Publishing Set-Valued Data Against Realistic Adversaries

Jun-Qiang Liu (刘君强)

School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou 310018, China
E-mail: ljq18@hotmail.com
Received July 11, 2011; revised November 24, 2011.

Abstract Privacy protection in publishing set-valued data is an important problem. However, privacy notions proposed in prior works either assume that the adversary has unbounded knowledge and hence provide over-protection that causes excessive distortion, or ignore the knowledge about the absence of certain items and do not prevent attacks based on such knowledge. To address these issues, we propose a new privacy notion, \((k, \ell)^{m,n}\)-privacy, which prevents both the identity disclosure and the sensitive item disclosure to a realistic privacy adversary who has bounded knowledge about the presence of items and the bounded knowledge about the absence of items. In addition to the new notion, our contribution is an efficient algorithm that finds a near-optimal solution and is applicable for anonymizing real world databases. Extensive experiments on real world databases showed that our algorithm outperforms the state of the art algorithms.

Keywords privacy, security, data mining, data management, set-valued data

1 Introduction

We consider the privacy protection problem in publishing set-valued data. Set-valued data is a collection of transactions, and each transaction consists of an arbitrary number of items. Examples include shopping transactions\(^1\), movie ratings\(^2\), and web query logs\(^3\)\(^4\). On one hand, set-valued data have a wide range of applications in data mining research. For example, customer behavior analysis is made possible by mining the shopping transactions, and movie rental companies have been successful in recommending movies to subscribers based on subscriber preference analysis obtained from the movie rating data. On the other hand, set-valued data may contain significant amount of sensitive information. The release of such data to a third party could breach privacy, as highlighted by a few recent incidents\(^2\)\(^4\), and as explained by the following motivating example.

1.1 Motivations

A supermarket released one week’s transactions shown in Table 1 to a data mining company for marketing strategy analysis to promote the sale of AdultToy, Viagra, and PregnancyTest. These items are profitable, but are also sensitive in that customers are unwilling to let their friends know that their shopping transactions contain such items, while other items, such as Beer, Wine, and Milk, are non-sensitive. Bob in the data mining company is analyzing the data, and he happened to know his colleague, Alice, bought Wine and Yogurt from this supermarket in the last week. Given the data, Bob is 100% sure that Alice’s transaction is \(t_1\) (identity disclosure), and Alice also bought AdultToy (sensitive item disclosure). Bob also knew that his friend Darth bought Beer but no Wine. Thus, Bob is sure that Darth’s transaction is \(t_2\), and Darth also bought AdultToy and Viagra.

Alice’s privacy is breached because of Bob’s knowledge about the presence of Wine and Yogurt in Alice’s transaction. Darth’s privacy is breached because of Bob’s knowledge about the absence of Wine in addition to Bob’s knowledge about the presence of Beer in Darth’s transaction.

<table>
<thead>
<tr>
<th>TID</th>
<th>Transactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t_1)</td>
<td>Wine, Milk, Yogurt, AdultToy</td>
</tr>
<tr>
<td>(t_2)</td>
<td>Beer, Jacket, Pants, AdultToy, Viagra</td>
</tr>
<tr>
<td>(t_3)</td>
<td>Yogurt, Jacket, Hose, Shoe, Viagra</td>
</tr>
<tr>
<td>(t_4)</td>
<td>Milk, Yogurt, Jacket, Geta, PregnancyTest</td>
</tr>
<tr>
<td>(t_5)</td>
<td>Beer, Wine, Milk, Jacket, Pants</td>
</tr>
</tbody>
</table>

It is quite surprising that most prior studies focus on the privacy protection problem in publishing relational data, only a handful of studies tackle the problem in publishing set-valued data\(^5\)\(^8\), even though most of the...
data published are set-valued data. In fact, the problem in publishing set-valued data, such as the one in the motivating example, is more challenging than the problem in publishing relational data. We observed three challenges that are not well addressed by the prior studies.

The first challenge is that assuming an omniscient adversary with unbounded knowledge is unrealistic due to high dimensionality. Set-valued data can be thought of as a relational table with hundreds even thousands of binary attributes. An approach that assumes the privacy adversary has unbounded knowledge about items in the target transaction, i.e., takes all the binary attributes (items) as a quasi-identifier [9], inevitably introduces excessive distortion to the anonymized data because of the curse of dimensionality [10].

