Knowl Inf Syst (2013) 36:153–210
DOI 10.1007/s10115-012-0544-3

REGULAR PAPER

Efficient and flexible anonymization of transaction data

Grigorios Loukides · Aris Gkoulalas-Divanis · Jianhua Shao

Received: 22 September 2011 / Revised: 30 May 2012 / Accepted: 17 August 2012 /
Published online: 9 September 2012

Abstract Transaction data are increasingly used in applications, such as marketing research and biomedical studies. Publishing these data, however, may risk privacy breaches, as they often contain personal information about individuals. Approaches to anonymizing transaction data have been proposed recently, but they may produce excessively distorted and inadequately protected solutions. This is because these approaches do not consider privacy requirements that are common in real-world applications in a realistic and flexible manner, and attempt to safeguard the data only against either identity disclosure or sensitive information inference. In this paper, we propose a new approach that overcomes these limitations. We introduce a rule-based privacy model that allows data publishers to express fine-grained protection requirements for both identity and sensitive information disclosure. Based on this model, we also develop two anonymization algorithms. Our first algorithm works in a top-down fashion, employing an efficient strategy to recursively generalize data with low information loss. Our second algorithm uses sampling and a combination of top-down and bottom-up generalization heuristics, which greatly improves scalability while maintaining low information loss. Extensive experiments show that our algorithms significantly outperform the state-of-the-art in terms of retaining data utility, while achieving good protection and scalability.

Keywords Anonymity · Privacy · Transaction data · Privacy requirements · Identity disclosure · Sensitive information disclosure · Efficiency · Scalability

G. Loukides · J. Shao
School of Computer Science and Informatics, Cardiff University, Cardiff, UK
e-mail: g.loukides@cs.cf.ac.uk

J. Shao
e-mail: j.shao@cs.cf.ac.uk

A. Gkoulalas-Divanis (✉)
Smarter Cities Technology Centre, IBM Research-Ireland, Dublin, Ireland
e-mail: arisdiva@ie.ibm.com

1 Introduction

Transaction datasets that contain information about individuals' behaviors or activities are used in a growing number of applications, including personalized web search, marketing analysis, and biomedical studies. These datasets are comprised of records, called *transactions*, and each transaction is a set of items, such as the goods purchased by or the diagnoses given to an individual. One of the key issues concerning the use of transaction datasets in applications is that individuals' private information contained within the data must be protected.

1.1 Motivation

Publishing transaction data safely requires the elimination of two types of potential privacy leak, namely identity and sensitive information disclosure. *Identity disclosure* occurs when an individual is linked to their transaction in the published data. Unfortunately, simply de-identifying transactions (i.e., removing personal identifiers) is not enough to prevent this type of disclosure. This is because combinations of items in transactions can still identify individuals, as in the AOL incident that resulted in disclosing an individual's identity based on their search engine query terms [11]. *Sensitive information disclosure*, on the other hand, occurs when an individual is linked not to a transaction, but to a set of items that is considered to be sensitive. For example, from de-identified transactions containing movie ratings given by an internet user, one may link this user to certain ratings to infer their religious views or political beliefs [62]. These two types of disclosure are illustrated in the following example.

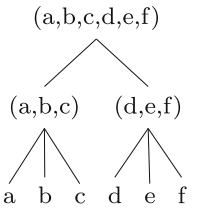
Example 1 Consider the de-identified data in Fig. 1a, which record items purchased by several individuals. Some items in these data are sensitive and denoted with bold letters. Knowing that *Mary* has purchased a, b and c, an attacker can link *Mary* to the first transaction of Fig. 1a, since this is the only one that contains all 3 items (identity disclosure). Furthermore, if the attacker knows that *Bob* has purchased e and f, then he can infer that *Bob* has purchased

Name	*Purchased items*
Mary	a b c d **g h i j**
Bob	e f **h i**
Tom	d **g j**
Anne	e f **g h**
Brad	a b d e **i**
Jim	c f **j**

(a)

Purchased items
(a, b, c) (d, e, f) **g h i j**
(d, e, f) **h i**
(d, e, f) **g j**
(d, e, f) **g h**
(a, b, c) (d, e, f) **i**
(a, b, c) (d, e, f) **j**

(b)



(c)

Purchased items
a (b, c) d **g h i j**
e f **h i**
d **g j**
e f **g h**
a (b, c) d e **i**
(b, c) f **j**

(d)

PS-rules
a $\rightarrow$ **j**
cd $\rightarrow$ **g**
d $\rightarrow$ **hi**

(e)

Fig. 1 **a** De-identified data, **b** 2^2-anonymous data [74], **c** hierarchy, **d** anonymized data based on our rule-based model, and **e** PS-rules

sensitive item **h**, even though he cannot link *Bob* to a specific transaction in Fig. 1a using e and f (sensitive information disclosure).

Approaches to guarding against these threats have been proposed recently. However, they consider scenarios that involve coarse privacy protection requirements, while, with the exception of [91], they seek to prevent either identity or sensitive information disclosure. First, they make some restrictive assumptions about protection requirements. More specifically, they assume that either all combinations of items [35] or itemsets of certain size [74,75,91] are known to an attacker, thus need to be protected from identity disclosure; or that all items that an attacker is not expected to know are sensitive, hence they must all be guarded against sensitive information disclosure [16,27,29]. However, in real-world applications (e.g. in biomedical and mobility data analysis [32,50,65,73]), only certain itemsets can lead to identity disclosure and only some items are actually sensitive. For instance, not all diagnoses given to a patient can be used to link up with an external data source to disclose their identity [50], and only diagnoses that can socially stigmatize patients, such as HIV, are treated as sensitive [65]. Thus, anonymizing data based on uniform privacy assumptions can result in *unnecessary* and often *excessive* data distortion. For example, if a, b, and c in Fig. 1a are the only items that may be linked to an external data source, then only these items need to be protected from identity disclosure. However, using the methods proposed in [74,75,91], protection will be extended to all 20 itemsets comprised of any 3 non-sensitive items (e.g., abc, abd, abe etc.),[1] causing unnecessary information loss.

Second, all existing approaches with the exception of [91] do not guarantee preventing both identity and sensitive information disclosure [27,29,33,35,51,74,75]. For example, when applied to Fig. 1a, the method given in [74] produces Fig. 1b by generalizing single items to (a, b, c) and (d, e, f). Clearly, identity disclosure is prevented in this case, but if we require that no sensitive information may be inferred using items a to f for any individual with a probability greater than $\frac{1}{2}$, then Fig. 1b is not protected from sensitive information disclosure. This is because knowing that *Mary* has purchased a, an attacker can infer that *Mary* has bought **j** with a probability of $\frac{2}{3}$. We argue that eliminating identity and sensitive information disclosure together is important. This is because, identity disclosure can have legal consequences [1] for the data publishers and endanger future data collection [37], for example, in applications related to the sharing of electronic medical records [50], statistical [37], and video rental data [62], even though the identified transactions contain no sensitive information.

1.2 Contributions

This paper proposes a general approach to anonymizing transaction data. To our knowledge, this is the first approach that allows fine-grained, flexible privacy requirements to be specified and enforced for both identity and sensitive information disclosure. The contributions of our work are summarized as follows.

First, we introduce a rule-based privacy model that is more general and flexible than the existing privacy principles [35,51,74,75,91]. Instead of imposing some uniform conditions on which items to be protected and seeking to prevent either identity or sensitive information disclosure, we allow data publishers to specify fine-grained protection requirements for both of these disclosures. This is achieved by specifying a set of *PS-rules* which naturally and intuitively captures the items to be protected and their inter-relationships. For example, $cd \rightarrow$ **g** is a PS-rule which specifies that identity disclosure through cd, as well as inference of

[1] We assume that these methods are applied to non-sensitive items. Otherwise, 56 itemsets would receive protection.

sensitive itemset **g** using *cd*, are to be prevented. Protection from identity disclosure is achieved by imposing that any subset of items in the antecedent of a PS-rule is contained in a minimum of k transactions in the anonymized dataset, while sensitive information disclosure is thwarted by imposing an upper bound c on the confidence of the rule. For example, $cd \rightarrow \mathbf{g}$ is protected in Fig. 1d with respect to $k = 2$ and $c = 0.5$. This implies that no individuals can be linked to a transaction through c or d or cd with a probability higher than $1/2$, and to sensitive item **g** using *cd* with a probability higher than 0.5. So, our model allows data publishers to capture precisely which items are to be protected from each type of disclosure. This, as we will see shortly, helps produce anonymizations with better protection and data utility.

Second, we develop two novel generalization-based algorithms to anonymize transactions with respect to specified PS-rules. Both algorithms work by generalizing non-sensitive items, as sensitive items need to be released intact in most applications [15,91]. The *Tree-based Anonymization* algorithm operates in a top-down fashion, recursively replacing generalized items with more refined ones in a way that helps data utility. For example, when applied to Fig. 1a using the PS-rules given in Fig. 1e and for $k = 2$ and $c = 0.5$, this algorithm produced an anonymized dataset shown in Fig. 1d. Compared to Fig. 1b, the dataset shown in Fig. 1d clearly incurred less information loss and offered acceptable protection w.r.t. the given rules. To see this, observe that the antecedent of any of the PS-rules in Fig. 1e is contained in at least 2 transactions in the dataset of Fig. 1d, while the confidence of all these rules is upper-bounded by 0.5. The *Sample-based Anonymization* algorithm is more scalable and can anonymize large and high-dimensional datasets with a large number of PS-rules efficiently. The main idea behind this algorithm is to use a sample to find an anonymized dataset in which most PS-rules are protected quickly, and then slightly adjust the level of generalization in this dataset so that all the rules become protected in it. Both algorithms employ efficient rule pruning strategies, developed based on the properties of PS-rules. These strategies are essential, because examining all PS-rules would incur a prohibitively high computational cost, for any realistic datasets. However, Sample-based Anonymization performs rule checking much faster than Tree-based Anonymization does, as only a small sample of the anonymized dataset is used in the checking process.

Our extensive experiments show that the proposed approach is able to offer effective protection and allow much more accurate analysis to be carried out on the anonymized data than that of [74] and [91]. In addition, they demonstrate that Sample-based Anonymization is significantly faster than Tree-based Anonymization is and results in anonymizations with better data utility due to the way it processes generalized items in the second stage of computation. A preliminary version of this work reporting a basic algorithm along the lines of Tree-based Anonymization can be found in [52].

The remainder of the paper is organized as follows. Related work is reviewed in Sect. 2. Section 3 formally defines our generalization model and utility metric, as well as our rule-based privacy model and the anonymization problem we consider. In Sect. 4, we present our rule checking strategy and algorithms. Section 5 discusses how PS-rules can be specified, and Sect. 6 reports experimental results. Finally, we conclude the paper in Sect. 7.

2 Related work

There are considerable research efforts for designing privacy-preserving methods [3,5,14,18,22,38,43,56,59,66,70–72,92,93,95]. Several of these methods focus on data sharing [3,5,14,26,44,54,56,59,66,70,71,93,95] and fall into two general categories [6,25].

The first category of methods attempts to prevent sensitive patterns, such as frequent itemsets [66] or sequences [2,30], or association rules [31,67,77], from being mined from the data, while the second one, referred to as privacy-preserving data publishing (PPDP), aims to protect the privacy of individuals whose information is contained in the data. Our work is related to PPDP, and in this section we review the existing PPDP techniques that are developed for publishing relational or transaction data. These techniques aim to publish the data in such a way that they remain protected from identity or sensitive information disclosure, but can still be analyzed at a record-level, which is essential to many applications [50,60,61]. We also discuss techniques for achieving *differential privacy* [13,22], which release noisy query results [13,24] or noisy summary statistics about the data [10,88] instead. Last, we discuss *algorithm-based* attacks, in which attackers use knowledge of the anonymization algorithm to increase their belief about individuals' sensitive values [21,81,82,87,94].

2.1 Preventing identity disclosure

Relational data publishing Identity disclosure may occur even when data, devoid of information that explicitly identifies individuals, are published. This is because released data may still be linked to external data containing individuals' identities, through seemingly innocuous attributes, such as age or gender. To prevent identity disclosure in such cases, k-anonymity was proposed in [68,70]. A relational table satisfying k-anonymity ensures that the probability of linking an individual to their record, based on potentially linkable attributes, is upper bounded by $\frac{1}{k}$.

Constructing a k-anonymous table is possible by transforming attribute values, but achieving this with minimal information loss is NP-hard [57]. Some algorithms employed suppression, a technique that removes values, to achieve k-anonymity [57] and others adopted generalization (i.e., they replace original values with more general but semantically consistent ones). Generalization-based methods employ evolutionary techniques [36], agglomerative and divisive clustering [89], and data partitioning [41,42]. LeFevre et al. for example, designed Mondrian, a generalization-based algorithm inspired by the kd-tree construction [42], as well as Rothko-S, an algorithm that introduced the use of sampling to anonymize datasets larger than the main memory based on the Mondrian framework [41].

These algorithms assume that all attributes are potentially linkable, hence all combinations of their values need to be protected. This can result in excessive, but unnecessary information loss [70] when the actual privacy requirements dictated by the intended applications are different [60,61,78]. An exception to this is the approach of [60,61], which protects combinations of up to a certain number of potentially linkable attributes. The authors of [60,61] proposed LKC-privacy to specify and enforce such privacy requirements, and they developed two generalization-based algorithms to satisfy LKC-privacy for centralized and distributed data sharing scenarios. Although we share the goal of enhancing data utility by taking into account detailed privacy requirements, our approach addresses a different problem than that of [60,61], as we consider anonymizing transaction data.

Transaction data have different characteristics than relational data [27,29,35,61,74,75,91]. Specifically, in transaction data, an individual is associated with a set of items of variable, and typically large, size, and some of these items may be sensitive. On the other hand, an individual in a relational dataset is associated with a fixed, and typically small, number of attribute values, one of which is sensitive. Due to these differences, most principles and algorithms that have been developed for anonymizing relational data are not applicable to transaction data, as discussed in [27,29,61,74,75,91]. For instance, Aggarwal [4] has proven theoretically that applying k-anonymity on high-dimensional data is likely to incur extremely

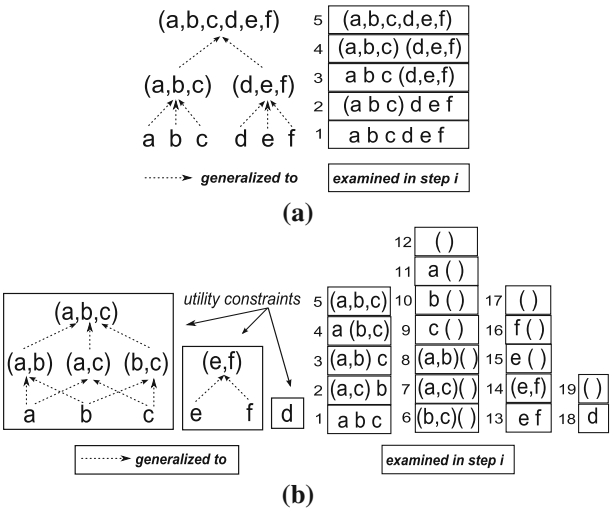


Fig. 2 Series of generalization decisions performed by **a** Apriori, and **b** COAT

high information loss, rendering the data useless. In addition, Xu et al. [91] have shown that anonymizing transaction data, using the relational data anonymization algorithm proposed in [26], may harm data utility. Furthermore, most measures that capture the utility of anonymized relational data, such as those that are based on ranges of values in anonymous groups of records [89,85], are not applicable to transaction data.

Transaction data publishing Privacy models for anonymizing transactions to forestall identity disclosure were considered in [35,51,74,75]. Terrovitis et al. [74] introduced k^m-anonymity, which prevents linking an individual to a transaction using combinations of any m items of this transaction, while He et al. [35] proposed a k-anonymity based principle to guard against attackers who may know any combination of items in an individual's transaction. Recently, Loukides et al. [51] suggested a constraint-based k-anonymity model to prevent identity disclosure through specific itemsets. The differences between these privacy models and ours were discussed in Introduction.

In terms of methods, Terrovitis et al. [74] designed Apriori, an effective algorithm to enforce k^m-anonymity. This algorithm replaces groups of items, represented as leaf nodes in a generalization hierarchy, to a generalized item that corresponds to one of their ascendants. For efficiency, Apriori seeks to protect increasingly larger combinations of items iteratively in a bottom-up fashion, from single items to combinations of m items. In each step, it examines all possible generalizations and finds one that incurs the least information loss and satisfies k^m-anonymity. For example, the generalization steps taken by Apriori when applied to Fig. 1a using the hierarchy of Fig. 1c are shown in Fig. 2a, where the release of original items is considered first, then the generalization of $\{a, b, c\}$ to (a, b, c), and finally the generalization of all items to (a, b, c, d, e, f).

Another bottom-up algorithm is COAT [51], which operates in a greedy fashion and performs both generalization and suppression. The selection of items that are generalized by COAT is governed by the utility constraints specified by data publishers. These constraints correspond to the most generalized items that can be used to replace a set of items, thus they limit the generalizations to those that are acceptable for intended applications. When an item cannot be generalized with respect to specified utility constraints, COAT suppresses it

to ensure privacy. In Fig. 2b, for example, COAT takes on three utility constraints and examines 19 possible transformations: 1–5, 13–14, and 18, corresponding to the generalizations considered for items in each utility constraint, and the remaining ones involving suppressed items, denoted by ().

Our algorithms are similar to Apriori and COAT in that an item is replaced with the same generalized item in all transactions. The algorithms proposed by He et al. [35] and Terrovitis et al. [75], however, allow the same item to be replaced by different generalized items in different transactions, as they partition transactions into sets and then generalize items in each set separately. While this may reduce data distortion, these algorithms generate anonymized datasets of limited utility in practice, as mining algorithms and analysis tools cannot work effectively on such datasets [25].

2.2 Preventing sensitive information disclosure

Relational data publishing Inferring sensitive information about individuals is possible even when released data prevent identity disclosure, for example, when a "large" number of records in a k-anonymous group have the same value in a sensitive attribute [54]. Several privacy principles were proposed to prevent this threat [44,54,78,80,83,86]. l-diversity requires each group of records to have at least l "well-represented" sensitive values [54], while m-invariance dictates that each group should have m distinct sensitive values and deals with the anonymization of dynamic datasets [86]. Li et al. [44] examined inferences that can be made when the distribution of sensitive values in an anonymous group is not similar to that in the sensitive attribute as a whole, and proposed t-closeness to limit the difference between these distributions.

While the aforementioned principles treat all values of an attribute as either potentially linkable or sensitive, Wang et al. [78–80] argued that less information loss will be incurred if this restriction is lifted. In [78,79], they introduced the concept of privacy template to model an association between a set of values in one or more potentially linkable attributes and a single sensitive value. A privacy template is satisfied when the probability of associating an individual with their sensitive value, based on the potentially linkable values, is limited. Thus, the privacy principle in [78,79] guarantees protection from sensitive information disclosure, but not from identity disclosure. Recently, Wang et al. [80] examined inferences of sensitive information in which any attribute value may be sensitive and proposed FF-anonymity to limit the probability of these inferences. Our model deals with detailed privacy requirements, which are common in transaction data publishing applications [50], but it offers protection from both types of disclosure.

A privacy principle, called (α, k)-anonymity, was proposed by Wong et al. [83] to thwart both identity and sensitive information disclosure. This principle demands that there exist at least k records with the same values in all potentially linkable attributes and at most $\alpha\%$ of these records have the same sensitive value. Yet, (α, k)-anonymity treats all non-sensitive information as potentially linkable. This is different from our principle, which allows the flexibility of specifying detailed privacy protection requirements for identity disclosure.

Several anonymization algorithms that focus on guarding against sensitive information disclosure have been developed [78,79,84]. Wang et al. [78,79] proposed a suppression-based algorithm that aims at preserving data utility for classification tasks. Our algorithms use generalization, which generally preserves data utility better than suppression, and release data that minimize a general-purpose information loss metric, which is useful when the utility requirements are not known or specified at the time of data publishing [25]. To enforce l-diversity, Xiao et al. [84] proposed an algorithm based on *anatomy*, a technique that releases

an anonymized table as two sub-tables, one containing potentially linkable attributes and the other sensitive ones. Methods [28,60,61,83] for preventing both identity and sensitive information disclosure have also been developed, such as the method that exploits space mapping techniques to construct k-anonymous and l-diverse relational tables [28].

Most principles and algorithms that have been designed for guarding against sensitive information disclosure in relational data publishing are not suited to transaction data publishing. This is due to the different semantics of these two types of data that are discussed in Sect. 2.1. Ghinita et al. [27,29], for example, have shown that generalization and anatomy-based algorithms, which are designed to enforce l-diversity on relational data, are likely to significantly reduce data utility when applied to transaction data, particularly when transactions have a large number of sensitive items.

Transaction data publishing Several works have considered the prevention of sensitive information disclosure in publishing transaction data. Xu et al. [91] introduced (h, k, p)-coherence that treats items an attacker is expected to know, referred to as *public* items, similarly to k^m-anonymity (p plays the same role as m in k^m-anonymity), and additionally limits the probability of inferring any non-public item using a parameter h. Enforcing (h, k, p)-coherence results in over-protection and over-distortion when only some specific privacy requirements are to be met [50,51], as in the case of several real-world applications [50,60,61]. Our model was developed to deal with such applications, and it has (h, k, p) coherence as its special case, as we will prove in Sect. 3.4.2. To enforce (h, k, p)-coherence, Xu et al. [91] proposed Greedy, a suppression-based algorithm which assumes that all items require protection against either identity or sensitive information disclosure. Another (h, k, p)-coherence algorithm, which is based on suppression and aims at preserving the counts of frequent itemsets, was proposed in [90]. Our algorithms employ generalization, and they preserve data utility significantly better than Greedy, even when they are configured to enforce (h, k, p)-coherence, as shown in our experiments.

Another group of methods treats all non-public items as sensitive and protects them all from disclosure. Ghinita et al. [27,29] suggested an anatomy-based method that releases groups of transactions, each of which contains public items in their original form and a summary of frequencies of non-public items. Concurrently with our work, Cao et al. [16] introduced ρ-uncertainty, which limits the probability of inferring any non-public item, and an algorithm which first iteratively suppresses non-public itemsets, and then generalizes public items using the model given in [74] to enforce this principle. This work is unique in that it guards against attackers who possess knowledge about sensitive items. On the contrary, our approach assumes that data publishers are aware of the items that can be used to infer sensitive information, which is often the case in applications, and blocks inferences that can be made using these items to enhance data utility. Contrary to ours, the methods given in [16,27,29] do not guarantee the prevention of identity disclosure, nor can they be easily modified to provide such guarantees, because their effectiveness and efficiency rely heavily on the intrinsic properties of their privacy models.

2.3 Preserving privacy through differential privacy

The *differential privacy* model ensures that the outcome of a calculation is insensitive to any particular record in the dataset [13,22]. This offers privacy, because the inferences an attacker can make about an individual will be (approximately) independent of whether any individual's record is included in the dataset or not. Differential privacy makes no assumptions about an attacker's background knowledge, as opposed to the models discussed in Sects. 2.1

and 2.2, and it has led to the development of many interesting approaches (see [23] for a survey). However, differential privacy may be overly restrictive [53], and its enforcement does not guarantee the prevention of all attacks [20,39]. Cormode, for example, showed that an attacker can infer the sensitive value of an individual fairly accurately, by applying a Naive Bayes classifier on differentially private data [20]. Furthermore, differential privacy cannot be satisfied by deterministic algorithms, and even randomized algorithms will almost certainly produce anonymized data that are totally unrepresentative of the original data [55]. Thus, either noisy answers to a limited number of queries posed by users [13,24] or noisy summary statistics have to be released instead [10,19,34,40,58,88]. Also, individuals may become associated with false information in the output of differential privacy methods [25]. Our approach focuses on a different setting, which involves the publication of data that can be analyzed at a record level and remain truthful (i.e., individuals are associated with more general, but not with false information in the published data). Producing such data is useful in several tasks, including visualization, auditing, and health and biomedical data analysis [7,50,60,61].

Concurrently with our work, Chen et al. [19] proposed a novel method to release noisy estimates of the answers to certain count queries involving itemsets in a transaction dataset. These queries are identified using a top-down, data partitioning strategy, which reduces the amount of added noise needed to satisfy differential privacy and improves utility. The partitioning strategy employs a taxonomy and starts by grouping together all transactions of the dataset, which initially contain a single item corresponding to the root of the taxonomy. Then, the group is recursively partitioned into a number of subgroups, similarly to [35], so that the transactions in each resultant subgroup share a common representation determined by the taxonomy. To satisfy differential privacy, a subgroup is partitioned if its noisy count (i.e., the noisy answer to the count query asking for the number of transactions contained in the subgroup) exceeds a threshold set by the data publisher. The partitioning ends when the transactions in all subgroups with non-zero, noisy counts contain only leaf-level nodes in the taxonomy. Subsequently, the algorithm computes the noisy count of each subgroup and releases it together with the node of the taxonomy that corresponds to the subgroup. Our approach is developed for a different publishing scenario than that of [19], which involves publishing truthful transactions that can be analyzed individually, as mentioned above. Consequently, our algorithms release such transactions, instead of the noisy counts of specific itemsets. Furthermore, the algorithms we propose employ generalization, and not noise addition, and they partition generalized items, without the need of a taxonomy, for the purpose of constructing an anonymized dataset.

2.4 Algorithm-based attacks

In algorithm-based attacks, sensitive information is inferred based on knowledge about the way the anonymization algorithm works [21,81,82,87,94]. The pioneering work of Wong et al. [81] introduced the *minimality attack*, in which an attacker is assumed to have some knowledge about the operation of a deterministic, relational data anonymization algorithm and its parameters. Moreover, the attacker must possess an *eponymous*, external dataset that contains the personal identifiers and all potentially linkable attributes of *all* the individuals contained in the published dataset. An algorithm is susceptible to minimality attack when it follows the *minimality principle* [81]. That is, when the algorithm produces an anonymized dataset that satisfies a privacy principle, and no other anonymized dataset, which satisfies the same principle, can be produced by specializing the values in a group of records of

the former dataset.[2] Wong et al. [81] also proposed an algorithm, called *MASK*, to prevent minimality attack. *MASK* is applicable to relational data and works by generalizing and distorting potentially linkable and sensitive values, respectively. Cormode et al. [21] provided a detailed analysis of the minimality attack. They identified properties that make relational data anonymization algorithms susceptible to the attack and showed how algorithms can be modified to prevent it.

