ρ-uncertainty: Inference-Proof Transaction Anonymization*

Jianneng Cao[#] Panagiotis Karras[#] Chedy Raïssi[§] Kian-Lee Tan[#]

[#]National University of Singapore
{caojianneng, karras, tankl}@comp.nus.edu.sg

[§]INRIA, Nancy Grand-Est, France
chedy.raissi@inria.fr

ABSTRACT

The publication of transaction data, such as market basket data, medical records, and query logs, serves the public benefit. Mining such data allows for the derivation of association rules that connect certain items to others with measurable confidence. Still, this type of data analysis poses a privacy threat; an adversary having partial information on a person's behavior may *confidently* associate that person to an item deemed to be *sensitive*. Ideally, an *anonymization* of such data should lead to an *inference-proof* version that prevents the association of individuals to sensitive items, while otherwise allowing for truthful associations to be derived. Original approaches to this problem were based on value *perturbation*, damaging data integrity. Recently, value *generalization* has been proposed as an alternative; still, approaches based on it have assumed *either* that all items are equally sensitive, *or* that some are sensitive and can be known to an adversary *only* by association, while others are non-sensitive and can be known directly. Yet in reality there *is* a distinction between sensitive and non-sensitive items, but an adversary may possess information on *any* of them. Most critically, no antecedent method aims at a clear *inference-proof* privacy guarantee. In this paper, we propose ρ-uncertainty, the *first*, to our knowledge, privacy concept that *inherently* safeguards against sensitive associations *without* constraining the nature of an adversary's knowledge and *without* falsifying data. The problem of achieving ρ-uncertainty with low information loss is challenging because it is *natural*. A trivial solution is to suppress all sensitive items. We develop more sophisticated schemes. In a broad experimental study, we show that the problem is solved non-trivially by a technique that combines generalization and suppression, which also achieves favorable results compared to a baseline perturbation-based scheme.

1. INTRODUCTION

We consider a data set $\mathcal{D}$ of *set-valued* or *transaction* data. Each entry in $\mathcal{D}$ is a set of items (*itemset*) $\{i_1, \ldots, i_n\}$ - purchased goods, query terms, or individual preferences, chosen from a universe $\mathcal{I}$.

Such data are instrumental in data mining applications, e.g., association rule mining [2, 3], query expansion [7], and predicting user behavior [1]; still, their publication poses privacy threats. A particular threat is posed by an adversary who has *partial* knowledge χ about a certain person's transaction t, and may use it, along with the published data, to derive additional knowledge about the contents of t, thus identifying previously unknown *sensitive* items in t. For example, a publicly released record of a bookstore's transactions reveals that people who bought a book titled "Romeo and Juliet" in the last month also bought a book titled "Free Alaska". Alice meets Bob reading "Romeo and Juliet", and learns that he bought it during the last month. Thus, she makes an inference that compromises Bob's privacy concerning his political persuasions.

Past research has examined the problem of transforming transaction data in a way that renders them proof against privacy threats. However, it never formulated the problem in a *natural* way that corresponds to a real-world setting. Realistically, we have to prevent against the determination of a *sensitive* item in a person's transaction, while preserving the utility of the data for mining other associations. Still, works providing the state of the art in the area [12, 30, 23, 14] fail to formulate the problem in this natural manner.

In particular, [12, 30] assume an a priori distinction between sensitive (private) and non-sensitive (public) items, postulating that *all* private items are *always* unknown to adversaries. Thus, sensitive items are disengaged from the transaction item set. However, in practice an attacker may possess *partial* knowledge of *some* of the sensitive items in a transaction. The methods of [12, 30] do not prevent such attackers from gaining extra sensitive knowledge.

On the other hand, [23, 14] move in the opposite direction, overcorrecting the drawback of [12, 30]. They consider all items in the universe of discourse to be equally sensitive. Thus, the distinction between sensitive and non-sensitive information is lost, even though it forms the main motivation for studying the problem in the first place. Besides, the distinction between what is sensitive and what is not also depends on individual preferences [29], which are not considered by [23, 14] either. After all, [23, 14] are inspired from the k-anonymity model for microdata anonymization [19], so they share its conceptual shortcomings: they do *not* effectively protect against the disclosure of a sensitive item. For example, assume that k shoppers have bought a dairy product, a detergent, some chocolate, and a pregnancy test-kit. A valid anonymization of shopping transaction data by the methods of [23, 14] could create an "anonymized" group out of these k transactions; this grouping still allows an adversary to infer that a neighbor who bought milk, dishwasher, and Belgian chocolate is considering being pregnant.

In this paper, we propose *ρ-uncertainty*, a novel, intuitive, and realistic model for transaction anonymization. To our knowledge, this is the *first* model that is *tailored* against the inference of sen-

*Work supported by Singapore's MOE AcRF grants T12-0702-P02 and T1 251RES0807.

Proceedings of the VLDB Endowment, Vol. 3, No. 1

sitive information by an adversary, does *not* impose artificial *constraints* on the nature of that adversary's knowledge, and neither *falsifies* the data nor *separates* sensitive items from the transactions they belong to. A ρ-*uncertain* transaction set $\mathcal{D}$ *does not allow* an attacker knowing *any* subset χ of a transaction $t \in \mathcal{D}$ to infer a *sensitive* item $\alpha \in t$ with confidence higher than ρ. Thus, ρ-uncertainty ensures that the confidence of any *sensitive association rule* (SAR) is at most ρ. Despite its semantic simplicity, the problem posed by ρ-uncertainty is particularly challenging compared to those posed by other anonymization models. We have to disassociate *all* sensitive items from *any* piece of information that they may be associated with, including both non-sensitive and other sensitive items. A trivial solution is to suppress *all* sensitive items. We look for alternatives that sacrifice *less* information.

Our contributions in relation to prior work [12, 23, 30, 14] are summarized as follows. We introduce ρ-uncertainty, a comprehensible, natural model that *inherently* protects against sensitive item inference. Contrary to [23, 14], we distinguish between public (non-sensitive) and private (sensitive) items; still, in a departure from [12, 30], we treat them as belonging to the same universe, and allow that an adversary may know *some* of the private items in a transaction. A private item already known to an adversary can function as a (quasi-)identifier for the association with another, unknown sensitive item. We define the problem in natural terms like no previous work does. We avoid reasoning that emanates from k-anonymity, wherein transactions are grouped together without offering protection against sensitive item disclosure per se. Last, while the most recent previous methods are based *either* on generalization [23, 14] *or* suppression [30], we employ *both*. We first develop a scheme that attempts to *suppress* a minimal amount of items; we use this as a building-block of a *mixed* scheme that *selectively* suppresses and *generalizes* items. We do *not* consider perturbation, so that no fake or distorted items are involved and *no false* associations are derived; we do not *disengage* sensitive items from the transactions they belong to, as in [12]. Our experiments show that the problem is solved non-trivially by our mixed approach.

2. RELATED WORK

A series of works has addressed the question of mining association rules [2] in a manner that *preserves privacy*. However, such works employ anonymization techniques based on *perturbation* [8, 21, 10, 18, 24, 4], thus damaging data *integrity* (i.e., use distorted and fake values) and generating *false* inferences [27], while they do *not* inherently disallow *sensitive* inferences. On the other hand, anonymization based on *global generalization and suppression* preserves the correctness of data and only compromises their *minuteness*; anonymized data are *less specific*, but *not false*.

To the best of our knowledge, no prior work applies generalization to prevent the mining of sensitive associations from transaction itemsets. Some recent works [30, 12, 23, 25, 14] address related problems, but do *not* arrive at a proper application of generalization for the *explicit* protection of sensitive information.

To our knowledge, [12] is the first work to anonymize transaction data without perturbation. Still, [12] assumes that adversaries can *only* know of non-sensitive items. For each group of transactions, [12] publishes the *exact* public items together with a summary of the frequencies of sensitive items in the group, so that it is not clear to which transaction therein a sensitive item belongs, as in [28]. Unfortunately, this method also impedes non-privacy-threatening mining involving sensitive items. Besides, its transparency renders it more vulnerable. For example, assume Timothy knows that (i) corn flakes were *only* bought together with Noam Chomsky's "Interventions", *and* that (ii) Theresa bought face lotion and chocolate. A transaction set published as in [12] may indicate that face lotion and chocolate were bought together in a *single* transaction, which also includes corn flakes. Therefore, there exists an association of face lotion and chocolate to "Interventions". Timothy did not know about this association in advance, but he can now confidently infer that Theresa bought that classified[1] book. A method that effectively hides all sensitive associations would protect the privacy of Theresa's transaction by publishing it less transparently.

The model of (h,k,p)-coherence [30] requires that any combination of p non-sensitive items is contained in (none or) *at least* k transactions, with *at most* $h\%$ thereof sharing the same sensitive item. Still, [30] applies *only* suppression; it assumes an *a priori* distinction between sensitive and non-sensitive items, like [12]; and postulates that an adversary can know *at most* p (non-sensitive) items in a transaction. Yet an adversary may know of more than p items in a transaction, including some sensitive item(s).

Two recent works [23, 14] employ generalization as a data transformation tool; [23] imposes constraints on an adversary's knowledge, but [14] does not. Still, these works do not distinguish between sensitive and non-sensitive information. They assume that all items are equally likely to be sensitive; this assumption does not conform to the real world. Even if we accepted this assumption as true, [23, 14] are ineffective in the task they set out to accomplish. Their declared objective is to prevent the inference of *potentially sensitive* information, assuming that the "sensitivity potential" is uniformly distributed among items. This objective is *not achieved*. What they do achieve is to hide a vulnerable transaction in a crowd of k others. As [17] has shown, such crowds do *not* effectively prevent sensitive inferences: if a group of k transactions already contain *identical* sensitive content, then [23, 14] can *still* group them together *without* concealing this content [17].

Recently, [25] suggested a relational privacy model that distinguishes between sensitive and non-sensitive information only at the *value* level. The extensibility of this model to transaction data is not considered. Besides, [25] assumes, as [30, 12] do, that *only* non-sensitive information is *observable* by an adversary, and that generalizing a sensitive value to a non-sensitive hierarchy level conceals its sensitivity. Yet such a generalization *reveals* that sensitivity is hidden behind it. For example, the act of generalizing `AIDS` to `virus` suggests that a sensitive value exists behind the generalized one. An akin argument is made by [26] in another context.

3. MODELING

This section presents our privacy concept and information loss metric, which form the two sides of the tradeoff we have to resolve.

