively. Moreover, the set $\{834.0, 401.0, 944.01\}$ is associated with the first 5 records in Fig. 4. Thus, an attacker who knows that a patient is diagnosed with the set of codes $\{296.01, 834.00\}$ cannot associate them with fewer than 3 records in Fig. 4. Thus, strong privacy requirements can be specified, without knowledge of potentially identifying diagnosis codes, and they can be enforced with low information loss. In addition, published data can still remain useful for intended analytic tasks. For instance, as can be seen in Fig. 4, applying our algorithm to the data in Fig. 1, using $k = 3$ and $m = 2$, achieves the same privacy, but significantly better data utility, than CBA, whose result is shown in Fig. 3(a). This is because, in contrast to CBA, our algorithm does not suppress diagnosis codes and retains the exact counts of 8 out of 13 codes (i.e., those in $u_1$ and $u_3$). Moreover, our algorithm is able to preserve the findings of the first two studies in Fig. 3(b).

Second, we experimentally demonstrate that our approach preserves data utility significantly better than CBA [35]. Specifically, when applied to a large EHR dataset [8], our approach allows more accurate query answering and generates data that are highly useful for supporting various types of clinical case count studies and general medical analysis tasks. In addition, our approach is more efficient and scalable than CBA.

### 1.3. Paper organization

The remainder of the paper is organized as follows. Section 2 reviews related work and Section 3 presents the concepts that are necessary to introduce our method and formulate the problem we consider. In Sections 4 and 6, we discuss and experimentally evaluate our algorithm, respectively. Subsequently, we explain how our approach can be extended to deal with different types of medical data and privacy requirements in Section 7. Last, Section 8 concludes the paper.

## 2. Related work

There are considerable research efforts for designing privacy-preserving methods [51,48,56,22,10,23,24,50,19,43,36]. Our work is closely related to methods which aim to publish patient-level data, in a way that prevents identity disclosure. Thus, we review these methods, in Section 2.1. We also discuss *differential privacy*,

Cluster $P_1$ $|P_1| = 5$

| | Record chunks | | Item chunk |
|---|---|---|---|
| | $C_1$ | $C_2$ | $C_T$ |
| $r_1$ | {296.00, 296.01, 296.02} | | |
| $r_2$ | {296.00, 296.01, 296.02} | {692.71, 695.10} | 834.0, 401.0, |
| $r_3$ | {296.00, 296.02} | {692.71, 695.10} | 944.01 |
| $r_4$ | {296.00, 296.01} | {692.71} | |
| $r_5$ | {296.00, 296.01, 296.02} | {692.71, 695.10} | |

Cluster $P_2$ $|P_2| = 5$

| | Record chunk | Item chunk |
|---|---|---|
| | $C_1$ | $C_T$ |
| $r_6$ | {296.03, 295.04} | |
| $r_7$ | {294.10, 296.03} | 404.00, |
| $r_8$ | {294.10, 295.04, 296.03} | 480.1, 834.0, 944.01 |
| $r_9$ | {294.10, 295.04} | |
| $r_{10}$ | {294.10, 295.04, 296.03} | |

**Fig. 4.** Anonymized dataset $D^A$ using our DISSASSOCIATION method. The dataset is comprised of two clusters, and each record is comprised of a number of subrecords, called *chunks*.

a privacy model that allows releasing noisy query results or noisy data summaries, in Section 2.2.

### 2.1. Preventing identity disclosure

The threat of identity disclosure in medical data publishing was firstly pointed out by Sweeney [50], and it has since attracted significant research interest [26,19,42,17,20,18,43]. Although other threats have been considered [41,40,58,39], "*all the publicly known examples of re-identification of personal information have involved identity disclosure*" [18].

The majority of works focus on preventing identity disclosure via relational data (i.e., data in which a patient is associated with a fixed, and typically small number of attributes), which naturally model patient demographics, and apply *generalization* [50,52,31] or *suppression* [50,52]. Different from this line of research, we consider data containing diagnosis codes, which require different handling than relational data, and apply disassociation, which generally incurs lower information loss than generalization and suppression.

Anonymizing diagnosis codes can be achieved by modeling them using a *transaction* attribute and enforcing a privacy model for transaction data [37,39,28,55,59,53,38]. The value in a transaction attribute is a set of *items* (itemset), which, in our case, corresponds to a patient's diagnosis codes. In [28], He et al. proposed a privacy model, called *complete k-anonymity*, and a generalization-based algorithm, called *Partition*. Terrovitis et al. [55] proposed a more flexible privacy model, called $k^m$-anonymity, and an algorithm, called *Apriori*. *Apriori* uses an effective way of generalizing values, referred to as *full-subtree*, *global generalization*, which was first proposed in [29]. A suppression-based algorithm for protecting identity disclosure was proposed in [59].

Loukides et al. [36] showed that the algorithms proposed in [28,55,59] are not suited to anonymizing diagnosis codes. This is because, they explore a small number of possible ways to anonymize diagnosis codes, and they are inadequate to generate data that support biomedical analysis tasks. In response, they proposed two algorithms; *Utility-Guided Anonymization of Clinical Profiles* (UGACLIP) [36] and *Clustering-Based Anonymizer* (CBA) [35]. Both algorithms apply generalization to certain sets of diagnosis codes and aim at preserving specific associations between diagnosis codes, which are modeled as *utility constraints*. However, CBA is more effective than UGACLIP in terms of supporting the specified associations and in terms of incurring low information loss. As discussed in Introduction, our approach is developed for a different data sharing scenario than that of [36,35], and it applies a different privacy model and data transformation technique.

### 2.2. Preserving privacy through differential privacy

Another privacy model, called *differential privacy* [16], has attracted significant attention [45,30,21] and has recently been applied to medical data [24]. Differential privacy ensures that the outcome of a calculation is insensitive to any particular record in the dataset. This offers privacy, because the inferences that can be made about an individual will be (approximately) independent of whether any individuals record is contained in the dataset or not. Differential privacy makes no assumptions about an attacker's background knowledge, unlike $k^m$-anonymity, although its enforcement does not guarantee the prevention of all attacks [12]. However, differential privacy allows either noisy answers to a limited number of queries, or noisy summary statistics to be released, and there are a number of limitations regarding its application on healthcare data [13]. In addition, differentially private data may be of much lower utility compared to $k^m$-anonymous data produced by disassociation, as shown in [54].

## 3. Background

In the previous sections, we highlighted how a patient can be identified by simply tracing records that contain unique combinations of diagnosis codes. Here, we present a concrete attack model and an effective data transformation operation, called *disassociation*. Disassociation can be used to guarantee patient privacy with respect to this model, while incurring minimal data utility loss. To quantify the loss of data utility caused by disassociation, we also discuss two measures that capture different requirements of medical data applications.

### 3.1. Attack model and privacy guarantee

We assume a dataset $D$ of records (transactions), each of which contains a set of diagnosis codes (items) from a finite domain $T$. The number of records in $D$ is denoted with $|D|$. Each record in $D$ refers to a different patient and contains the set of all diagnosis codes associated with them. An example of a dataset is shown in Fig. 1. Each record in this dataset contains some diagnosis codes, and the domain of diagnosis codes is shown in Fig. 2. In contrast to the traditional attack models for relational data [41,33], we do not distinguish between sensitive (unknown to the attacker) and non-sensitive items in a record. Instead, we assume that any item is a potential quasi-identifier and, hence, it may lead to identity disclosure. Besides the dataset $D$ we also assume a set of utility constraints $U$ [36], also referred to as *utility policy*. As discussed in Section 2, utility constraints model associations between

diagnosis codes that anonymized data are intended for. Each utility constraint $u$ in $U$ is a set of items from $T$, and all constraints in $U$ are disjoint. Fig. 3(b) illustrates an example of a set of utility constraints.

We now explain the attack model considered in this work. In this model, an attacker knows up to $m$ items of a record $r$ in $D$, where $m \geqslant 1$. The case of attackers with no background knowledge (i.e., $m = 0$) is trivial, and it is easy to see that the results of our theoretical analysis are applicable to this setting as well. Note that, different from the methods in [36,35], the items that may be exploited by attackers are considered unknown to data owners. Also, there may be multiple attackers, each of which knows a (not necessarily distinct) set of up to $m$ items of a record $r$. Other attacks and the ability of our method to thwart them are discussed in Section 7.

Based on their knowledge, an attacker can associate the identified patient with their record $r$, breaching privacy. To thwart this threat, our work employs the privacy model of $k^m$-*anonymity* [55]. $k^m$-anonymity is a conditional form of $k$-anonymity, which ensures that an attacker with partial knowledge of a record $r$, as explained above, will not be able to distinguish $r$ from $k-1$ other records in the published dataset. In other words, the probability that the attacker performs identity disclosure is upperbounded by $\frac{1}{k}$. More formally:

**Definition 1.** An anonymized dataset $D^A$ is $k^m$-anonymous if no attacker with background knowledge of up to $m$ items of a record $r$ in $D^A$ can use these items to identify fewer than $k$ candidate records in $D^A$.

For the original dataset $D$ and its anonymized counterpart $D^A$, we define two transformations $\mathcal{A}$ and $\mathcal{I}$. The anonymization transformation $\mathcal{A}$ takes as input a dataset $D$ and produces an anonymized dataset $D^A$. The inverse transformation $\mathcal{I}$ takes as input the anonymized dataset $D^A$ and outputs all possible (non-anonymized) datasets that could produce $D^A$, i.e., $\mathcal{I}(D^A) = \{D' \mid D^A = \mathcal{A}(D)\}$. Obviously, the original dataset $D$ is one of the datasets in $\mathcal{I}(\mathcal{A}(D))$. To achieve $k^m$-anonymity (Definition 1) in our setting, we enforce the following privacy guarantee (from [54]).

**Guarantee 1.** Consider an anonymized dataset $D^A$ and a set $\mathcal{S}$ of up to $m$ items. Applying $\mathcal{I}(D^A)$, will always produce at least one dataset $D' \in \mathcal{I}(D^A)$ for which there are at least $k$ records that contain all items in $\mathcal{S}$.

Intuitively, an attacker, who knows any set $\mathcal{S}$ of up to $m$ diagnosis codes about a patient, will have to consider at least $k$ candidate records in a possible original dataset. We provide a concrete example to illustrate this in the next subsection.

### 3.2. Overview of the disassociation transformation strategy

In this section, we present disassociation, a data transformation strategy that partitions the records in the original dataset $D$ into subrecords, following the basic principles of the strategy presented in [54]. The goal of our strategy is to "hide" combinations of diagnosis codes that appear few times in $D$, by scattering them in the subrecords of the published dataset. The particular merit of disassociation is that it preserves all original diagnosis codes in the published dataset, in contrast to generalization and suppression. This is important to preserve data utility in various medical analysis tasks that cannot be predetermined, as explained in the introduction and will be verified experimentally.