For example, [5] took this approach by requiring each transaction identical to at least \( k - 1 \) other transactions. To lessen the negative effect of high dimensionality, [5] reused the multi-dimensional partitioning [11], which however destroys the domain exclusiveness [12], e.g., Beer and Liquor may co-exist in the anonymized data with some Beer being represented by Liquor.

As another example, [6] also took this approach by requiring that every inference about a sensitive item has a certainty no more than a fixed threshold [13], which does not serve the purpose of publication since the anonymized data are used for analyzing trends and patterns regarding sensitive items, and any inference that is beyond the knowledge of adversaries should be retained accurately [8].

The key to defining privacy is to model the adversary’s knowledge. Assuming that the adversary has unbounded knowledge deprives the data publisher of the right to make tradeoff between privacy and utility [14-16].

The second challenge is that the bounded knowledge of a realistic adversary may bear the fact that some items are absent from the target transaction. Such a problem has not been tackled by prior works. In particular, [7-8] handle the privacy attacks based on the knowledge about the presence of no more than \( m \) items, but do not tackle the attacks based on the knowledge about the absence of items. For example, Darth’s privacy breach is not handled by [7-8].

The third challenge is that handling the privacy attacks from a realistic adversary with bounded knowledge as in [7-8] is computationally harder than tackling an omniscient adversary with unbounded knowledge as in [5-6]. In fact, even the heuristic algorithms AA [7] and MM [8] also suffer efficiency and scalability issues as shown in Subsections 6.1 and 6.3.1. Although [7] presented an optimal solution OA, the authors of [7] acknowledged that OA is inapplicable to large, realistic problems.

1.2 Contributions

First, we propose a new notion, \((k,\ell)^{(m,n)}\)-privacy, for privacy protection in publishing set-valued data, which addresses the first two challenges with the prior studies. We consider a realistic adversary who has bounded knowledge [7-8] about his/her target individual, and may know that some items are absent from the target transaction. In particular, our notion models the knowledge of an adversary by a logical expression, that is, a conjunction of \( m+n \) clauses with \( m \) clauses expressing the knowledge about the presence of items and \( n \) clauses expressing the knowledge about the absence of items in the target individual’s transaction, and prevents both the identity disclosure and the sensitive item disclosure to such a realistic adversary. Our notion offers tradeoff between privacy and utility [14-16].

Second, as it is computationally harder to verify the existence of privacy violations with set-valued data than with relational data, a novel hierarchy tagging technique is proposed to represent hierarchical data without exploding transactions, which addresses the scalability issue that is the bottleneck of algorithms.

Third, we propose an efficient and scalable algorithm to enforce \((k,\ell)^{(m,n)}\)-privacy by integrating a progressive anonymization strategy, depth-first search, and the hierarchy tagging technique, which finds a solution that is very near to the optimal.

Fourth, we show that our near-optimal solution has a significant gain in data utility relative to heuristic solutions. We evaluate the data utility in terms of an information loss metric and usefulness in frequent pattern mining. Extensive experiments show that our algorithm outperforms the heuristic generalization algorithm AA [7] and the heuristic suppression algorithm MM [8] in terms of information loss, frequent pattern mining, and efficiency. Our algorithm also outperforms the LG algorithm [5] in frequent pattern mining.

The rest of the paper is organized as follows. Section 2 surveys related work. Section 3 describes the privacy attack model and anonymization technique. Section 4 proposes our \((k,\ell)^{(m,n)}\)-privacy notion and defines the optimal anonymization problem. Section 5 proposes a near-optimal solution to the problem. Section 6 evaluates our work, and Section 7 concludes the paper.