3.1 Privacy Concept

Let $\mathcal{D}$ be a set of transaction or set-valued data. Every entry $t \in \mathcal{D}$ is a subset of items drawn from a universe $\mathcal{I}$ that is divided in two subsets: a set of sensitive items $\mathcal{I}_S$, and a set of non-sensitive[2] ones $\mathcal{I}_N$, with $\mathcal{I} = \mathcal{I}_S \cup \mathcal{I}_N$ and $\mathcal{I}_S \cap \mathcal{I}_N \neq \emptyset$. We assume that an attacker may know a subset[3] of items χ of a transaction t, $\chi \subset t$, which may include non-sensitive (in $\mathcal{I}_N$) as well as sensitive items (in $\mathcal{I}_S$). Our objective is to prevent such an attacker from inferring, with high certainty, that a transaction t containing $\chi \subset t$ also contains a sensitive item $\alpha \notin \chi$. Such an inference corresponds to

[1]Currently banned from the Guantánamo Bay prison camp library.
[2]In case of individual preferences about what is sensitive, as in [29], we combine all such preferences to a global list of sensitive items, and treat items that are non-sensitive for all users separately.
[3]An attacker may conceivably know *all* items in a transaction t; in this case the problem we address has no object.

mining [2] association rule $\chi \rightarrow \alpha$. In such an association rule, the left side χ is the *antecedent*, and the right side α the *consequent*. A rule whose consequent involves *at least one* sensitive item $\alpha \in \mathcal{I}_S$ is a *sensitive association rule* (SAR). We need to bring the data in a form that prevents the mining of high-confidence SARs.

TID	Transaction Items
1	$a_1 b_1 b_2 \alpha \gamma$
2	$a_1 a_2 b_2$
3	$a_2 b_2$
4	$a_2 \gamma$
5	$a_1 b_2 \alpha \gamma$

Table 1: A transactional dataset

For example, Table 1 shows five transactions, in which items a_1, a_2, b_1, and b_2 are non-sensitive, while α and γ are sensitive. Given this table, if Alice knows that Bob has bought b_1, she can infer that he also bought a_1, b_2, α, and γ. Furthermore, if Alice already knows that Bob has bought the private item α, she can infer that he also bought a formerly unknown sensitive item, γ. In both cases, Alice *gains* knowledge that compromises Bob's privacy.

To prevent such inferences, we would like the data to be rendered in a form that keeps the *confidence* of each SAR lower than a threshold ρ. Thus, we arrive at the following definition.

DEFINITION 3.1. *A transaction data set $\mathcal{D}$, drawing items from $\mathcal{I}$, is said to satisfy ρ-***uncertainty***, if and only if, for any transaction $t \in \mathcal{D}$, any subset of items $\chi \subset t$, $\chi \subset \mathcal{I}$, and any sensitive item $\alpha \notin \chi$, $\alpha \in \mathcal{I}_S$, the confidence of the Sensitive Association Rule (SAR) $\chi \rightarrow \alpha$ is less than a value $\rho > 0$.*

The privacy concept in Definition 3.1 *does not* limit the provenance of the items that make up χ to $\mathcal{I}_N$. *Sensitive* items from $\mathcal{I}_S$ may also form part of adversaries' prior knowledge. It poses no bound on an association rule's *support* either. *All* sensitive associations of high confidence need to be blocked, even if they involve a single transaction; on the other hand, innocuous appearances of sensitive items that imply *no* high-confidence associations are allowed. For example, assume that, out of 1000 transactions containing milk, one also contains hemorrhoid cream. By ρ-uncertainty, this transaction can be published, since no confident SAR is established thereby, hence nobody's privacy is compromised. On the other hand, assume that the SAR "those who buy blue cheese also buy strawberry-flavored condoms" can be mined from a few transactions. Then the data should be transformed so that this rule shall not be minable. These observations come in contrast to the view of privacy in perturbation-based models [4]. We solve the natural problem of rendering a given transaction data set ρ-uncertain:

PROBLEM 1. *Given a transaction data set $\mathcal{D}$, drawing items from $\mathcal{I}$, including a subset of sensitive items $\mathcal{I}_S \subset \mathcal{I}$,* transform $\mathcal{D}$ *to an anonymized form $\mathcal{D}'$ that satisfies ρ-uncertainty, maintaining the integrity of the data and as much of their utility as possible.*

This problem is *inherently* adopted to the privacy threat transaction data are exposed to, and *one of* the data mining tasks they are *likely* to be used for. [23, 14] also *assume* inference by association to be the risk the data are subject to, yet *do not* explicitly design their models for it. Instead, they adopt precepts of k-anonymization, which divides the data into homogeneous groups. [12, 30] have an inference threat in mind, yet also divide transactions into groups, and assume adversaries only have access to items in $\mathcal{I}_N$, as if the data were relational data with one or more unknown sensitive attributes. We discard such precepts of relational anonymization. Table 2 gathers together our notations.

The *support*, $sup(\chi)$, of a given a subset of items χ is the number of transactions $t \in \mathcal{D}$ such that $\chi \subset t$ [2]. We represent the union of two item sets (*itemsets*) or singleton items, χ_1 and χ_2, as $\chi_1 \chi_2$. The *confidence* of a SAR $\chi \rightarrow \alpha$ is $conf\,(\chi \rightarrow \alpha) = \frac{sup(\chi\alpha)}{sup(\chi)}$. To attain ρ-uncertainty, we need to ensure that the confidence of each SAR is lower than ρ. The following lemma shows that we only need to consider association rules with a singleton consequent.

LEMMA 3.1. *If an association rule $\chi \rightarrow \alpha$, where α is a single item, has confidence $conf\,(\chi \rightarrow \alpha) < \rho$, then any rule $\chi \rightarrow Z$ with $\alpha \in Z$ also has a confidence less than ρ.*

PROOF. Every transaction that contains χZ also contains $\chi\alpha$. Then $sup(\chi Z) \leq sup(\chi\alpha)$, hence

$$conf\,(\chi \rightarrow Z) = \frac{sup(\chi Z)}{sup(\chi)} \leq \frac{sup(\chi\alpha)}{sup(\chi)} = conf\,(\chi \rightarrow \alpha) < \rho$$

Thus, the confidence of rule $\chi \rightarrow Z$ is less than ρ. □

Notation	Meaning
$\mathcal{I}$	The domain of items
$\mathcal{I}_S$	The set of sensitive items in I
$\mathcal{I}_N$	The non-sensitive items in I; $\mathcal{I}_S \cup \mathcal{I}_N = \mathcal{I}$
$\mathcal{H}$	The hierarchy of non-sensitive items in $\mathcal{I}_N$
$\mathcal{D}$	A set of transactions
$\mathcal{D}'$	The anonymized form of $\mathcal{D}$
$sup(a)$	The number of transactions in D that contain item a

Table 2: Notations and their meanings

3.2 Information Loss Metric

A utility metric should indicate the expected performance in a task. We adapt the information loss metric used in [15, 23, 14]. This metric measures the fitness of the anonymized data for a variety of data mining tasks, including classification, analytical processing, and rule mining. We emphasize that the *nature* of this metric reflects an established consensus that antecedent research on anonymization using generalization and suppression has arrived at [12, 11, 30, 23, 14], which we follow. It would also be desirable to compare the association rules mined from anonymized data to those mined from the original data. Still, there is no straightforward way of doing so. The anonymized data can be used to mine *generalized* association rules [22]. Such rules are *correct*; they can be mined from the original data too; there is *no error* in them. After all, *mining generalized association rules* is a valid and self-contained data mining application in itself [22]. These rules are characterized by *less specificity*, as they view the data from a *higher* (but *not* less interesting) point, blurring details. The degree to which a generalized association rule correctly reflects more particular associations depends on the nature of the underlying data. Still, generalized rules can offer *bounds* on the confidence of the more particular rules they may represent. An exploration of this question would open a separate and worthwhile research question, following the footsteps of [22]. Here it is safe to claim that the *more* generalization or suppression we perform, the *less specific* rules we will derive. Our metric measures this *indistinctness* in the anonymized data.

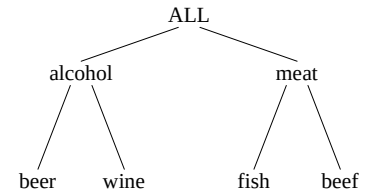


Figure 1: A hierarchy of non-sensitive items

As we discuss in Appendix B, we employ a generalization hierarchy for items in $\mathcal{I}_N$. A leaf node therein represents a non-sensitive item. An internal node is a generalized value of its descendants; the root stands for the generalized value of *all* items. Figure 1 shows

an example. Items `beer`, `wine`, `fish`, and `beef` are non-sensitive. The former two can be generalized to `alcohol`, and the latter two to `meat`. `alcohol` and `meat` can be further generalized to *ALL*.

Let $\mathcal{H}$ be a hierarchy of non-sensitive items, and n a node in it. We define the information loss of n as follows:

$$\mathcal{IL}_n = \frac{|\mathsf{leaves}(n)|}{|\mathcal{I}_N|}$$

where $\mathsf{leaves}(n)$ is the set of leaves under the subtree rooted at n in $\mathcal{H}$. If n is a leaf, then $\mathsf{leaves}(n) = \emptyset$. For the hierarchy in Figure 1, the information loss of node `alcohol` is $\frac{|\mathsf{leaves}(\texttt{alcohol})|}{4} = \frac{1}{2}$.

Given an item a, the information loss it involves is defined as:

$$\mathsf{infoLoss}(a) = \begin{cases} \mathcal{IL}_n, & \text{if } a \text{ is } \textit{generalized} \text{ to node } n \in \mathcal{H} \\ 1, & \text{if } a \text{ is } \textit{suppressed} \end{cases}$$

For example, if `beer` is generalized to `alcohol`, it incurs information loss $\mathsf{infoLoss}(\texttt{beer}) = \mathcal{IL}_{\texttt{alcohol}} = \frac{1}{2}$. As we will discuss in Sections 4 and 5, we opt for *global* suppression and generalization. That is, a suppressed item is deleted from *all* transactions containing it, and a generalized one is substituted by the same hierarchy node in all transactions where it appears. These choices ensure that *no false inferences* are drawn from the anonymized data.

For any item a, let $sup(a)$ be the number of transactions in a transaction data set $\mathcal{D}$ that contain a, *before* anonymization. Then the *average information loss* of an *anonymized* form $\mathcal{D}'$ of $\mathcal{D}$ is:

$$\mathsf{avgLoss}(\mathcal{D}') = \frac{\sum_{a \in \mathcal{I}} sup(a) \cdot \mathsf{infoLoss}(a)}{\sum_{a \in \mathcal{I}} sup(a)}$$

EXAMPLE 3.1. *Assume there are three transactions as follows:*

$\{\texttt{wine}, \texttt{fish}, \texttt{pregnancy test}, \texttt{viagra}\}$
$\{\texttt{wine}, \texttt{beef}\}$
$\{\texttt{beer}\}$

Assume a hierarchy (for non-sensitive items) as in Figure 1. If we generalize `fish` *and* `beef` *to* `meat` *and suppress* `viagra`*, then* $\mathsf{avgLoss}(\mathcal{D}') = \frac{1/2+1/2+1}{7} = \frac{2}{7}$.

4. SUPPRESSION METHOD

We now examine how value *suppression* can be applied to control the confidence of sensitive association rules (SARs). We start out by discussing what *type* of suppression is appropriate here.