To illustrate the main idea of disassociation, we use Fig. 4, which shows a disassociated dataset produced from the original dataset $D$ of Fig. 1. Observe that the dataset in Fig. 4 is divided into two *clusters*, $P_1$ and $P_2$, which contain the records $r_1 - r_5$ and $r_6 - r_{10}$, respectively. Furthermore, the diagnosis codes in a cluster are divided into subsets, and each record in the cluster is split into subrecords according to these subsets. For example, the diagnosis codes in $P_1$ are divided into subsets $T_1 = \{296.00, 296.01, 296.02\}$, $T_2 = \{692.71, 695.10\}$, and $T_T = \{834.0, 401.0, 944.01\}$, according to which $r_1$ is split into three subrecords; $\{296.00, 296.01, 296.02\}$, an empty subrecord $\{\}$, and $\{834.0, 944.01\}$. The collection of all (possibly empty) subrecords of different records that correspond to the same subset of diagnosis codes is called a *chunk*. For instance, the subrecord $\{296.00, 296.01, 296.02\}$ of $r_1$ goes into chunk $C_1$, the empty subrecord goes into chunk $C_2$, and the subrecord $\{834.0, 944.01\}$ goes into chunk $C_T$. In contrast to $C_1$ and $C_2$ which are *record chunks*, $C_T$ is a special, *item chunk*, containing a single set of diagnosis codes. In our example, $C_T$ contains the set $\{834.0, 401.0, 944.01\}$, which represents the subrecords from all $r_1 - r_5$ containing these codes. Thus, the number of times each diagnosis code in $C_T$ appears in the original dataset is completely hidden from the attacker, who can only assume that this number ranges from 1 to $|P_i|$, where $|P_i|$ is the number of records in $P_i$.

In addition, the order of the subrecords that fall into a chunk is randomized, which implies that the association between subrecords in different chunks is hidden from the attacker. In fact, the original dataset $D$ may contain any record that could be *reconstructed* by a combination of subrecords from the different chunks plus *any subset of diagnosis codes* from $C_T$. For example, $\{296.00, 296.01, 834.0, 944.01\}$ in Fig. 5 is a reconstructed record, which is created by taking $\{296.00, 296.01\}$ from $C_1$, the empty subrecord $\{\}$ from $C_2$, and $\{834.0, 944.01\}$ from $C_T$. Observe that this record does not appear in the original dataset of Fig. 1. The disassociated dataset $D^A$ amounts to the set of all possible original datasets $\mathcal{I}(D^A)$ (see Guarantee 1). In other words, the original dataset $D$ is *hidden*, among all possible datasets that can be reconstructed from $D^A$. A dataset, which is reconstructed from the disassociated dataset in Fig. 4, is shown in Fig. 5. Note that reconstructed datasets can be greatly useful to data analysts, because (i) they have similar statistical properties to the original dataset from which they are produced, and (ii) they can be analyzed directly, using off-the-shelf tools (e.g., SPSS), in contrast to generalized datasets that require special handling (e.g., interpreting a generalized code as an original diagnosis code, with a certain probability).

As an example, consider the dataset in Fig. 4, which satisfies Guarantee 1, for $k = 3$ and $m = 2$. Observe that an attacker, who knows up to $m = 2$ codes from a record $r$ of the original dataset in Fig. 1, must consider a reconstructed dataset that has at least 3 records containing the codes known to them. We emphasize that each of these codes can appear in any chunk of a cluster in $D^A$, including the item chunk. For instance, an attacker, who knows that the record of a patient contains 296.01 and 834.0, must consider the dataset in Fig. 5. In this dataset, the combination of these codes appears in the records $r_1, r_2$, and $r_3$.

| ID | Records |
|---|---|
| $r_1$ | {296.00, 296.01, 834.0, 944.01} |
| $r_2$ | {296.02, 296.01, 692.71, 834.0} |
| $r_3$ | {296.00, 296.01, 296.02, 692.71, 695.10, 834.0} |
| $r_4$ | {296.00, 296.02, 692.71, 695.10} |
| $r_5$ | {296.00, 296.01, 296.02, 692.71, 695.10, 401.0} |
| $r_6$ | {296.02, 295.04, 480.1} |
| $r_7$ | {294.10, 296.02, 404.00, 834.0, 944.01} |
| $r_8$ | {294.10, 295.04, 296.02, 480.1, 834.0} |
| $r_9$ | {294.10, 295.04, 404.00, 834.0} |
| $r_{10}$ | {294.10, 295.04, 296.02, 834.0, 944.01} |

**Fig. 5.** A possible dataset $D'$ reconstructed from $D^A$ of Fig. 4.

### 3.3. Measuring data utility

Different datasets that can be produced by an original dataset, using disassociation, do not offer the same utility. In addition, most existing measures for anonymized data using generalization and/or suppression, such as those proposed in [55,59,36,35], are not applicable to disassociated datasets. Therefore, we measure data utility using the accuracy of: (i) answering COUNT queries on disassociated data, and (ii) estimating the number of records that are associated with any set of diagnosis codes in a utility constraint (i.e., *matched* to the constraint). The first way to measure data utility considers a scenario in which data recipients issue queries to perform *case counting* (i.e., discover the number of patients diagnosed with a set of one or more diagnosis codes, using COUNT queries). Alike other transformation strategies, disassociation may degrade the accuracy of answering COUNT queries [36,54]. Thus, a utility measure must capture how accurately such queries can be answered using disassociated data. The second way to quantify data utility considers a scenario in which various analytic tasks, simulated through different utility policies, are performed by data recipients. To the best of our knowledge, there are no measures that can capture data utility in this scenario.

To quantify the accuracy of answering a workload of COUNT queries on disassociated data, we use the *Average Relative Error* (ARE) measure and queries of the following SQL-like form:

```
SELECT  COUNT r' (or r)
FROM    D' (or D)
WHERE   P is supported by r' in D' (or P supports r in D)
```

where $P'$ and $P$ are sets of diagnosis codes, in the anonymized dataset $D'$ and in the original dataset $D$, respectively. These sets retrieve sets of a fixed number of diagnosis codes. These queries are used by several prior works on data anonymization (e.g., [32,57,39,36,35,54]), and they are important, because they form the basis for more complex queries and various analytic tasks (e.g., frequent itemset mining and classification). ARE is a standard data utility indicator [36,35,54], which reflects the average number of transactions that are retrieved incorrectly as part of query answers. The following definition explains how ARE can be computed.

**Definition 2.** Let $\mathcal{W}$ be a workload of COUNT queries $q_1, \ldots, q_n$, and $C_A$ and $C_O$ be functions which count the number of records answering a query $q_i, i \in [1, n]$ on the anonymized dataset $D'$ and on the original dataset $D$, respectively. The ARE measure for $\mathcal{W}$ is computed as

$$ARE(\mathcal{W}) = avg_{\forall i \in [1,n]} \frac{| C_A(q_i) - C_O(q_i) |}{C_O(q_i)}$$

Thus, ARE is computed as the mean error of answering all queries in the query workload $\mathcal{W}$. Clearly, a zero ARE implies that the anonymized dataset $D'$ are as useful as the original dataset in answering the queries in $\mathcal{W}$, and low scores in ARE are preferred.

To capture data utility in the presence of specified utility policies, we propose a new measure, called *Matching Relative Error* (MRE). The computation of MRE is illustrated in the following definition.

**Definition 3.** Let $u$ be a utility constraint in $U$, and $M_A$ and $M_O$ be functions, which return the number of records that match $u$ in the anonymized dataset $D'$ and in the original dataset $D$, respectively. The MRE for $u$ is computed as

$$MRE(u) = \frac{M_O(u) - M_A(u)}{M_O(u)}$$

Thus, a zero MRE implies that an anonymized dataset can support $u$ as well as the original dataset does, and MRE scores close to zero are preferred. For clarity, we report MRE as a percentage (i.e., the *percent error*). For example, an MRE in the interval $[-5\%, 5\%]$ implies that the number of transactions that match the utility constraint in the anonymized dataset is no more than 5% different (larger or smaller) than the corresponding number in the original dataset.

## 4. Disassociation algorithm

This section presents our disassociation-based algorithm for anonymizing diagnosis codes, which is referred to as DISASSOCIATION. This algorithm performs three operations: (i) *horizontal partitioning*, (ii) *vertical partitioning*, and (iii) *refining*. Horizontal partitioning brings together similar records with respect to diagnosis codes into clusters. As will be explained, performing this operation is important to preserve privacy with low utility loss. Subsequently, the algorithm performs vertical partitioning. This operation, which is the heart of our method, disassociates combinations of diagnosis codes that require protection and creates chunks. DISASSOCIATION differs from the method of [54] in that it aims at producing data that satisfy utility constraints and hence remain useful in medical analysis. Specifically, the horizontal and vertical partitioning phases in our algorithm treat codes that are contained in utility constraints as first-class citizens, so that they are preserved in the published dataset to the largest possible extent. Last, our algorithm performs the refining operation, to further reduce information loss and improve the utility of the disassociated data, A high-level pseudocode of DISASSOCIATION is given in Fig. 6. In addition, Fig. 7 summarizes the notation used in our algorithm and in the algorithms that perform its operations.

In the following, we present the details of the horizontal partitioning, vertical partitioning, and refining operations of our algorithm.

**Horizontal partitioning.** This operation groups records of the original dataset $D$ into disjoint *clusters*, according to the similarity

**Algorithm**: DISASSOCIATION
**Input** : Original dataset $D$,
parameters $k$ and $m$
**Output** : Disassociated dataset $D^A$

1 Split $D$ into disjoint clusters by applying Algorithm HORPART;
2 **for** *every cluster* $P$ *produced* **do**
3 Split $P$ vertically into chunks by applying Algorithm VERPART;
4 Refine clusters;
5 **return** $D^A$;

**Fig. 6.** DISASSOCIATION algorithm.

| Symbol | Explanation |
|---|---|
| $D, D^A$ | Original, anonymized dataset |
| $T$ | The set of all diagnosis codes in $D$ |
| $U$ | Set of utility constraints |
| $T_U$ | The set of all diagnosis codes in $U$ |
| $s(\mathsf{a})$ | Support of diagnosis code $\mathsf{a}$ |
| $P, P_1 \ldots$ | Clusters |
| $T^P$ | Domain of cluster |
| $C, C_1, \ldots$ | Record chunks |
| $T_1, T_2, \ldots$ | Domain of record chunk |
| $C_T$ | Item chunk |
| $T_T$ | Domain of item chunk |

**Fig. 7.** Notation used in our DISASSOCIATION algorithm and in the algorithms HORPART and VERPART.

of diagnosis codes. For instance, cluster $P_1$ is formed by records $r_1 - r_5$, which have many codes in common, as can be seen in Fig. 4. The creation of clusters is performed with a light-weight, but very effective heuristic, called HorPart. The pseudocode of HorPart is provided in Fig. 8. This heuristic aims at creating coherent clusters, whose records will require the least possible disassociation, during vertical partitioning.