2 Related Work

There are prior studies that consider a realistic adversary and offer strong tradeoffs between privacy and utility in publishing relational data and graph data. [14] introduced \( \epsilon \)-privacy that guards against a realistic
adversary. [15] modeled the adversary’s background knowledge by a conjunction of k implications and proposed (c, k)-safety ensuring the maximum privacy disclosure in the worst case is less than c. [16] introduced (k, ℓ)-groupings for anonymizing bipartite graph data, which retains the underlying graph structure perfectly and masks the mapping between entities and graph nodes. However, in publishing set-valued data, prior studies provide either over-protection by assuming an omniscient adversary with unbounded knowledge,[5-6], or under-protection by ignoring the knowledge about absence of items[7-8]. We provide a strong tradeoff between privacy and utility for publishing set-valued data.

There are also studies in privacy preserving data mining. [17] presented randomization operators to limit privacy breaches. [18] santized association rules by reducing confidences and supports of sensitive rules. [19] considered privacy violations embedded in data mining results and suggested ways eliminating such violations by means of pattern distortion. We focus on publishing set-valued data that retains the semantics of each transaction, which is different from [17-19]. In other words, while [17-19] only consider a specific mining task, our approach has no such limitation, i.e., the anonymized data can be used for various tasks. Moreover, mining our anonymized data always produces true results while [17-19] allow to distort the mining result.

Recently, [20] proposed the differential privacy notion, which has no assumptions about an adversary’s background knowledge, and requires that the participation of an entity in the database should have no significant difference on the query answers. [21] employed differential privacy in publishing a query-click graph derived from a web query log. [22] proposed to release differentially private histogram that is useful for linear distributive queries by exploiting the query workload derived by an interactive differential privacy interface. [23] demonstrated that set-valued data can be efficiently released under differential privacy with guaranteed utility for counting queries. While differential privacy is a promising approach, it primarily aims at an interactive query model and its non-interactive variant is derived based on a query workload. We consider a general approach that does not target a specific class of queries.

3 Privacy Attack Model

A data publisher releases an anonymized version $D'$ of a set-valued database $D = \{t_1, t_2, \ldots, t_{|D|}\}$ that is a collection of transactions. Each transaction $t_i$ is a set of items from the universe $I = \{i_1, i_2, \ldots, i_{|I|}\}$. Among items in $I$, some are sensitive, such as Viagra, and some are non-sensitive, such as Beer. We denote the domain of all sensitive items as $I_S$, and the domain of all non-sensitive items as $I_N$.

An adversary has access to the published, anonymized data, and has acquired the knowledge about the presence of some non-sensitive items in and the knowledge about the absence of certain non-sensitive items from his/her target individual’s transaction. However, the adversary has no knowledge about sensitive items. The adversary tries to infer the target individual’s transaction and sensitive items with high probabilities.

3.1 Anonymization Technique

The data publisher employs generalization technique,[5,7] to generalize non-sensitive items in producing $D'$. Specifically, the full subtree generalization technique is employed, with which $D'$ is defined by a cut on the hierarchy tree of non-sensitive items used by the data publisher for generalization.

**Definition 1** (Generalization, Specialization, and Cut). Given a generalization hierarchy $H_N$ of non-sensitive items (also available to the adversary) and a set-valued database $D$, a cut is a set of items containing exactly one item on every root-to-leaf path on $H_N$. A set of items (a.k.a. itemset), $X$, is a generalization of itemset $Y$ ($Y$ is a specialization of $X$) if for each item $x$ in $X$ there is an item $y$ in $Y$ such that $x$ is $y$ or $x$ is an ancestor of $y$ ($y$ is a descendant of $x$) on $H_N$. Let $\text{Cut}$ be such a cut, the generalized version $D'$ of $D$, denoted as $D' = \text{gen}(D, \text{Cut})$, is derived by replacing each non-sensitive item in $D$ with its hierarchical ancestor in $\text{Cut}$.

For example, Fig.1 is such a hierarchy with cuts, cut_c, cut_s, and cut_w, on it. Table 2 is a generalized version $D' = \text{gen}(D, cut_c)$, where all Beer and Wine are

![Fig.1. Generalization hierarchy $H_N$ of non-sensitive items and cuts on $H_N$.](image)

<table>
<thead>
<tr>
<th>TID</th>
<th>Generalized Transactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>Liquor, Dairy, AdultToy</td>
</tr>
<tr>
<td>$t_2$</td>
<td>Liquor, Jacket, Pants, AdultToy, Viagra</td>
</tr>
<tr>
<td>$t_3$</td>
<td>Dairy, Jacket, Footwear, Viagra</td>
</tr>
<tr>
<td>$t_4$</td>
<td>Dairy, Jacket, Footwear, PregnancyTest</td>
</tr>
<tr>
<td>$t_5$</td>
<td>Liquor, Dairy, Jacket, Pants</td>
</tr>
</tbody>
</table>
generalized to Liquor, all Milk and Yogurt to Dairy, and all Geta, Hose and Shoe to Footwear.