Given a transactional dataset $\mathcal{D}$ and an item a participating in its transactions, a can be *partially* suppressed, i.e., deleted from only a subset of transactions in $\mathcal{D}$ that contain it. However, such partial suppression may have undesired side-effects, such as the generation of *false* association rules [24]. As [27] has shown, an attempt to prevent such effects limits our capacity to hide all sensitive rules. To avoid such repercussions, we opt for *global suppression*: a suppressed item is deleted from *all* transactions in $\mathcal{D}$ that contain it. Global suppression does *not* generate false rules, and does *not* perturb the confidence of rules other than those we intend to conceal.

Suppression can be applied on both sensitive and non-sensitive items. This flexibility comes in contrast to the usual assumptions about anonymizing microdata, where there is exactly one sensitive attribute whose values are not supposed to be suppressed [17]. Here, it may be advisable to suppress one sensitive item in order to decrease one's confidence for inferring the existence of another.

Our goal is to suppress a set of items $\mathcal{S} \subset \mathcal{I}$ so that the confidence of all minable SARs is kept lower than ρ, while $\mathsf{avgLoss}(\mathcal{D})$ is minimized. This is a *combinatorial optimization* problem. We could try all subsets $\mathcal{S} \subset \mathcal{I}$, check which ones yield a ρ-uncertain data set $\mathcal{D}'$ when suppressed, and select the one of them that achieves the lowest information loss; that would be the *optimal* solution. However, checking all subsets of $\mathcal{I}$ yields $O\big(2^{|\mathcal{I}|}\big)$ complexity. Given the hardness of the problem, we direct our efforts towards an effective *heuristic*. We opt for an *iterative greedy* heuristic. In a nutshell, it conceals SARs in iterations of increasing antecedent cardinality. In the i^{th} iteration, it considers all derivable SARs with antecedents of exactly i items, computes their confidences, and suppresses items so as to blanket those that violate ρ-uncertainty. Suppressions conducted at lower i save us from some confidence computations at higher i; we gain efficiency at the price of optimality. The process terminates after checking the longest minable SARs.

Let $\mathcal{SR}_i$ be the set of SARs that violate ρ-uncertainty, hence *have to* be concealed, in the i^{th} iteration. Given an item b, either sensitive or non-sensitive, let $C(b, \mathcal{SR}_i)$ be the number of rules in $\mathcal{SR}_i$ that contain it (either in their antecedents or in their consequents), and $sup(b)$ the number of transactions in $\mathcal{D}$ that contain it. We define the *payoff ratio* of item b with respect to $\mathcal{SR}_i$ as $\mathsf{payoff}(i, b) = \frac{C(b, \mathcal{SR}_i)}{sup(b)}$. This ratio expresses the amount of SARs concealed *per unit* of lost information when we suppress b, hence quantifies the suppression's *payoff*. Higher payoff is preferable.

Our heuristic first computes the payoff ratios of all items involved therein. Then, it progressively suppresses the *next* item of highest payoff ratio. After suppressing an item b, it deletes from $\mathcal{SR}_i$ all SARs that contain it. This process terminates when $\mathcal{SR}_i$ becomes empty; then we move to $\mathcal{SR}_{i+1}$. The pseudo-code of this SuppressControl algorithm is found in Section A.1.

Item	Transaction Cover
a_1	$1, 2, 5$
a_2	$2, 3, 4$
α	$1, 5$
b_1	1
b_2	$1, 2, 3, 5$
γ	$1, 4, 5$

(a) Vertical format of Table 1

Itemset	Cover Intersection	Support
a_1, a_2	2	1
a_1, γ	$1, 5$	2
a_1, b_2	$1, 2, 5$	3
a_1, b_2, γ	$1, 5$	2

(b) Support computation

Figure 2: Support computation by cover intersection

To calculate the confidence of SAR $\chi \rightarrow \alpha$, we have to compute the support of itemsets χ and $\chi\alpha$. We represent our transaction data set in *vertical format*, where each item is followed by its *transaction cover*, i.e., the list of transactions containing it [20, 32]. This format allows us to compute the support of an itemset χ by *intersecting* the transaction covers of any two subsets $\tau, \psi \subseteq \chi$, such that $\tau \cup \psi = \chi$. Figure 2(a) illustrates the vertical format of Table 1.

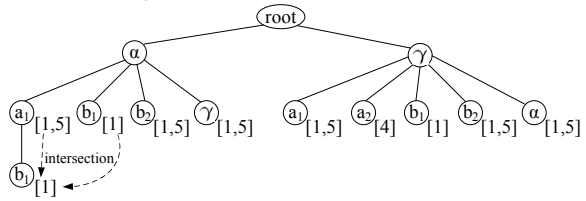


Figure 3: First nodes of a Sensitive Rules Trie (SRT)

To facilitate SAR generation and antecedents support counting, we employ two tries [5, 6]. The *Sensitive Rules Trie* (SRT) has nodes associated to SARs, conceptualized as k-itemsets, including the sensitive item in an SAR's consequent and $k-1$ items in its antecedent. The root corresponds to an empty itemset. Nodes attached to the root represent an SAR's sensitive consequent. Each subsequent node stores the last item in the antecedent of the SAR it stands for. A full SAR is reconstructed by following a path through the SRT. A node's *siblings* represent SARs that differ only at one antecedent item. A *join* of sibling nodes generates an SAR of antecedent larger by one item. Figure 3 shows a trie for all SARs derived from Table 1 with antecedent size 1, and one with antecedent size 2, namely $a_1 b_1 \rightarrow \alpha$; the latter is produced by joining sibling nodes a_1 (representing $a_1 \rightarrow \alpha$) and b_1 (representing $b_1 \rightarrow \alpha$), and

stored by appending a node of item b_1 as a child of node a_1; its transaction cover is an *intersection* of those of a_1 and b_1.

Similarly, the *Antecedents Trie* (AT) is used to generate candidate itemsets that serve as SAR antecedents. Now each node represents an antecedent itemset, and is labeled by the last item in the itemset it represents. A full itemset is reconstructed by following a path through the AT. Nodes at lower levels represent itemsets of higher cardinality. As with the SRT, larger antecedent itemsets are generated from existing ones by joining sibling nodes. Each node stores the transaction cover of the itemset it stands for. Thus, again, an intersection operation suffices to calculate the cover of a larger itemset and its support. Figure 4 illustrates an example of AT node generation and support computation for the data set in Table 1. The full AT is too large to show here, as it has 37 nodes over 5 levels.

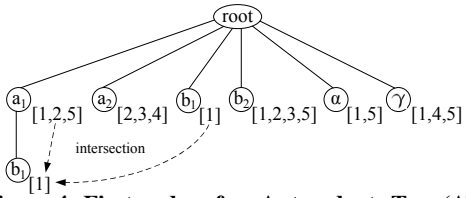


Figure 4: First nodes of an Antecedents Tree (AT)

Our algorithm uses these structures to identify SARs whose confidence exceeds ρ, expanding them by one level per iteration. To facilitate this operation, we use a structure akin to the Frequent Pattern tree [13], the *Payoff Tree* (PT). A path in a PT represents an SAR to be concealed. Now the sensitive consequent is represented by the leaf node where the path ends, and its ancestors stand for the antecedent. For each item, a header table records the number of SARs and transactions in which this item is involved. This information aids the calculation of payoff ratios. The worst-case complexity of these structures is $O\big(|D|m2^{m-1}\big)$, where m is the maximum transaction length in D. In practice, it is pruned by suppressions performed before high itemset cardinalities are reached, as in [31]. Section A.1 illustrates our method with an example.

5. GENERALIZATION METHOD

In this section we examine how value *generalization* can be applied to our problem. Like suppression, generalization comes in two variants: *local* and *global*. In *local* generalization [14], different instances of the same item may be generalized to different levels in a generalization hierarchy $\mathcal{H}$. However, as the case is with partial suppression, local generalization has unwelcome side-effects. It allows for *inconsistencies* in our generalization choices, from which privacy-threatening inferences can be made by an adversary, as in [26]. Besides, the extra precision it allows for does not translate to more precise association rule mining. When mining association rules from a locally generalized data set, no precise confidences can be calculated for rules involving locally generalized items, but only upper and lower bounds thereof; thereby, the estimated confidence of certain association may be inadvertently *increasing*, leading to the mining of *false positive* rules. On the other hand, *global* generalization [23] maps *all* instances of an item b, as well as *all* items of the *same* subtree of hierarchy $\mathcal{H}$, to the *same* level in $\mathcal{H}$. For example, using the hierarchy in Figure 1, if an instance of `beer` is generalized to `alcohol`, then all other instances of `beer`, as well as all instances of `wine`, are *consistently* generalized in the same manner. Even if *only* an instance of `beer` were generalized to `alcohol`, one would not be able to mine exact inferences involving `beer` and `wine`; *only* inferences at the generalization level of `alcohol` would be unambiguous. This state of affairs is the *same* with global generalization; local generalization does not offer a significant mining advantage. In conclusion, we opt for *global generalization*.

A further question is whether we should apply generalization on both sensitive and non-sensitive items, as we do with suppression in Section 4. We opt for generalizing *only* non-sensitive items. Appendix B presents our detailed argument. Its core is that a sensitive item should *never* be generalized at an intermediate hierarchy level; if it were, then that choice itself would reveal sensitive information. Again, this argument is reminiscent of the *minimality attack* argument made by [26] in the context of microdata anonymization.

In effect, we develop an algorithm that anonymizes $\mathcal{D}$ by applying *global generalization* over the hierarchy $\mathcal{H}$ of non-sensitive items in $\mathcal{I}_N$, along with *selective global suppression* of some items. In a nutshell, our algorithm performs a *top-down particularization* process. It starts out assuming that *all* items in $\mathcal{D}$ are generalized at the top level of $\mathcal{H}$, represented by the root node. This state of affairs satisfies ρ-uncertainty, but provides no information at all. In order to recover some information, we relax, or *particularize*, the generalization by moving along branches of $\mathcal{H}$ in a *greedy* manner. A particularization is a move from (i.e., *split* of) a node n in $\mathcal{H}$ to its children, which allows *all* items in $\mathcal{D}$ that belong to leaves of n to assume the more *particular* values in those children. For example, in the hierarchy of Figure 1, *splitting* node *ALL* to its children nodes `alcohol` and `meat` allows *all* instances of `fish` and `beef` (`beer` and `wine`) in our data to assume the value `meat` (`alcohol`) instead of *ALL*. A particularization move aims to *gain* as much information as possible. However, a particularization per se may uncover item values in a way that violates ρ-uncertainty. In a such a case, our algorithm examines whether the violation of ρ-uncertainty can be avoided by *suppressing* some items, and calculates the *net* information gain of the *combined* particularization-and-suppression move. At each step, the algorithm opts for the move of *maximum net information gain*. We now present the tools this algorithm employs.