To achieve this, the key idea is to split the dataset into two parts, $D_1$ and $D_2$, according to: (i) the *support* of diagnosis codes in $D$ (the support of a diagnosis code a, denoted with $s$(a), is the number of records in $D$ in which a appears), and (ii) the participation of diagnosis codes in the utility policy $U$. At each step, $D_1$ contains all records with the diagnosis code a, whereas $D_2$ contains the remaining records. This procedure is applied recursively, to each of the constructed parts, until they are small enough to become clusters. Diagnosis codes that have been previously used for partitioning are recorded in a set *ignore* and are not used again.

In each recursive call, Algorithm HorPart selects a diagnosis code a, in lines 3–10. In the first call, a is selected as the most frequent code (i.e., the code with the largest support), which is contained in a utility constraint. At each subsequent call, a is selected as the most frequent code, among the codes contained in $u$ (i.e., the utility constraint with the code chosen in the previous call) (line 4). When all diagnosis codes in $u$ have been considered, a is selected as the most frequent code in the set $\{T - ignore\}$, which is also contained in a utility constraint (line 6). Of course, if no diagnosis code is contained in a utility constraint, we simply select a as the most frequent diagnosis code (line 9).

Horizontal partitioning reduces the task of anonymizing the original dataset to the anonymization of small and independent clusters. We opted for this simple heuristic, because it achieves a good trade-off between data utility and efficiency, as shown in our experiments. However, we note that any other algorithm for creating groups of at least $k$ records could be used instead.

**Vertical partitioning.** This operation partitions the clusters into chunks, using a greedy heuristic that is applied to each cluster independently. The intuition behind the operation of this heuristic, called VerPartz, is twofold. First, the algorithm tries to distribute infrequent combinations of codes into different chunks to preserve privacy, as in [54]. Second, it aims at satisfying the utility constraints, in which the diagnosis codes in the cluster are contained. To achieve this, the algorithm attempts to create record chunks, which contain as many diagnosis codes from the same utility constraint as possible. Clearly, creating a record chunk that contains all the diagnosis codes of one or more utility constraints is beneficial, as tasks involving these codes (e.g., clinical case count studies) can be performed as accurately as in the original dataset.

The pseudocode of VerPart is provided in Fig. 9. This algorithm takes as input a cluster $P$, along with the parameters $k$ and $m$, and returns a set of $k^m$-anonymous record chunks $C_1, \ldots, C_v$, and the item chunk $C_T$ of $P$. Given the set of diagnosis codes $T^P$ in $P$, VerPart computes the support $s(t)$ of every code $t$ in $P$ and moves all diagnosis codes having lower support than $k$ from $T^P$ to a set $T_T$ (lines 2–4). As the remaining codes have support at least $k$, they will participate in some record chunk. Next, it orders $T^P$ according to: (i) $s(t)$, and (ii) the participation of the codes in utility constraints (line 5). Specifically, the diagnosis codes in $P$ that belong to the same constraint $u$ in $U$ form groups, which are ordered two times; first in decreasing $s(t)$, and then in decreasing $s(t)$ of their first (most frequent) diagnosis code.

Subsequently, VerPart computes the sets $T_1, \ldots, T_v$ (lines 6–20). To this end, the set $T_{remain}$, which contains the ordered, non-assigned codes, and the set $T_{cur}$, which contains the codes that will be assigned to the current set, are used. To compute $T_i (1 \leqslant i \leqslant v)$, VerPart considers all diagnosis codes in $T_{remain}$ and inserts a code $t$ into $T_{cur}$, only if the $C_{test}$ chunk, constructed from $T_{cur} \cup \{t\}$, remains $k^m$-anonymous (line 13). Note that the first execution of the for loop in line 10, will always add $t$ into $T_{cur}$, since $C_{test} = \{t\}$ is $k^m$-anonymous. If the insertion of $t$ to $T_{cur}$ does not render $T_{cur} \cup \{t\}$ $k^m$-anonymous, $t$ is skipped and the algorithm considers the next code. While assigning codes from $T_{remain}$ to $T_{cur}$, VerPart also tracks the utility constraint that each code is contained in (line 14). Next, VerPart iterates over each code $t$ in $T_{cur}$ and removes it from $T_{cur}$, if two conditions are met: (i) $t$ is contained in a utility constraint $u$ that is different from the constraint of the first code assigned to $T_{cur}$, and (ii) all codes in $u$ have also been assigned to $T_{cur}$ (lines 16–17). Removing $t$ enables the algorithm to insert the code into another record chunk (along with the remaining codes

**Algorithm**: HorPart
**Input** : Dataset $D$,
set of diagnosis codes $ignore$ (initially empty),
a utility constraint $u \in U$ (initially empty)
**Output** : A HORizontal PARTitioning of $D$, i.e., a set of clusters
**Param.** : The maximum cluster size $maxClusterSize$

1 Let $T$ be the set of diagnosis codes in $D$, and $T_U$ be the set of diagnosis codes appearing in the utility constraints of $U$;
2 **if** $|D| < maxClusterSize$ **then return** $\{\{D\}\}$;
3 **if** $\{T - ignore\} \bigcap u \neq \{\}$ **then**
4 Find the most frequent diagnosis code a in $\{T - ignore\} \bigcap u$;
5 **else if** $\{T - ignore\} \bigcap T_U \neq \{\}$ **then**
6 Find the most frequent diagnosis code a in $\{T - ignore\} \bigcap T_U$;
7 $u \leftarrow$ the constraint a belongs to;
8 **else**
9 Find the most frequent diagnosis code a in $\{T - ignore\}$;
10 $u \leftarrow \{\}$;
11 $D_1 \leftarrow$ the set of all records of $D$ that have a;
12 $D_2 \leftarrow D - D_1$;
13 **return** HorPart$(D_1, ignore \cup$ a$, u) \cup$HorPart$(D_2, ignore, \{\})$

**Fig. 8.** HorPart algorithm.

**Algorithm**: VERPART
**Input** : A cluster $P$, parameters $k$ and $m$
**Output** : A $k^m$-anonymous VERtical PARTitioning of $P$

1 Let $T^P$ be the set of diagnosis codes of $P$, i.e., the domain of $P$;
2 **for** *every diagnosis code* $t \in T^P$ **do**
  `//`$s(t)$ `is the number of records of` $P$ `the code` $t$ `appears in`
3  Compute the support $s(t)$;
4 Move all diagnosis codes with $s(t) < k$ from $T_P$ into $T_T$; `//`$T_T$ is finalized
5 Identify the groups of diagnosis codes in $T^P$ that belong to the same utility constraint of $U$, sort the diagnosis codes of each group in decreasing $s(t)$, and then sort the groups in decreasing support of their first diagnosis code;
6 $i \leftarrow 0$;
7 $T_{remain} \leftarrow T^P$;
8 **while** $T_{remain} \neq \{\}$ **do**
9  $T_{cur} \leftarrow \{\}$;
10  **for** *every diagnosis code* $t \in T_{remain}$ **do**
11  Create a chunk $C_{test}$ by projecting the records of $P$ to $T_{cur} \cup \{t\}$ ;
12  **if** $C_{test}$ *is* $k^m$*-anonymous* **then**
13  $T_{cur} \leftarrow T_{cur} \cup \{t\}$;
14  keep track of the constraint in which $t$ is contained (if any);
15  **for** *every diagnosis code* $t \in T_{cur}$ **do**
16  **if** $t$ *belongs to a constraint* $u$*, which is different from the constraint of the first diagnosis code added to* $T_{cur}$ **and** *not all diagnosis codes of* $u$ *are added to* $T_{cur}$ **then**
17  $T_{cur} \leftarrow T_{cur} - \{t\}$;
18  $i \leftarrow i + 1$ ;
19  $T_i \leftarrow T_{cur}$;
20  $T_{remain} \leftarrow T_{remain} - T_{cur}$;
21 Create record chunks $C_1, \ldots, C_v$ by projecting to $T_1, \ldots, T_v$;
22 Create item chunk $C_T$ using $T_T$;
23 **return** $\{C_1, \ldots, C_v, C_T\}$

**Fig. 9.** VERPART algorithm.

of $u$) in a subsequent step. After that, VERPART assigns $T_{cur}$ to $T_i$, removes the diagnosis codes of $T_{cur}$ from $T_{remain}$, and continues to the next set $T_{i+1}$ (lines 18–20).

Last, the algorithm constructs and returns the set $\{C_1, \ldots, C_v, C_T\}$ (lines 21–23). This set consists of the record chunks $C_1, \ldots, C_v$, and the item chunk $C_T$, which are created in lines 21 and 22, respectively.

In the following, we clarify the intuition behind lines 15–17. When VERPART starts creating a chunk in lines 10–14, it uses codes that may belong to different constraints. This aims at reducing the total number of record chunks in each cluster, by assigning as many codes as possible to a chunk (even from different constraints). Recall that the more record chunks we have in each cluster, the more disassociated the resulting dataset will be, and this is something we should avoid.

Thus, when codes from more than one utility constraint can be added into the same chunk, then there is no need to split them and create one chunk per constraint. Consider, for example, that VERPART created a chunk in lines 10–14, and that the codes in the chunk appear in two different utility constraints $u_1$ and $u_2$. Without loss of generality, we assume that the codes of $u_1$ are inserted into the chunk before those of $u_2$. The fact that all codes of the same constraint will be checked before the codes of a different constraint is ensured, by the sorting in line 5 of VERPART. Since codes of $u_1$ are inserted into the cluster first, we know that, if a code of $u_1$ was not inserted at that point, then this is because it breaks the $k^m$-anonymity guarantee, due to a combination with codes of the same constraint, i.e., $u_1$. Hence, there is nothing better to be done, for the set of codes of $u_1$, in this case. However, for the remaining codes of $u_2$, the situation requires a different treatment. If there is a code of $u_2$ that was not included in the chunk, then this may have happened because the $k^m$-anonymity was violated by a combination of this code and a code of $u_1$. Still, it may be possible that all codes of $u_2$ can be included in another chunk of the cluster, without violating the privacy guarantee. For this reason, the algorithm removes all codes of $u_2$ that were inserted in the chunk of the current step (lines 15–17) and tries to include all of them together (along with the previously excluded code) into a subsequent chunk of the cluster. In any case, the maximum set of codes from $u_2$ that does not violate the $k^m$-anonymity is guaranteed to be added into a chunk, when the algorithm considers this set first in the creation of a chunk, as in the case of codes from $u_1$ we described before.