Xiao et al. [87] considered attackers who know everything about the anonymization algorithm and possess an eponymous, external dataset as in [81]. Such attackers may infer an individual's sensitive value with "high" probability, even when l-diversity [54] holds. To see this, observe that, without knowledge of the algorithm, an attacker must consider all original datasets that could be anonymized to the published dataset as equally likely.[3] In this case, l-diversity guarantees that an individual's sensitive value will not be inferred with a probability that exceeds $\frac{1}{l}$. However, this bound does not hold for the type of attackers considered in [87]. This is because, such attackers can rule out the possible original datasets that the algorithm would not transform to the anonymized dataset, thereby limiting the number of possible original datasets in which an individual is associated with a specific sensitive value. These *reverse engineering* attacks may compromise the privacy protection offered by several l-diversity algorithms, which include deterministic and randomized algorithms, as well as *MASK* [81], and can be prevented by applying *transparent l-diversity* [87]. An algorithm satisfies transparent l-diversity, when all associations between an individual and a sensitive value hold in no more than a "sufficiently" small fraction of the possible original datasets that could have been produced by the algorithm. Xiao et al. [87] also proposed three transparent l-diverse algorithms to anonymize relational data; *Tailor*, a deterministic, generalization-based algorithm inspired by kd-tree construction, *Ace* a randomized, generalization-based algorithm, and *Hybrid*, which first partitions data using *Tailor* and then invokes *Ace* in each subpartition.

All works studying algorithm-based attacks assume a relational data publishing setting [21,81,82,87,94]. However, as discussed by Cao et al. [16], "there is nothing analogous to the eponymous, external dataset", which is necessary for both minimality and transparency attacks to succeed, in transaction data publishing. On the contrary, an attacker can use only the items about an individual they can observe [16], which often are a small fraction of all items in the dataset. Consider, for example, the release of an anonymized, transaction dataset that contains patient diagnoses derived from electronic medical records [49]. Obtaining an eponymous, external dataset is extremely difficult, even for attackers with access to the electronic medical record system [50]. Similar observations were also made for e-commerce data [73]. Thus, the applicability of algorithm-based attacks on transaction data publishing is questionable. We further discuss these attacks in relation to our algorithms in Sect. 4.4.

3 Background and problem statement

After introducing some notations, we describe the generalization model and utility metric we use in Sects. 3.2 and 3.3, respectively. Section 3.4 presents the rule-based privacy model we propose, and Sect. 3.5 formulates our problem.

[2] Specialization is the reverse operation of generalization [26].

[3] This is the statistically best strategy an attacker can follow [9].

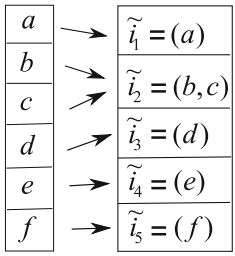


Fig. 3 An example of set-based item generalization

3.1 Notations

Let $\mathcal{I} = \{i_1, \ldots, i_M\}$ be a finite set of literals, called *items*, where M denotes the cardinality of the set. Any subset $I \subseteq \mathcal{I}$ is an *itemset* over $\mathcal{I}$, and it is represented as a concatenation of the items it contains. An itemset that has m items (or equivalently of size m) is called an m-itemset. The size of an itemset I is denoted by $|I|$.

Let $\mathcal{P}$ be the set of items in $\mathcal{I}$ that an attacker may use to link individuals to published transactions, $\mathcal{N}$ be the set of items in $\mathcal{I}$ that are not in $\mathcal{P}$, and $\mathcal{S}$ be a non-empty subset of $\mathcal{N}$ that represents some private information that needs to be protected. We require $\mathcal{P} \cup \mathcal{N} = \mathcal{I}$, $\mathcal{P} \cap \mathcal{N} = \varnothing$, and $\mathcal{S} \subseteq \mathcal{N}$. We call $\mathcal{P}$ *public*, $\mathcal{N}$ *non-public* and $\mathcal{S}$ *sensitive* sets of items, and denote their sizes with $|\mathcal{P}|$, $|\mathcal{N}|$ and $|\mathcal{S}|$, respectively. Following [27,29,54,83,91], we assume that attackers do not possess any background knowledge about items in $\mathcal{N}$.

A *transaction* T over $\mathcal{I}$ is a pair $T = \langle tid, I \rangle$, where I is an itemset and *tid* a unique identifier. A transaction dataset $\mathcal{D} = \{T_1, \ldots, T_N\}$ is a set of N transactions over $\mathcal{I}$, which contains all items of $\mathcal{I}$. The size of $\mathcal{D}$ is denoted by $|\mathcal{D}|$. A transaction $T = \langle tid, J \rangle$ *supports* an itemset I, if $I \subseteq J$. Given an itemset I over $\mathcal{I}$ in $\mathcal{D}$, we use $sup(I, \mathcal{D})$ to represent the number of transactions in $\mathcal{D}$ that support I. These transactions are called *supporting transactions* of I in $\mathcal{D}$ and denoted with $\mathcal{D}_I$.

3.2 Generalization model

We anonymize transactions through *item generalization*, an operation that replaces original items with more general items [74]. We generalize public items only as it is often necessary to keep sensitive items intact in applications [91], and we employ the set-based generalization model proposed in [51]. Set-based generalization models have been introduced in [45] and shown to be effective at retaining data utility due to their ability to represent a large number of generalizations [45,51,64].

Definition 1 (*Set-based generalization*) A set-based generalization is a partition $\tilde{\mathcal{P}}$ of $\mathcal{P}$ in which each item i in $\mathcal{P}$ is mapped to the member $\tilde{i}$ of $\tilde{\mathcal{P}}$ that contains i. We call $\tilde{i}$ a generalized item, and denote the number of items in $\tilde{i}$ by $|\tilde{i}|$.

So a set-based generalization is essentially a function that maps each item i in $\mathcal{P}$ to a unique subset $\tilde{i}$ of $\mathcal{P}$. To express this mapping, we write $\Phi(i) = \tilde{i}$ and call $\Phi : \mathcal{P} \rightarrow \tilde{\mathcal{P}}$ a *generalization function*. When the members of $\tilde{i}$ are known, we may also express $\tilde{i}$ by listing its item(s) in brackets. We interpret $\tilde{i}$ as representing any non-empty subset of its constituent items. For example, in Fig. 3, items b and c are both mapped to the same generalized item $\tilde{i_2} = (b, c)$, which represents either b, or c, or bc. For clarity, we may drop () when $\tilde{i}$ contains only one item.

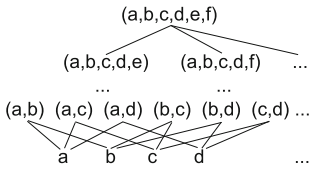


Fig. 4 All set-based generalizations for Fig. 1a

Applying Φ to the items of an original dataset $\mathcal{D}$ results in the transformation of $\mathcal{D}$ into a *generalized dataset* $\tilde{\mathcal{D}}$. It is easy to see that the support of a generalized item in $\tilde{\mathcal{D}}$ will always be greater than or equal to the support of any item mapped to it in $\mathcal{D}$. Consider Fig. 1a for example. If we assume that b and c are both mapped to the generalized item (b, c), then (b, c) is supported by 3 transactions in Fig. 1d, whereas b and c are only supported by 2 and 1 transactions in Fig. 1a. This helps prevent identity disclosure because an attacker will associate an individual with more transactions in $\tilde{\mathcal{D}}$ than in $\mathcal{D}$.

The set-based generalization model allows us to explore a significantly larger number of generalizations than the full-subtree generalization model can [35,74,75]. This is because each item $i \in \mathcal{P}$ can be independently mapped to a generalized item, so there are $2^{|\mathcal{P}|} - 1$ different generalized items possible, each corresponding to one of the itemsets in $\mathcal{P}$. For example, Fig. 4 shows the space of available set-based generalizations for the public items of Fig. 1a in a semilattice-like structure. As can be seen, there are $2^6 - 1 = 63$ generalized items possible. In comparison, the full subtree model will only consider 9 generalized items, as shown in the hierarchy given in Fig. 1c. A larger generalization space can help construct anonymized data with higher utility.

3.3 Utility metrics

There are numerous ways to generalize a dataset, and the one that harms data utility the least is typically preferred. Several criteria capture data utility based on the level of information loss incurred by generalization [75,89]. Normalized Certainty Penalty (*NCP*) [89], for example, is a variant of the LM measure introduced in [36]. NCP is employed in [74,75] and expressed as the weighted average of information loss incurred by all generalized items, each measured in terms of the number of descendants it has in the generalization hierarchy. Other measures, such as multiple level Mining Loss (ML^2) and differential multiple level Mining Loss (dML^2), express utility based on how well anonymized data support frequent itemset mining [75]. All these measures require items to be generalized according to hierarchies. We use the Utility Loss (*UL*) measure [51], which can be applied in the absence of hierarchies.

Definition 2 (*Utility loss*) The Utility Loss (*UL*) for a generalized item $\tilde{i}$ is defined as $UL(\tilde{i}) = \frac{2^{|\tilde{i}|}-1}{2^{|\mathcal{P}|}-1} \times w(\tilde{i}) \times \frac{sup(\tilde{i},\tilde{\mathcal{D}})}{N}$, where $w(\tilde{i}) \in [0, 1]$ is a weight assigned to $\tilde{i}$ according to its perceived usefulness. Accordingly, the Utility Loss for a generalized dataset $\tilde{\mathcal{D}}$ is defined as $UL(\tilde{\mathcal{D}}) = \sum_{\forall \tilde{i} \in \tilde{\mathcal{P}}} UL(\tilde{i})$.

The UL measure penalizes a generalized item $\tilde{i}$ in terms of its size, support, and a weight. The size is taken into account because $\tilde{i}$ can represent any of the $2^{|\tilde{i}|} - 1$ non-empty subsets of the items mapped to it. The larger $\tilde{i}$ is, the less certain we are about the value it represents. The support for $\tilde{i}$ also contributes to the loss of utility, as its generalization will affect more transactions, resulting in more distortion. The terms $2^{|\mathcal{P}|} - 1$ and N are used to normalize size and support, respectively. A weight is used to penalize generalizations exercised on more

"important" items. This weight is to be specified by the data publisher, but can be computed based on the semantic similarity of items that are mapped to a generalized item [51,89].

3.4 Rule-based privacy model

This section presents our rule-based privacy model. We first introduce PS-rules and then explain how they can be used to prevent both identity and sensitive information disclosure. We also show that our model is more general than the existing ones.

3.4.1 PS-rules and their protection

A PS-rule models an association between public and sensitive items that must be concealed from the published data in order to prevent identity and sensitive information disclosure.

Definition 3 (*PS-rule*) A *PS-rule* is an implication $I \rightarrow J$, where I is an itemset in $\mathcal{P}$ and J is an itemset in $\mathcal{S}$.

PS-rules can be used to specify fine-grained privacy requirements for both identity and sensitive information disclosure, as there are no restrictions on which itemsets of $\mathcal{P}$ and $\mathcal{S}$ are to be used as antecedents or consequents of the rules. We will discuss how PS-rules can be specified in Sect. 5.

We now give conditions under which a PS-rule is considered to be protected and explain how identity and sensitive information disclosure is prevented through protected PS-rules. We use $\bigcup$ as a generalized itemset constructor, and $\tilde{I} = \bigcup_{\forall i \in I} \Phi(i)$ to represent the generalized itemset constructed from itemset I.

Definition 4 (*PS-rule protection*) Given a generalized version $\tilde{\mathcal{D}}$ of $\mathcal{D}$, produced using the set-based generalization model, and parameters $k \in [2, N]$ and $c \in [0, 1]$, a PS-rule $I \rightarrow J$ is *protected* in $\tilde{\mathcal{D}}$ if (1) $sup(\tilde{I}, \tilde{\mathcal{D}}) \geq k$, and (2) $conf(\tilde{I} \rightarrow J, \tilde{\mathcal{D}}) \leq c$, where $conf$ is a function, called *confidence*, defined by $\frac{sup(\tilde{I} \cup J, \tilde{\mathcal{D}})}{sup(\tilde{I}, \tilde{\mathcal{D}})}$.

Since $\tilde{\mathcal{D}}$ is produced using set-based generalization, it is easy to see that an attacker would be unable to associate an individual with any transaction in $\tilde{\mathcal{D}}$ through $\tilde{I}$, if $\tilde{I}$ was unsupported in $\tilde{\mathcal{D}}$ [91]. We therefore assume that $sup(\tilde{I}, \tilde{\mathcal{D}}) > 0$. To explain why conditions (1) and (2) help prevent identity and sensitive information disclosure, we observe the following properties.

Property 1 The probability of linking an individual to their transaction in $\tilde{\mathcal{D}}$ using the antecedent of a *protected* PS-rule in $\tilde{\mathcal{D}}$ is at most $\frac{1}{k}$.

Property 2 The probability of associating an individual with the consequent of a *protected* PS-rule in $\tilde{\mathcal{D}}$, given that their transaction supports the generalized antecedent of this rule, is at most c.

Example 2 Consider the dataset of Fig. 1d and the PS-rule $cd \rightarrow \mathbf{g}$, which is protected for $k = 2$ and $c = 0.5$, as there are 2 transactions supporting the generalized antecedent $\bigcup_{\forall i \in \{c,d\}} \Phi(i) = (b, c)d$, and only one of them supports $(b, c)d \cup \mathbf{g}$. This implies that an attacker, who knows that *Mary* has purchased b and c, cannot infer *Mary*'s actual transaction with a probability that exceeds $\frac{1}{2}$, nor can the attacker infer that *Mary* has purchased $\mathbf{g}$ with a probability higher than $\frac{1}{2}$.

So far, we have only considered the attacks that use all of the items specified in the antecedent and consequent of a PS-rule $I \rightarrow J$. Following [66], we do not consider the attacks that use subsets of the items in J, since our intention is to satisfy specified privacy requirements only. Attacks based on subsets of I and on supersets of J, however, are prevented when $I \rightarrow J$ is protected. We will discuss this observation and its implication further in Sect. 4.1.2. Also, we do not consider negated items, such as *Bob has not purchased item a*, because such knowledge is not very useful in attacks, and it is rather difficult to obtain [75].

3.4.2 Generality of the PS-rule model

To illustrate the expressive power of our rule-based model, we show that existing privacy models for transaction data [35,51,74,75,91] are in fact the special cases of our model. We will show that for k^m-anonymity [74,75] and (h, k, p)-coherence [91] in Properties 3 and 4 below (proofs follow from the definitions of these models[4]). Showing that for the models proposed in [35] and [51] is trivial.

Property 3 For a given m, construct a set of PS-rules Θ that contains one PS-rule $I \rightarrow J$ for each m-itemset in $\mathcal{P}$ as I, and any itemset in $\mathcal{N}$ as J. A transaction dataset $\mathcal{D}$ in which all rules in Θ are protected, for a given k and any $c \in [0, 1]$, will also satisfy k^m-anonymity for the same k and m.

Property 4 For a given p, construct a set of PS-rules Θ that contains one PS-rule $I \rightarrow J$ for each p-itemset in $\mathcal{P}$ as I and for each 1-itemset in $\mathcal{N}$ as J. A transaction dataset $\mathcal{D}$ in which all rules in Θ are protected, for a given $k \in [2, N]$ and $c \in [0, 1]$, will also satisfy (h, k, p)-coherence for the same k and p, and for $h = c \times 100\,\%$.

3.5 Problem statement

The problem we consider in this paper is defined below.

Problem (PS-Rule-based anonymization) Given a transaction dataset $\mathcal{D}$, a set of PS-rules Θ, and parameters k and c, construct a generalized dataset $\tilde{\mathcal{D}}$ from $\mathcal{D}$ using the set-based generalization model such that: (1) each PS-rule in Θ is protected in $\tilde{\mathcal{D}}$ and (2) the amount of utility loss $UL(\tilde{\mathcal{D}})$ is minimal.

As we show in Theorem 1, our problem is NP-hard. Also, it may not have a solution, as explained in Theorem 2, unless a condition provided in Theorem 3 holds. The proofs of these theorems, as well as those in the rest of the paper, are provided in the "Appendix".

Theorem 1 *The PS-rule-based anonymization problem is NP-hard.*

Theorem 2 *Given a dataset* $\mathcal{D} = \{T_1, \ldots, T_N\}$*, a set of PS-rules* $\Theta = \{I_1 \rightarrow J_1, \ldots, I_s \rightarrow J_s\}$*, and parameters k and c, a generalized dataset* $\tilde{\mathcal{D}}$ *in which each PS-rule in* Θ *is protected, cannot be constructed from* $\mathcal{D}$ *using set-based generalization if* $sup(I_q \cup J_q, \mathcal{D}) > N \times c$*, for any rule* $I_q \rightarrow J_q, q \in [1, s]$*, in* Θ*.*

Theorem 3 *Given a dataset* $\mathcal{D} = \{T_1, \ldots, T_N\}$*, a set of PS-rules* $\Theta = \{I_1 \rightarrow J_1, \ldots, I_s \rightarrow J_s\}$*, and parameters k and c, a generalized dataset* $\tilde{\mathcal{D}}$ *in which each PS-rule in* Θ *is protected, can be constructed from* $\mathcal{D}$ *using set-based generalization if* $sup(\bigcup_{\forall i \in I_q} \Phi(i) \cup J_q, \tilde{\mathcal{D}}) \leq N \times c$*, for any rule* $I_q \rightarrow J_q, q \in [1, s]$*, in* Θ*.*

[4] We assume that k^m-anonymity is applied to public items.

Thus, a generalized dataset $\tilde{\mathcal{D}}$, in which all rules in Θ are protected, can be constructed if the support of none of the itemsets $\bigcup_{\forall i \in I_q} \Phi(i) \cup J_q$ in $\tilde{\mathcal{D}}$ exceeds $N \times c$. Note that it is possible to examine whether such a generalized dataset can be constructed without anonymizing $\mathcal{D}$. That is, it is sufficient to check whether $sup(J_r, \mathcal{D}) \leq N \times c$ holds, where J_r is the most frequently supported sensitive item in the original dataset $\mathcal{D}$. This is because $sup(\bigcup_{\forall i \in I_q} \Phi(i) \cup J_q, \tilde{\mathcal{D}}) \leq sup(J_r, \mathcal{D})$. Performing the check using $sup(J_r, \mathcal{D})$ requires $O(N \times |\mathcal{S}|)$ time, where $|\mathcal{S}|$ is the number of sensitive items in $\mathcal{D}$.

4 Efficient rule-based anonymization

In this section, we consider the problem of generalizing data to protect PS-rules, hence the dataset. We introduce several novel strategies to process PS-rules efficiently, and present two efficient and effective heuristic algorithms that utilize these strategies.

4.1 Handling PS-rules

Generalizing transaction data to protect a given set of PS-rules is not straightforward. First, it is computationally expensive to examine all possible generalizations to find an optimal one. To reduce this cost, we develop an efficient heuristic (Sect. 4.1.1) that explores only a subset of potentially useful generalizations. Second, determining whether specified PS-rules are protected in a generalized dataset is time-consuming. This is because checking a rule requires support computations that involve dataset scan, and there can be many rules to be checked. To address these issues, we design an efficient rule checking algorithm (Sect. 4.1.3), based on the properties of PS-rules. This algorithm can significantly reduce the computations required for rule checking.

4.1.1 Finding useful generalizations

To protect PS-rules, we generalize items contained in their antecedents. This is because doing so can increase the support of their antecedents, but will not affect the support of their consequents. That is, for a PS-rule $I \rightarrow J$ with $\tilde{I} = \bigcup_{\forall i \in I} \Phi(i)$, it holds that $sup(I, \mathcal{D}) \leq sup(\tilde{I}, \tilde{\mathcal{D}})$ and $sup(I \cup J, \mathcal{D}) \leq sup(\tilde{I} \cup J, \tilde{\mathcal{D}})$. However, as confidence is non-monotonic [16], it may be possible that $conf(I \rightarrow J, \mathcal{D}) \leq conf(\tilde{I} \rightarrow J, \tilde{\mathcal{D}})$. This implies that releasing I instead of $\tilde{I}$ may offer better privacy protection, even though more specific information is provided to the attacker. It is worth noting, however, that releasing this information will not lead to sensitive information disclosure, because our approach guarantees that the confidence of rules will not exceed c.

We perform generalization in a *top-down* fashion. That is, we start by assuming that every public item is mapped to a single, most generalized item, and then seek to split it recursively into smaller (less general) ones to enhance utility. This is achieved by using a heuristic, called *B-Split*, which is inspired by divisive clustering. *B-Split* seeks to divide a generalized item $\tilde{i}$ into two disjoint generalized items $\tilde{i_l}$ and $\tilde{i_r}$ that incur low information loss.

Algorithm 1 describes how *B-Split* works. Φ_{ID} used in the description is the *identity generalization function* which maps a set of items $\{i_1, \ldots, i_s\}$ to the generalized item $\tilde{i} = (i_1, \ldots, i_s)$. In step 1, we find two items i_l and i_r contained in $\tilde{i}$ that will incur a maximum amount of utility loss when generalized together by examining the *UL* scores of all pairs of items, breaking ties arbitrarily. Then, in steps 2–3, we use i_l and i_r as "seeds" to start two new itemsets I_1 and I_2. Following that, *B-Split* examines each item i_q, $q \notin \{l, r\}$ in $\tilde{i}$ in

Algorithm 1 *B-Split*($\tilde{i}$)

input: generalized item $\tilde{i}$, dataset $\mathcal{D}$, generalized dataset $\tilde{\mathcal{D}}$
output: generalized items $\tilde{i_l}$ and $\tilde{i_r}$
1. find $\{i_l, i_r\}$ in $\tilde{i}$ s.t. $UL(\Phi_{ID}(\{i_l, i_r\}))$ is maximum
2. $I_1 \leftarrow i_l$
3. $I_2 \leftarrow i_r$
4. **for each** ($i_q \in \tilde{i}$ and $q \notin \{l, r\}$)
5. $\tilde{i_1} \leftarrow \Phi_{ID}(I_1 \cup \{i_q\})$
6. $\tilde{i_2} \leftarrow \Phi_{ID}(I_2 \cup \{i_q\})$
7. **if** $(|I_1| \times w(\tilde{i_1}) \times \sum_{\forall i \in I_1 \cup \{i_q\}} sup(i, \mathcal{D}) \leq |I_2| \times w(\tilde{i_2}) \times \sum_{\forall i \in I_2 \cup \{i_q\}} sup(i, \mathcal{D}))$
8. $I_1 \leftarrow I_1 \cup \{i_q\}$
9. **else**
10. $I_2 \leftarrow I_2 \cup \{i_q\}$
11. **return** $\tilde{i_l} = \Phi_{ID}(I_1), \tilde{i_r} = \Phi_{ID}(I_2)$

turn (step 4), creates two temporary generalized itemsets $\tilde{i_1}$ and $\tilde{i_2}$ from $I_1 \cup i_q$ and $I_2 \cup i_q$ (steps 5–6), and uses them to determine whether i_q should be assigned to I_1 or I_2, depending on the information loss each would have incurred (steps 7–10). Finally, in step 11, two new generalized items $\tilde{i_l}$ and $\tilde{i_r}$, which are less general than $\tilde{i}$, are created from I_1 and I_2, and returned.

Notice that the computation of information loss in step 7 of Algorithm 1 is similar to that of *UL*, but more efficient. This is because, in this computation, the support of $\tilde{i_1}$ and $\tilde{i_2}$ is calculated approximately using items in $\mathcal{D}$ and thus can be pre-computed by scanning $\mathcal{D}$ once. The computation of *UL*, on the other hand, is based on the support of generalized items, which would need to be re-computed in each iteration of the loop of step 4 in Algorithm 1. Our experiments show that this simplification is largely effective.

Example 3 Consider the generalized version of the dataset in Fig. 1a that is shown in Fig. 1b and assume the following weights: $w((a, b)) = 1, w((a, c)) = 1$ and $w((b, c)) = 0.5$. *B-Split* produces generalized items (a) and (b, c) from (a, b, c) in two steps. First, it computes the *UL* scores for (a, b), (b, c) and (a, c), which are 0.016, 0.012 and 0.024, respectively. Then, it selects a and c as "seeds" to start I_1 and I_2. After that, it assigns b to I_2, since the amount of information loss for $I_2 \cup \{b\}$ is calculated as $2 \times 0.5 \times (2 + 2) = 4$, which is lower than $2 \times 1 \times (2 + 2) = 8$ for $I_1 \cup \{b\}$.

Notice also that *B-Split* only searches a subset of possible generalizations allowed by the set-based generalization model, hence is efficient. To illustrate this, consider the generalization of $\{i_1, i_2, \ldots, i_{|\mathcal{P}|}\}$. Following Algorithm 1, *B-Split* may successively create generalized items $\tilde{i} = (i_1, i_2, \ldots, i_{|\mathcal{P}|})$, then $\tilde{i_l} = (i_1, i_2, \ldots, i_l)$, $\tilde{i_r} = (i_{l+1}, \ldots, i_{|\mathcal{P}|})$, and eventually $\tilde{i_1} = (i_1), \tilde{i_2} = (i_2), \ldots, \tilde{i_{|\mathcal{P}|}} = (i_{|\mathcal{P}|})$, if necessary. These generalizations form a binary tree, henceforth referred to as *generalization tree*, as depicted in Fig. 5. Each node corresponds to a distinct generalized item, which may be used to replace the leaf nodes (items) reachable from it. For instance, $\tilde{i_{l1}}$ may replace $\{i_1, \ldots, i_{l1}\}$. It is easy to see that, for each item $i \in \mathcal{P}$ considered for generalization, *B-Split* will explore a maximum of $|\mathcal{P}|$ generalized items. This is in contrast to exploring the entire space of $2^{|\mathcal{P}|} - 1$ possible set-based generalizations.