3.2 Adversary’s Knowledge as a Logical Expression

We model an adversary’s knowledge about the presence and absence of a single, non-sensitive item by decorating the item with + and − respectively.

Definition 2 (Positive Item and Negative Item). For a non-sensitive item, we decorate it with + (−) to represent the presence (absence) of i, and call i+ (i−) the positive (negative) decoration of i or simply a positive (negative) item. We also call i+ (i−) a decorated item, and i itself an undecorated item.

We further model the adversary’s knowledge about his/her target individual by a conjunction of m + n clauses with m clauses expressing the presence of non-sensitive items and n clauses expressing the absence of non-sensitive items.

Definition 3 (Knowledge-Expression). The knowledge of an adversary about a target individual is a logical expression KE = (∨1x = 1...m j+) ∧ (∨n = 1...n k−) where disjunction (j+) and disjunction (k−) for x = 1,...,m represents the knowledge of the presence of non-sensitive items, and negative item i− for x = 1,...,n represents the knowledge of the absence of items.

In general, the bounded knowledge of a realistic privacy adversary may be at any level, which however can always be translated into a knowledge expression defined at the leaf level. This is because knowing the presence of a general item is equivalent to knowing the presence of at least one of its descendant leaf items (a disjunction of positive items), and knowing the absence of a general item is equivalent to knowing the absence of all its descendant leaf items (a conjunction of negative items).

Therefore, in Definition 3, KE is defined at the leaf level. The number m of disjunctions of positive items and the number n of negative items depend on the adversary’s power or effort in acquiring KE, which is denoted by power(KE) = (m, n).

For example, a simple form of knowledge-expression can be just a conjunction of decorated, non-sensitive items. Bob’s knowledge is “Wine + ∧ Yogurt +” regarding Alice with power being (2, 0), and “Beer + ∧ Wine−” regarding Darth with power (1, 1). A general form of knowledge could be “(Hose + ∨ Geta +) ∧ Milk−” with power (1, 1).

3.3 How Adversary Launching Attacks

For each (generalized) transaction, only (generalized) items present in the transaction will be published.

The adversary launches attacks by finding out the subset of published transactions that satisfy his/her knowledge-expression KE in the following two cases.

In the first case the published transactions are not generalized, the adversary simply follows Definitions 2 and 3 to determine transactions that satisfy his/her KE. For example, t1 in Table 1 satisfies Bob’s knowledge regarding Alice, “Wine + ∧ Yogurt +”, and t2 satisfies Bob’s knowledge about Darth, “Beer + ∧ Wine−”.

In the second case the published transactions are generalized by the data publisher, the adversary has to rely on the same semantics by Definition 1 to derive the most specific information available for determining possible satisfactions. For example, the generalized transactions t1′ and t2′ in D′ in Table 2 contain {Liquor, Dairy} that may be generalized from a transaction containing {Wine, Yogurt}. Thus, t1′ and t2′ satisfy “Wine + ∧ Yogurt +”, i.e., Bob’s knowledge about Alice. Similarly, t1′, t2′ and t3′ contain {Liquor}, each of which may be generalized from a transaction containing Beer but no Wine. Thus, t1′, t2′ and t3′ satisfy “Beer + ∧ Wine−”, i.e., Bob’s knowledge about Darth. Moreover, if Bob’s knowledge about a third person is “Milk−”, then the target transaction may be generalized into a transaction t4′ containing {Dairy} or t4′ not containing {Dairy} as Bob did not know whether Yogurt is present.