**Refining.** This operation focuses on further improving the utility of the disassociated dataset, while maintaining Guarantee 1. To this end, we examine the diagnosis codes that reside in the item chunk of each cluster. Consider, for example, Fig. 4. The item chunk of the cluster $P_1$ contains the diagnosis codes 834.0 and 944.01, because the support of these codes in $P_1$ is 2 (i.e., lower than $k = 3$). For similar reasons, these diagnosis codes are also contained in the item chunk of $P_2$. However, the support of these codes in both clusters $P_1$ and $P_2$ together is not small enough to violate privacy (i.e., the combination of 834.0 and 944.01 appears as many

| Record | | Item | Shared |
|---|---|---|---|
| $P_1$ cluster | | | |
| {296.00, 296.01, 296.02} | | | {834.0,944.01} |
| {296.00, 296.01, 296.02} | {692.71, 695.10} | 401.0 | {944.01} |
| {296.00, 296.02} | {692.71, 695.10} | | {834.0} |
| {296.00, 296.01} | {692.71} | | |
| {296.00, 296.01, 296.02} | {692.71, 695.10} | | |
| $P_2$ cluster | | | |
| {296.03, 295.04} | | | |
| {294.10, 296.03} | | 404.00, | {834.0,944.01} |
| {294.10, 295.04, 296.03} | | 480.1 | |
| {294.10, 295.04} | | | |
| {294.10, 295.04, 296.03} | | | {834.0,944.01} |

**Fig. 10.** Disassociation with a shared chunk.

times as the one of 296.03 and 294.10 which is in the record chunk of $P_2$).

To handle such situations, we introduce the notion of *joint clusters* by allowing different clusters to have common record chunks. Given a set $T^s$ of *refining codes* (e.g., 834.0 and 944.01 in the aforementioned example), which commonly appear in the item chunks of two or more clusters (e.g., $P_1$ and $P_2$), we can define a joint cluster by (i) constructing one or more *shared chunks* after projecting the original records of the initial clusters to $T^s$ and (ii) removing all diagnosis codes in $T^s$ from the item chunks of the initial clusters. Fig. 10 shows a joint cluster, created by combining the clusters $P_1$ and $P_2$ of Fig. 4, when $T^s = \{834.0, 944.01\}$.

Furthermore, large joint clusters can be built by combining smaller joint clusters. Note that the creation of shared chunks is performed similarly to the method of [54], but shared chunks are created by our VerPart algorithm, which also takes into account the utility constraints.

We now provide an analysis of the time complexity of our algorithm.

**Time complexity.** We first consider each operation of Disassociation separately. The worst-case time complexity of the horizontal partitioning operation is $O(|D|^2)$. This is because HorPart works similarly to the Quicksort algorithm, but instead of a pivot, it splits each partition by selecting the code a. Thus, in the worst case, HorPart performs $|D|$ splits and at each of them it re-orders $|D|$ records. The time complexity of vertical partitioning depends on the domain $T^P$ of the input cluster $P$, and not on the characteristics of the complete dataset. The most expensive operation of VerPart is to ensure that a chunk is $k^m$-anonymous, which requires examining $\binom{|T^P|}{m}$ combinations of diagnosis codes. Thus, VerPart takes $O(|T^P|!)$ time, where $T^P$ is small in practice, as we regulate the size of the clusters. Last, the complexity of the refining operation is $O(|D|^2)$. This is because, in the worst case, the number of passes over the clusters equals the number of the clusters in $D$. Thus, the behavior of Disassociation is dominated by that of HorPart, as the dataset size grows. Note that this analysis refers to a worst-case. In practice, our algorithm is as efficient as the method in [54], although it takes into account utility constraints.

## 5. Example of disassociation

This section presents a concrete example of applying Disassociation to the dataset $D$ of Fig. 1. The input parameters are $k = 3$ and $m = 2$, and that the *maxClusterSize* parameter of HorPart is set to 6.[4]

**Horizontal partitioning.** First, Disassociation performs the horizontal partitioning operation on the original dataset $D$, using the HorPart algorithm. The algorithm selects 296.00, which participates in constraint $u_1$ of Fig. 3(b) and has the largest support. It then splits $D$ into two parts, $D_1$ and $D_2$. $D_1$ consists of the records containing 296.00 (i.e., $r_1 - r_5$), whereas $D_2$ contains the remaining records $r_6 - r_{10}$. At this point, 296.00 is moved from the domain $T$ of $D_1$ into the set *ignore*, so that it will not be used in subsequent splits of $D_1$. Moreover, the next call of HorPart for $D_1$ (line 13) is performed with the utility constraint $u_1$ as input. Thus, HorPart tries to further partition $D_1$, using the codes of this constraint. On the contrary, an empty *ignore* set and no utility constraint are given as input to HorPart, when it is applied to $D_2$. As the size of both $D_1$ and $D_2$ is lower than *maxClusterSize* (condition in line 2 of 8), HorPart produces the dataset in Fig. 11. This dataset is comprised of the clusters $P_1$ and $P_2$, which amount to $D_1$ and $D_2$, respectively.

**Vertical partitioning.** Then, Disassociation performs vertical partitioning operation, by applying VerPart to each of the clusters $P_1$ and $P_2$. The latter algorithm computes the support of each code in $P_1$, and then moves 401.0, 834.0 and 944.01, from the cluster domain $T_P$ into the set $T_T$ (line 4 in VerPart). The codes are moved to $T_T$, which corresponds to the domain of the item chunk, because they have a lower support than $k = 3$. Thus, $T_P$ now contains {296.00, 296.01, 296.02, 692.71, 695.10}, and it is sorted according to the support of these codes in $P_1$ and their participation in a utility constraint of $U$. Specifically, for the utility constraints of Fig. 3(b), we distinguish two groups of codes in $T_P$; a group {296.00, 296.01, 296.02}, which contains the codes in $u_1$, and another group {692.71, 695.10} with the codes in $u_2$. Next, VerPart sorts the first group in descending order of the support of its codes. Thus, 296.00 is placed first and followed by 296.01 and 296.02. The second group is sorted similarly. After that, the two groups are sorted in descending order of the support of their first code. Thus, the final ordering of $T_P$ is {296.00, 296.01, 296.02, 692.71, 695.10}.

Subsequently, VerPart constructs the record chunks of $P_1$ (lines 10–14), as follows. First, it selects 296.00 and checks whether the set of projections of the records $r_1$-$r_5$ on this code is $3^2$-anonymous. This holds, as 296.00 appears in all records of $P_1$. Thus, VerPart places 296.00 into the set $T_{cur}$, which will later be used to define the record chunk $C_1$. Then, the algorithm selects 296.01 and checks whether the projections of all records $r_1 - r_5$ on {296.00, 296.01} are also $3^2$-anonymous. As this is true, 296.01 is moved to $T_{cur}$, and the same procedure is performed, for each of the codes 296.02, 692.71, and 695.10. When the projections of the records $r_1 - r_5$ are found to be $3^2$-anonymous, the corresponding code is added to $T_{cur}$. Otherwise, it is left in a set $T_{remain}$ to be used in a subsequent step. Notice that 296.02 and 692.71 are added into $T_{cur}$, but the code 695.10 is not. This is because the combination of codes 296.01 and 695.10 appears in only two records of $P_1$ (i.e., $r_2$ and $r_5$), hence,

[4] This parameter could be set to any value at least equal to the value of $k$. However, it is fixed to $2 \cdot k$, because we have observed that this leads to producing good clusters.

| ID | Records |
|---|---|
| $r_1$ | {296.00, 296.01, 296.02, 834.0, 944.01} |
| $r_2$ | {296.00, 296.02, 296.01, 401.0, 944.01, 692.71, 695.10} |
| $r_3$ | {296.00, 296.02, 692.71, 834.0, 695.10} |
| $r_4$ | {296.00, 296.01, 692.71, 401.0} |
| $r_5$ | {296.00, 296.01, 296.02, 692.71, 695.10} |
| **ID** | **Records** |
| $r_6$ | {296.03, 295.04, 404.00, 480.1} |
| $r_7$ | {294.10, 296.03, 834.0, 944.01} |
| $r_8$ | {294.10, 295.04, 296.03, 480.1} |
| $r_9$ | {294.10, 295.04, 404.00} |
| $r_{10}$ | {294.10, 295.04, 296.03, 834.0, 944.01} |

**Fig. 11.** Output of horizontal partitioning on $D$.

**Table 1**
Description of the dataset.

| Dataset | $\|D\|$ | Distinct codes | Max, Avg # codes/record |
|---|---|---|---|
| *INFORMS* | 58,302 | 631 | 43, 5.11 |

the projections of records $r_1 - r_5$ on $\{296.00, 296.01, 296.02, 692.71, 695.10\}$ are not $3^2$-anonymous.

After considering all codes in $T_P$, VERPART checks whether the codes of a constraint $u \in U$ are only partially added to $T_{cur}$. This is true for 692.71, which is separated from 695.10 of the same constraint $u_2$. Hence, 692.71 is moved from $T_{cur}$ back to $T_{remain}$ (line 17), so that it can be added to the chunk $C_2$ of $P_1$ along with 695.10. After that, the algorithm finalizes the chunk $C_1$, according to $T_{cur}$, empties the latter set, and proceeds to creating $C_2$. By following this procedure for the cluster $P_2$, VERPART constructs the dataset $D^A$ in Fig. 4.

**Refining.** During this operation, DISASSOCIATION constructs the shared chunks, which are shown in Fig. 10, as follows. It inspects the item chunks of $P_1$ and $P_2$ in Fig. 4, and it identifies that each of the codes 834.0 and 944.01 appears in two records of $P_1$, as well as in two records of $P_2$. Note that the actual supports of codes in item chunks are available to the algorithm after the vertical partitioning operation, although they are not evident from Fig. 4 (because they are completely hidden in the published dataset). Since the total support of 834.0 and 944.01 in both clusters is $2 + 2 = 4 > k = 3$, the algorithm reconstructs the projections of $r_1 - r_5$ and $r_6 - r_{10}$ on the item chunk domain of $P_1$ and $P_2$ respectively, and calls VERPART, which creates the shared chunk of Fig. 10.

## 6. Experimental evaluation

### 6.1. Experimental data and setup

We implemented all algorithms in C++ and applied them to the *INFORMS* dataset [8], whose characteristics are shown in Table 1. This dataset was used in INFORMS 2008 Data Mining contest, whose objective was to develop predictive methods for identifying high-risk patients, admitted for elective surgery. In our experiments, we retained the diagnosis code part of patient records only.

We evaluated the effectiveness of our DISASSOCIATION algorithm, referred to as *Dis*, by comparing to *CBA*, which employs generalization to prevent identity disclosure based on diagnosis codes. The default parameters were $k = 5$ and $m = 2$, and the hierarchies used in *CBA* were created as in [35]. All experiments ran on an Intel Xeon at 2.4 GHz with 12 GB of RAM.