Thus, for a given $\mathcal{P}$, its corresponding generalization tree will contain all the generalized items that can be constructed by *B-Split*. Clearly, only those generalized items that form a partition of $\mathcal{P}$ can actually be used to generalize the items. We say that this collection forms a *generalization cut*, as explained in Definition 5.

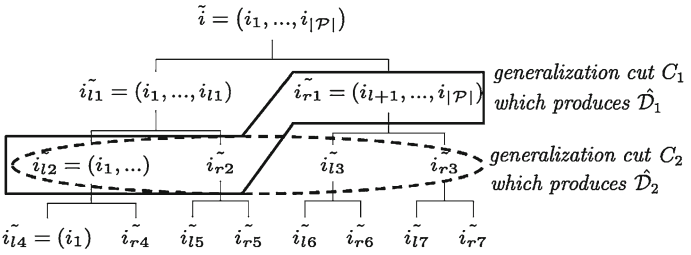


Fig. 5 A generalization tree

Definition 5 A *generalization cut C* is a set of generalized items corresponding to the nodes of a generalization tree, and for each and every $i \in \mathcal{P}$, there exists a unique $\tilde{i} \in C$ such that $\tilde{i} = \Phi(i)$.

Thus, a generalized dataset $\tilde{\mathcal{D}}$ can be constructed according to C by mapping each public item $i \in \mathcal{D}$ to a generalized item in C that contains i. For example, C_1 is a generalization cut in Fig. 5 which comprises of $\tilde{i_{l2}}$, $\tilde{i_{r2}}$ and $\tilde{i_{r1}}$, since every item in $\{i_1, \ldots, i_{|\mathcal{P}|}\}$ is mapped to one of the generalized items uniquely.

Given a generalization tree, multiple cuts are possible, and some may lead to producing more useful generalized datasets than the others. For example, C_1 and C_2 are two different generalization cuts in Fig. 5, which can be used to produce two different generalized datasets $\tilde{\mathcal{D}}_1$ and $\tilde{\mathcal{D}}_2$, but intuitively, C_2 is preferred as items in it are generalized less. To compare generalization cuts, we introduce a notion of *cut coverage*.

Definition 6 (*Cut coverage*) A generalization cut C_1 *covers* a cut C_2 if for each generalized item $\tilde{i} \in C_1$, we have either $\tilde{i} \in C_2$ or there exists a set of generalized items $\{\tilde{i_1}, \ldots, \tilde{i_s}\} \in C_2 \backslash C_1$ s.t. $\tilde{i_1} \cup \cdots \cup \tilde{i_s} = \tilde{i}$.

So a generalization cut C_1 covers another cut C_2 if every generalized item in C_1 is either a generalized item or an ancestor of some generalized items in C_2. For example, C_1 covers C_2 in Fig. 5 because $\tilde{i_{l2}}$ and $\tilde{i_{r2}}$ are in C_2, and $\tilde{i_{r1}}$ is an ancestor of $\tilde{i_{l3}}$ and $\tilde{i_{r3}}$ in C_2. Theorem 4 shows an important property concerning the coverage among generalization cuts. A similar property to that of Theorem 4 has been discussed in [35,74,75].

Theorem 4 *(*Monotonicity*) Given a generalization cut C_1 that covers a cut C_2, and generalized datasets $\tilde{\mathcal{D}}_1$ and $\tilde{\mathcal{D}}_2$, produced according to C_1 and C_2 respectively, $UL(\tilde{\mathcal{D}}_1) \geq UL(\tilde{\mathcal{D}}_2)$ holds when $w(\tilde{i}) \geq \sum_{\forall \tilde{i_q} \in desc(\tilde{i})} w(\tilde{i_q})$ for each $\tilde{i} \in C_1 \backslash C_2$, where $desc(\tilde{i})$ denotes the descendants of $\tilde{i}$ that appear in C_2.*

We also examine how the coverage among generalization cuts affects the protection of PS-rules. Consider two generalization cuts C_1 and C_2, such that C_1 covers C_2, and two generalized datasets $\tilde{\mathcal{D}}_1$ and $\tilde{\mathcal{D}}_2$ that are produced according to C_1 and C_2, respectively. Also, consider a PS-rule $I \rightarrow J$, and the generalized itemsets $\tilde{I}_1$ and $\tilde{I}_2$, which are constructed from I and are contained in C_1 and C_2, respectively. Then, as the support is monotonic [8], the following relations hold: (1) $sup(\tilde{I}_2, \tilde{\mathcal{D}}_2) \leq sup(\tilde{I}_1, \tilde{\mathcal{D}}_1)$, and (2) $sup(\tilde{I}_2 \cup J, \tilde{\mathcal{D}}_2) \leq sup(\tilde{I}_1 \cup J, \tilde{\mathcal{D}}_1)$. On the other hand, the confidence is non-monotonic [16], hence $conf(\tilde{I}_2 \rightarrow J, \tilde{\mathcal{D}}_2) \leq conf(\tilde{I}_1, \rightarrow J, \tilde{\mathcal{D}}_1)$ may not hold. Thus, the application of *B-Split* cannot increase the support of a rule, but may or may not decrease its confidence. If the confidence of a PS-rule is decreased, we can protect the rule with lower information loss, as we will discuss

in Sect. 4.2. So, applying *B-Split* can affect the protection of $I \rightarrow J$, as explained in the following cases:

1. $I \rightarrow J$ is protected in both $\tilde{\mathcal{D}}_1$ and $\tilde{\mathcal{D}}_2$. This occurs when $sup(\tilde{I}_2, \tilde{\mathcal{D}}_2)$ remains at least k and $conf(\tilde{I}_2 \rightarrow J, \tilde{\mathcal{D}}_2)$ is no more than c.
2. $I \rightarrow J$ is protected in $\tilde{\mathcal{D}}_1$, but it is not protected in $\tilde{\mathcal{D}}_2$. This occurs when $sup(\tilde{I}_2, \tilde{\mathcal{D}}_2)$ is less than k and/or $conf(\tilde{I}_2 \rightarrow J, \tilde{\mathcal{D}}_2)$ is greater than c.
3. $I \rightarrow J$ is not protected in $\tilde{\mathcal{D}}_1$, because $sup(\tilde{I}_1, \tilde{\mathcal{D}}_1)$ is less than k. In this case, $I \rightarrow J$ will not be protected in $\tilde{\mathcal{D}}_2$, because $sup(\tilde{I}_2, \tilde{\mathcal{D}}_2)$ will also be less than k.
4. $I \rightarrow J$ is not protected in $\tilde{\mathcal{D}}_1$, because $conf(\tilde{I}_1 \rightarrow J, \tilde{\mathcal{D}}_1)$ is greater than c. In this case, if $sup(\tilde{I}_2, \tilde{\mathcal{D}}_2)$ is at least k, then $I \rightarrow J$ will be protected in $\tilde{\mathcal{D}}_2$ when $conf(\tilde{I}_2 \rightarrow J, \tilde{\mathcal{D}}_2)$ is at most c, or not protected in $\tilde{\mathcal{D}}_2$, when $conf(\tilde{I}_2 \rightarrow J, \tilde{\mathcal{D}}_2)$ is greater than c.

Observe that, in Case 2, the application of *B-Split* may increase the confidence of a PS-rule. Thus, a rule that is protected in $\tilde{\mathcal{D}}_1$ will not be protected in $\tilde{\mathcal{D}}_2$, if its confidence in $\tilde{\mathcal{D}}_2$ is greater than c. Consider, for example, the PS-rule $cd \rightarrow \mathbf{g}$ and the datasets of Fig. 1b, d as $\tilde{\mathcal{D}}_1$ and $\tilde{\mathcal{D}}_2$, respectively. Given $k = 2$ and $c = \frac{1}{3}$, the rule is protected in $\tilde{\mathcal{D}}_1$ but not protected in $\tilde{\mathcal{D}}_2$, because its confidence in $\tilde{\mathcal{D}}_2$ is greater than $\frac{1}{3}$. On the other hand, in Case 4, applying *B-Split* may decrease the confidence of PS-rule. Thus, a rule that is not protected in $\tilde{\mathcal{D}}_1$ is protected in $\tilde{\mathcal{D}}_2$, if its confidence in $\tilde{\mathcal{D}}_2$ is no more than c. For instance, consider the PS-rule $a \rightarrow \mathbf{j}$, and that $k = 2$ and $c = \frac{1}{2}$. The rule is not protected in $\tilde{\mathcal{D}}_1$, but it is protected in $\tilde{\mathcal{D}}_2$, as it confidence in $\tilde{\mathcal{D}}_2$ is no more than $\frac{1}{2}$.

4.1.2 Identifying PS-rules that can be pruned

In this section, we establish a set of criteria that can be used to determine whether a PS-rule can be pruned. A prunable rule is one whose support and confidence do not need to be computed in order to decide whether it is protected in a generalized dataset or not. As we will show in our experiments, rule pruning is essential to designing algorithms that are applicable to realistically large datasets, based on our privacy model.

We start by considering PS-rules that are protected in any dataset $\tilde{\mathcal{D}}$ that can be constructed from $\mathcal{D}$, using set-based generalization. Such rules satisfy the conditions provided in Theorem 5.

Theorem 5 *Given a dataset $\mathcal{D}$, and parameters k and c, a PS-rule $I \rightarrow J$ is protected in any generalized dataset $\tilde{\mathcal{D}}$ that can be constructed from $\mathcal{D}$ using set-based generalization, if (1) $sup(I, \mathcal{D}) \geq k$, and (2) $sup(J, \mathcal{D}) \leq c \times k$.*

Consider, for instance, the original dataset of Fig. 1a, and a PS-rule $a \rightarrow \mathbf{ghij}$. Observe that the rule satisfies both conditions of Theorem 5, for $k = 2$ and $c = 0.5$, and it is protected in the generalized versions of the dataset of Fig. 1a that are shown in Fig. 1b, d. Clearly, the rules that satisfy the conditions of Theorem 5 do not need to be contained in Θ, as they will always be protected. It is easy to see that eliminating these rules is straightforward; thus, we henceforth assume that Θ does not contain such rules.

We then establish a theorem that helps identify those PS-rules that become unprotected as one of the generalized items in a cut gets split by *B-Split*.

Theorem 6 *Assume that two generalized datasets $\tilde{\mathcal{D}}_1$ and $\tilde{\mathcal{D}}_2$ are constructed from $\mathcal{D}$ according to cuts C_1 and C_2 respectively, and that C_2 is derived from C_1 by splitting a generalized item $\tilde{i}$ in C_1 into $\tilde{i}_l$ and $\tilde{i}_r$. If a PS-rule $I \rightarrow J$ is protected in $\tilde{\mathcal{D}}_1$ but not in $\tilde{\mathcal{D}}_2$, then I contains at least one item i s.t. $\Phi(i) = \tilde{i}_l$ or $\Phi(i) = \tilde{i}_r$.*

Example 4 Consider the dataset of Fig. 1d, and a PS-rule $cd \rightarrow \mathbf{g}$ that is protected in it w.r.t. $k = 2$ and $c = 0.5$. It is easy to verify that this rule is not protected in the dataset of Fig. 1a, in which (b, c) is split into (b) and (c) as a result of applying *B-Split* to (b, c). Observe that item c is contained in the antecedent of this rule.

While a large number of public items are typically included in a dataset, attackers are not expected to know many of them [74,91], so few items will actually be contained in the antecedent of a PS-rule. This suggests that, as generalized items are split by *B-Split* into smaller and smaller ones, it becomes more and more likely that the items contained in the antecedents of the PS-rules will not overlap with each other, which implies that these rules are prunable according to Theorem 6. This pruning is particularly effective at the later stages of our generalization process, when a large number of generalized items is created by *B-Split* and many PS-rules are to be checked against these items.

The PS-rules whose antecedents are subsets of the antecedent of a protected rule may also be prunable, as Theorem 7 states.

Theorem 7 *Given a generalized dataset $\tilde{\mathcal{D}}$ and two PS-rules $I \rightarrow J$ and $I' \rightarrow J$ s.t. (1) $I \rightarrow J$ is protected in $\tilde{\mathcal{D}}$, (2) $I' \subseteq I$, and (3) $\bigcup_{\forall i \in I} \Phi(i)$ is a single generalized item in $\tilde{\mathcal{D}}$, then $I' \rightarrow J$ is protected in $\tilde{\mathcal{D}}$.*

For example, $bc \rightarrow \mathbf{h}$ is protected in the dataset of Fig. 1(d), for $k = 2$ and $c = 0.5$. Based on Theorem 7, $b \rightarrow \mathbf{h}$ is also protected in the same dataset, as $b \subseteq bc$ and the items contained in the antecedent of the rule are mapped to a single generalized item (b, c).

In addition, when a PS-rule $I \rightarrow J$ is protected, then another rule with the same antecedent I may be prunable, if its consequent satisfies a condition that is provided in Theorem 8.

Theorem 8 *Given a generalized dataset $\tilde{\mathcal{D}}$ and two PS-rules $I \rightarrow J$ and $I \rightarrow J'$ such that (1) $I \rightarrow J$ is protected in $\tilde{\mathcal{D}}$, and (2) $sup(J, \tilde{\mathcal{D}}_{\tilde{I}}) \geq sup(J', \tilde{\mathcal{D}}_{\tilde{I}})$, where $\tilde{\mathcal{D}}_{\tilde{I}}$ is the set of supporting transactions of $\tilde{I} = \bigcup_{\forall i \in I} \Phi(i)$ in $\tilde{\mathcal{D}}$, then $I \rightarrow J'$ is also protected in $\tilde{\mathcal{D}}$.*

A special case to observe is that if J' is a superset of J, then $sup(J', \tilde{\mathcal{D}}) \leq sup(J, \tilde{\mathcal{D}})$. Thus $I \rightarrow J'$ can be pruned, if $I \rightarrow J$ is protected, based on Theorem 8. To illustrate this, we revisit Example 4. According to Theorem 8, $cd \rightarrow \mathbf{gh}$, constructed by adding $\mathbf{h}$ to the consequent of the protected rule $cd \rightarrow \mathbf{g}$, for $k = 2$ and $c = 0.5$, in the dataset of Fig. 1d, is also protected in this dataset for the same k and c, because the support of $\mathbf{gh}$ is smaller than that of $\mathbf{h}$.

4.1.3 Efficient checking for PS-rule protection

Based on Theorems 6–8, we develop the *Check* function which can be used to determine whether a set of PS-rules[5] are protected in a generalized dataset efficiently. Algorithm 2 explains how *Check* works.

We first construct and check a PS-rule, based on Theorems 7 and 8. Specifically, in steps 1–5, we assign the set of all items mapped to $\tilde{i}$ to I, and the most supported sensitive item in $\tilde{\mathcal{D}}_{\tilde{i}}$ to J. Next, we check whether $I \rightarrow J$ is protected in $\tilde{\mathcal{D}}$ (step 4). If $I \rightarrow J$ is protected, it can be used to prune other rules.

Steps 7–18 describe how rules are checked. First, we find all PS-rules in Θ, whose antecedent contains at least one item in $\tilde{i}$ and is potentially unprotected in $\tilde{\mathcal{D}}$ according

[5] Recall from Sect. 4.1.2 that Θ does not contain rules that are always protected.

Algorithm 2 *Check*$(\tilde{i}, \tilde{\mathcal{D}}, \Theta, k, c)$

input: Generalized item $\tilde{i}$, generalized version $\tilde{\mathcal{D}}$ of $\mathcal{D}$, a set of PS-rules Θ, k, c
output: 1 if $\forall r \in \Theta$,r is protected in $\tilde{\mathcal{D}}$,
0 if $\forall r \in \Theta$, r's support is at least k and $\exists r \in \Theta$, r's confidence exceeds c
-1 if $\exists r \in \Theta$, r's support is less than k

1. $I \leftarrow \bigcup_{\forall i \in \tilde{i}} i$ ◁ I contains each i mapped to $\tilde{i}$
2. $J \leftarrow \underset{\forall i \in \mathcal{S}}{\text{argmax}}\, sup(i, \tilde{\mathcal{D}}_{\tilde{i}})$
3. $pr \leftarrow false$
4. **if** ($I \rightarrow J$ is protected in $\tilde{\mathcal{D}}$)
5. $pr \leftarrow true$
6. $exceeds\_c \leftarrow false$
7. $\Theta' \leftarrow \{I' \rightarrow J' | I' \rightarrow J' \in \Theta$ and $\exists i \in I', \Phi(i) = \tilde{i}\}$
8. **foreach** ($I' \rightarrow J'$ in Θ')
9. **if** (pr =*true* and $I' \subseteq I$ and $sup(J', \tilde{\mathcal{D}}_{\tilde{i}}) \leq sup(J, \tilde{\mathcal{D}}_{\tilde{i}})$)
10. **continue** ◁ proceeds to the next iteration
11. **if** ($sup(I', \tilde{\mathcal{D}}) < k$)
12. **return** -1 ◁ return terminates execution
13. **if** ($conf(I' \rightarrow J', \tilde{\mathcal{D}}) > c$)
14. $exceeds\_c \leftarrow true$
15. **if** ($exceeds\_c$ =*true*)
16. **return** 0
17. **else**
18. **return** 1

to Theorem 6, and assign them to Θ' (step 7). How we do this efficiently will be explained shortly. Then, we iterate over the rules in Θ' to examine whether they are protected in $\tilde{\mathcal{D}}$ (step 8). In steps 9–10, we check whether pr=*true*, I' is a subset of I, and J' is supported by no more transactions in $\tilde{\mathcal{D}}_{\tilde{i}}$ than J is. If it is, we proceed to the next iteration, because $I' \rightarrow J'$ can be pruned (Theorems 7 and 8). In step 11, we compute $sup(I', \tilde{\mathcal{D}})$, because $I' \rightarrow J$ has not been pruned. If $sup(I', \tilde{\mathcal{D}}) < k$, we know that at least one rule is not protected in $\tilde{\mathcal{D}}$, so we return -1 (steps 11–12). Otherwise, we check and record if $conf(I' \rightarrow J', \tilde{\mathcal{D}}) > c$ (steps 13–14). When *Check* has finished with rule checking, it returns 0 if at least one rule violates the confidence requirement, or 1 otherwise (steps 15–18).

Efficient construction of Θ' *Check* can be executed more efficiently if Θ' in Algorithm 2 can be constructed quickly. For this purpose, we introduce *Rule-tree*, a data structure for organizing PS-rules, and *Find-rules*, a method for constructing Θ', based on the Rule-tree.

A Rule-tree for a set of PS-rules is comprised of an *item header table*, a *tree structure*, and a set of *consequent-lists*, as shown in Fig. 6. The item header table and tree structure contain the items of the antecedents of PS-rules sorted in a descending order of support, which is chosen because it results in a smaller tree [80] that allows faster rule retrieval. These are similar to the components of an FP-tree [80], but are built to retrieve PS-rules and not frequent itemsets. *Consequent-lists* contain the items of the consequents of the rules. The Rule-tree is constructed in two steps. First, the items in the antecedents of PS-rules in Θ are scanned and sorted. Then, all items of the rules in Θ are scanned to build the tree.

Each entry in the item header table corresponds to a distinct item i contained in the antecedents of rules in Θ, and points to a tree node labeled i via a *node-link*. We will refer to a node labeled i as "node i", when it is clear from the context. The labels of nodes correspond to items in the antecedent of rules, except for the root which is "null". Each node has a *node-link* that points to the next item with the same label, if such an item exists, so that all nodes with the same label are linked in sequence via node-links. Also, the node labeled with the last item of the antecedent of a rule has a *consequent-link*, which points to a *consequent-list*. A path from the root to the last item of the consequent-list represents a PS-rule, and the

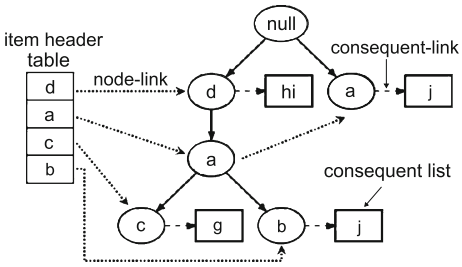


Fig. 6 Example of a Rule-tree

antecedents of rules that share a prefix of items are compressed (i.e., represented using the same nodes). Example 5 illustrates the Rule-tree.

Example 5 Consider the dataset of Fig. 1a and a set of PS-rules $\{a \rightarrow \mathbf{j}, acd \rightarrow \mathbf{g}, adb \rightarrow \mathbf{j}, d \rightarrow \mathbf{hi}\}$. The Rule-tree built from these rules is shown in Fig. 6. Items d, a, c and b in the item header table are connected to the nodes labeled with these items via node-links, and the link between two nodes a indicates that the antecedents of two rules contain a. The path from d to g corresponds to $adc \rightarrow \mathbf{g}$, and node d precedes node a in this path, since the support of d is greater than that of a in Fig. 1a. Also observe that each consequent of the rules appears as a consequent-list in the tree structure.

Traversing the branches of a Rule-tree allows all rules whose antecedents contain an item i to be found efficiently for two reasons. First, all nodes i in the tree are connected through node-links, which implies that examining rules that do not contain i in their antecedents is avoided. Second, since the antecedents of rules that share a common prefix of items correspond to nodes that lie on the same path in the tree, we do not need to examine these nodes more than once.

Algorithm 3 *Find-rules*$(\mathcal{R}, \tilde{i_l}, \tilde{i_r})$

input: Rule-tree $\mathcal{R}$ for Θ, generalized items $\tilde{i_l}, \tilde{i_r}$
output: Set of PS-rules Θ' s.t. $\Phi(i) = \tilde{i_l}$ or $\Phi(i) = \tilde{i_r}$, for any i in the antecedent of these rules

$\Theta' \leftarrow \varnothing$
foreach (i such that $\Phi(i) = \tilde{i_l}$ or $\Phi(i) = \tilde{i_r}$)
  find the node v labeled i in $\mathcal{R}$ following the node-link from i in the item header table of $\mathcal{R}$
  $\Lambda_{low} \leftarrow \varnothing$
  $\Lambda_{up} \leftarrow \varnothing$
  while (*true*)
    foreach (path p_r from v to a node $v' \in \mathcal{R}$ that has a consequent-link)
      $\lambda_a(p_r) \leftarrow$ the set of labels of the nodes in p_r
      $\lambda_c(p_r) \leftarrow$ the set of items of the consequent-list to v'
      $\Lambda_{low} \leftarrow \Lambda_{low} \cup \{\lambda_a(p_r), \lambda_c(p_r)\}$
    $\Lambda_{up} \leftarrow$ the set of nodes in the path from the immediate ascendant of v to the root of $\mathcal{R}$
    for each (pair $\{\lambda_a(p_r), \lambda_c(p_r)\}$ in Λ_{low})
      $I \leftarrow$ the set of items of the lists $\lambda_a(p_r)$ and Λ_{up}
      $J \leftarrow$ the set of items of $\lambda_c(p_r)$
      $\Theta' \leftarrow \Theta' \cup \{I \rightarrow J\}$
    if (v has a node-link)
      $v \leftarrow$ the node pointed by the node-link of v
    else
      break
return Θ'

To construct Θ' by traversing a Rule-tree, we provide *Find-rules*, described in Algorithm 3. We first find i that maps to either $\tilde{i_l}$ or $\tilde{i_r}$ in the item header table of a Rule-tree $\mathcal{R}$, and follow

its node-link to the first node ν labeled i in the tree (step 3). Then, in steps 7–10, we traverse each path p_r from ν to a node $\nu' \in \mathcal{R}$ that has a consequent-link and insert it into Λ_{low} as a pair of lists $\lambda_a(p_r)$ and $\lambda_c(p_r)$, containing the labels of the tree nodes in p_r and all items in the consequent-list of ν', respectively. Next, we follow the path from the immediate ascendant of ν to the root and insert the labels of all these nodes into Λ_{up} (step 11). Following that, we iterate over each pair of lists $\{\lambda_a(p_r), \lambda_c(p_r)\}$ in Λ_{low} to create a rule whose antecedent contains all items of $\lambda_a(p_r)$ and Λ_{up} and consequent all items of $\lambda_c(p_r)$, and add this rule to Θ' (steps 12–15). If $\mathcal{R}$ contains another node with the same label as ν, this node is assigned to ν and the process is repeated.

Example 6 Consider applying *Find-rules* using the Rule-tree given in Fig. 6, $\tilde{i_l} = (b, c)$ and $\tilde{i_r} = (a)$. We first find rules whose antecedent contains a. This is achieved by finding a in the item header table of the Rule-tree, and examining the path (p_1) from the leftmost node a to node c. From this path, we create $\lambda_a(p_1)$ to contain a and c and $\lambda_c(p_1)$ to contain $\{\mathbf{g}\}$, and add the pair to Λ_{low}. We then do the same for the path p_2 from the leftmost node labeled a to node b and add the pair $\lambda_a(p_2) = \{a, b\}$ and $\lambda_c(p_2) = \{\mathbf{j}\}$ to Λ_{low}. Then, the ascendant nodes of a are followed up to the root and added to Λ_{up}. Now we iterate over Λ_{low} to construct PS-rules: $acd \rightarrow \mathbf{g}$ and $abd \rightarrow \mathbf{j}$, and add them to Θ'. All other rules for Θ' are obtained similarly.

4.2 Anonymization algorithms

In this section, we present two anonymization algorithms. The first one, called *Tree-based Anonymization*, works in a top-down fashion. It progressively builds a generalization tree, checking the rules each time the tree is expanded. We also propose a more scalable algorithm, called *Sample-based Anonymization*. Using sampling and a combination of top-down and bottom-up generalization strategies, this algorithm avoids the large number of expensive dataset scans performed by Tree-based Anonymization, while enhancing data utility.