Clearly, the adversary can be thought of making attacks in two steps. The adversary first generalizes his/her knowledge-expression to the same level as the published data, and then finds out those generalized transactions that satisfy the generalized knowledge-expression by Definitions 2 and 3. For example, Bob can first generalize his knowledge “Wine + ∧ Yogurt +” and “Beer + ∧ Wine−” into “Liquor + ∧ Dairy−” and “Liquor +” respectively, and then make attacks. But, Bob cannot generalize “Milk−” into “Dairy−”.

Therefore, we have an important theorem.

Theorem 1 (Generalized KE and Its Power). A generalized form gKE of an original knowledge expression KE, with power(KE) = (m, n), contains decorated, generalized items. Each disjunction p of positive items in gKE is derived from a disjunction p in KE such that for every item i whose positive decoration is in p, the positive decoration of a hierarchical descendant of i is in p. For an item i whose negative decoration is in gKE, the negative decoration of every hierarchical descendant of i is in KE. The number #p of disjunctions of positive items in gKE is no more than m, and the total number of hierarchical leaf descendants of items whose negative decorations are in gKE is no more than n. Therefore, we have power(gKE) = (#gKE − ∑i ∈ KE #leaves(i)) ≤ power(KE).
4 Privacy Notion

We propose a new privacy notion that prevents both identity disclosure and sensitive item disclosure to an adversary whose knowledge $KE$ is bounded by $\text{power}(KE) \leq (m, n)$, and try to enforce such a privacy notion with minimum information loss.

4.1 $(k, \ell)^{(m,n)}$-Privacy

Although the power of $KE$ is defined at the leaf level of the generalization hierarchy, $KE$ somehow may be of general nature, i.e., quite vague and not specific. For example, “(Beer + ∨ Wine +)” is vaguer, less specific, and hence less powerful than “Beer +”. As Beer and Wine are the only children of Liquor, “(Beer + ∨ Wine +)” is equivalent to “Liquor +”, meaning that any children of Liquor may be present in the target transaction. In other words, $KE$ may be equivalent to a conjunction of decorated, generalized, non-sensitive items by Theorem 1, which can be denoted by an itemset.

To safe-guard the privacy, we prevent such an itemset $X$ (i.e., a conjunction) of decorated, generalized, non-sensitive items from becoming a privacy violation. In particular, we assure that the probability to identify any individual’s transaction by $X$ is no more than $1/k$, and the probability to guess any individual’s sensitive items by $X$ is no more than $1/\ell$.

**Definition 4** $(k, \ell)^{(m,n)}$-Privacy. A set-valued database $D$ observes $(k, \ell)^{(m,n)}$-privacy if for any itemset $X$ of decorated, generalized, non-sensitive items with $\text{power}(X) \leq (m, n)$ by Theorem 1, 1) either no transaction or more than $m$ transactions in $D$ support $X$, and 2) for each sensitive item $s$ at most $1/\ell$ of transactions in $D$ that support $X$ also contain $s$. We say that a transaction $t$ supports $X$ if $t$ or the generalization of $t$ satisfies $X$ by Definitions 2 and 3. We use $sup(X)$ to denote the number of transactions supporting $X$. Thus, we can rewrite 1) $\text{sup}(X) = 0$ or $\text{sup}(X) \geq k$, and 2) $\text{conf}(X \rightarrow \{s\}) = \frac{\text{sup}(X \cup \{s\})}{\text{sup}(X)} \leq 1/\ell$. If $X$ violates 1) or 2), $X$ is a privacy violation.

For example, both $D$ in Table 1 and $D' \in$ Table 2 violate $(2,2)^{(2,1)}$-privacy. However, $D'$ observes $(2,3)^{(1,0)}$-privacy.

We consider a realistic adversary whose knowledge $KE$ is bounded by $\text{power}(KE) \leq (m, n)$. Similar assumptions are also made in [14-16]. Nevertheless, our notion is more flexible and yet more general than prior notions in publishing set-valued data. [5] considers only the identity disclosure with $m+n = |I_N|$ (i.e., the size of the domain of non-sensitive items), [6] considers only the sensitive item disclosure with $m+n = |I_S|$, and [7] and [8] are special cases with $n = 0$.