5.1 The Particularization Tree

We represent particularizations (i.e., from a bottom-up view, generalizations) using a *particularization tree* $\mathcal{T}$. A node n of $\mathcal{T}$ maps a node of $\mathcal{H}$, which represents a set of items $v(n)$, as follows:

$$v(n) = \begin{cases} \{n\}, & \text{if } n \text{ is a leaf node in } \mathcal{H} \\ leaves(n), & \text{otherwise} \end{cases}$$

The *leaves* of $\mathcal{T}$ (which may be *internal* nodes of $\mathcal{H}$) show the *level* of generalization to which *all* instances of items beneath them in $\mathcal{H}$ are generalized. A leaf node $\ell \in \mathcal{T}$ that maps a leaf node in $\mathcal{H}$ indicates that the related item is not generalized at all. A leaf node ℓ in $\mathcal{T}$ that maps an *internal* node in $\mathcal{H}$ prescribes that *any* item $a \in v(\ell)$ that appears in *any* transaction $t \in \mathcal{D}$ is generalized to ℓ. This *generalization at* ℓ, applied on $\mathcal{D}$, yields information loss:

$$\mathcal{GIL}(\ell) = \textstyle\sum_{a \in v(\ell)} sup(a) \cdot \mathcal{IL}_\ell \qquad (1)$$

Coming back to the example using the hierarchy $\mathcal{H}$ in Figure 1, a particularization tree $\mathcal{T}$ built over $\mathcal{H}$ initially consists of the root node ALL. The *split* of this node, provided it incurs positive information gain, adds `alcohol` and `meat` as *leaves* in $\mathcal{T}$. Next, assume that splitting `alcohol` brings about positive net information gain, but splitting `meat` does not. Then `alcohol` is split into `beer` and `wine`. Eventually, $\mathcal{T}$ contains three leaf nodes, `beer`, `wine`, and `meat`, denoting the adopted generalization rules (GRs). According to these GRs, all instances of `fish` and `beef` in $\mathcal{D}$ are mapped to `meat`. More details on this algorithm are discussed in Section 5.3.

5.2 Potential Net Information Gain

As we discussed, our algorithm aims to achieve as much *net information gain* as it can at each step. When faced with more than

one option, it greedily carries out the move that offers the greatest information gain benefit. Information gain is *negative information loss*. When particularization is combined with suppression, we calculate their *net effect* on data utility. *Negative* information gain (i.e., positive information loss) can also arise from a move.

To faciliate the greedy choice of one among the leaf nodes of $\mathcal{T}$ that are candidate for splitting, we define the *Potential Net Information Gain* (PNIG) a node's split will effect, *if* carried out:

$$\mathsf{pnig}(x) = \left(\mathcal{GIL}(x) - \textstyle\sum_{C \in \mathsf{children}(x)} \mathcal{GIL}(C)\right) - \mathsf{loss} \quad (2)$$

This function calculates the information that would be gained by *splitting* a leaf node x in $\mathcal{T}$ as the *difference* of the information loss by generalization involving x *before* the split (first term) from that *after* the split (second term), *minus* the *loss* of information caused by item suppressions required to maintain ρ-uncertainty (third term). A need for such suppressions can arise from SARs violating ρ-uncertainty, whose antecedents contain the *newly particularized* children of x. In order to find items that *would have* to be suppressed to maintain ρ-uncertainty, we use a variant of suppressControl that duly *selects* items for suppression, but does not suppress them; it only returns the third term in Equation 2.

Still, in order to properly calculate the loss incurred by suppressions of *generalized* items, we have to enhance the definition of *payoff* ratio. The information *loss* incurred by suppressing a generalized item g is *not* simply equal to the number of transactions where it is found; it is only the *difference* between the loss of a wholesale suppression and that *already* incurred by generalization to g. Suppressing g amounts to suppressing all items in $\mathcal{D}$ already mapped to g. The number of such items is $sup(g) = \sum_{c \in v(g)} sup(c)$. The information lost by this mapping is $sup(g) \cdot \mathcal{IL}_g$, hence the information surviving after it is $sup(g) \cdot (1 - \mathcal{IL}_g)$. Therefore, we redefine the *payoff* ratio as $\frac{C(g,\mathcal{SR})}{sup(g)\cdot(1-\mathcal{IL}_g)}$, where $\mathcal{SR}$ is the set of SARs we try to conceal and $C(g, \mathcal{SR})$ the number of rules in $\mathcal{SR}$ that contain g. Our suppression mechanism again selects the item of highest *payoff* ratio. The pseudo-code for our PNIG calculation procedure infoGain is found in Section A.2.

5.3 The Algorithm

We now give more details on our top-down global generalization algorithm, TDControl. As we discussed, it constructs a *particularization tree* $\mathcal{T}$ in a greedy manner, aiming to achieve the highest information gain possible in each move. It terminates when no move of positive net information gain is possible any more; i.e., when any possible particularization move would violate ρ-uncertainty, and the suppressions required to safeguard it would cause *more* loss of information than the gain the particularization offers in the first place. In such circumstances, Equation 2 would acquire negative values for all candidate nodes. These leaf nodes in the final particularization tree $\mathcal{T}$ define the final GRs applied to anonymize $\mathcal{D}$.

In more detail, $\mathcal{T}$ is initialized as the root ALL of $\mathcal{H}$, which holds all items in $\mathcal{I}_N$. Then, at each iteration, we calculate the PNIG of all leaf nodes of $\mathcal{T}$, and split the node ℓ of highest PNIG. This *split* adds to $\mathcal{T}$ the children of ℓ, as found in $\mathcal{H}$, which become leaf nodes in $\mathcal{T}$ themselves, and hence candidates for splitting in the next iteration. Section A.2 provides the pseudo-code of this TDControl algorithm and a full example of its operation.

6. EXPERIMENTAL EVALUATION

Now we evaluate the effectiveness of our algorithms. We clarify that SuppressControl is a stand-alone scheme that achieves ρ-uncertainty by suppression. TDControl obtains ρ-uncertainty by both generalization and suppression, while it uses SuppressControl as a component. Our schemes are not comparable to previous works in transaction anonymization, because *such works do not provide ρ-uncertainty guarantees*. Thus, we compare against a baseline ρ-uncertainty-attaining method, denoted as Simple, which suppresses all sensitive items and preserves others without generalization. Qualitatively, this suppression incurs a distinctive damage, as it disables all mining related to sensitive items. The desired outcome is to enable such mining to the extent allowed by ρ-uncertainty. Our methods are designed with this goal in mind.

We assess performance in terms of runtime and the information loss incurred to achieve ρ-uncertainty. Algorithms were implemented in C++ and ran on an Intel Core 2 Duo 2.33GHz machine with 4GB RAM running Windows XP. We used benchmark data that are popular in related research [23, 14], described in Appendix C. As no distinction of sensitive from non-sensitive items is provided in the data, we randomly select sensitive items, tuning their *percentage*; its default value is 40%. Given the set of non-sensitive items $\mathcal{I}_N$, a *fan-out* parameter determines the shape of hierarchy $\mathcal{H}$. Our default fan-out is 4. The problem posed by ρ-uncertainty becomes computationally harder as the maximum transaction length increases. Even so, our best method can process 99% of the largest test dataset. We also conduct detailed experiments on transactions of at most 5 items. Last, our default value of ρ is 0.5.

Varying dataset. We first run all algorithms with default parameters on all datasets. Figure 5 illustrates the results. *WV1* stands for *BMS-WebView-1*, *WV2* for *BMS-WebView-2*, and *POS* for *BMS-POS*. SuppressControl performs poorly in both information quality and time. Checking the anonymized datasets it produces, we find that SuppressControl suppresses almost all the sensitive items, and it also suppresses a few non-sensitive items. For instance, SuppressControl suppresses 90% of sensitive items in *WV1*, as well as 60% of non-sensitive ones, but TDControl suppresses 80% percent of sensitive items only. Thus, the output of SuppressControl is worse than Simple, even as it tries to minimize information loss at each iteration. SuppressControl performs worse than TDControl in terms of efficiency too, as the former needs to consider *all* SARs in the *whole* dataset at each iteration, while the latter only processes SARs restricted to a subset of items.

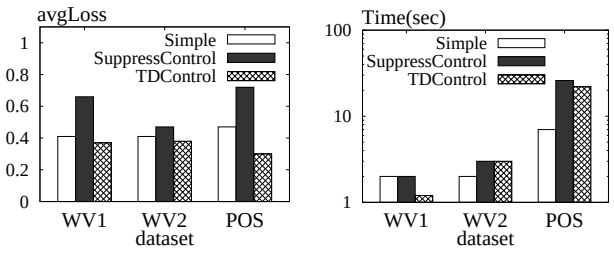


Figure 5: Varying datasets

In the following experiments we use the *BMS-POS* dataset. Results with the other two datasets exhibit similar trends. Furthermore, *BMS-POS* is much larger than the others. Thus, we consider *BMS-POS* as a better ground for testing the strengths and weaknesses of the algorithms we compare.

Processing transactions of increasing length. In our previous experiments we have set the maximum length at 5. This setting allows us to illustrate the practical feasibility of our strong privacy objective with real-world data of reasonable transaction length, for which our techniques are most suitable. This choice has processed $306,983$ out of $515,597$ transactions in *BMS-POS*, thus excluded 40.5% of the transactions, which have length more than 5. It is interesting to examine to what extent our ρ-uncertainty algorithms can operate on data of *unbounded* transaction length. To this end,

we have experimented as follows. We sorted the transaction in *BMS-POS* by length in ascending order, and processed *prefixes* of this ordered dataset, using the default settings of $\rho = 0.5$, fan-out 4, and the percentage of sensitive items at 40%.

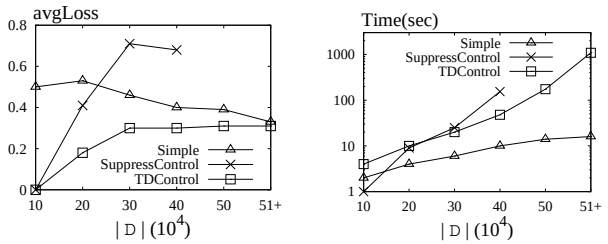


Figure 6: Varying the size of dataset

Figure 6 shows the results as a function of the size of the dataset prefix involved. Given the sorted order, larger size involves not only *more* transactions, but also increasingly *longer* transactions. The maximum transaction lengths for the six prefixes in the experiment are 2, 3, 5, 9, 23, and 77, respectively. SuppressControl can process up to $400,000$ tuples before running out of memory. However, our method of choice, TDControl, can satisfactorily process up to $515,500$ tuples (i.e., 99% percent of data in *BMS-POS*), involving all transactions of length no more than 77, i.e., allowing an adversary's knowledge of up to 76 items in a transaction, and maintains its information loss advantage with respect to Simple. This result confirms that ρ-uncertainty is a feasible privacy objective even for data sets containing very long transactions and a high percentage of sensitive items. Previous works dealing with sensitive item disclosure do *not* experiment with such settings, even while they define the problem in a *less general* manner. [30] assumes as a default setting that an adversary may know only up to $p = 4$, only non-sensitive, items in a transaction; [12] assumes as a default setting that there are only 10 sensitive items in the data and also postulates that an adversary can only know about non-sensitive items. Simple is the fastest algorithm in this experiment, but its information loss does not present a clear pattern, as sensitive items are randomly selected. The amount of SARs to conceal grows as a function of the maximum transaction length, which increases as the prefix is enlarged. Thus, it is increasingly harder to reduce the confidence of SARs lower than ρ. Therefore, the information loss of TDControl and SuppressControl grows with prefix size. Last, the runtime of all three algorithms grows with the prefix size as expected.