4.2.1 Tree-based anonymization

Tree-based Anonymization starts by mapping all public items of the dataset to be generalized to a single (most) generalized item, and then attempts to split this item as far as possible to enhance utility. When a generalized item $\tilde{i}$ is split into $\tilde{i_l}$ and $\tilde{i_r}$, we check whether the specified PS-rules that are protected with respect to $\tilde{i}$ are still protected with respect to $\tilde{i_l}$ and $\tilde{i_r}$. There can be three possible outcomes. If all rules are protected, we simply continue to split $\tilde{i_l}$ and $\tilde{i_r}$, constructing our generalization tree recursively. If there is at least one rule whose support is less than k, then we do not replace $\tilde{i}$ by $\tilde{i_l}$ and $\tilde{i_r}$ in the generalized data, nor split these generalized items any further. If the support of every rule is at least k, but there are some rules whose confidence exceeds c, then we explore if further splitting $\tilde{i_l}$ or $\tilde{i_r}$ may help reduce confidence. Although it is safe to stop before splitting $\tilde{i_l}$ or $\tilde{i_r}$, as the algorithm of [52] does, Tree-based Anonymization further splits these generalized items. This allows constructing anonymized datasets that protect rules with significantly better utility, particularly when c is large.

The pseudocode of Tree-based Anonymization is given in Algorithm 4. In steps 1–2, we check whether a single item is mapped to $\tilde{i}$. If so, we return the generalized dataset $\tilde{\mathcal{D}}$, assuming, for simplicity, that more than 1 items are mapped to the most generalized item $\tilde{i}$ with which the algorithm is initially called. In steps 3–4, we split $\tilde{i}$ into $\tilde{i_l}$ and $\tilde{i_r}$, using *B-Split*, and construct a temporary dataset $\mathcal{D}'$ using *Update*, which maps items in $\mathcal{D}$ to $\tilde{i_l}$ or

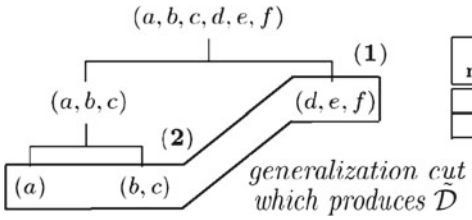
Algorithm 4 Tree-based($\tilde{i}, \tilde{\mathcal{D}}, \mathcal{D}, \Theta, k, c$)

input: Generalized item $\tilde{i}$, set of PS-rules Θ, k, c
original dataset $\mathcal{D}$, generalized dataset $\tilde{\mathcal{D}}$
output: Anonymized dataset $\tilde{\mathcal{D}}$

1. **if** ($|\tilde{i}| = 1$)
2. **return** $\tilde{\mathcal{D}}$
3. $\{\tilde{i_l}, \tilde{i_r}\} \leftarrow$ *B-Split*($\tilde{i}$)
4. $\mathcal{D}' \leftarrow$ *Update*($\tilde{\mathcal{D}}, \mathcal{D}$) ◁ temporary dataset
5. **if** (*Check*($\tilde{i_l}, \mathcal{D}', \Theta, k, c$) $= -1$ or *Check*($\tilde{i_r}, \mathcal{D}', \Theta, k, c$) $= -1$)
6. $u \leftarrow -1$
7. **else**
8. $u \leftarrow$ *Check*($\tilde{i_l}, \mathcal{D}', \Theta, k, c$) $\times$ *Check*($\tilde{i_r}, \mathcal{D}', \Theta, k, c$)
9. **if** ($u = 1$) ◁ rules are protected
10. $\tilde{\mathcal{D}} \leftarrow \mathcal{D}'$
11. $\tilde{\mathcal{D}}' \leftarrow$ *Tree-based*($\tilde{i_l}, \tilde{\mathcal{D}}, \mathcal{D}, \Theta, k, c$)
12. $\tilde{\mathcal{D}}'' \leftarrow$ *Tree-based*($\tilde{i_r}, \tilde{\mathcal{D}}, \mathcal{D}, \Theta, k, c$)
13. **return** *Replace*($\tilde{i_l}, \tilde{i_r}, \tilde{\mathcal{D}}, \tilde{\mathcal{D}}', \tilde{\mathcal{D}}'', \mathcal{D}$)
14. **else if** ($u = -1$) ◁ the support of a rule is less than k
15. **return** $\tilde{\mathcal{D}}$
16. **else** ◁ the confidence of a rule exceeds c
17. create an empty queue Q
18. $Q.insert(\tilde{i_l})$
19. $Q.insert(\tilde{i_r})$
20. **while** (Q is not empty)
21. $\{\tilde{i_l'}, \tilde{i_r'}\} \leftarrow$ *B-Split*($Q.delete()$)
22. $\mathcal{D}'' \leftarrow$ *Update*($\mathcal{D}', \mathcal{D}$) ◁ temporary dataset
23. **if** (*Check*($\tilde{i_l'}, \mathcal{D}'', \Theta, k, c$) $= 1$ and *Check*($\tilde{i_r'}, \mathcal{D}'', \Theta, k, c$) $= 1$)
24. $\tilde{\mathcal{D}} \leftarrow \mathcal{D}''$
25. delete all elements of Q
26. **return** $\tilde{\mathcal{D}}$
27. **else if** (*Check*($\tilde{i_l'}, \mathcal{D}'', \Theta, k, c$) $= -1$ or *Check*($\tilde{i_r'}, \mathcal{D}'', \Theta, k, c$) $= -1$)
28. delete all elements of Q
29. **return** $\tilde{\mathcal{D}}$
30. **else**
31. **if** ($|\tilde{i_l'}| > 1$)
32. $Q.insert(\tilde{i_l'})$
33. **if** ($|\tilde{i_r'}| > 1$)
34. $Q.insert(\tilde{i_r'})$
35. **return** $\tilde{\mathcal{D}}$

$\tilde{i_r}$. Then, we apply *Check* on $\tilde{i_l}$ and $\tilde{i_r}$ to determine whether the PS-rules in Θ are protected in $\mathcal{D}'$ (steps 5–8). If they are, we assign $\mathcal{D}'$ to the generalized dataset $\tilde{\mathcal{D}}$ and recursively apply the algorithm first to $\tilde{i_l}$ and then to $\tilde{i_r}$ (steps 9–12). Next, we use *Replace*, which constructs an anonymized dataset in which the items in $\mathcal{D}$ that were mapped to $\tilde{i_l} \in \tilde{\mathcal{D}}$ and $\tilde{i_r} \in \tilde{\mathcal{D}}$ are replaced by their corresponding generalized items in $\tilde{\mathcal{D}}'$ and $\tilde{\mathcal{D}}''$, respectively, and return the result (step 13). Otherwise, if there is at least one rule whose support is less than k in $\mathcal{D}'$, we return $\tilde{\mathcal{D}}$ (steps 14–15). This is because, as explained in Sect. 4.1.1, further splitting the generalized items in $\tilde{\mathcal{D}}'$ and $\tilde{\mathcal{D}}''$ cannot increase the support of rules.

Steps 16–34 deal with the case when all rules have acceptable support but some have a larger confidence than c in $\mathcal{D}'$. In this case, we explore if further splitting $\tilde{i_l}$ or $\tilde{i_r}$ can reduce the confidence of these rules. Recall from Sect. 4.1.1 that this may help the protection of PS-rules. This is because, a PS-rule whose antecedent contains $\tilde{i_l}$ or $\tilde{i_r}$, and it is not protected due to its confidence, can become protected after further splitting these generalized items. We use an initially empty queue Q to store the generalized items $\tilde{i_l}$ and $\tilde{i_r}$ (steps 17–19), and then process them in turn (steps 20–34). Specifically, the first element on Q (returned by the *delete* operation) is split into two new generalized items $\tilde{i_l'}$ and $\tilde{i_r'}$ by *B-Split* (step 21) and



Split number	set of PS-rules Θ′ returned by *Find-rules*	*Check* returns
(1)	{a → j, cd → g, d → hi}	0
(2)	{a → j, cd → g}	1

Fig. 7 Example of Tree-based anonymization

Purchased items
(a, b, c) (d, e, f) **g h i j**
(d, e, f) **h i**
(d, e, f) **g j**
(d, e, f) **g h**
(a, b, c) (d, e, f) **i**
(a, b, c) (d, e, f) **j**

(a)

Purchased items
a (b, c) (d, e, f) **g h i j**
(d, e, f) **h i**
(d, e, f) **g j**
(d, e, f) **g h**
a (b, c) (d, e, f) **i**
(b, c) (d, e, f) **j**

(b)

Fig. 8 Temporary datasets **a** $\mathcal{D}'$, and **b** $\mathcal{D}''$

a temporary dataset $\mathcal{D}''$ is constructed based on them (step 22). If the rules are protected in $\mathcal{D}''$, then we assign $\mathcal{D}''$ to $\tilde{\mathcal{D}}$ and exit the current recursion after emptying Q (steps 23–26). Otherwise, if there is a rule with support less than k in $\mathcal{D}''$, we exit the current recursion, returning $\tilde{\mathcal{D}}$ after emptying Q (steps 27–29). Finally, if all rules have acceptable support but there is one with a confidence larger than c in $\mathcal{D}''$, we insert $\tilde{i_{ll}}$ and $\tilde{i_{lr}}$ into Q so that they can be considered for further split (steps 20–34), and return $\tilde{\mathcal{D}}$ when every item on Q has been processed (step 35).

Example 7 Consider applying Algorithm 4 to the data of Fig. 1a w.r.t. the PS-rules of Fig. 1e, $k = 2$ and $c = 0.5$. We start with $\tilde{i} = (a, b, c, d, e, f)$ and obtain $\tilde{i_l} = (a, b, c)$ and $\tilde{i_r} = (d, e, f)$ by applying *B-Split* to $\tilde{i}$ (split (1) in Fig. 7), after setting the weights using the hierarchy of Fig. 1c, as in [51]. Based on $\tilde{i_l}$ and $\tilde{i_r}$, the temporary dataset of Fig. 8a is constructed, and *Check* is called to examine whether the rules are protected in it. According to *Find-Rules*, the rules shown in the first row in Fig. 7 need to be checked in this case and *Check* returns 0. This means that we need to explore (a, b, c) and (d, e, f) further, so we add them to Q. Assuming that (a, b, c) is the first element of Q, (a, b, c) is split by *B-Split* into (a) and (b, c) (split (2) in Fig. 7), and the dataset of Fig. 8b is created based on this split. *Check* is then called to examine the rules in the second row in Fig. 7. This time, *Check* returns 1. Thus, (a), (b, c) and (d, e, f) form a generalization cut that produces the dataset of Fig. 8b. The rules are protected in the latter dataset, so it is returned.

The worst-case time complexity of Algorithm 4 is $O(2^{|\mathcal{P}|} \times |\mathcal{S}| \times N)$ and the space complexity is $O(2^{|\mathcal{P}|} \times |\mathcal{S}| + N \times |\mathcal{I}|)$, when all possible PS-rules are specified (see Theorem 10 in "Appendix"). However, attackers are unlikely to possess knowledge about a large number of items in $\mathcal{P}$ [16,74,75,91] so the number of specified rules is expected to be significantly smaller. Note that the time complexity of Apriori is also exponential in the domain size and parameter m [74], but Algorithm 4 was faster in our experiments.

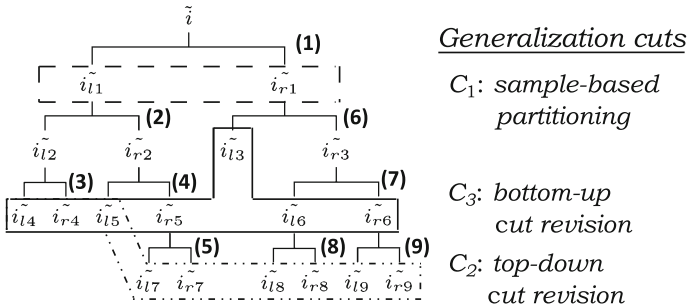


Fig. 9 Example of generalization tree

4.2.2 Sample-based Anonymization

The main idea of *Sample-based Anonymization* is to use sampling to quickly construct a generalized dataset $\tilde{\mathcal{D}}$ in which most PS-rules will be protected and then to slightly modify some of the generalized items in it to protect the remaining rules. Details of Sample-based Anonymization are given in Algorithm 5.

The algorithm starts with a generalized dataset $\tilde{\mathcal{D}}$ that contains a single (most) generalized item and works in three phases. In the *sample-based partitioning* phase, it progressively builds a generalization tree just like Tree-based Anonymization does, but *Check* is efficiently performed on a random sample of $\tilde{\mathcal{D}}$. However, the generalized dataset derived from this sample-based process may be overly generalized or insufficiently protected. Thus, *Sample-based Anonymization* proceeds to revising the generalization cut that produced $\tilde{\mathcal{D}}$. In the first, *top-down cut revision* phase, generalized items are further split to enhance utility, while, in the next, *bottom-up cut revision* phase, we iteratively merge generalized items to ensure protection. Figure 9 shows these phases and the generalization cuts derived: C_1 is derived by *sample-based partitioning*, C_2 is produced by further splitting some generalized items of C_1 in *top-down cut revision*, and C_3 is obtained by merging those of C_2 in *bottom-up cut revision*.

Sample-based partitioning The two key functions performed in this phase (steps 1–3) are *Create-sample* and *Sample-Based-Partition*. *Create-sample* takes a random sample $\mathcal{D}_s$ with replacement from $\mathcal{D}$ by selecting $ln(\frac{2}{\delta})/(2 \times \epsilon^2)$ transactions from $\mathcal{D}$, where δ and ϵ are parameters selected by data publishers. As explained in Theorem 9, a sample of this size ensures that, for a PS-rule $I \rightarrow J$, the difference between the fraction of transactions supporting I in $\mathcal{D}$ and $\mathcal{D}_s$ will be bounded by an $\epsilon > 0$ with a probability at most δ.

Theorem 9 *Let $\mathcal{D}$ be an original dataset, $\mathcal{D}_s$ be a random sample constructed from $\mathcal{D}$ with replacement and with transactions projected over $\mathcal{P}$, and $I \rightarrow J$ be a PS-rule. It holds that the probability $Pr\left(|\frac{sup(I,\mathcal{D}_s)}{|\mathcal{D}_s|} - \frac{sup(I,\mathcal{D})}{|\mathcal{D}|}| \geq \epsilon\right)$ is at most δ, for $\delta \in [0, 1]$ and $\epsilon > 0$, when $|\mathcal{D}_s| = ln(\frac{2}{\delta})/(2 \times \epsilon^2)$.*

It is easy to see that this bound is independent of the size of $\mathcal{D}$ and that to estimate support more accurately by using a smaller δ and/or ϵ would result in a larger sample [33]. In addition, the same bound holds for the difference between the fraction of transactions supporting $\tilde{I} = \bigcup_{\forall i \in I} \Phi(i)$ in $\tilde{\mathcal{D}}_s$ and $\tilde{\mathcal{D}}$, if the same mapping function Φ is used. Clearly, when this difference is small, we could expect most of the specified PS-rules to be protected

Algorithm 5 Sample-based($\tilde{i}$, $\tilde{\mathcal{D}}$, $\mathcal{D}$, Θ, k, c)

input: Generalized item $\tilde{i}$, set of PS-rules Θ, k, c, original dataset $\mathcal{D}$, generalized dataset $\tilde{\mathcal{D}}$
output: Anonymized dataset $\tilde{\mathcal{D}}$

```
    /* Sample-based partitioning phase */
1.  D_s ← Create-sample(δ, ε, D)
2.  R ← Rule-tree for Θ the set of PS-rules Θ
3.  G ←Sample-Based-Partition(ĩ, D̃, D_s, Θ, k, c, R)
    /* Top-down cut revision phase */
4.  create an empty queue Q
5.  foreach(leaf-level node u ∈ G from left to right)
6.      ĩ_g ← generalized item that corresponds to u
7.      Q.insert(ĩ_g)
8.  if(Check-all-leaves(Q, D̃, Θ, k, c, R)=true)
9.      while(Q is not empty)
10.         ĩ_q ← Q.delete()
11.         {ĩ_l, ĩ_r} ← Split(ĩ_q)
12.         D' ←Update(D̃, D)          ◁ temporary dataset
13.         if (Find-rules(R, ĩ_l, ĩ_r) ≠ ∅)
14.             if(Check-leaves(ĩ_l, ĩ_r, D', Θ, k, c)=true)
15.                 if (|ĩ_l| > 1)
16.                     Q.insert(ĩ_l)
17.                 if (|ĩ_r| > 1)
18.                     Q.insert(ĩ_r)
19.         else
20.             if (|ĩ_l| > 1)
21.                 Q.insert(ĩ_l)
22.             if (|ĩ_r| > 1)
23.                 Q.insert(ĩ_r)
    /* Bottom-up cut revision phase */
24. foreach(leaf-level node u ∈ G from left to right)
25.     ĩ_q ← generalized item that corresponds to u
26.     if (Check(ĩ_q, D', Θ, k, c)=1)
27.         Q.delete()
28.         D̃ ← D'
29.     else
30.         Merge-siblings(ĩ_q, D', G)
31. return D̃
```

in the generalized dataset constructed in this phase, and few generalized items in it to be modified during the cut revision phases.

Sample-based-partition is similar to Algorithm 4, except for the following. First, it uses the sample $\mathcal{D}_s$ in place of $\mathcal{D}$, thus *Check* is performed on a much smaller dataset, hence much more efficiently. Second, it performs rule checking using only the support of PS-rules in $\tilde{\mathcal{D}}$. Thus, it constructs generalization cuts that often contain a larger number of generalized items than those created by Algorithm 4. This is because Sample-based Anonymization always splits a generalized item, when the rules have acceptable support but excessive confidence, but Algorithm 4 may not do so (see steps 16–34). Third, instead of just finding a generalization cut, it constructs and returns a generalization tree $\mathcal{G}$. This is necessary for the *bottom-up cut revision* phase.

Top-down cut revision In the top-down cut revision phase (steps 4–23), we attempt to find a generalization cut that lies as low as possible in the generalization tree, in order to enhance data utility. We first populate an initially empty queue Q with the generalized items that correspond to leaf nodes in $\mathcal{G}$ (steps 4–7), and then check whether the support of rules in $\tilde{\mathcal{D}}$ is at least k, using *Check-all-leaves*. This function is similar to *Check* except that it is based on computing only the support of rules in $\tilde{\mathcal{D}}$ and that it takes all generalized items in Q as input. If the rules have acceptable support, then we split each generalized item on

Q, and construct a temporary dataset $\mathcal{D}'$ accordingly (steps 8–12). Which rules to check is determined by *Find-rules* (step 13). If there are rules to be checked and they have a support of at least k in $\mathcal{D}'$, then we add $\tilde{i}_l$ and $\tilde{i}_r$ to Q for further split consideration (steps 14–18). If *Find-rules* finds no rules, then it means that all the rules are protected, so we add $\tilde{i}_l$ and $\tilde{i}_r$ to Q for further split (steps 19–23). Thus, in this phase, the algorithm does not take into account the confidence of PS-rules, but continues splitting generalized items based only on the support of rules. This strategy helps utility (see Theorem 4), while it may lead to protecting rules that are not protected in a more heavily generalized dataset due to their confidence (see Case 4 in Sect. 4.1.1).

Bottom-up cut revision The bottom-up cut revision phase (steps 24–30) begins by examining all leaf-level nodes of the generalization tree $\mathcal{G}$ from left to right (step 24). For each such generalized item $\tilde{i}_q$, we use *Check* to determine whether the rules are protected in the temporary dataset $\mathcal{D}'$ (steps 25–26). If all the rules are protected, then $\tilde{i}_q$ is deleted from Q and $\mathcal{D}'$ is assigned to the generalized dataset $\tilde{\mathcal{D}}$ (steps 26–28). Otherwise, the node representing $\tilde{i}_q$ is merged with its sibling node in $\mathcal{G}$, using the *Merge-siblings* function (steps 29–30). Note that the support remains acceptable, while merging may decrease the confidence of PS-rules, which can result in their protection, as discussed in Sect. 4.1.1. *Merge-siblings* is fairly straightforward. It begins by constructing an itemset I that contains all items of $\tilde{i}_q$ and $sibling(i_q)$ (the sibling of i_q in $\mathcal{G}$), and then creates a generalized item $\tilde{i}$ based on I. After that, it updates $\mathcal{G}$, by deleting the node that corresponds to $\tilde{i}_q$ and those in the subtree rooted at $sibling(i_q)$, and updates $\mathcal{D}'$ by substituting $\tilde{i}_q$ and $sibling(i_q)$ with $\tilde{i}$ in all transactions of $\mathcal{D}'$.

The worst-case time complexity of Algorithm 5 is $O(2^{|\mathcal{P}|} \times |\mathcal{S}| \times N)$ and the space complexity is $O(2^{|\mathcal{P}|} \times |\mathcal{S}| + N \times |\mathcal{I}|)$ (see Theorem 11 in the "Appendix"), when all possible PS-rules are specified and the tallest possible $\mathcal{G}$ is created in the top-down cut revision phase. This occurs when *Sample-Based-Partition* fails to split the single generalized item contained in the most generalized version of $\mathcal{D}$, and all the nodes of $\mathcal{G}$ are finally merged to the root of $\mathcal{G}$ in the bottom-up cut revision phase. Thus, while Sample-based Anonymization has the same time complexity as Tree-based Anonymization in the worst case, the former is significantly more scalable in practice, as shown in our experiments.

4.3 Comparison to Tree-based anonymization

Sample-based Anonymization is significantly more scalable than Tree-based Anonymization, because *Check* is performed efficiently on a random sample of $\tilde{\mathcal{D}}$ during the *sample-based partitioning* phase, and it can also retain data utility better. To see this, recall that Tree-based Anonymization splits a generalized item $\tilde{i}$ as far as required to protect rules, when all rules have an acceptable support, but some have an unacceptable confidence (see steps 17–35 of Algorithm 4). This strategy is adopted for efficiency, but may not help data utility when the antecedent of a rule contains some items that are mapped to $\tilde{i}$ and others that are not, because only $\tilde{i}$ is considered for further splitting. Thus, the confidence of the rule may never become acceptable, due to the items not mapped to $\tilde{i}$, in which case Algorithm 4 does not split $\tilde{i}$.

Sample-based Anonymization, on the other hand, performs top-down cut revision, which allows more splits to be considered, because only the support of rules is checked. For example, assuming that the items that are not mapped to $\tilde{i}$ are mapped to $\tilde{i}'$, this algorithm splits both $\tilde{i}$ and $\tilde{i}'$, as long as the support of rules is acceptable, and then merges the resultant generalized items, if required, to ensure that the confidence is acceptable as well. This gives more chance to protect the rules with less information loss, as it is illustrated in Example 8.

Example 8 Consider a rule $i_1 i_2 \rightarrow \mathbf{j}$ such that the items i_1 and i_2 can be mapped to $\tilde{i_{l7}}$ and its ascendants and $\tilde{i_{l8}}$ and its ascendants, respectively, as shown in Fig. 9, and its support remains acceptable after each of the splits (1)–(9) in Fig. 9. Assuming that the rule remains protected after Tree-based Anonymization has split $\tilde{i}$ into $\tilde{i_{l1}}$ and $\tilde{i_{r1}}$ (split (1) in Fig. 9), the algorithm will try to split $\tilde{i_{l1}}$. If the confidence of the rule exceeds c after the split, the algorithm will try to perform splits (3), (4), and (5) without considering splitting $\tilde{i_{r1}}$. However, since confidence is not monotonic, it is possible that the rule remains unprotected even after all these splits have been attempted. In this case, the algorithm will decide not to split $\tilde{i_{l1}}$ and proceed to split $\tilde{i_{r1}}$. Similarly, if the confidence of the rule remains unacceptable after considering splits (6) to (9), the algorithm will produce a dataset in which all items are mapped to $\tilde{i_{l1}}$ and $\tilde{i_{r1}}$. Now consider applying Sample-based Anonymization. Assuming that this algorithm performs split (1) during the sample-based partition phase, it then performs top-down cut revision in which only the support of the rule is checked using the anonymized dataset. That is, it performs splits (2), (6), (3), (4), (7), (5), (8), and (9) and finds the generalization cut C_2 in Fig. 9. Note that the rule can be protected in the anonymized dataset constructed according to this cut, because confidence is not monotonic, and that this dataset incurs significantly less information loss than the one produced by Tree-based Anonymization.

4.4 Protection from algorithm-based attacks

As discussed in Sect. 2.4, the minimality [81,82] and transparency [87] attacks are possible only when an attacker possesses an eponymous, external dataset. In transaction data publishing, the latter dataset must contain the personal identifiers and all public items of *all* individuals represented in the published dataset. Such a dataset is extremely difficult for an attacker to obtain [16,50], which implies that the aforementioned attacks are not likely to compromise the privacy offered by our algorithms.

We now consider the unlikely case in which an attacker possesses an eponymous, external dataset. Identity disclosure can be prevented in this case, by specifying one PS-rule for each possible information in $\mathcal{P}$. The consequent of the rule can be any of the itemsets induced by items in $\mathcal{N}$. In this setting, minimality attack is not applicable, as neither of our anonymization algorithms follow the minimality principle (see Sect. 2.4). This is because, Tree-based Anonymization does not consider splitting all generalized items, as explained in Sect. 4.3, while Sample-based Anonymization does not consider splitting generalized items after the bottom-up cut revision phase. Thus, generalized datasets, in which the specified PS-rules are protected, can be produced by further splitting generalized items that are contained in the output of our algorithms.

Furthermore, in our approach, a small fraction of the possible itemsets in $\mathcal{N}$ are typically specified as sensitive. This is different from the assumption made in [87], according to which all values in a specified attribute are sensitive. The specified sensitive itemsets in our approach are not contained in the eponymous, external dataset and are not known to the attacker.[6] Thus, an attacker cannot perform a transparency attack by applying Tree-based Anonymization or Sample-based Anonymization to a possible original dataset, after having obtained the anonymized dataset $\tilde{\mathcal{D}}$ and inferred the parameters k and c from it, as in [87].

[6] These itemsets represent individuals' sensitive information, which is unknown to an attacker. A similar assumption was made in [87], for individuals' sensitive values.

5 Deriving PS-rules

Our rule-based privacy model offers data publishers the flexibility of specifying privacy requirements to avoid over-protecting and over-distorting data. In the following, we discuss how such requirements can be translated to PS-rules in scenarios involving data publishers with different amounts of domain knowledge.