To evaluate data utility, we employed the ARE and MRE measures, discussed in Section 3.3. For the computation of ARE, we used two different types of query workloads. The first workload type, referred to as $\mathcal{W}_1$, contains queries asking for sets of diagnosis codes that a certain percentage of all patients have. In other words, these queries retrieve *frequent itemsets* (i.e., sets of diagnosis codes that appear in at least a specified percentage of records (transactions), expressed using a *minimum support threshold*). Answering such queries accurately is crucial in various biomedical data analysis applications [35], since frequent itemsets serve as building blocks in several data mining models [32]. The second workload type we considered is referred to as $\mathcal{W}_2$ and contains 1000 queries, which retrieve sets of diagnosis codes, selected uniformly at random. These queries are important, because it may be difficult for data owners to predict many of the analytic tasks that will be applied to anonymized data by data recipients.

In addition, we evaluated MRE using three classes of utility policies: *hierarchy-based*, *similarity-based*, and *frequency-based*. The first two types of policies have been introduced in [35] and model semantic relationships between diagnosis codes. For hierarchy-based policies, these relationships are formed using the ICD hierarchy. Specifically, hierarchy-based utility policies are constructed by forming a different utility constraint for all 5-digit ICD codes that have a common ancestor (other than the root) in the ICD hierarchy. The common ancestor of these codes is a 3-digit ICD code, *Section*, or *Chapter*,[5] for the case of *level* 1, *level* 2, and *level* 3 hierarchy-based policies, respectively. For example, consider a utility constraint $u$ for *Schizophrenic disorders*. The 5-digit ICD codes in $u$ are of the form 295.*xy*, where $x = \{0, \ldots, 9\}$ and $y = \{0, \ldots, 5\}$, and they have the 3-digit ICD code 295 as their common ancestor. The utility constraint $u$ is shown in the first row of Table 2. By forming a different utility constraint, for each 3-digit ICD code in the hierarchy (e.g., 296, 297, etc.), we construct a *level* 1, hierarchy-based policy. Alternatively, the common ancestor of the codes in the utility constraint $u$ may be a *Section*. For example, $u$ is comprised of *Psychoses*, whose common ancestor is 295 – 299, in the second row of Table 2. In this case, $u$ will be contained in a *level* 2, hierarchy-based policy. In another case, the common ancestor for the codes in $u$ may be a *Chapter*. For example, $u$ may correspond to *Mental disorders* that have 290 – 319 as their common ancestor (see the last row of Table 2). In this case, $u$ is contained in a *level* 3, hierarchy-based policy.

The similarity-based utility policies are comprised from utility constraints that contain the same number of sibling 5-digit ICD codes in the hierarchy. Specifically, we considered similarity-based constraints containing 5, 10, 25, and 100 codes and refer to their associated utility policies as *sim* 5, 10, 25, and 100, respectively. Consider, for instance, a utility constraint that contains 5 sibling ICD codes; 295.00, 295.01, 295.02, 295.03, and 295.04. This constraint, as well as all other constraints that contain 5 sibling ICD codes (e.g., a utility constraint that contains 296.00, ..., 296.04), will be contained in a *sim* 5 utility policy. Last, we considered frequency-based utility policies that model *frequent itemsets*. We mined frequent itemsets using the *FP-Growth* algorithm [27], which was configured with a varying minimum support threshold in $\{0.625, 1.25, 2.5, 5\}$. Thus, the generated utility constraints contain sets of diagnosis codes that appear in at least $0.625\%, 1.25\%, 2.5\%$, and $5\%$ percent of records, respectively. The utility policies associated with such constraints are denoted with *sup* 0.625, 1.25, 2.5, and 5, respectively. Unless otherwise stated, we use *level 1*, *sim 10*, and *sup 0.625*, as the default hierarchy, similarity, and frequency based utility policy, respectively.

### 6.2. Feasibility of identity disclosure

The risk of performing identity disclosure was quantified by measuring the number of records that share a set of $m$ diagnosis

[5] Sections and Chapters are internal nodes in the ICD hierarchy, which model aggregate concepts http://www.cdc.gov/nchs/data/icd9/icdguide10.pdf.

**Table 2**
Examples of hierarchy-based utility constraints.

| Type | Codes in utility constraint |
|---|---|
| *level* 1 | {295.00, 295.01, . . . , 295.95} |
| *level* 2 | {295.00, 295.01, . . . , 295.95, 296.00, . . . , 299.91} |
| *level* 3 | {290.10, 295.00, 295.01, . . . , 295.95, 296.00, . . . , 299.91, . . . , 299.91, . . . , 319} |

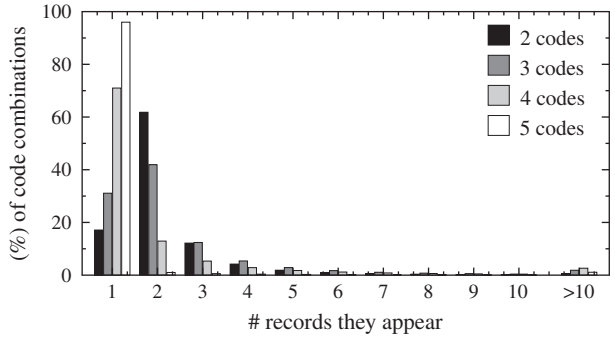


**Fig. 12.** Number of records in which a percentage of combinations, containing 2 to 5 diagnosis codes, appears.

codes. This number equals the inverse of the probability of performing identity disclosure, using these codes. Fig. 12 shows that more than 17% of all sets of 2 diagnosis codes appear in one record. Consequently, more than 17% of patients are uniquely re-identifiable, if the dataset is released intact. Furthermore, fewer than 5% of records contain a diagnosis code that appears at least 5 times. Thus, approximately 95% of records are *unsafe*, based on the probability threshold of 0.2 that is used typically [19]. Moreover, the number of times a set of diagnosis codes appears in the dataset increases with $m$. For example, 96% of sets containing 5 diagnosis codes appear only once. As we will see shortly, our algorithm can thwart attackers with such knowledge, by enforcing $k^m$-anonymity with $m = 5$, while preserving data utility.

### 6.3. Comparison with CBA

In this set of experiments, we demonstrate that our method can enforce $k^m$-anonymity, while allowing more accurate query answering than *CBA*. We also examine the impact of dataset size on the effectiveness and efficiency of both methods.

We first report ARE for query workloads of type $\mathcal{W}_1$ and for the following utility policies: *level* 1 (hierarchy-based), *sim* 10 (similarity-based), and *sup* 1.25 (frequency-based). For a fair comparison, the diagnosis codes retrieved by all queries are among those that are not suppressed by *CBA*. Fig. 13(a) illustrates the results for the *level* 1 policy. On average, the ARE scores for *Dis* and *CBA* are 0.055 and 0.155, respectively. This shows that the use of disassociation instead of generalization allows enforcing $k^m$-anonymity with low information loss. Figs. 13(b) and (c) show the corresponding results for the similarity-based and frequency-based policies, respectively. Again, our method outperformed *CBA*, achieving ARE scores that are substantially lower (better). Of note, the difference between the ARE scores for *Dis* and *CBA*, in each of the experiments in Figs. 13(a)–(c), was found to be statistically significant, according to Welch's $t$-test ($p < 0.01$). It is also worth noting that the difference of *Dis* and *CBA* with respect to ARE, increases as the utility policies get less stringent. For instance, the ARE scores for *Dis* and *CBA* are 0.129 and 0.163, respectively, for *level* 1, hierarchy-based policies, but 0.006 and 0.1, respectively, for *level* 3, hierarchy-based policies. This suggests that both algorithms perform reasonably well with respect to query answering, for restrictive constraints. However, *CBA* does so by suppressing a large number of diagnosis codes, as will be discussed later. Thus, the result of *CBA* is generally of lower utility (e.g., it does not allow queries on suppressed codes to be answered). Quantitatively similar results were obtained for query workloads of type $\mathcal{W}_2$ (omitted, for brevity).

Next, we report the number of distinct diagnosis codes that are suppressed when $k$ is set to 5, $m$ is 2 or 3, and the utility policies of the previous experiment are used. The results in Fig. 14 show that *CBA* suppressed a relatively large number of diagnosis codes, particularly when strong privacy is required and the utility constraints are stringent. For instance, 23.6% (i.e., 149 out of 631) of distinct diagnoses codes were suppressed, when $m = 3$ and the *level* 1 utility policy was used. On the contrary, our method released all diagnoses codes intact, as it does not employ suppression. This is particularly useful for medical studies (e.g., in epidemiology), where a large number of codes are of interest.

Then, we examined the effect of dataset size on ARE, by applying *Dis* and *CBA* on increasingly larger subsets of the dataset. The smallest and largest of these subsets contain the first 2.5$K$ and 25$K$ records of the dataset, respectively. In addition, we used query workloads of type $\mathcal{W}_2$ and the *sup* 1.25 utility policy. The results in Fig. 15(a) show that both algorithms incurred more information loss to anonymize smaller datasets. This is expected, because all datasets contain at least 79% of the diagnosis codes of the entire dataset, and many sets of diagnosis codes have a lower support than $k$. The ARE scores for *Dis* were always low, and substantially lower than those for *CBA*, for smaller datasets. For example, for the smallest dataset, the ARE scores for *Dis* and *CBA* were 0.95 and 11.57. The difference between the ARE scores for *Dis* and *CBA*, in all the results in Fig. 15(a), was found to be statistically significant, according to Welch's $t$-test ($p < 0.01$). Furthermore, as shown in Fig. 15(b), *CBA* suppressed a relatively large percentage of diagnosis codes, which decreased as the dataset size grows, for the reason explained before.

Last, we compared the runtime of *Dis* to that of *CBA*. We used the same parameters as in Fig. 15(b), and report the results in Fig. 15(c). As can be seen, both algorithms required less than 5 s. However, *Dis* is more efficient than *CBA*, and the performance gain increases with the dataset size. Specifically, *Dis* needed 1.2 s to anonymize the largest dataset, while *Dis* needed 4.9 s. In addition, the computation cost of *Dis* remained sub-quadratic, for all tested datasets.

Having established that our method outperforms *CBA*, we do not include results for *CBA* in the remainder of the section.

### 6.4. Supporting clinical case count studies

In the following, we demonstrate the effectiveness of our method at producing data that support clinical case count studies.

Fig. 16(a) illustrates the results for all three hierarchy-based policies and for query workloads of type $\mathcal{W}_2$. These workloads require retrieving a randomly selected set of 1 to 4 diagnosis codes. For consistency, we add a random code to a set of $c$ diagnosis codes to produce a larger set of $c + 1$ codes. For instance, a random code is added to a set of 1 diagnosis code to obtain a set containing 2 diagnosis codes. Observe that the error in query answering is fairly small and increases with the size of sets of diagnosis codes. This is because larger sets appear in few records and are more difficult to be made $k^m$-anonymous. Furthermore, low ARE scores are achieved, even for the *level 1* utility policy, which is difficult to satisfy using generalization. Similar observations can be made for other types of constraints, as can be seen in Figs. 16(b) and (c).