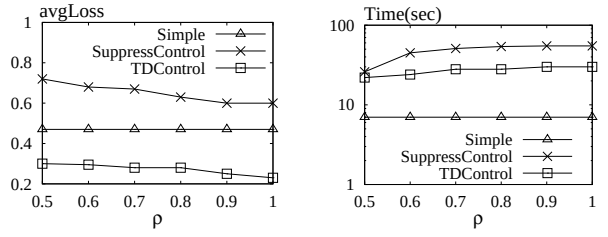


Figure 7: Varying the value of ρ

Tuning ρ. Next, we use the prefix with $300,000$ tuples as the experiment dataset, denoted as POS_30k. We randomly mark 40% percent of the items in POS_30k as sensitive, denoted as $\mathcal{I}_S$_30k_40, and vary ρ. Figure 7 shows our results. As expected, the information loss of both TDControl and SuppressControl *decreases* as a function of ρ. This is due to the fact that higher ρ requires less SARs to be concealed. However, as ρ grows, the *amount* of suppressed sensitive items decreases, therefore more SARs need to be checked at each step. In effect, runtime *grows* with ρ. As in previous experiments, TDControl achieves the best information quality, while its time efficiency stands between those of Simple and SuppressControl. The performance of Simple is not affected by ρ.

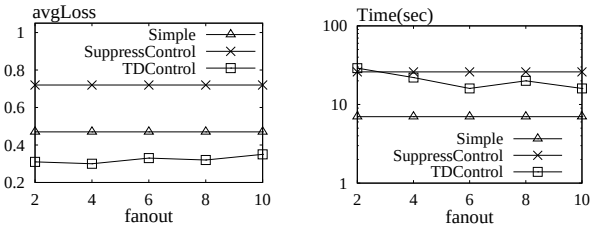


Figure 8: Varying the fanout of the hierarchy

Tuning fan-out. Figure 8 presents our results as a function of the fan-out value in the hierarchy $\mathcal{H}$, using dataset POS_30k and the set of sensitive items $\mathcal{I}_S$_30k_40. This parameter only affects the performance of TDControl, yet the information quality achieved by TDControl does not change uniformly as a function of fanout. Still, overall the curve suggests that a smaller value of fanout tends to preserve more information. After all, when the fan-out is higher, then more nodes are generalized to a single node in one step, hence the likelihood of over-generalization is increased. On the other hand, for lower fan-out the hierarchy $\mathcal{H}$ is deeper, hence the number of possible intermediate generalization levels for each leaf node is increased, and it becomes more likely that an item is generalized to a level where more information is preserved. For similar reasons, runtime tends to drop for higher fan-out.

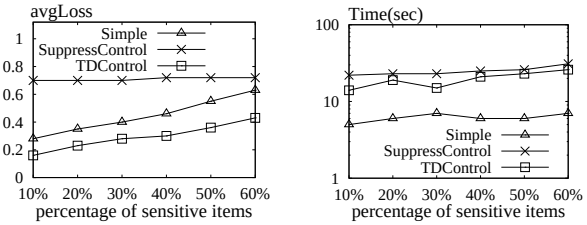


Figure 9: Varying the percentage of sensitive items

Tuning percentage of sensitive items. We now tune the percentage p of items selected as sensitive with POS_30k. Six sets of sensitive items are generated. For consistency, we ensure that the set of sensitive items for a smaller percentage is a *subset* of that for larger. Figure 9 shows the results. The number of minable SARs that have to be concealed grows with p. In effect, the time required to calculate the confidences of such SARs increases, hence information loss and runtime grow. TDControl is again the clear winner in terms of information preserved with a satisfactory runtime.

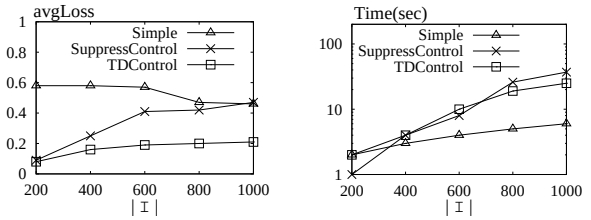
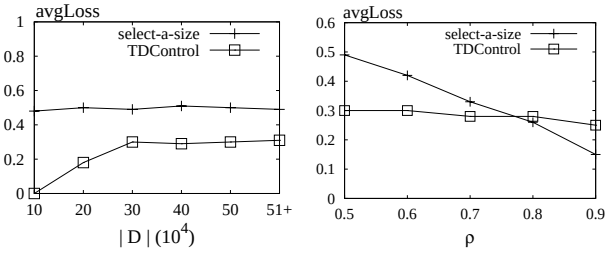


Figure 10: Varying the number of items

Tuning $|\mathcal{I}|$. Using POS_30k, we prepared datasets of different numbers of distinct items. Let $\mathcal{I}$_30k be the item domain of POS_30k. We randomly selected subsets of items $\mathcal{I}'$ from $\mathcal{I}$_30k, and generated a dataset by removing from POS_30k all items not appearing in $\mathcal{I}'$. For each generated dataset we randomly select its set of sensitive items, ensuring that smaller sets of selected items (sensitive items) are subsets of larger ones. Figure 10 shows our results. Runtime grows with $|\mathcal{I}'|$, as growing $\mathcal{I}'$ implies more data and more SARs to handle. The information loss curve of Simple is unstable, as the number of instances of each suppressed item varies. However, since the amount of SARs to conceal grows as a function

of $|\mathcal{I}'|$, the information loss of TDControl and SuppressControl grows with it as well. TDControl clearly outperforms the other two methods in information quality in this experiment too.

A comparison to ρ_1-to-ρ_2 privacy. We introduce a new privacy model, ρ-uncertainty, and *deterministic*, generalization-based algorithms to achieve it. Still, the same privacy guarantee can in principle be provided, *in expectation*, by probabilistic perturbation-based models. While the work needed in this area goes beyond the scope of this paper, we can safely say that a starting point for a perturbation-based scheme that achieves ρ-uncertainty in expectation would be the mechanism of ρ_1-to-ρ_2 privacy [9, 4], which aims to hide the exact contents of transactions. Therefore, we conduct a basic comparison to this technique in terms of information loss. We adopt a basic version of ρ_1-to-ρ_2 privacy, using a simple form of its select-a-size randomization operator; more sophisticated variants may achieve higher privacy (which we do not measure) at the price of more information loss (which we measure). This operator [9] features two parameters: a randomization level, $0 \leq \beta \leq 1$ that determines the amount of new items added in a transaction, and a transaction subset size selection probability distribution, $\{p[j]\}_{j=0}^{|t|}$ [10, 9]. For each transaction t, an integer $0 \leq j \leq |t|$ is selected with probability $p[j]$, and j randomly chosen items from t are kept in its perturbed version, t'. Then, each item $a \notin t$ is added to t' with probability β. We assume a "default" probability distribution, where $p[j] = \binom{|t|}{j} \cdot \beta^{|t|-j}(1-\beta)^j$, as each item in t is kept with probability $1-\beta$. We measure the information loss incurred by each *noisy* item in $t \setminus t'$ as $\mathcal{IL}_n$, where n is the lowest common ancestor of all items in $(t' \setminus t) \cup (t \setminus t')$, i.e. items added in, and missing from, t', in a hierarchy $\mathcal{H}'$ containing all items; when sensitive items are removed from $\mathcal{H}'$, we get the hierarchy used by TDControl.



(a) Varying dataset size (b) Varying ρ

Figure 11: A comparison to ρ_1-to-ρ_2 privacy

We first set $\rho = \beta = 0.5$ and study the effect of dataset size with prefixes of *BMS-POS* sorted by length. Figure 11(a) shows the average information loss results. The loss of select-a-size roughly depends on the amount of items kept, while that of TDControl is less for smaller data size and converges robustly to a value lower than that of select-a-size as the size grows. This result indicates that our deterministic generalization framework can effectively contain the amount of information sacrificed to achieve ρ-uncertainty in a way competitive to randomization. Then we process transactions of size at most 5, and study the effect of ρ, setting $\beta = 1 - \rho$. This setting forms a baseline for a perturbation scheme aiming at ρ-uncertainty. Figure 11(b) shows our results. Higher ρ allows for more items to be preserved, hence better accuracy. TDControl achieves higher information accuracy for practically significant values of ρ.

7. CONCLUSIONS

This paper introduced ρ-uncertainty, the *first*, to our knowledge, anonymization model that naturally safeguards against mining sensitive inferences, *without* imposing constraints on an adversary's prior knowledge, and *without* falsifying data. Despite, or because of, its naturalness, the model poses a non-trivial problem. Rendering a transaction data set ρ-uncertain in a way better than suppressing all sensitive items is a challenging task. We apply sophisticated techniques on this problem, with an algorithm that couples global generalization over non-sensitive items with selective global suppression of some items, *without* relying on precepts of relational anonymization. We show that this algorithm consistently leads to anonymizations of higher quality than the naive approach, and favorable results compared to a baseline perturbation-based scheme. Appendix D discusses questions peripheral to our work.

8. REFERENCES

[1] E. Adar, D. S. Weld, B. N. Bershad, and S. S. Gribble. Why we search: visualizing and predicting user behavior. In *WWW*, 2007.

[2] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD*, 1993.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, 1994.

[4] S. Agrawal, J. R. Haritsa, and B. A. Prakash. FRAPP: A framework for high-accuracy privacy-preserving mining. *Data Min. Knowl. Discov.*, 18(1):101–139, 2009.

[5] A. Amir, R. Feldman, and R. Kashi. A new and versatile method for association generation. *Information Systems*, 22(6-7):333–347, 1997.

[6] R. J. Bayardo, Jr. Efficiently mining long patterns from databases. In *SIGMOD*, 1998.

[7] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Probabilistic query expansion using query logs. In *WWW*, 2002.

[8] E. Dasseni, V. S. Verykios, A. K. Elmagarmid, and E. Bertino. Hiding association rules by using confidence and support. In *IHW*, 2001.

[9] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *PODS*, 2003.

[10] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *KDD*, 2002.

[11] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis. A framework for efficient data anonymization under privacy and accuracy constraints. *ACM TODS*, 34(2):1–47, 2009.