We first consider data publishers who know which itemsets are potentially linkable or sensitive. Such knowledge is application-specific and often derived from policies and regulations. An example of a real-world application involving this type of knowledge is electronic medical record data sharing [50]. In this application, the set of diagnosis codes assigned to a patient, during a single hospital visit, is treated as potentially linkable, as explained in [50]. Meanwhile, diagnosis codes that can socially stigmatize patients, such as HIV, are regarded as sensitive (e.g., according to the policy of [65]). Using our privacy model, a data publisher can specify a PS-rule, whose antecedent is a potentially linkable set of diagnosis codes and consequent a sensitive diagnosis code. Another example comes from the field of e-commerce and involves the Octopus card,[7] which is used for payment at various sites (e.g., at convenience stores and service stations) in Hong Kong. Before publishing credit holders' data, the Octopus card company identifies certain items purchased by card holders as sensitive, based on policies or credit card holders' preferences, and partitions all other items into subsets, each containing the items sold to card holders by a different retailer [73]. This is because, a retailer, who may act as an attacker, will possess information only for items sold by them. Our privacy model can be used to specify a PS-rule that contains a set of items a card holder purchased from a retailer as its antecedent, and the card holder's sensitive itemset as its consequent. The ability to specify such detailed privacy requirements can lead to anonymized data that are significantly less distorted than those produced by approaches that impose coarse privacy requirements [74,75,91], as shown in our experiments.

However, not all data publishers possess knowledge about which specific itemsets are potentially linkable or sensitive. Yet, they often know that a class of items falls into either of these categories. For instance, a data publisher may expect that "dvds" are likely to be used in identity disclosure attacks and that "pills" are sensitive, despite being uncertain about specific dvd titles that an attacker can use to link an individual to their transaction, or about types of pills that are sensitive. Liu et al. [46,47] studied how users with limited domain knowledge can identify interesting classification [46] and association [47] rules. Following their work, we present a method to help data publishers with less specific knowledge generate PS-rules. We model data publishers' knowledge using hierarchies, which are common for many types of transaction data, including market-basket [69], lexical [12], and medical data [17], and they can also be derived automatically based on machine learning techniques [76]. Xiao et al. [85] proposed the use of a hierarchy to help individuals select the class of sensitive values they are not willing to be associated with. In contrast, the requirements we consider are specified by data publishers and are expressed as associations between public and sensitive itemsets.

Our method is based on the use of two hierarchies $\mathcal{H}_{\mathcal{P}}$ and $\mathcal{H}_{\mathcal{S}}$, whose leaf-level nodes correspond to the items in $\mathcal{P}$ and $\mathcal{S}$, respectively.[8] The internal nodes of $\mathcal{H}_{\mathcal{P}}$ and $\mathcal{H}_{\mathcal{S}}$ represent classes of public and sensitive items, respectively, while the root of $\mathcal{H}_{\mathcal{P}}$ and $\mathcal{H}_{\mathcal{S}}$ corresponds to the most general class of items, which is labeled as *any public item* and *any sensitive item*,

[7] http://www.octopus.com.hk.

[8] We employ two hierarchies, for simplicity. Using a single hierarchy containing all items of $\mathcal{I}$ is possible, and it requires trivial changes in the process of generating PS-rules.

respectively. Given $\mathcal{H}_\mathcal{P}$ and $\mathcal{H}_\mathcal{S}$, data publishers can express their knowledge that a class of items is expected to be potentially linkable by selecting a node $u_\mathcal{P}$ in $\mathcal{H}_\mathcal{P}$ and, similarly, that a class of items must be treated as sensitive by selecting a node $u_\mathcal{S}$ in $\mathcal{H}_\mathcal{S}$. Assume, for the moment, that an ordered pair $\langle u_\mathcal{P}, u_\mathcal{S}\rangle$ is selected. This pair is used to construct a set of PS-rules, which, when protected, prevent an attacker from inferring an individual's identity using *any* itemset induced by the items in the subtree of $\mathcal{H}_\mathcal{P}$ rooted at $u_\mathcal{P}$, as well as their sensitive information using *any* itemset induced by the items in the subtree of $\mathcal{H}_\mathcal{S}$ rooted at $u_\mathcal{S}$. Data publishers, who are unable to specify classes of items that should be treated as potentially linkable or sensitive,or believe that any association between these two categories of items needs protection, can simply select the root nodes of $\mathcal{H}_\mathcal{P}$ and $\mathcal{H}_\mathcal{S}$. This will lead to constructing a set of PS-rules that model a requirement for maximum privacy protection.

To automatically construct a set of PS-rules from an ordered pair $\langle u_\mathcal{P}, u_\mathcal{S}\rangle$, we present the *PS-rule EXtractor* (*PEX*) algorithm, which is illustrated in Algorithm 6.

Algorithm 6 PEX($\mathcal{D}, k, c, \mathcal{H}_\mathcal{P}, \mathcal{H}_\mathcal{S}, \langle u_\mathcal{P}, u_\mathcal{S}\rangle$)

input: Original dataset $\mathcal{D}$, parameters k, c, hierarchies $\mathcal{H}_\mathcal{P}, \mathcal{H}_\mathcal{S}$, ordered pair $\langle u_\mathcal{P}, u_\mathcal{S}\rangle$
output: Rule-tree $\mathcal{R}$, set of PS-rules Θ

1. $\mathcal{L} \leftarrow \varnothing$
2. **foreach**(transaction $T = \langle tid, T_\mathcal{P} \cup T_\mathcal{N}\rangle \in \mathcal{D}$)
3. $I_\mathcal{P} \leftarrow$ itemset induced by all items in $T_\mathcal{P} \cap subt(\mathcal{H}_\mathcal{P}, u_\mathcal{P})$
4. **foreach**(itemset I that can be induced by items in $I_\mathcal{P}$)
5. $J \leftarrow \underset{\forall i_q \in T_\mathcal{N} \cap subt(\mathcal{H}_\mathcal{S}, u_\mathcal{S})}{\text{argmax}} sup(I \cup i_q, \mathcal{D})$
6. **if**($sup(I, \mathcal{D}) < k$ or $\frac{sup(I \cup J, \mathcal{D})}{sup(I, \mathcal{D})} > c$)
7. $\mathcal{L} \leftarrow \mathcal{L} \cup \{I \rightarrow J\}$
8. construct Rule-tree $\mathcal{R}$ based on $\mathcal{L}$
9. $\Theta \leftarrow$ PS-rules obtained by traversing $\mathcal{R}$ in a depth-first search manner
10. **return**$\{\mathcal{R}, \Theta\}$

PEX iterates over each transaction $T = \langle tid, T_\mathcal{P} \cup T_\mathcal{N}\rangle$ in $\mathcal{D}$, where $T_\mathcal{P}$ and $T_\mathcal{N}$ contain the public and the non-public items of T, respectively (steps 1–2).[9] In each iteration, it constructs the itemset $I_\mathcal{P}$, which is induced by all public items contained in both T and $subt(\mathcal{H}_\mathcal{P}, u_\mathcal{P})$, where $subt(\mathcal{H}_\mathcal{P}, u_\mathcal{P})$ denotes the subtree of $\mathcal{H}_\mathcal{P}$ rooted at $u_\mathcal{P}$ (step 3). Then, in steps 4–5, the algorithm creates a PS-rule $I \rightarrow J$ from T, for each I that can be induced by items in $I_\mathcal{P}$, where J is the most supported *item* contained in both $T_\mathcal{S}$ and $subt(\mathcal{H}_\mathcal{S}, u_\mathcal{S})$ (i.e., the subtree of $\mathcal{H}_\mathcal{S}$ rooted at $u_\mathcal{S}$). If $sup(I, \mathcal{D}) < k$ or $\frac{sup(I \cup J, \mathcal{D})}{sup(I, \mathcal{D})} > c$, then we add $I \rightarrow J$ into $\mathcal{L}$, because this rule is not protected in $\mathcal{D}$ (steps 6–7). Note that the construction of the rule $I \rightarrow J$ guarantees that all rules whose consequent is a superset of J will be protected when $I \rightarrow J$ is protected, according to Theorem 8. Next, in steps 8 and 9, *PEX* creates a Rule-tree $\mathcal{R}$ based on $\mathcal{L}$, and uses a depth-first traversal of $\mathcal{R}$ to construct the set of PS-rules Θ. Last, the algorithm returns $\mathcal{R}$ and Θ (step 10).

Example 9 Consider applying *PEX* on the dataset of Fig. 1a, using $k = 2$ and $c = \frac{2}{3}$, and the ordered pair $\langle [D - F], \mathbf{i}\rangle$. The latter pair is constructed by selecting the node labeled $[D - F]$ from the hierarchy $\mathcal{H}_\mathcal{P}$, shown in Fig. 10a, and the node labeled **i** from the hierarchy $\mathcal{H}_\mathcal{S}$, shown in Fig. 10b. *PEX* starts by considering the first transaction of the dataset and creating $I_\mathcal{P} = \{d\}$, because d is the only item contained in the transaction and the subtree of $\mathcal{H}_\mathcal{P}$ rooted at $[D - F]$ (steps 1–3). Then, the algorithm sets J to **i**, as **i** is the only item contained

[9] We do not consider empty $T_\mathcal{P}$ and $T_\mathcal{S} \subseteq T_\mathcal{N}$, where $T_\mathcal{S}$ is the set of sensitive items in T, as they are unlikely to lead to meaningful privacy attacks. Our approach can be trivially modified to deal with such transactions.

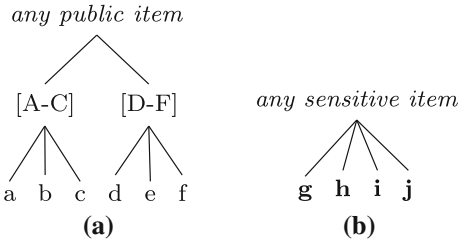


Fig. 10 Hierarchies for **a** the public items, and **b** the sensitive items

in both $T_{\mathcal{N}}$ and **i**, and it checks whether $d \rightarrow \mathbf{i}$ has support less than k or confidence greater than c in the dataset of Fig. 1a (steps 5–6). As neither of these conditions hold, $d \rightarrow \mathbf{i}$ is not added into $\mathcal{L}$, and *PEX* proceeds into considering the second transaction of the dataset of Fig. 1a. The algorithm creates $I_{\mathcal{P}} = \{e, f, ef\}$ and constructs $e \rightarrow \mathbf{i}$, which is not added into $\mathcal{L}$, as its support and confidence in the dataset of Fig. 1a are acceptable. Next, $f \rightarrow \mathbf{i}$ and $ef \rightarrow \mathbf{i}$ are considered, but none of these rules are added into $\mathcal{L}$, for the same reason. After all transactions of the dataset shown in Fig. 1a are considered, $\mathcal{L} = \{de \rightarrow \mathbf{i}\}$ and a rule-tree based on $\mathcal{L}$ are constructed. Last, *PEX* returns $\mathcal{L}$ together with the rule-tree $\mathcal{R}$.

The worst-case time complexity of *PEX* is $O(2^{|\mathcal{P}|} \times N \times |\mathcal{S}|)$, when $I_{\mathcal{P}}$ contains all items in $\mathcal{P}$ and all possible PS-rules are specified. This is because $\mathcal{D}$ contains N transactions, and computing $sup(I \cup i_q)$, for each item $i_q \in T_{\mathcal{N}} \cap subt(\mathcal{H}_{\mathcal{S}}, u_{\mathcal{S}})$ takes $O(N \times |\mathcal{S}|)$ time. In our experiments, the overhead of *PEX* was comparable to the time required for anonymization.

Data publishers may expect more than one class of items to be potentially linkable or sensitive. In this case, multiple ordered pairs of nodes from $\mathcal{H}_{\mathcal{P}}$ and $\mathcal{H}_{\mathcal{S}}$ are constructed by pairing together each node from $\mathcal{H}_{\mathcal{P}}$ with every node from $\mathcal{H}_{\mathcal{S}}$. This strategy requires minimal input from data publishers; however, a set of ordered pairs of nodes $\{\langle u^1_{\mathcal{P}}, u^1_{\mathcal{S}}\rangle, \ldots, \langle u^s_{\mathcal{P}}, u^s_{\mathcal{S}}\rangle\}$ may contain *redundant* pairs. These are pairs $\langle u^q_{\mathcal{P}}, u_{\mathcal{S}}\rangle$, for which there exists a pair $\langle u^r_{\mathcal{P}}, u_{\mathcal{S}}\rangle$ such that $u^r_{\mathcal{P}}$ is an ascendant of $u^q_{\mathcal{P}}$ in $\mathcal{H}_{\mathcal{P}}$, as well as pairs $\langle u_{\mathcal{P}}, u^q_{\mathcal{S}}\rangle$, for which there exists a pair $\langle u_{\mathcal{P}}, u^r_{\mathcal{S}}\rangle$ such that $u^r_{\mathcal{S}}$ is an ascendant of $u^q_{\mathcal{S}}$ in $\mathcal{H}_{\mathcal{S}}$. The use of a redundant pair as input to *PEX* would cause unnecessary computational overhead, because all the resultant PS-rules are obtainable by at least another pair. Therefore, we first remove all redundant pairs from $\{\langle u^1_{\mathcal{P}}, u^1_{\mathcal{S}}\rangle, \ldots, \langle u^s_{\mathcal{P}}, u^s_{\mathcal{S}}\rangle\}$, and then apply *PEX* to each ordered pair $\langle u^j_{\mathcal{P}}, u^j_{\mathcal{S}}\rangle$ in the set to obtain a set of PS-rules Θ_j, where $j \in [1, s]$. Last, we construct a set of PS-rules $\Theta = \Theta_1 \cup \ldots \cup \Theta_s$ that is given as input to our anonymization algorithms.

Note that our approach can be used to create PS-rules with a varying number of items in their antecedent and/or with specific items in their consequents. In contrast, the approach of Xu et al. [90,91] considers all public itemsets containing up to p items as requiring protection for identity disclosure, and all non-public items as sensitive. For instance, assuming that a data publisher's requirements are as in Example 9, the method of Xu et al. would need to consider all 20 possible 3-itemsets induced by $\{a, b, c, d, e, f\}$ and to prevent the association between each of these itemsets and any item in $\{\mathbf{g}, \mathbf{h}, \mathbf{i}, \mathbf{j}\}$. Thus, data would be distorted unnecessarily. When data publishers want to protect any association between potentially linkable and sensitive items, the application of *PEX*, using an ordered pair containing the root nodes of $\mathcal{H}_{\mathcal{P}}$ and $\mathcal{H}_{\mathcal{S}}$, achieves the same protection as applying (h, k, p)-coherence [90,91] with the same k, $h = c \times 100\,\%$, and $p = |\mathcal{P}|$. However, it is worth noting that our algorithms preserve data utility better than the algorithm of Xu et al. [91] in this setting, as shown in our experiments.

Table 1 Characteristics of used datasets

Dataset	N	$\|\mathcal{I}\|$	Max. size of T	Avg. size of T
BMS1	59,602	497	267	2.5
BMS2	77,512	3,340	161	5.0
POS	515,597	1,657	164	6.5

6 Experimental evaluation

We have conducted extensive experiments to evaluate our algorithms. Our results show that they significantly outperform the state-of-the-art in terms of data utility, while achieving good protection and efficiency.

6.1 Experiment data and setup

We used BMS-WebView-1 (BMS1), BMS-WebView-2 (BMS2), and BMS-POS (POS), three real-world datasets that have been used in evaluating previous work [29,27,35,74,75]. BMS1 and BMS2 contain click-stream data, while POS contains sales transaction data. Table 1 shows the properties of these datasets.

We compared our *Tree-based* and *Sample-based* algorithms to Greedy [91], which guarantees protection from both identity and sensitive information disclosure, and Apriori [74], which, unlike other generalization-based algorithms [35,51,75], produces data that can be mined effectively, because it maps instances of the same item to the same generalized item in all transactions, and it does not require data publishers to limit the amount of generalization that may be exercised. We did not include the algorithms for enforcing ρ-uncertainty [16] and differential privacy [19] in our evaluation, because they are not directly comparable to our methods (see Sects. 2.2, 2.3). The weights used in our methods were configured based on a notion of semantic distance [89] computed according to the hierarchies described in [74]. To demonstrate the effectiveness of the PS-rule handling and anonymization strategies we propose, we compared our algorithms against Baseline, a basic algorithm that splits generalized items based on *UL* instead of the approximation adopted by *B-Split*, and checks all rules in Θ according to Definition 4.

The default values for parameters used in evaluation were $k = 5$ and $c = 0.7$, and the set of sensitive items $\mathcal{S}$ was comprised of 5 % of the most frequent items in $\mathcal{I}$. Unless stated otherwise, Apriori was applied to public items with $m = 2$, Greedy with $p = 2$ and $h = 70\,\%$, and Sample-based using $\delta = 0.05$ and $\epsilon = 0.025$ (or equivalently a sample of 2,952 records). We report the worst results derived by executing Sample-based 5 times, to minimize the impact of randomness in sampling, although this difference was negligible. Furthermore, we used different sets of PS-rules, and selected the antecedents and consequents of rules uniformly at random. The PS-rules in all the sets we used were not always protected (i.e., none of them satisfied the conditions of Theorem 5). The default PS-rule set we used was S_1, which was comprised of $5K$ PS-rules whose antecedent is a 2-itemset in $\mathcal{P}$ and consequent an item in $\mathcal{S}$.

6.2 Preventing sensitive information disclosure

To measure protection for sensitive information disclosure, we counted the percentage of generalized items for which there is at least one sensitive item i_q such that $\frac{sup(\tilde{i} \cup i_q, \tilde{\mathcal{D}})}{sup(\tilde{i}, \tilde{\mathcal{D}})} > c$,

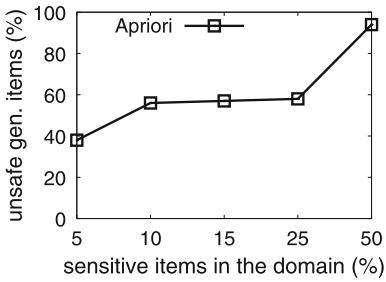


Fig. 11 Unsafe gen. items versus sens. items (BMS1)

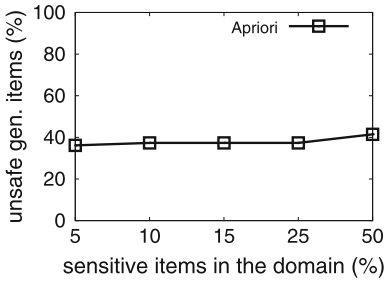


Fig. 12 Unsafe gen. items versus sens. items (BMS2)

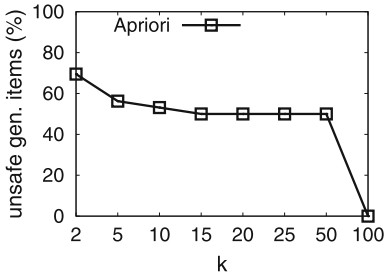


Fig. 13 Unsafe gen. items versus k (BMS1)

where $\tilde{i}$ is a generalized item in $\tilde{\mathcal{D}}$. We call these generalized items *unsafe* and use them because their number does not depend on the number of specified PS-rules. We do not report results for our algorithms and Greedy, since they are designed to eliminate unsafe generalized items.

Figures 11 and 12 report the results with respect to protection for BMS1 and BMS2, respectively, where the percentage of sensitive items in $\mathcal{I}$ varies from 5 to 50, $k = 5$, and $c = 0.2$. Notice that Apriori failed to prevent sensitive information disclosure, as the percentage of unsafe generalized items was at least 38 and 36.1 %, for BMS1 and BMS2, respectively. However, no generalized item was unsafe when Apriori ran on POS, because of the large amount of generalization incurred. The lack of protection was observed when different values of m were used, but the results were omitted for space reasons.

Next, we considered the 10 % of items in $\mathcal{I}$ as sensitive and investigated the impact of using various k and c values on privacy. Figure 13 shows the result for various k values and $c = 0.2$. As can be seen, Apriori constructed anonymizations that contained at least 50 % of unsafe generalized items, even when it ran with $k = 50$. The result for varying c and $k = 5$ is reported in Fig. 14. Apriori constructed anonymized datasets with up to 78 % of unsafe generalized items, and the percentage of unsafe generalized items remained non-zero for c up to 0.8. This confirms that anonymized data produced by Apriori are still at risk of sensitive information disclosure.

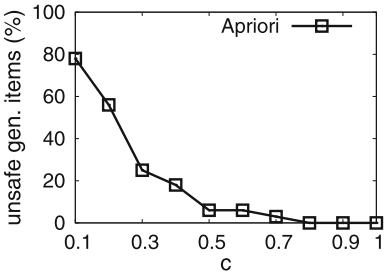


Fig. 14 Unsafe gen. items versus c (BMS1)

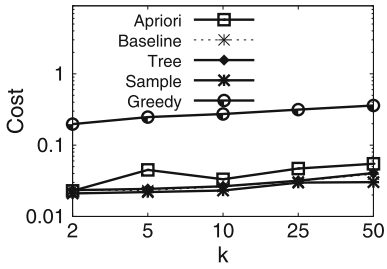


Fig. 15 Cost versus k (BMS1)

6.3 Data utility

In this section, we show the superiority of our algorithms over Apriori and Greedy for retaining data utility. We measured data utility using the *UL* and *Cost* [48] measures. *Cost* captures the information loss caused by generalization and suppression and is expressed as the sum of the value of *LM* [36] for $\tilde{\mathcal{D}}$ and the fraction of the items that are suppressed to produce $\tilde{\mathcal{D}}$. In addition, we measured the accuracy of answering query workloads on anonymized data using the *Average Relative Error* (ARE) measure [42], which is widely used [27,29,42] and invariant of the way all tested algorithms work. To simulate different data analysis requirements, we used two query workloads: W_1 and W_2. The details of the queries we considered in W_1 and W_2 and the computation of *ARE* are given in the "Appendix". The queries contained in W_2 are, intuitively, more difficult to answer accurately, because they involve more items [51]. They also pose additional difficulty for our algorithms in comparison with Apriori, because our algorithms constrain the combinations of generalized and sensitive items that may be released.

6.3.1 Enforcing (h, k, p)-coherence

We compared the information loss incurred by Greedy and our algorithms, when they are configured to enforce (h, k, p)-coherence. We also tested Apriori, using the same k and $m = p$, to compare the effect of using (h, k, p)-coherence versus k^m-anonymity on data utility.

Impact of k The *Cost* scores for the three datasets, for $h = 50\,\%$, $p = 2$, and k from 2 to 50 are shown in Figs. 15, 16, and 17. As can be seen, our methods can provide the same protection as Greedy does, while incurring significantly less information loss. Greedy suppressed at least 19.6 % of items in BMS1 and up to 72 % of items in the more sparse BMS2 dataset. Thus, its scores were at least 8.4 and up to 81 times higher than those of our algorithms. Tree-based was slightly worse than Baseline, because, for efficiency reasons, it splits generalized items based on the approximation of *UL* used in *B-Split* and not on *UL*

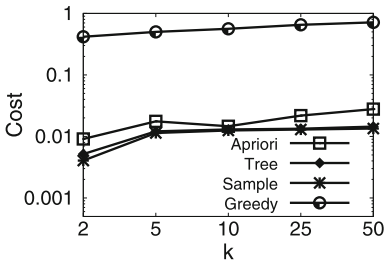


Fig. 16 Cost versus k (BMS2)

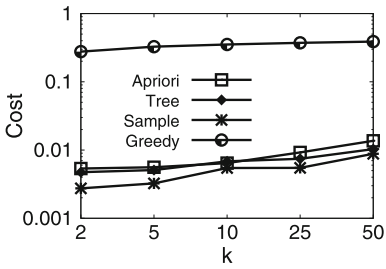


Fig. 17 Cost versus k (POS)

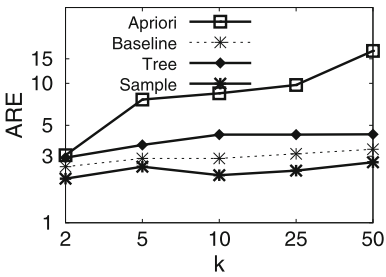


Fig. 18 ARE for W_1 versus k (BMS1)

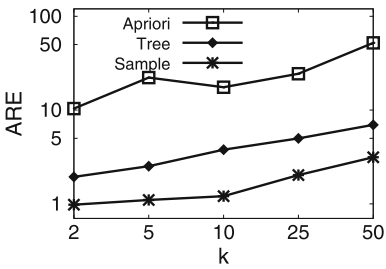


Fig. 19 ARE for W_1 versus k (BMS2)

itself, as Baseline does. Sample-based outperformed both Tree-based and Baseline, for the reasons given in Sect. 4.2.2. We also found that Apriori outperforms Greedy, confirming the results of [48]. However, Apriori incurred more information loss than our algorithms did, despite not preventing sensitive information disclosure. This shows that the set-based generalization model employed by our algorithms is better for data utility.

We report the *ARE* scores for BMS1 and BMS2 and for workload W_1 in Figs. 18 and 19, respectively. Greedy was not included in the experiments reported in this section, since it incurred a large amount of suppression, which prevented answering many queries. The scores for Apriori were at least 2.3 and 3.7 times larger than those of Tree-based and Sample-based for BMS1 and much larger for BMS2, while the relative performance among our

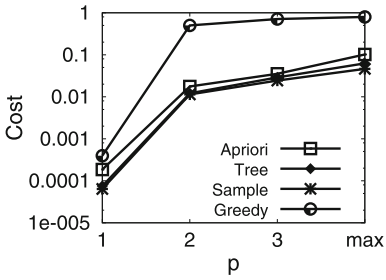


Fig. 20 Cost versus p (BMS2)

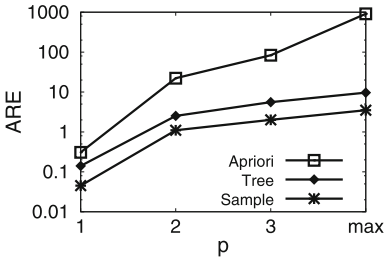


Fig. 21 ARE for W_1 versus p (BMS2)

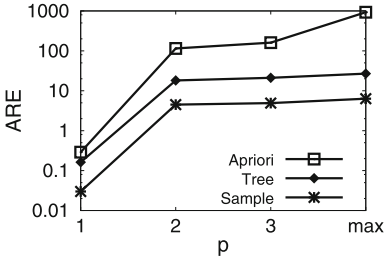


Fig. 22 ARE for W_2 versus p (BMS2)

algorithms was the same as in the above experiments. This confirms that using the set-based generalization model together with *UL*, computed exactly or approximately, helps query answering accuracy.