The $(k, \ell)^{(m,n)}$-privacy is monotone in that if a privacy violation is supported by a solution defined by $Cut$, then the privacy violation is also supported by any solution defined by a specialization of $Cut$, i.e., the specialization of an invalid cut is also invalid.

4.2 Information Loss and Optimal Anonymization

We employ the full subtree generalization technique[24] to generalize non-sensitive items to enforce the $(k, \ell)^{(m,n)}$-privacy as we want to keep the domain exclusiveness[12], which is an important property required by data mining applications. Sensitive items will not be generalized as we want to analyze the trends and patterns regarding the sensitive items. The suppression technique is not suitable because suppression could introduce false absence of items and hence create artificial privacy violations, which is unacceptable for victims associated with the artificial violations.

The generalization process will cause information loss, which can be measured by various metrics. Our approach applies to any well-behaved metric, which has the following properties: 1) it is item-based, i.e., it charges information loss per item rather than per transaction, since transactions are of different sizes; 2) it is monotonic, i.e., it charges more on a general item than a specific item. Most metrics are well-behaved, e.g., loss metric (LM)[24] and normalized cardinality penalty (NCP)[25].

We use $\text{infoloss}(D, Cut)$ to denote the information loss incurred by the generalization solution defined by $Cut$, which can be decomposed into that of constituent items of $Cut$, i.e., $\text{infoloss}(D, Cut) = \sum_{x \in Cut} \text{infoloss}(D, x)$.

Our objective is to find an optimal anonymization solution that achieves our $(k, \ell)^{(m,n)}$-privacy with minimum information loss.

**Definition 5** (Optimal Anonymization Problem). Given a transaction database $D$, the hierarchy $H_N$ for non-sensitive items $I_N$, the set $I_S$ of sensitive items, and the privacy parameters $(k, \ell, m, n)$, find a generalization solution defined by $Cut_{min}$ that is a valid cut by Definition 4 and $\text{infoloss}(D, Cut_{min})$ is the minimum among all valid cuts.

5 Near-Optimal Anonymization Solution

The $(k, \ell)^{(m,n)}$-privacy is monotone. Conceptually, any top-down search algorithm, e.g., $k$-Optimize[26] and $\ell^*$-Optimize[12], could be employed for finding an optimal solution. However, while these algorithms work well with structured, relational data, they are not suitable for unstructured, set-valued data as the search space for verifying the existence of privacy violations with set-valued data is much larger than that with
In the early rounds, SingleRoundDF is efficient as it produces an optimal solution on the given hierarchy \( H \). First, by producing \( \text{sol}(i,0) \) for \( i = 1, \ldots, m \) and then \( \text{sol}(m,j) \) for \( j = 1, \ldots, n \) such that \( D^{(i,j)} = \text{gen}(D, \text{sol}(i,j)) \) satisfies \((k, \ell)(i,j)\)-privacy. In the early rounds, SingleRoundDF is efficient as it only verifies itemsets of small sizes, whose number is limited even with a large number of distinct items (a large hierarchy). In the later rounds, SingleRoundDF is also efficient as it works with a reduced number of distinct items (on a reduced hierarchy), which results in a limited number of itemsets to be verified although the itemsets are of larger sizes.

### 5.1 Progressive Anonymization

To find an optimal solution, we have to verify all the generalized itemsets that are within the adversary’s power \((m,n)\). The number of such itemsets depends on \( m \) and \( n \) as well as the size of the generalization hierarchy. It is computationally intractable to verify such itemsets when \( m, n \), and the size of the generalization hierarchy are all large.

Therefore, we first find a solution cut with small \( m \) and \( n \), i.e., a less restrictive privacy requirement, on a large (original) hierarchy; and then find a solution cut with large \( m \) and \( n \) on a small hierarchy derived by cutting off the portion under the previous solution cut; and so on until we get a solution that satisfies the full requirement.

For example, to find a solution observing \( (2,2)(2,1) \)-privacy, we may first find a solution satisfying \( (2,2)(2,0) \)-privacy, which yields \( \text{cut}_0 \) on the hierarchy in Fig. 1. A solution satisfying \( (2,2)(2,1)\)-privacy can be derived by searching the reduced hierarchy above \( \text{cut}_0 \), which is \( \text{cut}_a \). This idea is summarized by our algorithm, MOANA, namely Multi-round Optimal Anonymization against realistic Adversaries.