[12] G. Ghinita, Y. Tao, and P. Kalnis. On the anonymization of sparse high-dimensional data. In *ICDE*, 2008.

[13] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *SIGMOD*, 2000.

[14] Y. He and J. F. Naughton. Anonymization of set-valued data via top-down, local generalization. *PVLDB*, 2(1):934–945, 2009.

[15] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *KDD*, 2002.

[16] D. Kifer. Attacks on privacy and deFinetti's theorem. In *SIGMOD*, 2009.

[17] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. *ACM TKDD*, 1(1):3, 2007.

[18] S. J. Rizvi and J. R. Haritsa. Maintaining data privacy in association rule mining. In *VLDB*, 2002.

[19] P. Samarati. Protecting respondents' identities in microdata release. *IEEE TKDE*, 13(6):1010–1027, 2001.

[20] A. Savasere, E. Omiecinski, and S. B. Navathe. An efficient algorithm for mining association rules in large databases. In *VLDB*, 1995.

[21] Y. Saygin, V. S. Verykios, and C. Clifton. Using unknowns to prevent discovery of association rules. *SIGMOD Rec.*, 30(4):45–54, 2001.

[22] R. Srikant and R. Agrawal. Mining generalized association rules. In *VLDB*, 1995.

[23] M. Terrovitis, N. Mamoulis, and P. Kalnis. Privacy-preserving anonymization of set-valued data. *PVLDB*, 1(1):115–125, 2008.

[24] V. S. Verykios, A. K. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni. Association rule hiding. *IEEE TKDE*, 16(4):434–447, 2004.

[25] K. Wang, Y. Xu, A. W. C. Fu, and R. C. W. Wong. FF-anonymity: When quasi-identifiers are missing. In *ICDE*, 2009.

[26] R. C.-W. Wong, A. W.-C. Fu, K. Wang, and J. Pei. Minimality attack in privacy preserving data publishing. In *VLDB*, 2007.

[27] Y.-H. Wu, C.-M. Chiang, and A. L. Chen. Hiding sensitive association rules with limited side effects. *IEEE TKDE*, 19(1):29–42, 2007.

[28] X. Xiao and Y. Tao. Anatomy: simple and effective privacy preservation. In *VLDB*, 2006.

[29] X. Xiao and Y. Tao. Personalized privacy preservation. In *SIGMOD*, 2006.

[30] Y. Xu, K. Wang, A. W.-C. Fu, and P. S. Yu. Anonymizing transaction databases for publication. In *KDD*, 2008.

[31] G. Yang. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *KDD*, 2004.

[32] M. J. Zaki. Scalable algorithms for association mining. *IEEE TKDE*, 12(3):372–390, 2000.

[33] Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In *KDD*, 2001.

APPENDIX

A. PSEUDO-CODES AND EXAMPLES

A.1 Suppression Method

Algorithm 1 presents our suppression algorithm SuppressControl, operating on a transactional dataset $\mathcal{D}$.

Algorithm 1: SuppressControl($\mathcal{D}$)

```
1  Initialize i = 1;
2  Initialize loss = 0;
3  while true do
4      SR'_i = {χ → α | α ∈ I_S, |χ| = i};
5      if SR'_i = ∅ then
6          break;
7      SR_i = {r | r ∈ SR'_i, conf(r) ≥ ρ} ;
8      while SR_i ≠ ∅ do
9          Find the item b of maximum payoff(i, b);
10         Suppress b;
11         Delete all rules in SR_i containing b;
12         loss = loss + sup(b);
13     i = i + 1;
14 Return loss;
```

The algorithm operates iteratively with increasing i. In the i^{th} round, it finds the set $\mathcal{SR}'_i$ of all SARs whose antecedents contain exactly i items. It then extracts from $\mathcal{SR}'_i$ the set of those SARs that have confidence at least ρ (Lines 4-7). Then it goes about concealing the SARs in $\mathcal{SR}_i$ by suppressing the highest-payoff item until $\mathcal{SR}_i$ becomes empty (Lines 8-12). For that purpose, we maintain a priority queue of item payoffs. For each item, we keep a bag of rules that contain it, and for each rule the items in it. After each item suppression, we update accordingly the payoff ratios of items affected by the rule's deletion, and the queue. The cost of Step 9 is logarithmic on the number of items. The algorithm terminates when there is no SAR with an antecedent of i items, hence neither one for larger values of i (Lines 5-6).

An example of our suppression strategy is shown below.

EXAMPLE A.1. *We wish to render the transaction data set in Table 1 in a ρ-uncertain form with $\rho = 0.7$. Assume that α and γ are sensitive items, while all the remaining ones are non-sensitive. The tree in Figure 12 depicts the generated SRT for all the SARs processed in the first round of the algorithm, for $i = 1$. Each path in the tree starts out with the sensitive consequent of the rule it forms, and continues with the nodes of all antecedent items.*

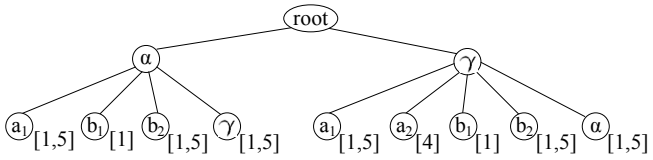


Figure 12: Sensitive rules with $|\chi| = 1$

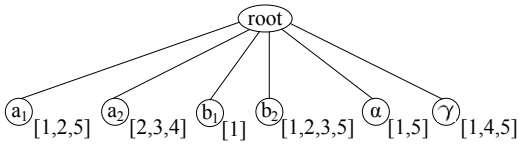


Figure 13: Antecedents with $|\chi| = 1$

In order to calculate the confidences of the SARs in the SRT of Figure 12, we also maintain an AT containing all possible SAR antecedents, as in Figure 13. Using the information contained in these two trees, we calculate the confidence of all SARs for $|\chi| = 1$, and find the set of SARs whose confidence violates the threshold $\rho = 0.7$, namely $\mathcal{SR}_1 = \{\alpha \to \gamma, b_1 \to \gamma, b_1 \to \alpha\}$. We need to conceal the rules in $\mathcal{SR}_1$.

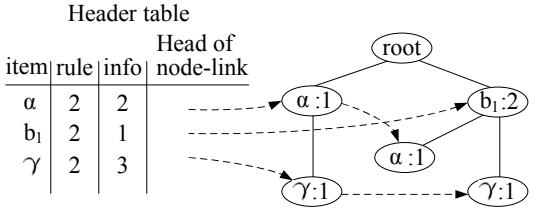


Figure 14: Payoff tree for $|\chi| = 1$

In order to carry out this concealment operation, we first construct the payoff tree and the associated header table for these SARs, as illustrated in Figure 14. The header table records the number of SARs and transactions in which each item is involved. The payoff tree paths *represent the three SARs to be concealed. The values beside each node label in this payoff tree denote the number of paths (i.e., SARs) in the tree that pass through that node. For example, the node label $b_1 : 2$ indicates that there are two paths passing through the node of item b_1, hence two SARs involving that item. Moreover, all nodes with the same item-label are linked by dashed lines in the figure. Following these links, we can calculate the aggregate number of SARs involving a certain item in the header table. Thus, using the transaction number information in the header table, we calculate, for example, that the* payoff *ratio for item b_1 is $\frac{2}{1}$. This payoff ratio is the highest among the three items in our payoff tree. Thus, our algorithm first suppresses b_1.*

After suppressing b_1, the SAR counters for the remaining two items are updated accordingly. In particular, the SAR counter of α becomes 1, since the SAR $b_1 \to \alpha$ has been now concealed and only $\alpha \to \gamma$ remains. Likewise, the SAR counter of γ also assumes the value 1, as $b_1 \to \gamma$ has been suppressed and only $\alpha \to \gamma$ remains. In fact, a single *SAR involves* both *of these two remaining items. The highest payoff ratio is now achieved by α, hence we suppress it. Thus, we have now concealed all rules in $\mathcal{SR}_1$.*

After we have suppressed b_1 and α, the only SAR with $i = 2$ items in its antecedent that we can derive in the second round is $b_2a_1 \to \gamma$ with transaction cover $[1, 2, 5]$ and $conf(b_2a_1 \to \gamma) = \frac{sup(b_2a_1\gamma)}{sup(b_2a_1)} = \frac{2}{3} < \rho$. Since this confidence does not violate our threshold ρ, no SAR needs to be hidden. Eventually, in the third round, we find there is no SAR with 3 items in its antecedent, hence the suppression operation ends. Table 3 shows the resulting published ρ-uncertain dataset.

TID	Transactions
1	$a_1b_2\gamma$
2	$a_1a_2b_2$
3	a_2b_2
4	$a_2\gamma$
5	$a_1b_2\gamma$

Table 3: The published form of Table 1 after suppression

A.2 Generalization Method

Function 2 shows our PNIG calculation procedure infoGain. It calculates the effect of splitting node x, receiving three input parameters: $\mathcal{L}$, the set of leaf nodes in the particularization tree; $x \in \mathcal{L}$, the leaf node to be split; and $\mathcal{H}$, the hierarchy over non-sensitive items. It is assumed that the anonymized dataset generated by applying on $\mathcal{D}$ the generalization rules defined by $\mathcal{L}$ does not give rise to SARs violating the ρ threshold; by induction, the result of the algorithm will satisfy ρ-uncertainty too. The information

that *would* be gained *if* we split x is calculated as the difference of the information loss before the split from that after the split (Lines 3-5). The splitting of x would transform $\mathcal{L}$ to $\mathcal{L}_x$, in which x is replaced by its children (Lines 6-9).

Function infoGain ($\mathcal{L}$, x, $\mathcal{H}$)

```
1  if x is a leaf node in H then
2      Return -1;
3  oldcost = Σ_{a∈v(x)} sup(a) × IL_x;
4  newcost = Σ_{C∈children(x)} ( Σ_{d∈v(C)} sup(d) × IL_C );
5  gain = oldcost − newcost;
6  Copy L to a local variable L_x;
7  Remove x from L_x;
8  foreach child C of x in H do
9      Add C to L_x;
10 D_x = Particularize(trans(x), L_x);
11 loss = suppressTest(D^x);
12 s_x = set of items selected by suppressTest;
13 pnig(x) = gain − loss;
14 Return pnig(x);
```

Still, the particularization carried out to gain information may allow the derivation of SARs that violate the confidence threshold ρ. In order to preserve ρ-uncertainty, we should perform some *suppression* in combination with particularization. A violation of ρ-uncertainty caused by splitting x can only arise from sensitive association rules whose antecedents contain the newly particularized children of x. Thus, in order to hide SARs that violate the threshold ρ after splitting x, we only need to process transactions that contain children of x. To facilitate this task, we need to know the set $\mathsf{trans}(x)$ of transactions containing items in $v(x)$; we discuss how we do this in the overall algorithm later. Then, all transactions in $\mathsf{trans}(x)$ are particularized (i.e., the children of x in them are mapped to more specific generalization levels) according to the (now relaxed) generalization rules defined by $\mathcal{L}_x$ (Line 10). Then we find those items that *would need* to be suppressed in the ensuing data set $\mathcal{D}_x$ so as to maintain ρ-uncertainty. For this purpose we use Function suppressTest, a variant of suppressControl that selects items for suppression, but does not actually suppress them (Line 11). We store the set of items s_x that would have to be suppressed (Line 12), so that we can retrieve them if we actually opt for splitting x and incurring the calculated PNIG. Besides, this suppression, if opted for by the algorithm, would cause information loss, which is deducted from gain (Line 13).