Impact of p We examined how varying p affects *Cost* and show the result for BMS2, $k = 5$, and $h = 50\,\%$, in Fig. 20. Observe that all algorithms traded-off information loss for stronger protection, as p increased, and that Greedy performed the worst. Our methods retained more information than Apriori, although they additionally prevent sensitive information disclosure. This can be seen in Figs. 21 and 22, which illustrate the *ARE* scores for W_1 and W_2 in BMS2. This result suggests that protecting incrementally larger itemsets, as Apriori does, leads to more generalization compared to applying generalization to protect each rule, as our algorithms do. The results on the other datasets were similar and omitted for the sake of space.

6.3.2 Enforcing our rule-based model

Impact of k We first considered BMS1 with k varied from 2 to 50. The results for workload W_1, given in Fig. 23, show that Tree-based outperformed Apriori, achieving up to 3 times better *ARE* scores. However, since Tree-based captures information loss based on an approximation introduced in *B-Split*, rather than the more accurate *UL* measure, in order to achieve

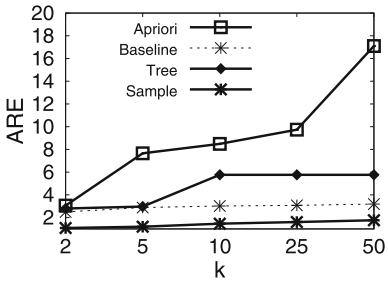


Fig. 23 ARE for W_1 versus k (BMS1)

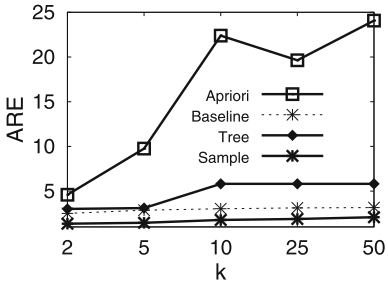


Fig. 24 ARE for W_2 versus k (BMS1)

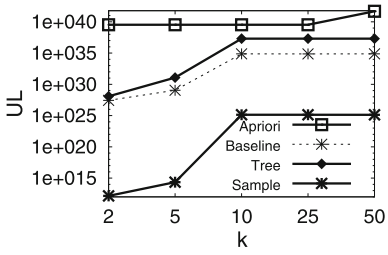


Fig. 25 UL versus k (BMS1)

efficiency, it performed worse than Baseline for $k > 5$. Interestingly, Sample-based outperformed all other algorithms, achieving up to 10 and 4 times better *ARE* scores than Apriori and Tree-based did, respectively. This confirms our theoretical analysis in Sect. 4.2.2. Our algorithms outperform Apriori in this experiment, because they provide no more protection than that required to protect the specified rules, while they adopt a more flexible generalization model. The results w.r.t. unnormalized *UL* scores for the same experiments are provided in Fig. 25 and verify the observations made using *ARE* as a data utility indicator. The results from the same experiments using W_2, reported in Fig. 24, are similar to those of Fig. 23. In this set of experiments, we do not report additional results based on *UL*, as they are all consistent with *ARE* scores, nor do we include the results for Baseline, since Sample-based outperformed it in data utility and efficiency, while achieving the same level of protection.

Figure 26 shows the *ARE* scores for POS using W_1. Tree-based and Sample-based achieved *ARE* scores that were 1.9 and 2.6 times better on average than those for Apriori, which again confirms the superiority of our algorithms.

We also measured data utility when anonymized data are produced based on different privacy requirements, which are specified using our hierarchy-based method (see Sect. 5). We considered 5 privacy requirements: $PR_1,\ldots,PR_5$, which are specified to protect different associations between classes of items. Each of these requirements was modeled by selecting nodes in the hierarchies $\mathcal{H}_\mathcal{P}$ and $\mathcal{H}_\mathcal{S}$, as shown in Table 2, and resulted in a set of PS-rules.

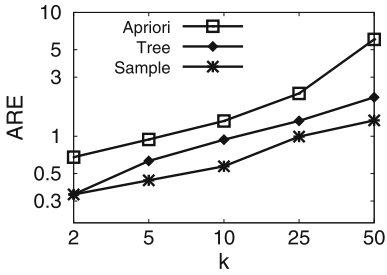


Fig. 26 ARE for W_1 versus k (POS)

Table 2 Summary of privacy requirements (BMS2)

Privacy requirement	Selected nodes of $\mathcal{H}_\mathcal{P}$	Selected nodes of $\mathcal{H}_\mathcal{S}$
PR_1	Root	Root
PR_2	3 (50 % of) level-3	16 (48 % of) level-1
PR_3	5 (15 % of) level-2	5 (15 % of) level-1
PR_4	3 (50 % of) level-3	7 (22 % of) level-1
PR_5	3 (50 % of) level-3	2 (6 % of) level-1

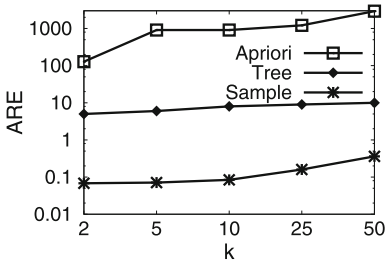


Fig. 27 ARE for W_1 and PR_1 versus k (BMS2)

This set of PS-rules was produced by applying *PEX* to BMS2, and it was subsequently provided as input to our algorithms. The default hierarchy for BMS2 was used as $\mathcal{H}_\mathcal{P}$, and a hierarchy produced based on the method described in [74] as $\mathcal{H}_\mathcal{S}$. $\mathcal{H}_\mathcal{P}$ and $\mathcal{H}_\mathcal{S}$ have 6 and 4 levels, respectively, and their leaves lie in level 0. For each privacy requirement, a number of nodes from a level of $\mathcal{H}_\mathcal{P}$ and $\mathcal{H}_\mathcal{S}$, respectively, were selected uniformly at random. The number and percentage of the selected nodes, as well as the level of these nodes in the hierarchy they were selected from, are shown in Table 2. For consistency, a set of nodes, selected from a certain level of a hierarchy to model a privacy requirement, were contained in all larger sets of nodes, which lie in the same level of the hierarchy, and were selected to model a different requirement. For example, the 2 level-1 nodes selected from $\mathcal{H}_\mathcal{S}$ to model PR_5 were also selected in the case of PR_2.

The results for PR_1 are shown in Fig. 27. These results were obtained for BMS2, W_1 and various k values in [2, 50]. In this setting, Apriori is configured with $m = |\mathcal{P}|$. Note that Apriori produced data that are practically useless, since the *ARE* scores for this algorithm were in the order of hundreds or more, as opposed to those of Tree-based and Sample-based that were no more than 9.1 and 0.4, respectively, even though these algorithms offer more protection for sensitive information disclosure.

Figure 28 reports the results for Sample-based when it is configured with the PS-rules corresponding to PR_1, PR_2, and PR_3. All other parameters were set as in the previous experiment. The *ARE* scores in the case of PR_2 and PR_3 were on average 1.5 and 6 times

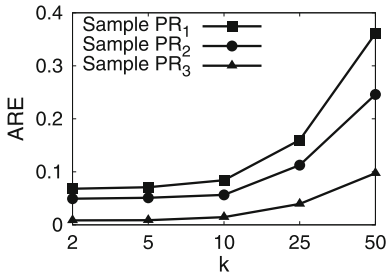


Fig. 28 ARE for W_1 and PR_1 to PR_3 versus k (Sample-based, BMS2)

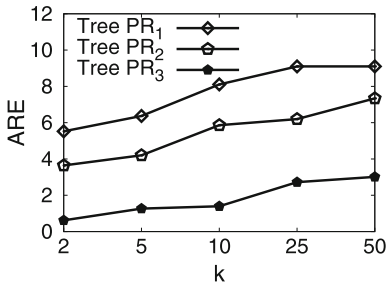


Fig. 29 ARE for W_1 and PR_1 to PR_3 versus k (Tree-based, BMS2)

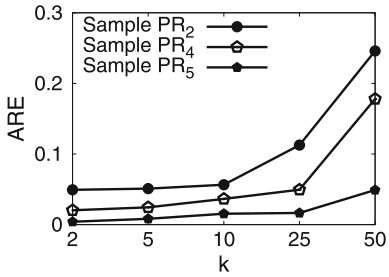


Fig. 30 ARE for W_1 and PR_2, PR_4, and PR_5 versus k (Sample-based, BMS2)

smaller than those corresponding to PR_1. This is because PR_3 is less strict than PR_2, which, in turn, is less strict than PR_3. That is, PR_3 requires protecting fewer public itemsets from identity disclosure, and fewer associations between public and sensitive itemsets from sensitive information disclosure than PR_2 and PR_1. These results demonstrate that the specification of different privacy requirements, using our hierarchy-based method, can affect the amount of information loss incurred by anonymization. The results from the same experiment, using Tree-based, are shown in Fig. 29, and they are similar to those of Fig. 28. The *ARE* scores for Apriori were significantly larger than those of both our algorithms and are not reported. This is expected, as Apriori had to generalize all m-itemsets, where m is the size of the largest itemset used as antecedent in each set of PS-rules our algorithms were configured with.

Furthermore, we tested how *ARE* is affected by the use of requirements in which a varying number of nodes in $\mathcal{H}_S$ are selected. Figure 30 shows the results for Sample-based when it is configured with PS-rules that correspond to PR_2, PR_4, and PR_5, and with all other parameters as in the previous experiment. The *ARE* scores corresponding to PR_4 and PR_5 were on average 1.9 and 7 times smaller than those for PR_2. This is because PR_5 contains fewer itemsets that need protection from sensitive information disclosure than PR_4, which, in turn, contains fewer such itemsets than PR_2. These results demonstrate that our

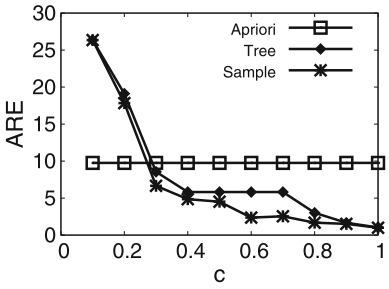


Fig. 31 ARE for W_2 versus c (BMS1)

Table 3 Summary of privacy requirements (BMS1)

Privacy requirement	Selected nodes of $\mathcal{H}_\mathcal{P}$	Selected nodes of $\mathcal{H}_\mathcal{S}$
PR_6	Root	Root
PR_7	2 (50 % of) level-3	3 (60 % of) level-1
PR_8	4 (20 % of) level-2	5 (20 % of) level-0
PR_9	2 (50 % of) level-3	2 (40 % of) level-1
PR_{10}	2 (50 % of) level-3	1 (20 % of) level-1

hierarchy-based method allows the production of PS-rules, which can avoid over-distortion and over-protection of data. Similar trends were observed for Tree-based, while Apriori was outperformed by both our algorithms, for the reasons explained above. The results for Tree-based and Apriori are not reported, for brevity.

Impact of c We studied how varying c in [0.1, 1] affects *ARE* and provide the results for BMS1 and W_2 in Fig. 31. As can be seen, our algorithms incurred more information loss than Apriori for $c < 0.3$, because they had to trade-off some utility for stronger privacy, but outperformed Apriori by an increasingly large margin for $c \geq 0.3$. Observe the result for $c = 1$, a setting in which our algorithms become similar to Apriori in that they prevent identity disclosure only. In this case, Tree-based and Sample-based allowed 6.2 and 25 times more accurate query answering than Apriori. Similar results were observed for BMS2 and W_2, but are not reported here for brevity.

We also measured how *ARE* is affected by the use of different privacy requirements, which are expressed based on our hierarchy-based method. We considered 5 privacy requirements: $PR_6,\ldots,PR_{10}$, whose specification is similar to those reported in the previous set of experiments, and it is explained in Table 3. The default hierarchy for BMS1 was used as $\mathcal{H}_\mathcal{P}$, and $\mathcal{H}_\mathcal{S}$ was derived by the method described in [74]. Each of these privacy requirements resulted in a set of PS-rules that was produced by applying *PEX* and provided as input to our algorithms.

The results for PR_6, which corresponds to the strictest privacy requirement that can be specified by our privacy model, are shown in Fig. 32. In these experiments, Apriori ran with $m = |\mathcal{P}|$ and all other parameters were set as in the experiment of Fig. 31. As can be seen from Fig. 32, both our algorithms outperformed Apriori by many orders of magnitude, across all tested c values. This shows that our algorithms can provide the same protection from identity disclosure than Apriori and additionally prevent sensitive information disclosure while incurring less information loss.

Figures 33 and 34 report the results for PR_6 to PR_8, for Sample-based and Tree-based, respectively. The *ARE* scores achieved by either of these algorithms for PR_7 and PR_8 were many times smaller than those corresponding to PR_6. This is because PR_8 is less strict

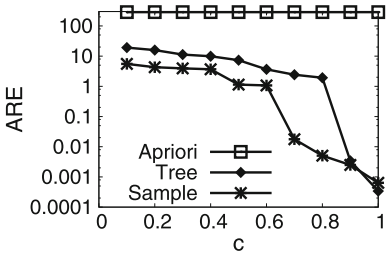


Fig. 32 ARE for W_2 and PR_6 versus c (BMS1)

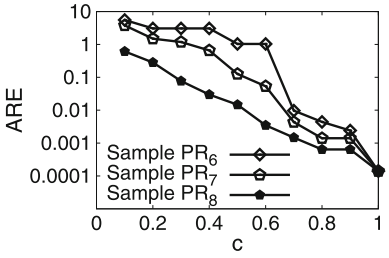


Fig. 33 ARE for W_2 and PR_6 to PR_8 versus c (Sample-based, BMS1)

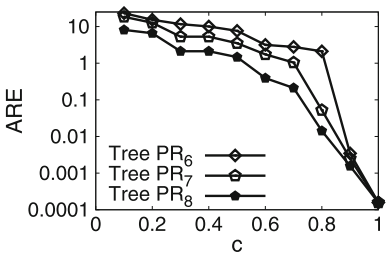


Fig. 34 ARE for W_2 and PR_6 to PR_8 versus c (Tree-based, BMS1)

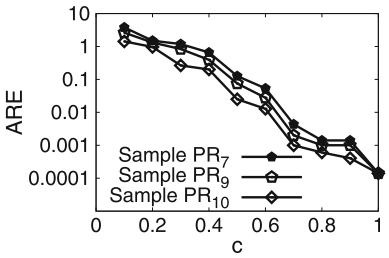


Fig. 35 ARE for W_2 and PR_7, PR_9, and PR_{10} versus c (Sample-based, BMS1)

than PR_7, which, in turn, is less strict than PR_6. For instance, configuring Sample-based with PR_8 instead of PR_6, resulted in *ARE* scores that were 55 times smaller, on average. Figure 35 illustrates the *ARE* scores for Sample-based, when it is configured with the PS-rules corresponding to PR_7, PR_9, and PR_{10}. All other parameters were set as in the experiment reported in Fig. 33. Note that in the case of PR_9 and PR_{10}, the *ARE* scores for Sample-based were up to 2.2 and 4.4 times smaller compared to the case of PR_7. These results demonstrate that the specification of privacy requirements, using our approach, affects the information loss incurred by anonymization. The *ARE* scores for Apriori were significantly larger than those of both our algorithms, for the reasons discussed above, and are not reported here.

Impact of number of PS-rules The results for the workload W_2 on BMS1 and BMS2 with a set of specified PS-rules, whose antecedents and consequents are 2-itemsets and

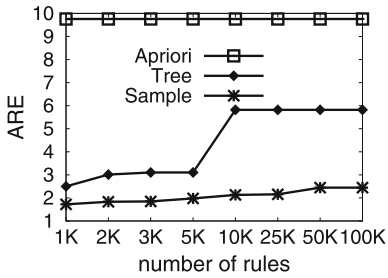


Fig. 36 ARE for W_2 versus #*rules* (BMS1)

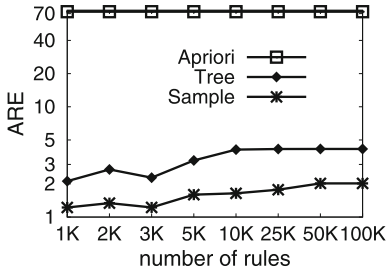


Fig. 37 ARE for W_2 versus #*rules* (BMS2)

number varies from 1 to 100 K, are shown in Figs. 36 and 37, respectively. Our algorithms significantly outperformed Apriori, although the difference in their *ARE* scores decreased as the number of rules increased. This is expected as the stronger protection required, the more data utility is compromised. Note also that the *ARE* scores of our algorithms remained steady when the number of specified rules is 50 K or more, because the consequents of the rules have collectively covered all sensitive items appeared in transactions. Thus, no additional generalization was needed to protect more than 50 K PS-rules and the *ARE* scores did not increase after this point.

Impact of different PS-rules We used sets that contain 5 K rules and refer to a set by the size of antecedents and consequents it contains. For example, 2–2 refers to a set of rules whose antecedents and consequents are 2-itemsets. Apriori was configured by setting m to the size of antecedent of rules in the set.

The result for BMS1 and W_2 is illustrated in Fig. 38. Note that Apriori incurred much more information loss for rules with larger antecedents than our algorithms did, since it is forced to generalize all m-public itemsets, not only those specified by the rules. The median ratio between the *ARE* scores for Apriori and Tree-based is 4.5 and 9.5 for W_1 and W_2, respectively. Sample-based consistently outperformed Tree-based achieving a result up to 3 and 9 times better for W_1 and W_2, which again indicates its effectiveness in retaining data utility.

Note that the *ARE* scores for our algorithms increased with the size of consequents of rules, as more information loss was incurred to achieve stronger protection. The *ARE* scores for Apriori, on the other hand, remained steady, as it does not consider sensitive items. However, our algorithms significantly outperformed Apriori, even when the 4–4 rule set, which requires a large amount of generalization to be protected, was used. These observations are also confirmed by the results for BMS2, shown in Figs. 39 and 40 for W_1 and W_2, respectively.

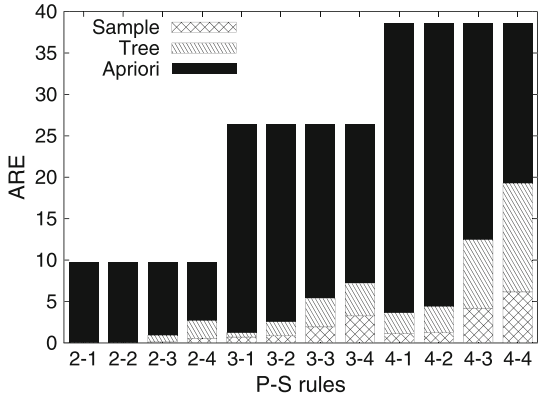


Fig. 38 ARE for W_2 versus rules in S_1 (BMS1)

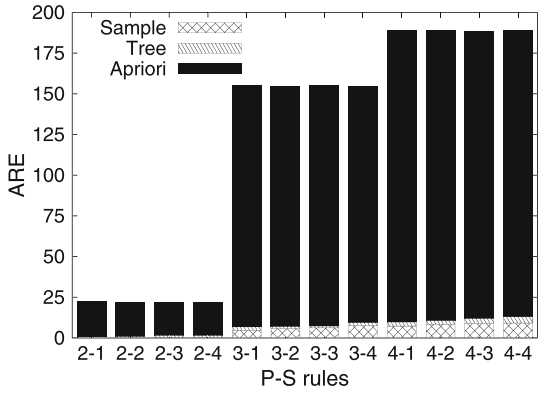


Fig. 39 ARE for W_1 versus rules in S_1 (BMS2)

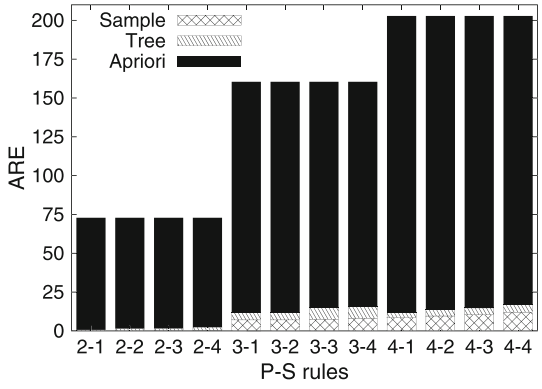


Fig. 40 ARE for W_2 versus rules in S_1 (BMS2)

6.4 Efficiency of computation

We evaluate the runtime and memory requirements of our algorithms, as well as the performance of our rule checking strategy.

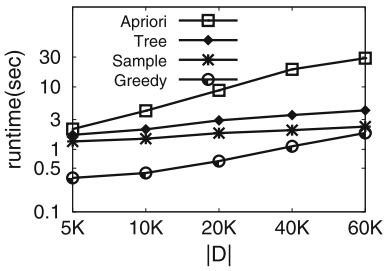


Fig. 41 Efficiency versus $|D|$ (subsets of BMS2)

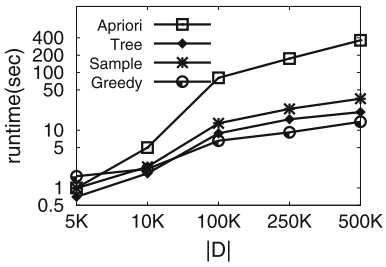


Fig. 42 Efficiency versus $|D|$ (subsets of POS)

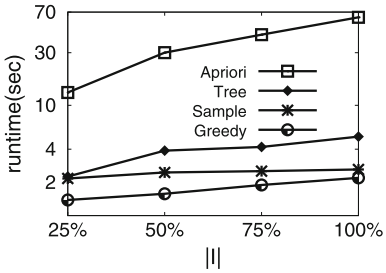


Fig. 43 Efficiency versus $|\mathcal{I}|$ (subsets of BMS2)

6.4.1 Runtime evaluation

Impact of dataset size Figures 41 and 42 report the results for datasets constructed by selecting increasingly larger subsets of BMS2 and POS. The transactions in each subset were selected uniformly at random but contained in all larger sets. Greedy was the most efficient of the algorithms, requiring 55 and 23.6 % less time on average than Sample-based for BMS2 and POS, respectively. Sample-based was the second fastest and scaled better than Greedy, because Greedy required increasingly more time to identify all itemsets that need protection. Both our algorithms are more scalable than Apriori, which is up to 10.3 times and 17 times less efficient than Tree-based and Sample-based on POS. This shows that exploring the space of generalizations in a top-down fashion, as our algorithms do, is more efficient than the bottom-up strategy used in Apriori. Sample-based was faster than Tree-based, because it performed rule checking based on a sample. The performance gain increased with the dataset size. For reference, we report that Baseline was several orders of magnitude slower, requiring 10 h to anonymize BMS2, because it performs a large number of support computation to split items based on UL and to check all PS-rules.

Impact of domain size The results for datasets comprised of an increasingly larger percentage of items of BMS2 and POS are reported in Figs. 43 and 44, respectively. Greedy was the most efficient algorithm, followed by Sample-based, which needed 19 and 12 % more time

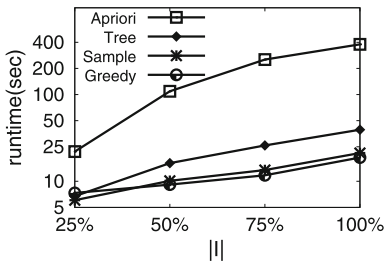


Fig. 44 Efficiency versus $|\mathcal{I}|$ (subsets of POS)

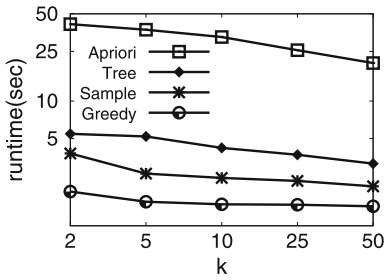


Fig. 45 Efficiency versus k (BMS2)

to anonymize BMS2 and POS. Both of these algorithms scaled similarly with the increase of the domain size and much better than other algorithms, because Greedy suppressed a large fraction of items, quickly achieving the privacy requirement, and Sample-based performed most rule checks efficiently based on a sample. Note also that Tree-based and Sample-based are up to 12 and 24 times faster than Apriori, and are more scalable. For instance, they require less than 40 and 21 s to anonymize the larger POS dataset, while Apriori needs 6.3 min. This indicates that splitting generalized items in a top-down fashion can cope better with large domain sizes than the bottom-up generalization used by Apriori. Also, Sample-based scaled much better and was up to 2 times faster than Tree-based in this experiment. This is because the time spent in the top-down and bottom-up cut revision phases of Sample-based became an increasingly small fraction of total execution time as domain size increased.

Impact of parameter k As can be seen in Fig. 45, all algorithms required less time to execute as k increased. This is attributed to the top-down item splitting strategy of our algorithms, which requires fewer iterations to create generalized items with large support, to the recoding model of Apriori, and to the fact that Greedy did not suppress significantly more items for $k \geq 5$ compared to $k = 2$. Greedy was the most efficient of the algorithms, being 39 and 63 % faster than Sample-based and Tree-based, which in turn were at least 6 times faster than Apriori, although they check the confidence of rules to prevent sensitive itemset disclosure.

Impact of parameter c Figure 46 shows the result of applying our algorithms with various c values. The result for Apriori is not reported here, since its efficiency is not affected by c. Tree-based is fairly insensitive to c, for $c \leq 0.9$, because it attempted to find generalizations with higher data utility when rules had a support of at least k, but a larger confidence than c. For $c = 1$, however, it ran faster by 97 %, as it only had to compute the support of rules. On the contrary, the runtime of Sample-based decreased with c, which is expected, as only a small number of iterations were needed to revise the generalized dataset constructed during the sample-based partitioning phase. Tree-based and Sample-based were slower than Greedy, which needed more time to discover itemsets that require protection for smaller c values.