Fig. 17(a) shows the results, for hierarchy-based constraints and query workloads of type $\mathcal{W}_1$. The corresponding results for similarity-based and frequency-based constraints are reported in

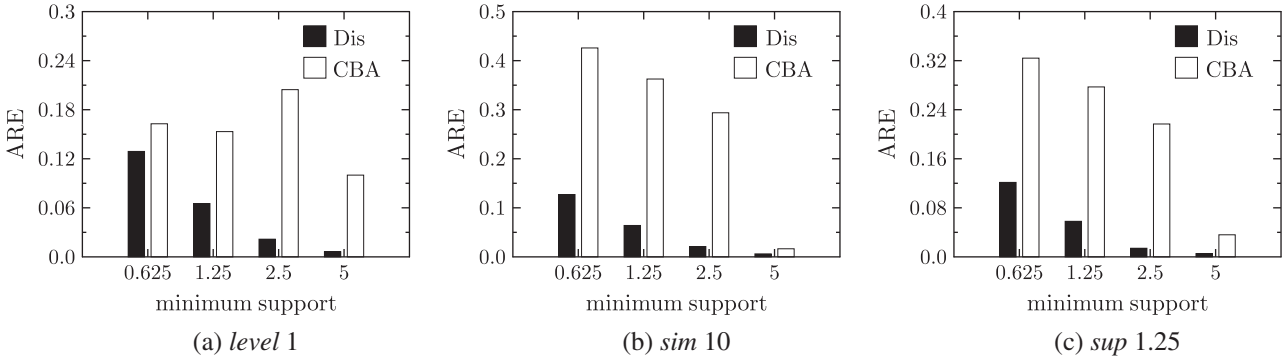


**Fig. 13.** Comparison with *CBA* with respect to ARE for query workloads of type $\mathcal{W}_1$ and for different utility policies.

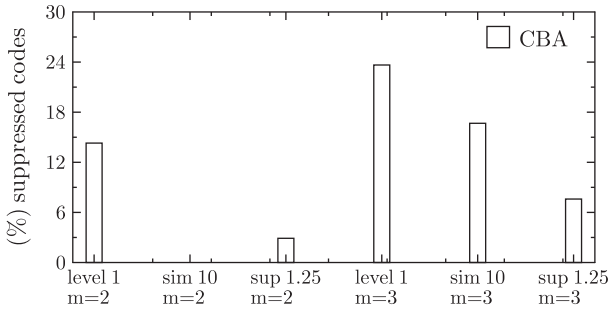


**Fig. 14.** Percentage of distinct diagnosis codes that are suppressed by *CBA* (no diagnosis codes are suppressed by our method, by design).

Figs. 17(b) and (c), respectively. Note that ARE scores are very low. In addition, queries involving more frequent sets of diagnosis codes can be answered highly accurately.

Next, we examined the impact of $k$ on ARE, by varying this parameter in $[5, 25]$, and considering the *level 1*, *sim* 10, and *sup* 2.5 utility policies. As can be seen in Fig. 18, ARE increases with $k$, as it is more difficult to retain associations between diagnosis codes, when clusters are large. However, the ARE scores are low, which indicates that our method permits accurate query answering.

### 6.5. Effectiveness in medical analytic tasks

In this set of experiments, we evaluate our method in terms of its effectiveness at supporting different utility policies. Given a utility policy, we measure MRE, for all constraints in the policy, and report the percentage of constraints, whose MRE falls into a certain interval. Recall from Section 6.1 that intervals whose endpoints are close to zero are preferred.

Fig. 19 reports the results, for the *level 1* utility policy. The MRE of all constraints in this policy is in $[-24\%, 5\%)$, while the MRE of the vast majority of constraints falls into much narrower intervals. Furthermore, the percentage of constraints with an MRE score close to zero is generally higher compared to those with MRE is far from zero. This confirms that the data produced by our method can support the intended analytic tasks, in addition to permitting accurate query answering.

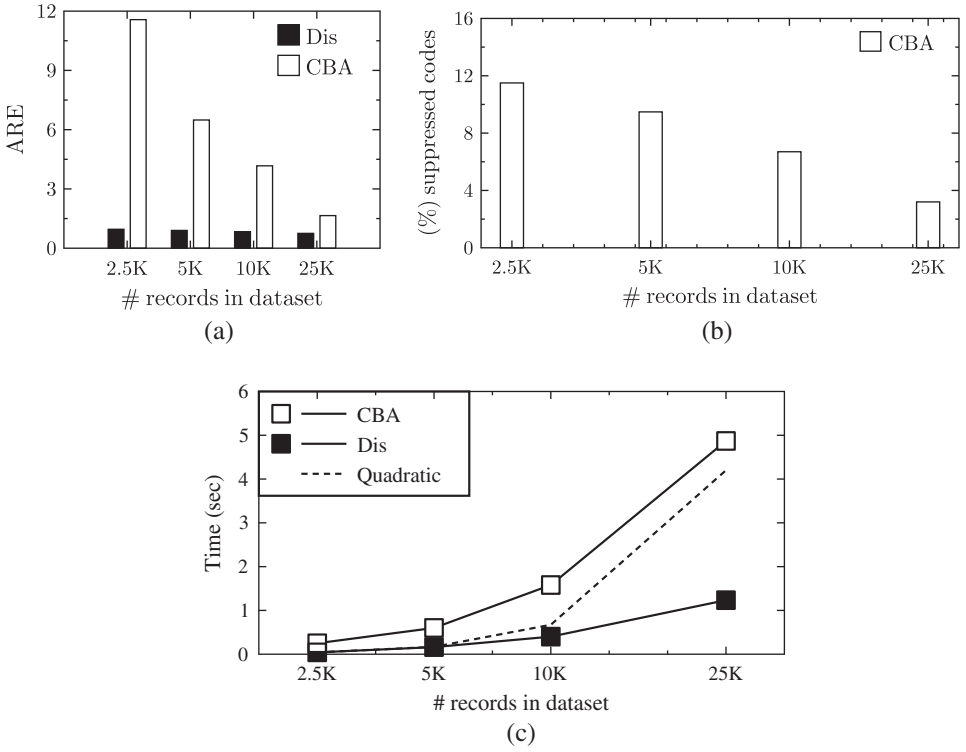


**Fig. 15.** Impact of dataset size on (a) ARE, (b) percentage of distinct codes that are suppressed by *CBA*, and (c) efficiency.

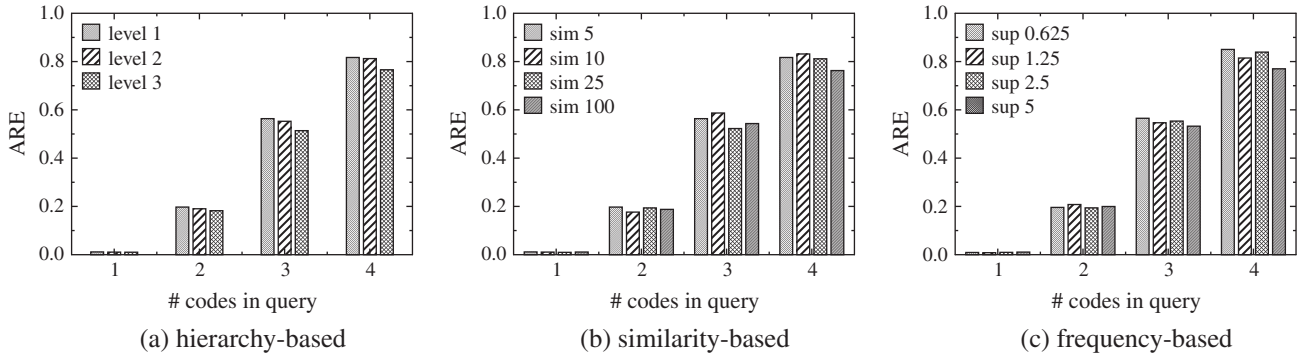


**Fig. 16.** ARE for query workloads of type $\mathcal{W}_2$ that retrieve 1 to 4 diagnosis codes and for different utility policies.