Algorithm 3: TDControl($\mathcal{D}, \mathcal{H}$)

```
1  Let SD be a set of transactions, initialized to be empty;
2  foreach t ∈ D do
3      Let st be the set of sensitive items in t;
4      if st is non-empty then
5          Insert st into SD;
6  SuppressControl(SD);
7  L = set of leaf nodes in T, initialized to empty;
8  Add ALL to L;
9  trans(ALL) = D;
10 pnig(ALL) = infoGain(L, ALL, H);
11 while true do
12     m = node of maximum pnig in L;
13     if pnig(m) ≤ 0 then
14         break;
15     Suppress from D all items in s_m;
16     Remove m from L;
17     foreach child C of m in H do
18         Add C to L;
19         trans(C) = collect(trans(m), C);
20     foreach node x ∈ L do
21         pnig(x) = infoGain(L, x, H);
22 Generalize(D, L);
```

Next, Algorithm 3 illustrates the TDControl scheme. At the onset, this algorithm examines whether there are any SARs whose antecedents consist only of sensitive items. Generalization of non-sensitive items will have no effect on the confidence of such rules. Therefore, in a pre-processing step, we use our suppression technique to control the confidence of such rules. We identify such rules by collecting transactions containing at least one sensitive item (Lines 1-5). We then apply our suppression technique to enforce the ρ thershold on any SAR composed of sensitive items only. The call to SuppressControl in Line 6 pertains to the suppression of such rules only, since it receives a special data set $\mathcal{SD}$ containing only the sensitive-item subset of each transaction t in $\mathcal{D}$.

In its main part, the algorithm progressively constructs the particularization tree $\mathcal{T}$, while keeping track of the set of its leaf nodes $\mathcal{L}$. The leaf nodes in $\mathcal{L}$ are stored in a priority queue that provides the node m that offers the maximum potential information gain (Line 12). The while loop of Lines 11-21 recursively expands $\mathcal{T}$, greedily splitting the maximum-gain node m at each iteration, until no positive gain can be harnessed any more. As we discussed, when a maximum-gain node m is split, we may have to suppress some items in order to preserve ρ-uncertainty; these items are provided by the set s_m calculated by infoGain (Line 15). Moreover, the children of a split node replace it in $\mathcal{L}$ (Lines 16-19).

As we have seen before, to facilitate the PNIG computation for a certain node x, we need to know the set $\mathsf{trans}(x)$ of all transactions that contain items in $v(x)$. Thus, for each child C of a split node m, we collect the set $\mathsf{trans}(C)$ from the already known set $\mathsf{trans}(m)$, where $\mathsf{trans}(C) \subseteq \mathsf{trans}(m)$ (Line 19). Besides, after $\mathcal{L}$ is updated, the PNIGs of *all* nodes in $\mathcal{L}$ are re-calculated by the infoGain process (Lines 20-21). This recalculation is needed because the suppressions that may be needed after splitting *any* node in $\mathcal{L}$ mutually depend on the positions of all other nodes in $\mathcal{L}$ (Lines 10-11 in infoGain). Thus, they all have to be re-calculated. Eventually, after the particularization process is finished, the generalization specified by $\mathcal{L}$ is applied on $\mathcal{D}$ (Line 22). By construction, the anonymized data set $\mathcal{D}'$ so generated does not contain SARs violating ρ-uncertainty.

The following example illustrates our generalization strategy.

EXAMPLE A.2. *Assume we wish to anonymize the data in Table 1 with $\rho = 0.7$. Items α and γ are sensitive, while all others are non-sensitive. The hierarchy of non-sensitive items $\mathcal{H}$ is given in Figure 15, ignoring the dotted lines in it: the children of A are a_1 and a_2 only, and those of B are b_1 and b_2 only. Then $\mathcal{SD} = \{\alpha\gamma, \gamma, \alpha\gamma\}$ (Lines 1-5 in algorithm 3). Then* suppressControl *will suppress item α (Line 6). $\mathcal{L}$ and $\mathsf{trans}(ALL)$ are initialized as $\mathcal{L} = \{ALL\}$ and $\mathsf{trans}(ALL) = \{1, 2, 3, 4, 5\}$. By the application of the generalization rules in $\mathcal{L}$, $\mathsf{trans}(ALL)$ assumes the form $\{ALL\gamma, ALL, ALL, ALL\gamma, ALL\gamma\}$. The only sensitive rule that can be derived is $ALL \rightarrow \gamma$, with a confidence of 0.6, which does not violate ρ-uncertainty. Thus, we proceed to particularization. For the splitting of ALL into A and B, it is $\mathsf{oldcost}_{ALL} = (sup(a_1) + sup(a_2) + sup(b_1) + sup(b_2)) \cdot \mathcal{IL}_{ALL} = 11$ and $\mathsf{newcost}_{ALL} = (sup(a_1)+sup(a_2)) \cdot \mathcal{IL}_A + (sup(b_1)+sup(b_2)) \cdot \mathcal{IL}_B = 5.5$. Thus, $\mathsf{gain}_{ALL} = \mathsf{oldcost}_{ALL} - \mathsf{newcost}_{ALL} = 5.5$. If the split is carried out, then it will be $\mathcal{L}_{ALL} = \{A, B\}$. $\mathcal{D}_{ALL}$, the form of data generated by applying the generalization specified by $\mathcal{L}_{ALL}$, is $\{AB\gamma, AB, AB, A\gamma, AB\gamma\}$. No sensitive rule that violates the threshold ρ can be generated, hence $\mathsf{loss}_{ALL} = 0$ and $\mathsf{pnig}(ALL) = 5.5$. Therefore the split is allowed, and $\mathcal{L} = \mathcal{L}_{ALL} = \{A, B\}$. Now we have two candidates in $\mathcal{L}$ for splitting. $v(B) = \{b_1, b_2\}$, so $\mathsf{trans}(B) = \{1, 2, 3, 5\}$ and $\mathcal{L}_B = \{A, b_1, b_2\}$. $\mathcal{D}_B$, generated by applying the generalization specified by $\mathcal{L}_B$ on $\mathsf{trans}(B)$, is $\{Ab_1b_2\gamma, Ab_2, Ab_2, Ab_2\gamma\}$. $\mathsf{oldcost}_B = (sup(b_1) + sup(b_2)) \cdot \mathcal{IL}_B = (1 + 4) \cdot 1/2 = 2.5$. $\mathsf{newcost}_B = 0$, since b_1 and b_2 incur no generalization on*

$\mathsf{trans}(B)$, *hence* $\mathsf{gain}_B = 2.5$. *However, the sensitive rule* $b_1 \rightarrow \gamma$ *will then have confidence* 1. *Therefore, we need to suppress* b_1, *and the real PNIG is* $\mathsf{pnig}(B) = \mathsf{gain}_B - \mathsf{loss}_B = 2.5 - 1 = 1.5$. *On the other hand, the PNIG for splitting* A *is* 3, *thus we opt for splitting* A *first, and get* $\mathcal{L} = \{a_1, a_2, B\}$. *Finally, we also split* B, *while suppressing* b_1, *hence* $\mathcal{L} = \{a_1, a_2, b_2\}$. *We do not include* b_1 *in* $\mathcal{L}$, *since it has been globally suppressed. Applying the generalization implied by* $\mathcal{L}$ *on Table 1, we eventually get* $\mathcal{D}' = \{a_1 b_2 \gamma, a_1 a_2 b_2, a_2 b_2, a_2 \gamma, a_1 b_2 \gamma\}$. *This result is by coincidence the same as that of Table 3.*

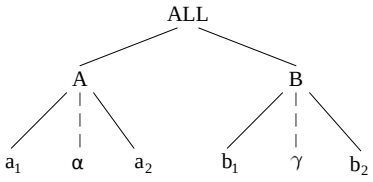


Figure 15: A simplified item hierarchy

B. DELIMITATION OF THE GENERALIZATION DOMAIN

This Section discusses our generalization choices. We assume an adversary who has knowledge of the published data, as well as the generalization hierarchy that we employ. Such knowledge needs to be made public, in order for the published form of the data to retain its usefulness.

We first discuss whether we should adopt a generalization hierarchy that combines sensitive and non-sensitive items (i.e., $\mathcal{I}_N$ and $\mathcal{I}_S$). A straightforward application of global generalization would suggest generalizing all items in $\mathcal{I}$ so as to achieve low confidences for sensitive rules, assuming that a generalization hierarchy defines mixed categories containing sensitive along with non-sensitive items. Still, we should examine the ramifications of such an approach. We start our study with the following lemma.

LEMMA B.1. *Let* $\mathcal{D}$ *be a transactional data set, and* $\mathcal{H}'$ *be a generalization hierarchy over its domain* $\mathcal{I}$, *including both sensitive and non-sensitive items. If an item* β, *either sensitive or non-sensitive, is generalized to one of its ancestors in* $\mathcal{H}'$, Γ, *then, for any itemset* χ, $conf(\chi \rightarrow \Gamma) \geq conf(\chi \rightarrow \beta)$.

PROOF. Let $t \in \mathcal{D}$ be a transaction that contains $\chi\beta \subseteq t$. Assume t is transformed to $\bar{t}$ when β is generalized to Γ. Then $\chi\Gamma \subseteq \bar{t}$. Thus, any transaction containing $\chi\beta$ prior to generalization, contains $\chi\Gamma$ after it. Then $sup(\chi\Gamma) \geq sup(\chi\beta)$, from which it follows that $conf(\chi \rightarrow \Gamma) = \frac{sup(\chi\Gamma)}{sup(\chi)} \geq \frac{sup(\chi\beta)}{sup(\chi)} = conf(\chi \rightarrow \beta)$. □

Lemma B.1 appears superficially similar to Lemma 3.1, but they are quite different. Lemma 3.1, deals with rule $\chi \rightarrow Z$, where Z is a *combination* of items. Thus, for a transaction to contain Z, it should contain *each* of the items combined in Z; this requirement is expressed as a *conjunction*. On the other hand, in Lemma B.1, we deal with rule $\chi \rightarrow \Gamma$, where Γ is a *generalization*. Thus, for a transaction to contain Γ, it should contain *any* of the items generalized to Γ; this requirement is expressed as a *disjunction*.