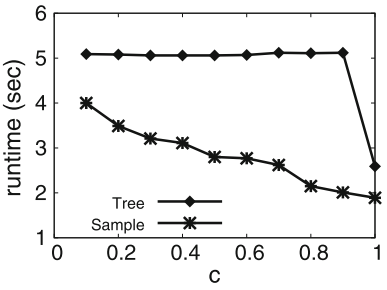


Fig. 46 Efficiency versus c (BMS2)

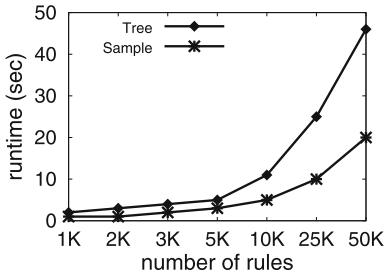


Fig. 47 Efficiency versus # rules (BMS2)

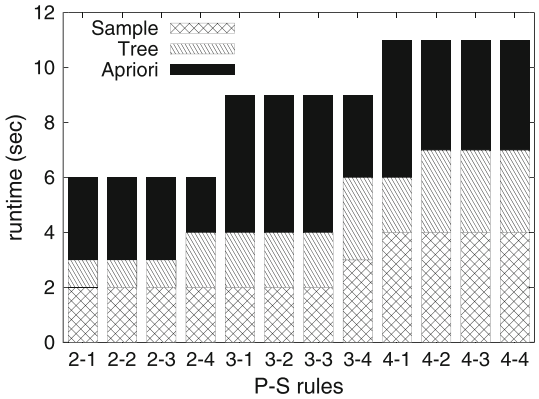


Fig. 48 Efficiency versus different rules (BMS1)

However, this computation did not substantially affect the runtime of Greedy, which is not reported here.

Impact of number of PS-rules The result for BMS2 using the rules, whose antecedents and consequents are 2-itemsets and number varies from 1 to 50 K, is shown in Fig. 47. The strong protection from sensitive information disclosure provided by our algorithms when a large number of rules are to be protected resulted in an increase in their runtime. However, their computation cost remained sublinear in all tested cases. Furthermore, due to the use of sampling, Sample-based was up to 2.5 times faster than Tree-based was.

Impact of different PS-rules The results for sets of PS-rules whose antecedents and consequents have different sizes are shown for BMS1 and BMS2 in Figs. 48 and 49. While all algorithms needed more time when rules had larger antecedents, the efficiency of Apriori was affected the most, since it unnecessarily protected all possible m-items. This was more evident when rules with antecedents that contain up to 10 items were used. As can be seen in Fig. 50, the runtime of Apriori increased 19 times when it was executed on the 10–2 rule set

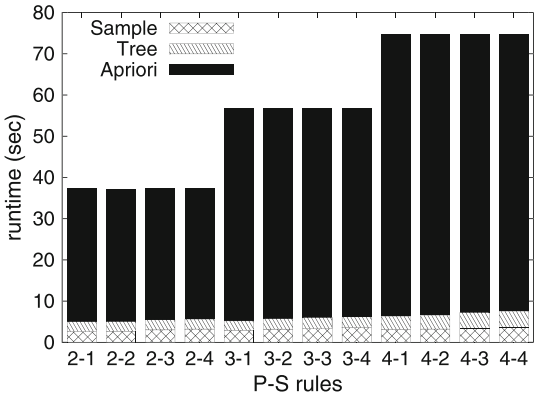


Fig. 49 Efficiency versus different rules (BMS2)

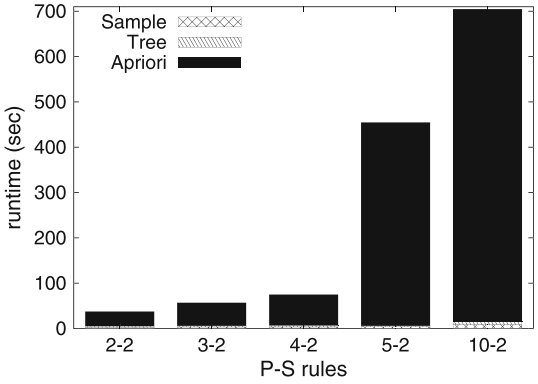


Fig. 50 Efficiency versus rules with increasingly large antecedents (BMS2)

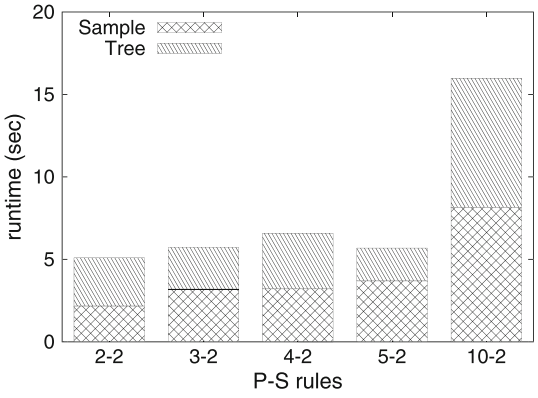


Fig. 51 Efficiency versus rules with increasingly large antecedents (BMS2)

instead of 2–2, while the runtime of our algorithms was only 3 times higher. Furthermore, despite the fact that our algorithms needed more time to protect rules with larger consequents, they were still faster than Apriori in all tested cases. Tree-based was at least 44 times faster than Apriori for both BMS1 and BMS2. In addition, Sample-based outperformed Tree-based by at least 60 %, and was more scalable with respect to the size of antecedents and consequents, due to the use of sampling. The comparative runtime of our algorithms in this experiment is shown more clearly in Fig. 51.

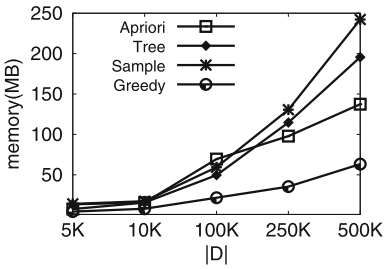


Fig. 52 Memory versus $|D|$ (POS)

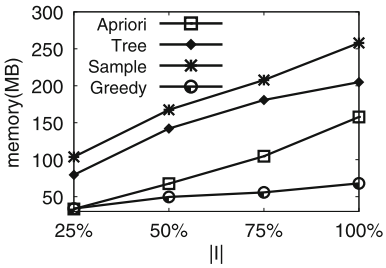


Fig. 53 Memory versus $|I|$ (POS)

6.4.2 Memory requirements

We compare the memory requirements of the tested algorithms with respect to their peak memory consumption, since they employ different anonymization strategies and data structures.

Impact of dataset size Figure 52 shows the results for the increasingly larger subsets of POS used in Sect. 6.4.1. All tested algorithms scaled sublinearly to the dataset size and Greedy required the least memory; up to 73 and 69 % less than our algorithms and Apriori. This is because, Greedy processes *minimal moles*, i.e., itemsets that do not satisfy (h, k, p)-coherence but all of their proper subsets do, eliminating the item contained in most of them. These itemsets are typically fewer than all moles and stored in a compressed form, which reduces storage requirements. Tree-based and Sample-based needed up to 30 and 45 % more memory than Apriori, since they need to store generalized items and check rules, which involves creating a temporary dataset for computing their support after item splitting. Tree-based consumed up to 19 % less memory than Sample-based, which needs to store the entire generalization tree to enable bottom-up cut revision.

Impact of domain size Figure 53 shows the results for datasets comprised of an increasingly larger percentage of items of POS used in Sect. 6.4.1. All algorithms scaled sublinearly to the domain size, with Greedy being the most memory efficient, for the reasons discussed above. Tree-based and Sample-based required 2.4 and 3.1 times more memory than Apriori when $|I| = 25\,\%$, as a large fraction of the total memory is spent for generalized item storage and rule checking, but only 30 and 63 % more to anonymize the POS dataset.

Impact of number of PS-rules The result for POS using the rules, whose antecedents and consequents are 2-itemsets and number varies from 5 to 200 K, is shown in Fig. 54. Both algorithms consumed slightly more memory when the number of PS-rules was larger, which implies that the amount of memory required is manageable even when the number of rules is large. Figure 55, for example, shows the results for the three datasets when Greedy and our algorithms enforced (70 %, 5, 2)-coherence and Apriori 2^5-anonymity. Our algorithms needed a similar amount of memory to that of Apriori to anonymize BMS1 and BMS2 and

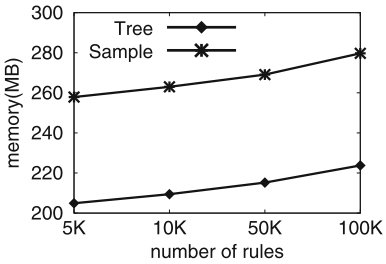


Fig. 54 Memory versus # rules (POS)

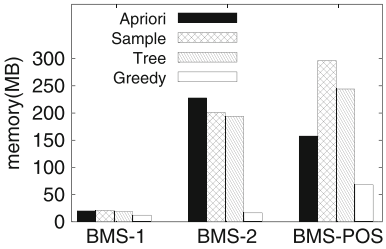


Fig. 55 Memory versus datasets

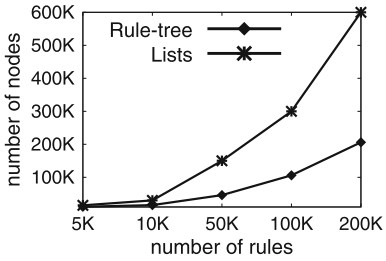


Fig. 56 Nodes of Rule-tree versus rules (POS)

a larger one, which did not exceed 300 MB, for POS. This is because the memory needed to store rules is a small fraction of the total memory required, most of which is spent to store generalized items and the temporary dataset used in rule checking. For instance, storing 200 K rules needed less than 3 MB of memory, while the temporary dataset needed more than 100 MB. Also, as can be seen in Fig. 56, which shows the number of nodes of the Rule-tree, used by our algorithms, the number of nodes needed by the Rule-tree increased sublinearly with the number of PS-rules and is much smaller than the number of nodes that would be needed if lists, each containing the items of a rule [52], were used instead.

6.4.3 Performance of PS-rule checking strategy

We evaluate the performance of our rule checking strategy by comparing the total number of rules that our algorithms check to that of Baseline. For brevity, we report the results for BMS1 only, omitting the results for BMS2 and POS, which were similar.

Impact of parameter k Figure 57 shows the number of checks performed for BMS1 when k varies in [2, 10]. All algorithms checked fewer rules as k increased, since they could construct an anonymized dataset after just a few iterations. However, our algorithms checked at least 3 and up to 5.5 times less rules than Baseline, which confirms that a large number of PS rules were pruned. Note also that Sample-based checked slightly more rules than Tree-based,

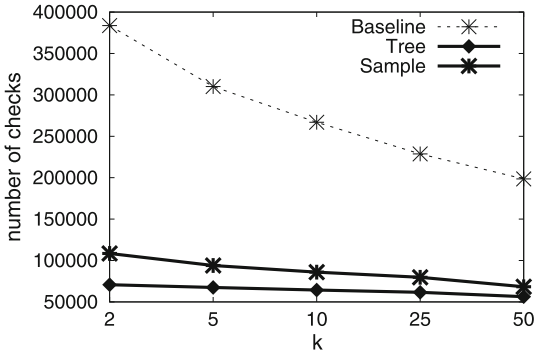


Fig. 57 # checks versus k (BMS1)

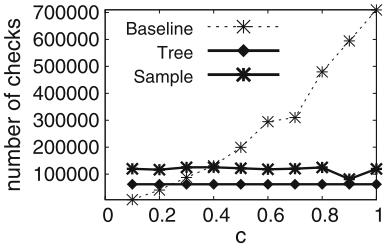


Fig. 58 # checks versus c (BMS1)

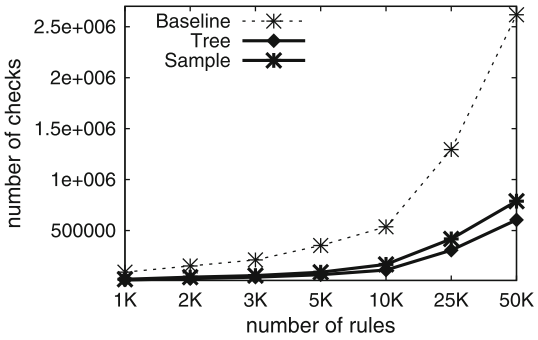


Fig. 59 # checks versus # rules (BMS1)

because it constructed less generalized items than Tree-based did. Moreover, Sample-based had to check some rules twice.

Impact of parameter c The result for measuring the number of rules checked against different c values in [0.1, 1] is given in Fig. 58. Baseline performed fewer checks for smaller c values, because it stops the first time the confidence of a rule exceeds c, which occurred at the very first iterations. The number of checks maximized when $c = 1$, since all checks then correspond to evaluating the support of rules. In contrast, the number of checks performed by Tree-based was not affected by varying c, because c was not used as a criterion to stop the exploration and expansion of the generalization tree. Sample-based performed slightly more checks than Tree-based did for the reasons explained above.

Impact of number of PS-rules Figure 59 shows that all algorithms perform more checks as the number of rules increases. This is because the random selection of the rules makes it unlikely for a rule to be protected as a result of protecting another. Both our algorithms scaled better than Baseline, which demonstrates the effectiveness of our rule checking strategy.

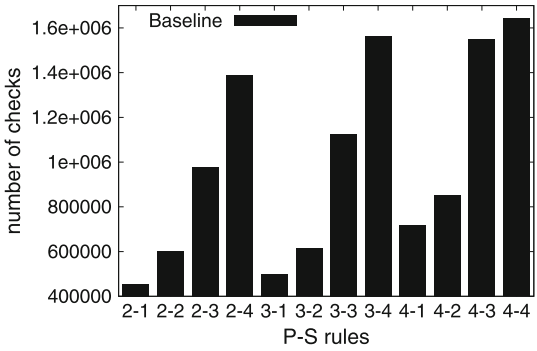


Fig. 60 # checks versus rules in S_1 (BMS1)

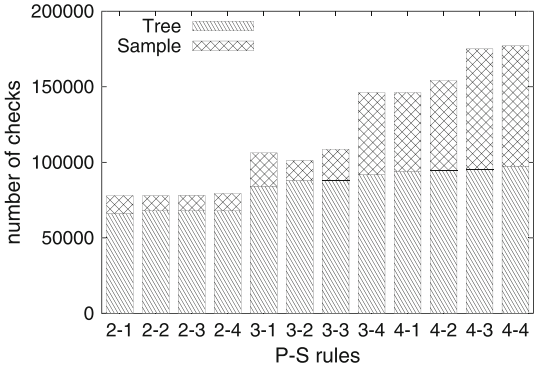


Fig. 61 # checks versus rules in S_1 (BMS1)

The number of checks performed by Sample-based was on average 3.5 times smaller than that of Baseline, but 41 % larger than that of Tree-based.

Impact of different PS-rules The number of checks performed for sets of rules, whose antecedents and consequents have a varying size, is in Figs. 60 and 61. By comparing these figures, it can be seen that our algorithms significantly outperformed Baseline. Sample-based checked at least 6 and up to 17 times less rules than Baseline, and Tree-based performed slightly better. We also observed that all algorithms performed an increasing number of checks when rules have large antecedents or consequents. This is because rules in these sets tend to have large confidence, so it took more iterations for all algorithms to protect them.

7 Conclusions and future work

Existing transaction data anonymization techniques often produce excessively distorted and inadequately protected data, because they do not capture fine-grained privacy requirements that are common in real-world data sharing scenarios and do not prevent identity and sensitive information disclosure together. In this work, we overcome these limitations by making two important contributions. First, we introduced a novel privacy model that allows data publishers to express a wide range of fine-grained privacy requirements for both identity and sensitive information disclosure and is more general than existing ones. Second, we designed two efficient and effective anonymization algorithms to enforce our privacy model. Our extensive experiments confirmed that our algorithms significantly outperform the state-of-the-art

method in terms of data utility, while they are scalable and can prevent both identity and sensitive information disclosure.

We consider our approach as an important step toward capturing and enforcing privacy requirements to effectively anonymize transaction data for real-world applications. We also believe that it provides a foundation for some future studies. First, while identity and sensitive information disclosure are the main concerns in data publishing, it is worth examining membership disclosure, in which inferring whether an individual's record is contained in the published data is to be prevented [63]. Second, we aim to extend our approach to anonymize disk-resident data with small memory consumption and I/O overhead.

Acknowledgments We would like to thank the handling editor and anonymous reviewers for their insightful comments, as well as the authors of [74] and [91] for providing the implementations of Apriori and Greedy, respectively. The first author was supported by a Research Fellowship from the Royal Academy of Engineering.

Appendix

Proof of Theorem 1 We observe that our problem can be reduced from the NP-hard problem of optimal privacy-constrained anonymization (OPC) [51]. The latter problem seeks to transform $\mathcal{D}$ into $\tilde{\mathcal{D}}$ that has a minimum $UL(\tilde{\mathcal{D}})$, using the set-based item generalization model, w.r.t. two sets of constraints $\mathscr{U}$ (on utility) and $\mathscr{P}$ (on privacy), and parameters k and s. We can map an instance of OPC to an instance of our problem in polynomial time by constructing Θ in such a way that, for each $p \in \mathscr{P}$, we specify a PS-rule whose antecedent contains all the items in p and no other items, $s = 0\,\%$ to disallow suppression, and $c = 1$. Then, $\tilde{\mathcal{D}}$ is a solution to our problem if and only if it is a solution to the OPC problem. □

Proof of Theorem 2 Assume that a $\tilde{\mathcal{D}}$ in which the PS-rules in Θ are protected can be constructed from $\mathcal{D}$, for given k and c, when $sup(I \cup J, \mathcal{D}) > N \times c$, for a rule $I \rightarrow J$ in Θ. Then, from Definition 4, we have $\frac{sup(\bigcup_{\forall i \in I} \Phi(i) \cup J, \tilde{\mathcal{D}})}{sup(\bigcup_{\forall i \in I} \Phi(i), \tilde{\mathcal{D}})} \leq c$. Since $sup(\bigcup_{\forall i \in I} \Phi(i) \cup J, \tilde{\mathcal{D}}) \geq sup(I \cup J, \mathcal{D})$ from the way Φ works, we have $\frac{sup(\bigcup_{\forall i \in I} \Phi(i) \cup J, \tilde{\mathcal{D}})}{sup(\bigcup_{\forall i \in I} \Phi, \tilde{\mathcal{D}})} \geq \frac{sup(I \cup J, \mathcal{D})}{N}$. Thus, $\frac{sup(\bigcup_{\forall i \in I} \Phi(i) \cup J, \tilde{\mathcal{D}})}{sup(\bigcup_{\forall i \in I} \Phi, \tilde{\mathcal{D}})} > c$, which contradicts our assumption and proves the theorem true. □

Proof of Theorem 3 Consider a dataset $\tilde{\mathcal{D}}$ constructed using a generalization function Φ that maps each and every $i \in \mathcal{P}$ to the same $\tilde{i} \in \tilde{\mathcal{P}}$. Since we have $sup(\tilde{i}, \tilde{\mathcal{D}}) = N \geq k$ for all rules in Θ, these rules satisfy Condition (1) of Definition 4 for $\tilde{\mathcal{D}}$. Also, as $sup(\bigcup_{\forall i \in I_q} \Phi(i) \cup J_q, \tilde{\mathcal{D}}) \leq N \times c$ and $sup(\tilde{i}, \tilde{\mathcal{D}}) = N$ hold, we have $\frac{sup(\bigcup_{\forall i \in I_q} \Phi(i) \cup J_q, \tilde{\mathcal{D}})}{sup(\bigcup_{\forall i \in I_q} \Phi(i), \tilde{\mathcal{D}})} \leq c$. So the rules in Θ also satisfy Condition (2) of Definition 4, and are all protected in $\tilde{\mathcal{D}}$. Thus, $\tilde{\mathcal{D}}$ is a generalized dataset that is constructed from $\mathcal{D}$ as required. □

Proof of Theorem 4 Assume that $UL(\tilde{\mathcal{D}}_1) < UL(\tilde{\mathcal{D}}_2)$. For each generalized item $\tilde{i}$ in both C_1 and C_2, we have $UL(\tilde{i}, \tilde{\mathcal{D}}_1) = UL(\tilde{i}, \tilde{\mathcal{D}}_2)$. Thus, there must be sets of generalized items $C_1' = C_1 \backslash C_2$ and $C_2' = C_2 \backslash C_1$ such that $\sum_{\tilde{i_x} \in C_1'} UL(\tilde{i_x}, \tilde{\mathcal{D}}_1) < \sum_{\tilde{i_y} \in C_2'} UL(\tilde{i_y}, \tilde{\mathcal{D}}_2)$ or $\sum_{\forall \tilde{i_x} \in C_1'} ((2^{|\tilde{i_x}|} - 1) \times w(\tilde{i_x}) \times sup(\tilde{i_x}, \tilde{\mathcal{D}}_1)) < \sum_{\forall \tilde{i_y} \in C_2'} ((2^{|\tilde{i_y}|} - 1) \times w(\tilde{i_y}) \times sup(\tilde{i_y}, \tilde{\mathcal{D}}_2))$, in order for our assumption to hold. However, this cannot be true because for each $\tilde{i_x} \in C_1'$ we have $|\tilde{i_x}| = \sum_{\forall \tilde{i_q} \in desc(\tilde{i_x})} |\tilde{i_q}|$ (by definition), $sup(\tilde{i_x}, \tilde{\mathcal{D}}_1) \geq \sum_{\forall \tilde{i_q} \in desc(\tilde{i_x})} sup(\tilde{i_q}, \tilde{\mathcal{D}}_2)$

(by *B-Split*), and $w(\tilde{i_x}) \geq \sum_{\forall \tilde{i_q} \in desc(\tilde{i_x})} w(\tilde{i_q})$ (the condition given in Theorem 4). Thus, we cannot have $UL(\tilde{\mathcal{D}}_1) < UL(\tilde{\mathcal{D}}_2)$. □

Proof of Theorem 5 Consider an arbitrary generalized dataset $\tilde{\mathcal{D}}$ that is constructed from $\mathcal{D}$ using a set-based generalization function Φ. In the following, we prove that $I \rightarrow J$ is protected in $\tilde{\mathcal{D}}$ by showing that it satisfies both conditions of Definition 4. From $sup(I, \mathcal{D}) \leq sup(\bigcup_{\forall i \in I} \Phi(i), \tilde{\mathcal{D}})$ and the condition (1) of the Theorem 5, we get that $sup(\bigcup_{\forall i \in I} \Phi(i), \tilde{\mathcal{D}}) \geq k$. Thus, $I \rightarrow J$ satisfies the condition (1) of Definition 4. Also, from $sup(J, \mathcal{D}) \geq sup(\bigcup_{\forall i \in I} \Phi(i) \cup J, \tilde{\mathcal{D}})$ and the condition (2) of Theorem 5, we have that $sup(\bigcup_{\forall i \in I} \Phi(i) \cup J, \tilde{\mathcal{D}}) \leq c \times k$. By combining the latter relation with $sup(\bigcup_{\forall i \in I} \Phi(i), \tilde{\mathcal{D}}) \geq k$, we get $\frac{sup(\bigcup_{\forall i \in I} \Phi(i) \cup J, \tilde{\mathcal{D}})}{sup(\bigcup_{\forall i \in I} \Phi(i), \tilde{\mathcal{D}})} \leq c$. Thus, $I \rightarrow J$ also satisfies the condition (2) of Definition 4. □

Proof of Theorem 6 Assume that $I \rightarrow J$ is protected in $\tilde{\mathcal{D}}_1$, not protected in $\tilde{\mathcal{D}}_2$ and there is no item $i \in I$ that maps to $\tilde{i_l}$ or $\tilde{i_r}$. By the construction of C_2, i is not mapped to $\tilde{i}$ either. Thus, we have $sup(\bigcup_{\forall i \in I} \Phi(i), \tilde{\mathcal{D}}_1) = sup(\bigcup_{\forall i \in I} \Phi(i), \tilde{\mathcal{D}}_2)$, since other than $\tilde{i_l}$ and $\tilde{i_r}$, the support of each generalized item in $\tilde{\mathcal{D}}_2$ and $\tilde{\mathcal{D}}_1$ is the same. This implies $\frac{sup(\bigcup_{\forall i \in I} \Phi(i) \cup J, \tilde{\mathcal{D}}_1)}{sup(\bigcup_{\forall i \in I} \Phi(i), \tilde{\mathcal{D}}_1)} = \frac{sup(\bigcup_{\forall i \in I} \Phi(i) \cup J, \tilde{\mathcal{D}}_2)}{sup(\bigcup_{\forall i \in I} \Phi(i), \tilde{\mathcal{D}}_2)}$. Thus, $I \rightarrow J$ is protected in $\tilde{\mathcal{D}}_2$, which contradicts our assumption and proves the theorem true. □

Proof of Theorem 7 Assume that $I' \rightarrow J$ is not protected in $\tilde{\mathcal{D}}$. Due to condition (1), we have $sup(\bigcup_{\forall i \in I} \Phi(i), \tilde{\mathcal{D}}) \geq k$ and $\frac{sup(\bigcup_{\forall i \in I} \Phi(i) \cup J, \tilde{\mathcal{D}})}{sup(\bigcup_{\forall i \in I} \Phi(i), \tilde{\mathcal{D}})} \leq c$, and, from conditions (2) and (3), we have $sup(\bigcup_{\forall i' \in I'} \Phi(i'), \tilde{\mathcal{D}}) = sup(\bigcup_{\forall i \in I} \Phi(i), \tilde{\mathcal{D}})$ since all items in I' are contained in I and all items in I are mapped to $\bigcup_{\forall i \in I} \Phi(i)$. Thus, $I' \rightarrow J$ is protected by Definition 4, which contradicts our assumption. □

Proof of Theorem 8 Assume that $I \rightarrow J'$ is not protected in $\tilde{\mathcal{D}}$. $sup(\bigcup_{\forall i \in I} \Phi(i), \tilde{\mathcal{D}}) \geq k$ holds from condition (1), hence $\frac{sup(\bigcup_{\forall i \in I} \Phi(i) \cup J', \tilde{\mathcal{D}})}{sup(\bigcup_{\forall i \in I} \Phi(i), \tilde{\mathcal{D}})} > c$ must hold, for our assumption to be true. From $sup(J', \tilde{\mathcal{D}}_{\tilde{I}}) = sup(\bigcup_{\forall i \in I} \Phi(i) \cup J', \tilde{\mathcal{D}})$, $sup(J, \tilde{\mathcal{D}}_{\tilde{I}}) = sup(\bigcup_{\forall i \in I} \Phi(i) \cup J, \tilde{\mathcal{D}})$, and condition (2), we get $\frac{sup(\bigcup_{\forall i \in I} \Phi(i) \cup J', \tilde{\mathcal{D}})}{sup(\bigcup_{\forall i \in I} \Phi(i), \tilde{\mathcal{D}})} \leq \frac{sup(\bigcup_{\forall i \in I} \Phi(i) \cup J, \tilde{\mathcal{D}})}{sup(\bigcup_{\forall i \in I} \Phi(i), \tilde{\mathcal{D}})} < c$, which contradicts our assumption. □

Proof of Theorem 9 Let $X_1, \ldots, X_n$ be independent Poisson trials and the probability $Pr(X_i) = p_i$. Let also $X = \sum_{\forall i \in [1,n]} X_i$ and $\mu = E[X]$. For any $\gamma > 0$, the additive Chernoff bound $Pr(|X - \mu| \geq \gamma) \leq (2 \times e^{-2n\gamma^2})$ holds. By setting $|\mathcal{D}_s| = n$, $\gamma = \epsilon$, $X = \frac{sup(I, \mathcal{D}_s)}{|\mathcal{D}_s|}$, and $\mu = \frac{sup(I, \mathcal{D})}{|\mathcal{D}|}$, we get $Pr(|\frac{sup(I, \mathcal{D}_s)}{|\mathcal{D}_s|} - \frac{sup(I, \mathcal{D})}{|\mathcal{D}|}| \geq \epsilon) \leq (2 \times e^{-2|\mathcal{D}_s|\epsilon^2})$. After setting $\delta \geq 2 \times e^{-2|\mathcal{D}_s|\epsilon^2}$ and some calculations, we get $|\mathcal{D}_s| \geq \frac{ln(\frac{2}{\delta})}{2 \times \epsilon^2}$, which proves the theorem true. □

Theorem 10 *The time complexity of Algorithm 4 is* $O(2^{|\mathcal{P}|} \times |\mathcal{S}| \times N)$ *and the space complexity is* $O(2^{|\mathcal{P}|} \times |\mathcal{S}| + N \times |\mathcal{I}|)$.