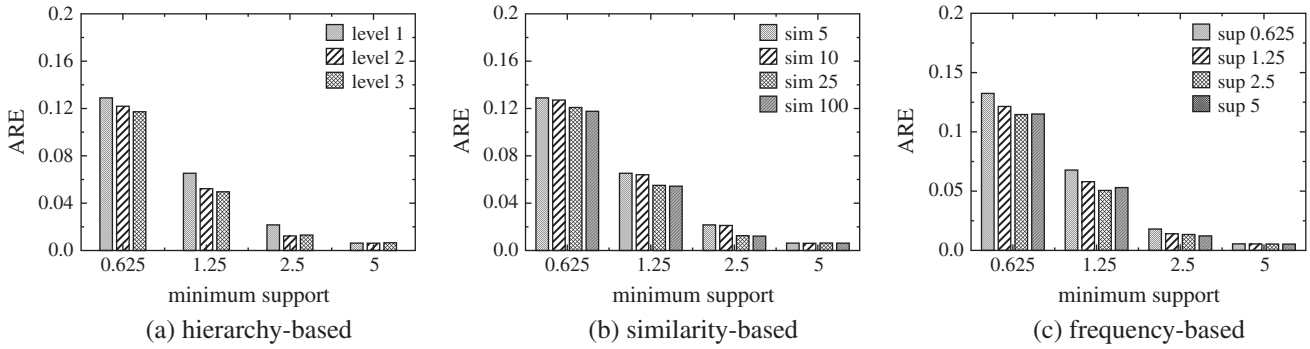


**Fig. 17.** ARE for query workloads of type $\mathcal{W}_1$ and for different utility policies.

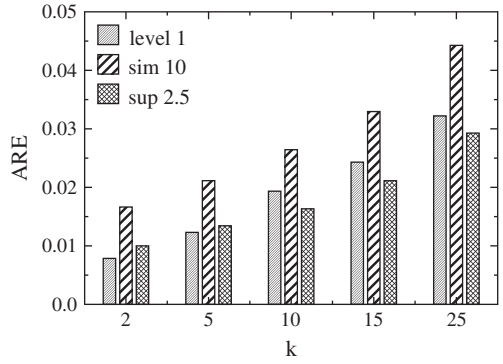


**Fig. 18.** ARE for varying $k$ in $[5, 25]$ and for different utility policies.

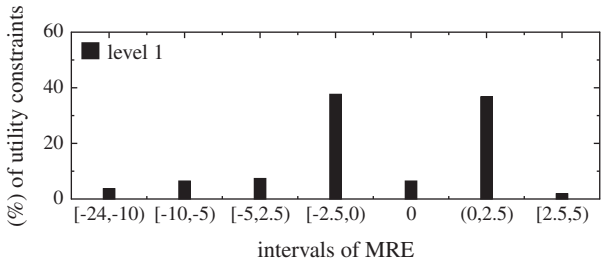


**Fig. 19.** MRE for *level 1* utility policy.

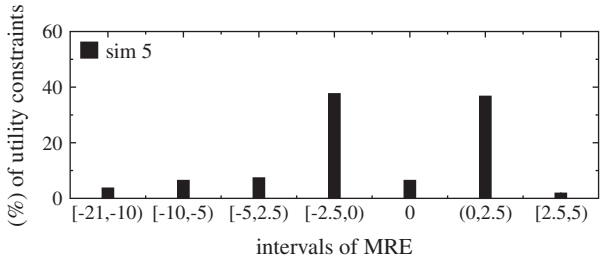


**Fig. 20.** MRE for the *sim 5* utility policy.

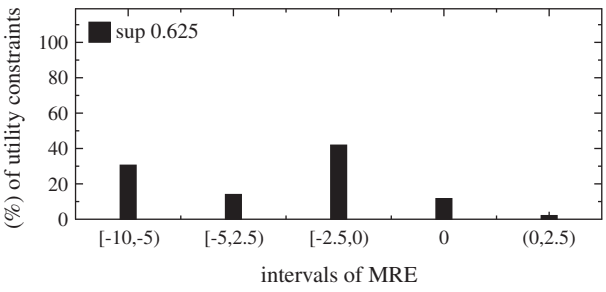


**Fig. 21.** MRE for the *sup 0.625* utility policy.

Next, we performed a similar experiment for similarity-based and frequency-based utility policies. The results for the *sim 5* policy are shown in Fig. 20. Note that 81% and 90% of the utility constraints in this policy have an MRE in $[-2.5\%, 2.5\%]$ and in $[-5\%, 5\%)$, respectively and only 3.6% of them have an MRE in $[-21\%, -10\%)$. The results for the *sup 0.625* utility policy are quantitatively similar, as can be seen in Fig. 21. These results together with those in Figs. 19 and 20 demonstrate the effectiveness of our method at supporting utility policies.

In addition, we examined the impact of $k$ on MRE, for different classes of utility policies. Figs. 22–24 illustrate the results for hierarchy-based, similarity-based, and frequency-based policies, respectively. It can be seen that, lowering $k$, helps the production of data that support the specified utility policies. For instance,

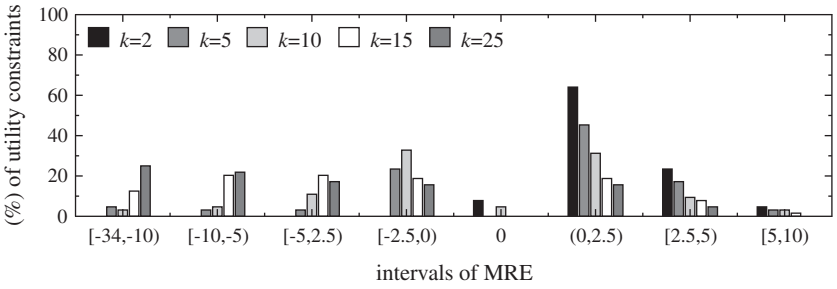


**Fig. 22.** MRE for hierarchy-based utility policies and for varying $k$ in $[2, 25]$.

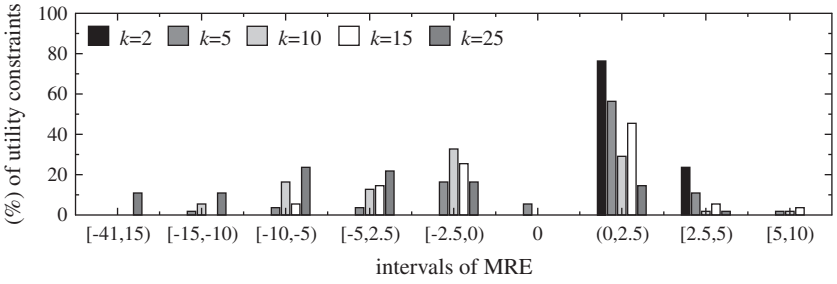


**Fig. 23.** MRE for similarity-based utility policies and for varying $k$ in $[2, 25]$.

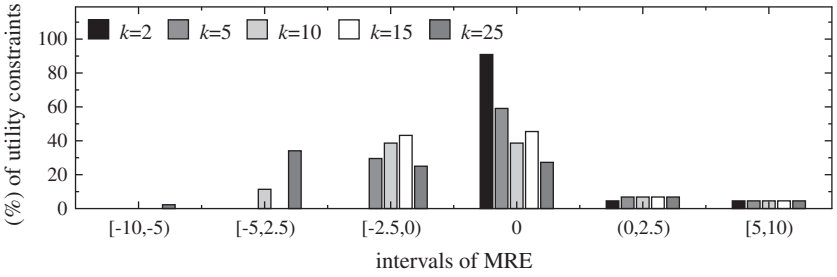


**Fig. 24.** MRE for frequency-based utility policies and for varying $k$ in $[2, 25]$.

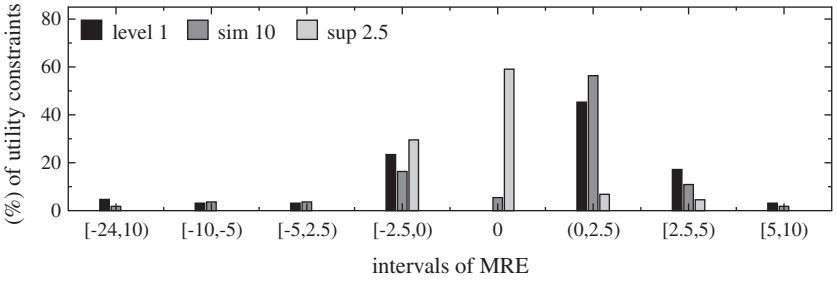


**Fig. 25.** MRE for different types of utility policies and for $m = 5$.

95.3% of hierarchy-based constraints have an MRE in $[-5\%, 5\%)$ when $k = 2$, but 53% of such constraints have an MRE in this interval when $k = 25$. This is expected due to the utility/privacy trade-off. However, the MRE of most of the constraints falls into $[-5\%, 5\%)$. Thus, our method is effective at supporting the intended medical analytic tasks.

Last, we investigated the effectiveness of our method, when $m = 5$. It is interesting to examine data utility in this setting, because a patient's record in discharge summaries, which may be used in identity disclosure attacks, often contains 5 diagnosis codes, which are assigned during a single hospital visit. Thus, enforcing $k^5$-anonymity provides protection from such attacks, assuming a worst case scenario in which data owners do not know which diagnoses codes may be used by attackers. In our experiments, we considered different classes of utility policies (namely, *level* 1, *sim* 10, and *sup* 2.5) and report the results in Fig. 25. Notice that the data produced by our method remain useful for supporting the utility policies, as 89%, 93%, and 100% of the tested hierarchy-based, similarity-based, and frequency-based constraints have an MRE in $[-5\%, 5\%)$, respectively.

## 7. Discussion

This section explains how our approach can be extended to deal with different types of medical data and privacy requirements. In addition, it discusses the limitations of our approach, which suggest opportunities for further research.

### 7.1. Dealing with data containing repeated diagnosis codes

Our work considers records comprised of a set of diagnosis codes, following [34,36,35]. However, some applications that aim at identifying phenotypes in the context of genetic association studies require data, in which a record contains repeated diagnosis codes (i.e., a multiset of diagnosis codes). Dealing with these applications is straightforward, as it requires a pre-processing in which different instances of the same diagnosis code in the dataset, and the utility constraints, are mapped to different values (e.g., the first occurrence of 250.01 is mapped to 250.011, the second to 250.012 etc.) [35].

### 7.2. Dealing with different privacy requirements

Our work focuses on preventing *identity disclosure* which is the most important privacy requirement in the healthcare domain. It ensures that an attacker with background knowledge of up to $m$ codes in a record cannot associate this record with fewer than $k$ candidate patients. However, the anonymization framework we propose is not restricted to guarding against attackers with only partial knowledge of the codes in a record in $D$. In fact, by setting $m$ to the maximum number of codes in a record of $D$, data owners can prevent attacks based on knowledge of all codes in a record. This is because the dataset that is produced by our method in this case satisfies Guarantee 1. Regardless of the specific values of $k$ and $m$, we do not consider collaborative attacks, where two or more attackers combine their knowledge in order to re-identify a patient nor attackers with background knowledge of multiple records in $D$. Such powerful attack schemes can only be handled within stronger privacy principles, such as differential privacy (see Section 2). However, applying these principles usually results in significantly lower utility, compared to the output of our method, which offers a reasonable tradeoff.

Furthermore, we do not assume any distinction between sensitive and non-sensitive diagnosis codes (see Section 3). Instead, we treat all codes as potentially identifying. However, when there is clear distinction between sensitive and non-sensitive codes in a record, i.e., data owners know that some codes (the sensitive ones) are not known to any attacker, then our framework allows thwarting *attribute disclosure* as well. An effective principle for preventing attribute disclosure is $\ell$-diversity [41]. Enforcing $\ell$-diversity using our framework is rather straightforward, as it simply requires (i) ignoring all sensitive codes during the horizontal partitioning operation, and (ii) placing all sensitive codes in the item chunk during vertical partitioning. This produces a dataset $D^A$, in which all sensitive codes are contained in the item chunks. This dataset limits the probability of any association between sensitive codes and any other subrecord or code to $\frac{1}{|P|}$, where $|P|$ is the size of the cluster. Clearly, the desired degree of $\ell$-diversity can be achieved in this case, by adjusting the size of the clusters.

In general, protection from attribute disclosure within our framework tends to incur higher information loss than simply protecting from identity disclosure. This is because sensitive codes are not necessarily infrequent, i.e., they may appear more than $k$ times in a cluster. Thus, the frequent sensitive codes that would be placed in a $k^m$-anonymous record chunk, when only identity disclosure is prevented, are now placed in the item chunk and each of them is completely disassociated from any other. In this case, the utility constraints that include sensitive codes are not preserved in the published dataset to the extent they would be preserved when only guarding against identity disclosure is required. Of course, this does not hold for the remaining utility constraints. The evaluation of our method with protection from both identity and attribute disclosure is left as future work.

### 7.3. Limitations

The proposed approach is limited in three main aspects. First, it considers data containing diagnosis codes. Some applications, however, require releasing data that contains both diagnosis codes and demographics. Anonymizing such data has been considered very recently by Poulis et al. [49] and by Mohammed et al. [46]. The method in [49] employs generalization and is not directly applicable to publishing patient information, while the method in [46] employs differential privacy and releases noisy summary statistics. Extending our approach, so that it can deal with such data is interesting but very challenging. This is because: (i) minimizing the information loss for both demographics and diagnosis codes is computationally infeasible, and (ii) existing optimization strategies do not achieve a "good" trade-off between the information loss in these two attribute types. For example, as proved in [49], to construct a dataset with the minimum information loss in demographics, we need to apply generalization to "small" groups of records independently, but we must apply generalization to all records in the dataset, to minimize information loss in diagnosis codes. The reason is that demographics and diagnosis codes have different semantics; a patient is associated with a fixed, typically small, number of demographics, but with a large number of diagnosis codes, that is not the same for all patients.