We proceed to examine the consequences of generalization applied on mixed hierarchies of sensitive and non-sensitive items in $\mathcal{I}$. Let $\mathcal{D}$ be a transaction data set over $\mathcal{I}$ with a mixed generalization hierarchy $\mathcal{H}'$. Assume an item $b \in \mathcal{I}$ and sensitive item $\gamma \in \mathcal{I}_S$, such that the SAR $b \rightarrow \gamma$ has $conf(b \rightarrow \gamma) \geq \rho$. Then, we should decrease the confidence of this rule by generalization. Suppose that we generalize b (and its siblings) to an ancestor Γ in $\mathcal{H}'$, $b \in leaves(\Gamma)$, so that the confidence of the rule $\Gamma \rightarrow \gamma$ becomes less than ρ. This reduction of confidence can be effectively achieved, because now $sup(\Gamma)$ is counted over all items generalized to Γ along with b.

However, an attempt to decrease the confidence of a certain SAR via generalization may inadvertently blur the confidence of *another* SAR. Assume there exists a sensitive item α among the leaves of Γ as well, $\alpha \in leaves(\Gamma)$. Then, after generalizing b to node Γ, any SAR $\chi \rightarrow \alpha$ is generalized to the rule $\chi \rightarrow \Gamma$. According to Lemma B.1, this generalized rule has *higher or equal* confidence in relation to its more specific form.

Nevertheless, a suspicious adversary would assume that we have generalized α to Γ in order to conceal its sensitive nature, and assume the higher confidence of rule $\chi \rightarrow \Gamma$ as an upper bound for the confidence of the hidden rule $\chi \rightarrow \alpha$. However, our intention is to allow interested parties who know the published form of the data and the generalization hierarchy that we employ to derive useful associations, while preventing them from inferring sensitive associations. Employing a hierarchy that mixes sensitive with non-sensitive items does not help our objectives. It complicates the problem without offering an advantage.

EXAMPLE B.1. *Assume the transaction data set of Table 1, with the item hierarchy of Figure 15, and* $\rho = 0.65$. *Items* α *and* γ *are sensitive. SAR* $a_1 \rightarrow \gamma$ *has confidence* $\frac{2}{3}$, *higher than* ρ. *If we generalize* a_1 *to* A, *then* $conf(A \rightarrow \gamma) = \frac{3}{5} < \rho$. *However, this generalization blurs the confidence of another SAR, namely* $b_2 \rightarrow \alpha$; *its confidence before generalization is* $\frac{1}{2} < \rho$, *but after generalization it is inadvertently submerged by rule* $b_2 \rightarrow A$ *with a confidence of* 1.

Still, one could argue that generalizing sensitive items to a hierarchy node deemed non-sensitive provides a handy method of protection. In fact, [25] suggests this approach as an anonymization method. However, this approach defeats its own purpose. Generalizing at a specific non-sensitive level directly reveals that there has been a need to do so, hence sensitivity is unveiled.

For example, assume we employ a hierarchy that generalizes `Flu` and `AIDS` to `Virus`. An adversary who knows we employ this hierarchy immediately assumes a high likelihood of AIDS in case we have adopted such a generalization. Our generalization may in fact be a side-effect of the need to generalize a non-sensitive sibling of the sensitive item. We would prefer this kind of side-effects to be avoided. After all, the problem at hand is to conceal sensitive pieces of information by limiting an attacker's ability to infer them by association. Concealing sensitivity by merely mapping a sensitive value to a non-sensitive ancestor in a generalization hierarchy is only a solution of last resort. As we will discuss, the form in which we employ such a solution of last resort is *complete suppression*, which does not leave any room for ambiguity.

From the preceding discussion we conclude that it is preferable to disengage the generalization hierarchy of non-sensitive items ($\mathcal{I}_N$) from that of sensitive items ($\mathcal{I}_S$). Thus, we employ generalization hierarchy $\mathcal{H}$ consisting of non-sensitive items in $\mathcal{I}_N$ only. Sensitive items in $\mathcal{I}_S$ should be treated separately.

We now discuss whether we should apply separate generalization on $\mathcal{I}_S$ itself. A generalization hierarchy over $\mathcal{I}_S$ can be of two kinds: Either intermediate nodes are considered as sensitive, or as non-sensitive. Still, in both cases, the very generalization of a sensitive value at a higher level does not conceal its sensitivity from an adversary; it would *always* reveal the sensitivity we attempt to hide, defeating our purpose. This state of affairs is reminiscent of the minimality attack problem encountered in microdata anonymization [26].

To effectively prevent this problem, we opt for generalization for $\mathcal{I}_N$ only *combined with* selective global suppression. In this approach, a sensitive item may still appear in an SAR's antecedent, and it may be globally suppressed; but it will never be generalized to an intermediate level. An SAR in the anonymized data set $\mathcal{D}'$ may then only appear in the form $Y \rightarrow \alpha$, where Y may contain intact sensitive items and (possibly) generalized non-sensitive items, and α is a non-generalized sensitive item. For example, assume that a data set $\mathcal{D}$ over the $\mathcal{I}_N$ hierarchy of Figure 1 contains the transaction $\{\texttt{beer}, \texttt{beef}, \texttt{viagra}\}$. If we generalize `beer` to `alcohol`, then a mined SAR can be `alcohol` $\rightarrow$ `viagra`.

C. DESCRIPTION OF DATA

dataset	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{I}\rvert$	$\max \lvert t\rvert$	$\text{avg}\lvert t\rvert$
BMS-POS	515,597	1,657	164	6.5
BMS-WebView-1	59,602	497	267	2.5
BMS-WebView-2	77,512	3,340	161	5.0

Table 4: Dataset characteristics

We have used 3 datasets: *BMS-POS*, *BMS-WebView-1*, and *BMS-WebView-2*. All are introduced in [33] and are common benchmarks in the data mining community. The transactions in *BMS-POS* are several years worth of point-of-sale data, collected from a large electronics retailer. *BMS-WebView-1* and *BMS-WebView-2* are composed of the click-stream data generated in several months from two e-commerce web sites. Table 4 summarizes the characteristics of these data sets, where $\max |t|$ and $\text{avg}|t|$ represent the maximum and average transaction size, respectively.

These data sets do not specify which items in them are sensitive. Thus, we randomly choose a subset of items from the item domain $\mathcal{I}$ as sensitive items $\mathcal{I}_S$. We consider the remaining items to be non-sensitive, and build a hierarchy $\mathcal{H}$ over them. We sort non-sensitive items in the ascending order of their IDs, and assign neighboring items to a common parent. The number of children per internal node is controlled by the *fan-out* of the hierarchy.

D. DISCUSSION

To the best of our knowledge, this paper is the *first* to propose generalization as a data anonymization technique aiming to conceal sensitive association rules. Previous works *either* used generalization in the context of differently defined objectives that do not protect against the disclosure of sensitive information [23, 14], *or* defined their models differently and did not use generalization as a tool either [10, 8, 24, 12, 30, 12, 25].

Still, a discussion of the effects of generalization as an anonymization technique in the context of the problem we examine is due. Figure 16(a) shows 4 transactions, in which α is a sensitive item.

TID	Transactions
1	$a_1\alpha$
2	$a_2\alpha$
3	$a_1a_2\alpha$
4	a_1a_2

(a) A non-monotonic data set

Rule	Confidence
$a_1 \rightarrow \alpha$	$\frac{2}{3}$
$a_2 \rightarrow \alpha$	$\frac{2}{3}$
$a_1a_2 \rightarrow \alpha$	$\frac{1}{2}$

(b) Original Confidences

Figure 16: A non-monotonic dataset and original confidences

The confidences of three derivable SARs are presented in Figure 16(b). Still, if we generalize items a_1 and a_2 to A, then the only derivable sensitive rule $A \rightarrow \alpha$ has a confidence of $\frac{3}{4}$, higher than that of any of the original SARs with α as the consequent.

We conclude that *generalization does not obey the monotonicity property*. That is, generalizing to a higher level in the generalization hierarchy does not necessarily lead to generalized forms of derivable SARs with confidence equal to or lower than the specific SARs they stand for; it may also lead to generalized [22] SARs of *higher* confidences. The confidence of a generalized SAR, like $A \rightarrow \alpha$ in this example, provides to an adversary only an *upper bound* to the true confidence value of the more specific SARs $a_1 \rightarrow \alpha$, $a_2 \rightarrow \alpha$, and $a_1a_2 \rightarrow \alpha$ it represents. Thus, generalization does not allows for *exact* calculation of the confidences of *specific* SARs, but *only* for the derivation of an upper bound therefor. Our algorithm considers these *upper* bounds of confidence when it tries to conceal SARs violating the ρ-uncertainty principle. Thus, it provides *safe* guarantees.

We clarify that our algorithms are meant for application where the number of distinct items per transaction is relatively small, while the number of transactions is large. For example, that is the case for e-commerce store or movie theater transactions. Our algorithm SuppressControl, which forms a component of TDControl, needs to iteratively compute all SARs formed for increasing antecedent size. This process can be efficiently preformed for relative small antecedent sizes [3].

Another question arises from the termination condition of TDControl. This algorithm terminates when further particularization would incur information *loss*, instead of *gain*, due to suppressions required to prevent a breach of ρ-uncertainty. In effect, particularization ends at, and hence *reveals*, the point where a privacy risk was at stake if it were carried forward. This revelation is *reminiscent* of a state of affairs that arises in the problem of microdata anonymization [26]. Throughout this paper, we have made arguments *akin* to the ones made in [26] to justify our choices of global suppression and generalization, and of not generalizing sensitive items. Still, a discussion about the relevance of the *exact* argument made by [26] is also due. We provide such a discussion here.

In the microdata anonymization problem, an adversary has access to an *eponymous* (i.e., not anonymous) external table of *all* personal non-sensitive attribute values. Furthermore, anonymized microdata fully *preserve* the values of a sensitive attribute in the published table, hence the adversary has access to those too. On top of that, it is *assumed* that an adversary knows the privacy goal the employed anonymization algorithm aims at, *as well as* the anonymization technique it uses. Armed with all this information, and observing the generalization choices the anonymization algorithm has made, adversaries can deduce sensitive information. The reasoning they apply in such a *minimality attack* is based on counting the total appearances of each sensitive value in the published table [26].

It is tempting to examine whether such an attack, or, for that matter, a deFinetti-style attack [16], could apply to transaction data anonymization, as they apply in the context, and under the particular assumptions, of *microdata anonymization*. Nevertheless, the transaction anonymization problem does *not* involve the collateral publication of *all* non-sensitive transaction contents in an *eponymous* rendering. Thus, there is nothing analogous to the eponymous external table of microdaa anonymization. Adversaries only have as much information about individual itemsets as they can themselves observe. The equivalent state of affairs in microdata anonymization is to know non-sensitive attribute values of specific persons, not a full table. Furthermore, *not all* sensitive transaction items are preserved in the published data, hence their full frequency distribution cannot be reconstructed from the published data, as the case is in microdata anonymization. In conclusion, the core of the reasoning behind the type of attacks raised in [26, 16] is rendered moot in the case of transaction data anonymization.