Proof We first examine the time complexity of *B-Split*, *Update*, *Replace*, and *Check*. *B-Split* requires $O\left(|\mathcal{P}|^2 + |\mathcal{P}|\right) \approx O(|\mathcal{P}|^2)$ time, where $|\mathcal{P}|$ is the size of the largest possible generalized item, since it examines all pairs of items mapped to this generalized item and then assigns

all public items to the created seeds. *Update* scans $\tilde{\mathcal{D}}$ once to replace $\tilde{i}$ with $\tilde{i_l}$ and $\tilde{i_r}$, hence it is computed in $O(|\mathcal{P}| \times N)$ time, where N is the number of transactions in $\tilde{\mathcal{D}}$. *Replace* has the same time complexity with *Update* and *Check* needs $O(|\Theta| \times (N + |\mathcal{P}|))$ time; $O(N)$ time to form $I \rightarrow J$ and determine if it is protected, $O(|\mathcal{P}| \times (log(|\mathcal{P}|) + |\Theta|))$ time to construct Θ' using *Find-rules* in the worst-case in which all rules have at least one item mapped to $\tilde{i}$ in their antecedent, and $O(|\Theta'| \times N)$ time to compute the support and confidence of these rules. Since Algorithm 4 performs up to $\lfloor \frac{2\times|\mathcal{P}|-1}{2} \rfloor \approx |\mathcal{P}|$ recursive calls, one for each edge in the generalization tree, it needs $O\left(|\mathcal{P}| \times \left(|\mathcal{P}|^2 + |\mathcal{P}| \times N + |\Theta| \times (N + |\mathcal{P}|)\right)\right)$ time, which can be approximated as $O(2^{|\mathcal{P}|} \times |\mathcal{S}| \times N)$ in the worst case when $|\Theta| = (2^{|\mathcal{P}|} - 1) \times |\mathcal{S}|$ and assuming that $N > |\mathcal{P}|$. Note that $|\Theta| = (2^{|\mathcal{P}|} - 1) \times |\mathcal{S}|$ when it contains all possible rules and there is no rule whose antecedent is the same with another rule and its consequent is a superset of that of the second rule (otherwise the former rule is redundant due to its lower confidence).

We then examine the space complexity of *Update*, *Replace*, and *Check*. *Update* and *Replace* need to store $O(N \times |\mathcal{I}|)$ items each. The Rule-tree used in *Check* is created once and stores $O(2^{|\mathcal{P}|} \times |\mathcal{S}|)$ items, as inserting a rule adds one item as a tree node and $|\mathcal{S}|$ items in the consequent-list. Q needs to store $O(|\mathcal{P}|^2)$ items when the generalization tree is the tallest one, hence Algorithm 4 requires $O(2^{|\mathcal{P}|} \times |\mathcal{S}| + N \times |\mathcal{I}|)$ space. □

Theorem 11 *The time complexity of Algorithm 5 is* $O(2^{|\mathcal{P}|} \times |\mathcal{S}| \times N)$ *and the space complexity is* $O(2^{|\mathcal{P}|} \times |\mathcal{S}| + N \times |\mathcal{I}|)$.

Proof The time complexity of the sample-based partitioning phase of Algorithm 5 is $O(h \times (|\mathcal{P}|^2 + |\mathcal{P}| \times N + |\Theta| \times (|\mathcal{P}| + |\mathcal{D}_s|)))$, where h is the height of the generalization tree $\mathcal{G}$ returned by *Sample-Based-Partition*. This is because the latter function calls *Split* and *Update*, whose cost was examined in Theorem 10, as well as *Check*, whose cost is $O(|\Theta| \times (|\mathcal{P}| + |\mathcal{D}_s|))$. The top-down cut revision phase needs $O(h' \times (|\mathcal{P}|^2 + |\mathcal{P}| \times N + |\mathcal{P}| \times (log(|\mathcal{P}) + \Theta) + \Theta \times (|\mathcal{P}| + N)))$, where h' is the number of executions of the while loop of step 9 in Algorithm 5. Finally, steps 24–29 need $O(h'' \times (|\Theta| \times (|\mathcal{P}| + N) + |\mathcal{P}| + N))$ time, assuming that all nodes of $\mathcal{G}$ are merged to their ancestors that lie h'' levels above them, since *Check* and *Merge-siblings* take $O(|\Theta| \times (|\mathcal{P}| + N))$ and $O(|\mathcal{P}| + N)$ time, respectively. It also holds $h' + h \leq |\mathcal{P}|$ and $h'' \leq |\mathcal{P}|$, as $\mathcal{G}$ has up to $|\mathcal{P}|$ levels. Thus, Algorithm 5 takes $O(|\mathcal{P}| \times (|\mathcal{P}|^2 + |\mathcal{P}| \times N + |\Theta| \times (|\mathcal{P}| + N)))$ time, or approximately $O(2^{|\mathcal{P}|} \times |\mathcal{S}| \times N)$, in the worst case when $|\Theta| = O(2^{|\mathcal{P}|} \times |\mathcal{S}|)$ and assuming that $N > |\mathcal{P}|$. The space complexity of *Update* and *Check* was examined in Theorem 10 and the cost of storing $\mathcal{G}$ is $O(|\mathcal{P}|^2)$ items. Thus, Algorithm 5 requires $O(2^{|\mathcal{P}|} \times |\mathcal{S}| + N \times |\mathcal{I}|)$ space. □

COUNT() Queries and ARE Computation

Consider an SQL-like COUNT() query
Q : `SELECT COUNT(`$\tilde{T}_n$`(or` T_n`)) FROM` $\tilde{\mathcal{D}}$ `(or` $\mathcal{D}$`)`
`WHERE` $\tilde{I}J$ `supports` $\tilde{T}_n$ `in` $\tilde{\mathcal{D}}$ `(or` IJ `supports` T_n `in` $\mathcal{D}$`)`

where $\tilde{I}$ and I are itemsets comprised of public items in $\tilde{\mathcal{D}}$ and $\mathcal{D}$, respectively, and J is an itemset comprised of sensitive items. Since data recipients have only access to $\tilde{\mathcal{D}}$, they need to estimate an answer for Q. This can be performed by computing the probability $P(\tilde{T}_n, Q)$ that $\tilde{T}_n$, the anonymized version of a transaction T_n in $\mathcal{D}$, satisfies Q. Assume that a generalized item $\tilde{i_m}$ in $\tilde{T}_n$ is interpreted as any possible subset of the items mapped to it with equal probability, and that there are no correlations among generalized items [27,29,42].

$P(\tilde{T}_n, Q)$ is given by $\Pi_{\forall \tilde{i_m} \in \tilde{T}_n} \frac{2^{|\tilde{i_m}|-c(I)}}{2^{|\tilde{i_m}|}-1}$ where $c : \mathcal{I} \rightarrow [0, |i_m|]$ is a function that, given I, returns the number of items in I that are mapped to $\tilde{i_m}$ in $\tilde{\mathcal{D}}$. An approximate answer $e(Q)$ to Q is then derived by summing the corresponding probabilities across all transactions $\tilde{T}_n$ in $\tilde{\mathcal{D}}$ that support J.

We considered two workloads W_1 and W_2, each comprised of 1,000 queries, whose items were selected uniformly at random. Queries in W_1 involve a 2-itemset in $\mathcal{P}$, whereas those in W_2 require retrieving a 3-itemset comprised of 2 public and 1 sensitive items.

References

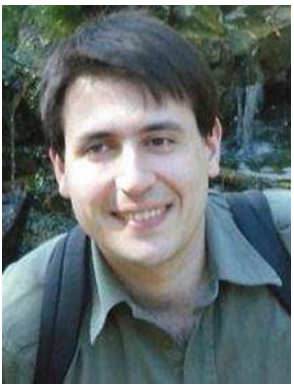
1. Health Insurance Portability and Accountability Act of 1996 United States Public Law
2. Abul O, Bonchi F, Giannotti F (2010) Hiding sequential and spatiotemporal patterns. TKDE 22(12):1709–1723
3. Abul O, Bonchi F, Nanni M (2008) Never walk alone: uncertainty for anonymity in moving objects databases. In: ICDE, pp 376–385
4. Aggarwal CC (2005) On k-anonymity and the curse of dimensionality. In: VLDB, pp 901–909
5. Aggarwal CC, Li Y, Yu PS (2011) On the hardness of graph anonymization. In: ICDM, pp 1002–1007
6. Aggarwal CC, Yu PS (2008) Privacy-preserving data mining: models and algorithms. Springer, Berlin
7. Agrawal R, Johnson CM (2007) Securing electronic health records without impeding the flow of information. Int J Med Inform 76(5–6):471–479
8. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules in large databases. In: VLDB, pp 487–499
9. Bacchus F, Grove AJ, Halpern JY, Koller D (1996) From statistical knowledge bases to degrees of belief. Artif Intell 87(1–2):75–143
10. Barak B, Chaudhuri K, Dwork C, Kale S, McSherry F, Talwar K (2007) Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In: PODS, pp 273–282
11. Barbaro M, Zeller T (2006) A face is exposed for aol searcher no. 4417749. New York Times, Aug
12. Basu S, Mooney RJ, Pasupuleti KV, Ghosh J (2001) Evaluating the novelty of text-mined rules using lexical knowledge. In: KDD, pp 233–238
13. Blum A, Dwork C, McSherry F, Nissim K (2005) Practical privacy: the sulq framework. In: PODS, pp 128–138
14. Bonchi F, Lakshmanan LVS (2011) Trajectory anonymity in publishing personal mobility data. SIGKDD Explor 13(1):30–42
15. Brickell J, Shmatikov V (2008) The cost of privacy: destruction of data-mining utility in anonymized data publishing. In: KDD, pp 70–78
16. Cao J, Karras P, Raïssi C, Tan K (2010) *rho*-Uncertainty: inference-proof transaction anonymization. PVLDB 3(1):1033–1044
17. Centers for Medicare and Medicaid Services (2011) International classification of diseases, 9th revision, clinical modification (icd-9-cm). http://www.cdc.gov/nchs/icd/icd9cm.htm. Accessed 20 March 2011
18. Chen K, Liu L (2011) Geometric data perturbation for privacy preserving outsourced data mining. Knowl Inf Syst 29(3):657–695
19. Chen R, Mohammed N, Fung BCM, Desai BC, Xiong L (2011) Publishing set-valued data via differential privacy. PVLDB 4(11):1087–1098
20. Cormode G (2011) Personal privacy vs population privacy: learning to attack anonymization. In: KDD, pp 1253–1261
21. Cormode G, Li N, Li T, Srivastava D (2010) Minimizing minimality and maximizing utility: analyzing method-based attacks on anonymized data. PVLDB 3(1):1045–1056
22. Dwork C (2006) Differential privacy. In: ICALP, pp 1–12
23. Dwork C (2008) Differential privacy: a survey of results. In: TAMC, pp 1–19
24. Friedman A, Schuster A (2010) Data mining with differential privacy. In: KDD, pp 493–502
25. Fung BCM, Wang K, Chen R, Yu PS (2010) Privacy-preserving data publishing: a survey on recent developments. ACM Comput Surv 42(4):1–53
26. Fung BCM, Wang K, Chen R, Yu PS (2005) Top-down specialization for information and privacy preservation. In: ICDE, pp 205–216

27. Ghinita G, Kalnis P, Tao Y (2011) Anonymous publication of sensitive transactional data. IEEE TKDE 23(2):161–174
28. Ghinita G, Karras P, Kalnis P, Mamoulis N (2007) Fast data anonymization with low information loss. In: VLDB, pp 758–769
29. Ghinita G, Tao Y, Kalnis P (2008) On the anonymization of sparse high-dimensional data. In: ICDE, pp 715–724
30. Gkoulalas-Divanis A, Loukides G (2011) Revisiting sequential pattern hiding to enhance utility. In: KDD, pp 1316–1324
31. Gkoulalas-Divanis A, Verykios VS (2009) Hiding sensitive knowledge without side effects. Knowl Inf Syst 20(3):263–299
32. Gkoulalas-Divanis A, Verykios VS, Mokbel MF (2009) Identifying unsafe routes for network-based trajectory privacy. In: SDM, pp 942–953
33. Hagerup T, Rüb C (1990) A guided tour of chernoff bounds. Inf Process Lett 33(6):305–308
34. Hay M, Li C, Miklau G, Jensen D (2009) Accurate estimation of the degree distribution of private networks. In: ICDM, pp 169–178
35. He Y, Naughton JF (2009) Anonymization of set-valued data via top-down, local generalization. PVLDB 2(1):934–945
36. Iyengar VS (2002) Transforming data to satisfy privacy constraints. In: KDD, pp 279–288
37. Karr AF, Kohnen CN, Oganian A, Reiter JP, Sanil AP (2006) A framework for evaluating the utility of data altered to protect confidentiality. Am Stat 60(3):224–232
38. Khoshgozaran A, Shahabi C, Shirani-Mehr H (2011) Location privacy: going beyond k-anonymity, cloaking and anonymizers. Knowl Inf Syst 26(3):435–465
39. Kifer D, Machanavajjhala A (2011) No free lunch in data privacy. In: SIGMOD, pp 193–204
40. Korolova A, Kenthapadi K, Mishra N, Ntoulas A (2009) Releasing search queries and clicks privately. In: WWW, pp 171–180
41. LeFevre K, DeWitt DJ, Ramakrishnan R (2008) Workload-aware anonymization techniques for large-scale datasets. TODS 33(3):1–47
42. LeFevre K, DeWitt DJ, Ramakrishnan R (2006) Mondrian multidimensional k-anonymity. In: ICDE, p 25
43. Li J, Liu J, Baig MM, Wong RC (2011) Information based data anonymization for classification utility. DKE 70(12):1030–1045
44. Li N, Li T, Venkatasubramanian S (2007) t-closeness: Privacy beyond k-anonymity and l-diversity. In: ICDE, pp 106–115
45. Li T, Li N (2006) Optimal k-anonymity with flexible generalization schemes through bottom-up searching. In: ICDMW, pp 518–523
46. Liu B, Hsu W, Chen S (1997) Using general impressions to analyze discovered classification rules. In: KDD, pp 31–36
47. Liu B, Hsu W, Wang K, Chen S (1999) Visually aided exploration of interesting association rules. In: PAKDD, pp 380–389
48. Liu J, Wang K (2010) Anonymizing transaction data by integrating suppression and generalization. In: PAKDD, pp 171–180
49. Loukides G, Denny JC, Malin B (2010) The disclosure of diagnosis codes can breach research participants' privacy. J Am Med Inform Assoc 17(3):322–327
50. Loukides G, Gkoulalas-Divanis A, Malin B (2010) Anonymization of electronic medical records for validating genome-wide association studies. Proc Natl Acad Sci 17(107):7898–7903
51. Loukides G, Gkoulalas-Divanis A, Malin B (2011) COAT: Constraint-based anonymization of transactions. Knowl Inf Syst 28(2):251–282
52. Loukides G, Gkoulalas-Divanis A, Shao J (2010) Anonymizing transaction data to eliminate sensitive inferences. In: DEXA, pp 400–415
53. Machanavajjhala A, Gehrke J, Götz M (2009) Data publishing against realistic adversaries. PVLDB 2(1):790–801
54. Machanavajjhala A, Gehrke J, Kifer D, Venkitasubramaniam M (2006) l-diversity: privacy beyond k-anonymity. In: ICDE, p 24
55. Machanavajjhala A, Kifer D, Abowd JM, Gehrke J, Vilhuber L (2008) Privacy: theory meets practice on the map. In: ICDE, pp 277–286
56. Medforth N, Wang K (2011) Privacy risk in graph stream publishing for social network data. In: ICDM, pp 437–446
57. Meyerson A, Williams R (2004) On the complexity of optimal k-anonymity. In: PODS, pp 223–228
58. Mohammed N, Chen R, Fung BCM, Yu PS (2011) Differentially private data release for data mining. In: KDD, pp 493–501

59. Mohammed N, Fung BCM, Debbabi M (2009) Walking in the crowd: anonymizing trajectory data for pattern analysis. In: CIKM, pp 1441–1444
60. Mohammed N, Fung BCM, Hung PCK, Lee C (2009) Anonymizing healthcare data: a case study on the blood transfusion service. In: KDD, pp 1285–1294
61. Mohammed N, Fung BCM, Hung PCK, Lee C (2010) Centralized and distributed anonymization for high-dimensional healthcare data. TKDD 4(4):18
62. Narayanan A, Shmatikov V (2008) Robust de-anonymization of large sparse datasets. In: IEEE S&P, pp 111–125
63. Nergiz ME, Atzori M, Clifton C (2007) Hiding the presence of individuals from shared databases. In: SIGMOD'07, pp 665–676
64. Nergiz ME, Clifton C, Nergiz AE (2009) Multirelational k-anonymity. TKDE 21(8):1104–1117
65. Texas Dept. of State Health Services (2008) User manual of texas hospital inpatient discharge public use data file. http://www.dshs.state.tx.us/THCIC/
66. Oliveira SRM, Zaïane OR (2003) Protecting sensitive knowledge by data sanitization. In: ICDM, pp 613–616
67. Qiu L, Li Y, Wu X (2008) Protecting business intelligence and customer privacy while outsourcing data mining tasks. Knowl Inf Syst 17(1):99–120
68. Samarati P (2001) Protecting respondents identities in microdata release. TKDE 13(9):1010–1027
69. Srikant R, Vu Q, Agrawal R (1997) Mining association rules with item constraints. In: KDD, pp 67–73
70. Sweeney L (2002) k-Anonymity: a model for protecting privacy. IJUFKS 10(5):557–570
71. Tai C, Yu P, Yang D, Chen M (2011) Privacy-preserving social network publication against friendship attacks. In: KDD, pp 1262–1270
72. Teng Z, Du W (2009) A hybrid multi-group approach for privacy-preserving data mining. Knowl Inf Syst 19(2):133–157
73. Terrovitis M, Mamoulis N (2008) Privacy preservation in the publication of trajectories. In: MDM, pp 65–72
74. Terrovitis M, Mamoulis N, Kalnis P (2008) Privacy-preserving anonymization of set-valued data. PVLDB 1(1):115–125
75. Terrovitis M, Mamoulis N, Kalnis P (2011) Local and global recoding methods for anonymizing set-valued data. VLDB J 20(1):83–106
76. Velardi P, Cucchiarelli A, Petit M (2007) A taxonomy learning method and its application to characterize a scientific web community. TKDE 19(2):180–191
77. Verykios V, Elmagarmid AK, Bertino E, Saygin Y, Dasseni E (2004) Association rule hiding. TKDE 16(4):434–447
78. Wang K, Fung BCM, Yu PS (2007) Handicapping attacker's confidence: an alternative to k-anonymization. Knowl Inf Syst 11(3):345–368
79. Wang K, Fung BCM, Yu PS (2005) Template-based privacy preservation in classification problems. In: ICDM, pp 466–473
80. Wang K, Xu Y, Fu A, Wong RCW (2009) FF-anonymity: when quasi-identifiers are missing. In: ICDE, pp 1136–1139
81. Wong RCW, Fu A, Wang K, Pei J (2007) Minimality attack in privacy preserving data publishing. In: VLDB, pp 543–554
82. Wong RCW, Fu A, Wang K, Pei J (2009) Anonymization-based attacks in privacy-preserving data publishing. TODS 34(2):1–46
83. Wong RCW, Li J, Fu A, Wang K (2006) Alpha-k-anonymity: an enhanced k-anonymity model for privacy-preserving data publishing. In: KDD, pp 754–759
84. Xiao X, Tao Y (2006) Anatomy: simple and effective privacy preservation. In: VLDB, pp 139–150
85. Xiao X, Tao Y (2006) Personalized privacy preservation. In: SIGMOD, pp 229–240
86. Xiao X, Tao Y (2007) M-invariance: towards privacy preserving re-publication of dynamic datasets. In: SIGMOD, pp 689–700
87. Xiao X, Tao Y, Koudas N (2010) Transparent anonymization: Thwarting adversaries who know the algorithm. TODS 35(2):1–48
88. Xiao X, Wang G, Gehrke J (2010) Differential privacy via wavelet transforms. In: ICDE, pp 225–236
89. Xu J, Wang W, Pei J, Wang X, Shi B, Fu AW-C (2006) Utility-based anonymization using local recoding. In: KDD, pp 785–790
90. Xu Y, Fung BCM, Wang K, Fu AW, Pei J (2008) Publishing sensitive transactions for itemset utility. In: ICDM, pp 1109–1114
91. Xu Y, Wang K, Fu AW-C, Yu PS (2008) Anonymizing transaction databases for publication. In: KDD, pp 767–775

92. Yakut I, Polat H (2012) Privacy-preserving hybrid collaborative filtering on cross distributed data. Knowl Inf Syst 30(2):405–433
93. Ying X, Wu X (2011) On link privacy in randomizing social networks. Knowl Inf Syst 28(3):645–663
94. Zhang L, Jajodia S, Brodsky A (2007) Information disclosure under realistic assumptions: privacy versus optimality. In: CCS, pp 573–583
95. Zhou B, Pei J (2011) The k-anonymity and l-diversity approaches for privacy preservation in social networks against neighborhood attacks. Knowl Inf Syst 28(1):47–77

Author Biographies



Grigorios Loukides is currently an Assistant Professor in the School of Computer Science & Informatics, Cardiff University, and a Royal Academy of Engineering Research Fellow. Before joining Cardiff University, he was a Postdoctoral Research Fellow in the Health Information Privacy LABoratory (HIPLAB), Vanderbilt University. Grigorios received a Diploma from the University of Crete, in 2005, and a Ph.D. degree from Cardiff University, in 2009, both in Computer Science. Grigorios' research interests are in privacy-preserving data mining and biomedical informatics. His papers have appeared in prestigious conferences and journals, including the *Proceedings of the National Academy of Sciences*, and received three best paper awards. Grigorios serves in the program committee of major conferences and as a reviewer for top journals. He is a professional member of the ACM.

Aris Gkoulalas-Divanis received the B.S. degree from the University of Ioannina (2003), the M.S. from the University of Minnesota (2005) and the Ph.D. from the University of Thessaly (2009), all in Computer Science. His Ph.D. dissertation was awarded the Certificate of Recognition and Honorable Mention in the 2009 ACM SIGKDD Dissertation Award. From March 2009 until February 2010, he was appointed as a Postdoctoral Research Fellow in the Department of Biomedical Informatics, Vanderbilt University, working on medical data privacy. On March 2010, he joined IBM Research-Zurich, in the capacity of a Research Staff Member. Since March 2012, he is working in the Smarter Cities Technology Center of IBM Research-Ireland, conducting research on data privacy. Aris is a regular reviewer for several prestigious journals and serves in the program committee of major conferences. He is a professional member of ACM, IEEE and SIAM, and an at-large member of UPE and Sigma-Xi.

Jianhua Shao received his B.Sc. degree from Shanghai University of Science and Technology, and his Ph.D. from the University of Ulster, all in Computer Science. He is currently a senior lecturer in the School of Computer Science & Informatics at Cardiff University. Previously, he worked as a research associate and a lecturer at the University of Ulster. His recent research interests include privacy protection (protecting private and identity information contained in datasets), and quality of service data analysis (establishing service quality in a service-oriented computing environment through the analysis of historical QoS data). He serves as a reviewer for international journals and on program committees of international conferences.