Second, as is true of all data anonymization methods, our approach assumes that data owners are able to select appropriate values for $k$ and $m$. Configuring these parameters to find the "best" trade-off between data utility and privacy is, however, not straightforward. For example, if the dataset is "too" small and contains a "large" number of "rare" diagnosis codes, applying $k^m$-anonymity with "large" $k$ and $m$ values may incur high information loss. It is therefore important to develop tools that help data owners in assessing data utility and privacy, so as to achieve a "good" utility/privacy trade-off.

Third, although our approach improves upon existing work in terms of minimizing information loss, it does not guarantee that the information loss will be bounded from the optimal. The design of approximation algorithms that offer such guarantees is an important open problem, which is challenging due to its computational hardness [37].

## 8. Conclusions

Ensuring that diagnosis codes cannot be used in identity disclosure attacks is necessary but challenging, particularly when data need to be shared broadly and to support a range of medical analytic tasks that may not be determined prior to data dissemination. To this end, we proposed a novel, disassociation-based approach that enforces $k^m$-anonymity with low information loss. Our approach does not require data owners to specify diagnosis codes, as existing methods do, and takes into account analytic tasks that published data are intended for. Extensive experiments using EHR data show that our approach can produce data that permit various types of clinical case count studies and general medical analysis tasks to be performed accurately.

## References


[1] National Institutes of Health. Policy for sharing of data obtained in NIH supported or conducted genome-wide association studies. NOT-OD-07-088; 2007.
[2] The Clinical Practice Research Datalink. <http://www.cprd.com/>.
[3] EU Directive 95/46/EC of the European Parliament and of the Council. <http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:en:HTML>.
[4] Health data initiative. Institute of Medicine and the U.S. Department of Health and Human Services (HHS). <http://www.hhs.gov/open/initiatives/hdi/>.
[5] The Secure Anonymised Information Linkage (SAIL) Databank. <http://www.ehi2.swansea.ac.uk/en/sail-databank.htm>.
[6] UK Anonymization Code. Information Commissioner's Office. <http://www.ico.org.uk/for_organisations/data_protection/topic_guides/anonymisation/>.
[7] Widening the use of Electronic Health Record data for research. National Center for Research Resources (US); 2009. <http://videocast.nih.gov/summary.asp?live=8062>.
[8] INFORMS Data Mining Contest; 2008. <https://sites.google.com/site/informsdataminingcontest/>.
[9] Atreya RV, Smith JC, McCoy AB, Malin B, Miller RA. Reducing patient re-identification risk for laboratory results within research datasets. J Am Med Inform Assoc 2013;20(1):95–101.
[10] Canim M, Kantarcioglu M, Malin B. Secure management of biomedical data with cryptographic hardware. IEEE Trans Inform Technol Biomed 2012;16(1):166–75.
[11] Cattell J, Chilukuri S, Levy M. How big data can revolutionize pharmaceutical R&D. McKinsey Company, Insights and Publications; 2013. <http://www.mckinsey.com/insights/health_systems_and_services>.
[12] Cormode G. Personal privacy vs. population privacy: learning to attack anonymization. In: KDD; 2011. p. 1253–61.
[13] Dankar FK, El Emam K. The application of differential privacy to health data. In: EDBT/ICDT workshops; 2012. p. 158–66.
[14] Denny JC. Chapter 13: mining electronic health records in the genomics era. PLoS Comput Biol 2012;8(12):e1002823. 12.
[15] U.S. Department of Health & Human Services. Breaches affecting 500 or more individuals; 2013. <http://www.hhs.gov/ocr/privacy/hipaa/administrative/breachnotificationrule/breachtool.html>.
[16] Dwork C. Differential privacy. In: ICALP; 2006. p. 1–12.
[17] Elliot M, Purdam K, Smith D. Statistical disclosure control architectures for patient records in biomedical information systems. J Biomed Inform 2008;41(1):58–64.
[18] El Emam K. Methods for the de-identification of electronic health records. Genome Med 2010;3(25).
[19] El Emam K, Dankar FK. Protecting privacy using k-anonymity. J Am Med Inform Assoc 2008;15(5):627–37.
[20] El Emam K, Kamal Dankar F, Issa R, Jonker E, Amyot D, Cogo E, et al. A globally optimal k-anonymity method for the de-identification of health data. J Am Med Inform Assoc 2009;16(5):670–82.
[21] Fan L, Xiong L, Sunderam VS. Fast: differentially private real-time aggregate monitor with filtering and adaptive sampling. In: SIGMOD; 2013. p. 1065–8.
[22] Fienberg S, Fulp W, Slavkovic A, Wrobel T. Secure log-linear and logistic regression analysis of distributed databases. In: Privacy in statistical databases; 2006. p. 277–90.
[23] Fung BCM, Wang K, Chen R, Yu PS. Privacy-preserving data publishing: a survey of recent developments. ACM Comput Surv 2010;42(4):14:1–14:53.
[24] Gardner JJ, Xiong L, Xiao Y, Gao J, Post AR, Jiang X, et al. Share: system design and case studies for statistical health information release. J Am Med Inform Assoc 2013;20(1):109–16.
[25] Ghinita G, Tao Y, Kalnis P. On the anonymization of sparse high-dimensional data. In: ICDE; 2008. p. 715–24.
[26] Gkoulalas-Divanis A, Loukides G. Anonymization of Electronic Medical Records to Support Clinical Analysis. Springer; 2013.
[27] Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation. SIGMOD Rec 2000;29(2):1–12.
[28] He Y, Naughton JF. Anonymization of set-valued data via top-down, local generalization. Proc VLDB Endow 2009;2(1):934–45.
[29] Iyengar VS. Transforming data to satisfy privacy constraints. In: KDD; 2002. p. 279–88.
[30] Kifer D, Machanavajjhala A. No free lunch in data privacy. In: SIGMOD; 2011. p. 193–204.
[31] LeFevre K, DeWitt DJ, Ramakrishnan R. Incognito: efficient full-domain k-anonymity. In: SIGMOD; 2005. p. 49–60.
[32] LeFevre K, DeWitt DJ, Ramakrishnan R. Workload-aware anonymization. In: KDD; 2006. p. 277–86.
[33] Li N, Li T, Venkatasubramanian S. t-closeness: Privacy beyond k-anonymity and l-diversity. In: ICDE; 2007. p. 106–15.
[34] Loukides G, Denny JC, Malin B. The disclosure of diagnosis codes can breach research participants' privacy. J Am Med Inform Assoc 2010;17:322–7.
[35] Loukides G, Gkoulalas-Divanis A. Utility-aware anonymization of diagnosis codes. IEEE J Biomed Health Inform 2013;17(1):60–70.
[36] Loukides G, Gkoulalas-Divanis A, Malin B. Anonymization of electronic medical records for validating genome-wide association studies. Proc Natl Acad Sci USA 2010;107:7898–903.
[37] Loukides G, Gkoulalas-Divanis A, Malin B. COAT: constraint-based anonymization of transactions. Knowl Inform Syst 2011;28(2):251–82.
[38] Loukides G, Gkoulalas-Divanis A, Shao J. Anonymizing transaction data to eliminate sensitive inferences. In: DEXA; 2010. p. 400–15.
[39] Loukides G, Gkoulalas-Divanis A, Shao J. Efficient and flexible anonymization of transaction data. Knowl Inform Syst 2013;36(1):153–210.
[40] Loukides G, Shao J. Capturing data usefulness and privacy protection in k-anonymisation. In: SAC; 2007. p. 370–4.
[41] Machanavajjhala A, Gehrke J, Kifer D, Venkitasubramaniam M. l-diversity: Privacy beyond k-anonymity. In: ICDE; 2006. p. 24.
[42] Malin B, Sweeney L. How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems. J Biomed Inform 2004;37(3):179–92.
[43] MartíNez Sergio, SáNchez David, Valls Aida. A semantic framework to protect the privacy of electronic health records with non-numerical attributes. J Biomed Inform 2013;46(2):294–303.
[44] Mailman MD MD, Feolo M M, Jin Y, et al. The ncbi dbgap database of genotypes and phenotypes. Nature Genet 2007;39:1181–6.
[45] Mohammed N, Chen R, Fung BCM, Yu PS. Differentially private data release for data mining. In: KDD; 2011. p. 493–501.
[46] Mohammed N, Jiang X, Chen R, Fung BCM, Ohno-Machado L. Privacy-preserving heterogeneous health data sharing. J Am Med Inform Assoc 2013;20:462–9.
[47] National Committee on Vital and Health Statistics. HIPAA code set rule. <http://www.ncvhs.hhs.gov/091210p06b.pdf>.
[48] Pinkas B. Cryptographic techniques for privacy-preserving data mining. ACM Special Interest Group Knowl Disc Data Min Explorat 2002;4(2):12–9.
[49] Poulis G, Loukides G, Gkoulalas-Divanis A, Skiadopoulos S. Anonymizing data with relational and transaction attributes. In: ECML/PKDD; 2013. p. 353–69.
[50] Samarati P. Protecting respondents identities in microdata release. IEEE Trans Knowl Data Eng 2001;13(9):1010–27.
[51] Sandhu RS, Coyne EJ, Feinstein HL, Youman CE. Role-based access control models. IEEE Comput 1996;29(2):38–47.
[52] Sweeney L. k-anonymity: a model for protecting privacy. Int J Uncertain Fuzziness Knowl-based Syst 2002;10:557–70.
[53] Tamersoy A, Loukides G, Denny JC, Malin B. Anonymization of administrative billing codes with repeated diagnoses through censoring. In: Proceedings of the 2010 AMIA annual symposium; 2010. p. 782–6.
[54] Terrovitis M, Liagouris J, Mamoulis N, Skiadopoulos S. Privacy preservation by disassociation. Proc VLDB Endow 2012;5(10):944–55.
[55] Terrovitis M, Mamoulis N, Kalnis P. Privacy-preserving anonymization of set-valued data. Proc VLDB Endow 2008;1(1):115–25.
[56] Wang S, Jiang X, Wu Y, Cui L, Cheng S, Ohno-Machado L. Expectation propagation logistic regression (explorer): distributed privacy-preserving online model learning. J Biomed Inform 2013;46(3):480–96.
[57] Xiao X, Tao Y. Anatomy: simple and effective privacy preservation. In: VLDB; 2006. p. 139–50.
[58] Xiao X, Tao Y. Personalized privacy preservation. In: SIGMOD; 2006. p. 229–40.
[59] Xu Y, Wang K, Fu AW-C, Yu PS. Anonymizing transaction databases for publication. In: KDD; 2008. p. 767–75.