#### Algorithm 1. MOANA\((D, H_N, k, \ell, m, n)\)

1. \( \text{sol}_\text{round} \leftarrow \) the set of the leaves of \( H_N \)
2. \( H_\text{round} \leftarrow \) the original \( H_N \)
3. for \((i, j) = (1, 0)\) to \((m, 0)\), \((m, 1)\) to \((m, n)\) do
4. \( H_\text{round} \leftarrow \) the upper portion of \( H_\text{round} \) above \( \text{sol}_\text{round} \) (inclusive)
5. \( \text{Cut} \leftarrow \text{Cut}_\text{min} \leftarrow \) the root of \( H_\text{round} \)
6. \( \text{sol}_\text{round} \leftarrow \text{SingleRoundDF}(\text{Cut}, \text{Cut}_\text{min}, D, H_\text{round}, k, \ell, i, j) \)
7. return \( \text{sol}_\text{round} \)

MOANA runs in \( m + n \) rounds with each round calling SingleRoundDF to produce an optimal solution on the given hierarchy \( H_\text{round} \), i.e., first to produce \( \text{sol}(i,0) \) for \( i = 1, \ldots, m \) and then \( \text{sol}(m,j) \) for \( j = 1, \ldots, n \) such that \( D^{(i,j)} = \text{gen}(D, \text{sol}(i,j)) \) satisfies \((k, \ell)(i,j)\)-privacy. In the early rounds, SingleRoundDF is efficient as it only verifies itemsets of small sizes, whose number is limited even with a large number of distinct items (a large hierarchy). In the later rounds, SingleRoundDF is also efficient as it works with a reduced number of distinct items (on a reduced hierarchy), which results in a limited number of itemsets to be verified although the itemsets are of larger sizes.

#### 5.2 Depth First Search in One Round

SingleRoundDF enumerates possible solutions, i.e., cuts on \( H_\text{round} \) to find an optimal one. Such a process can be conceptually organized by a solution enumeration tree where a child cut is derived by specializing exactly one constituent item of its parent cut. For example, Fig. 2 shows part of such a tree that enumerates all possible cuts on the item hierarchy in Fig. 1.

SingleRoundDF depth-first searches the subtree of the solution enumeration tree rooted at \( \text{Cut} \) and updates the running best cut \( \text{Cut}_\text{min} \) recursively. At the very beginning, \( \text{Cut} \) and \( \text{Cut}_\text{min} \) are initialized to the root of \( H_\text{round} \). The depth-first search is facilitated with two kinds of pruning, i.e., pruning based on lower bounding the information loss and pruning based on privacy violations. The pseudo code of SingleRoundDF is as follows.

The \( \text{LB}(D, \text{Cut}) \) function estimates the lower bound of information loss of any solution in the subtree of the solution enumeration tree rooted at \( \text{Cut} \) by the following formula, based on the fact that the information loss with items that will not specialize in the subtree is a component of the solution. If the lower bound is greater

![Fig.2. Solution enumeration tree.](image-url)
than the information loss of the running best cut, the subtree is pruned.

\[ LB(D, Cut) = \sum_{x \in Cut \cap \text{specialize}(x)} \text{infoloss}(D, x). \]

The violated\((Cut, D, H_{\text{round}}, k, \ell, i, j)\) function verifies if \(D' = \text{gen}(D, Cut)\) violates \((k, \ell)^{(i,j)}\)-privacy and will be discussed in the next subsection. If the function returns true, the subtree rooted at \(Cut\) can be pruned since this privacy notion is monotone in that if there is a privacy violation in \(D' = \text{gen}(D, Cut)\), then the violation also exists in \(D'' = \text{gen}(D, Cut')\) for any \(Cut'\) that is a specialization of \(Cut\).

If the information loss of \(Cut_{\text{min}}\) is greater than that of \(Cut\), \(Cut\) becomes the new running best cut. For each child cut, the subtree rooted at the child cut will be recursively searched. At the end, the running best cut is returned.

### 5.3 Discovering Privacy Violations

The most challenging step is violated\((Cut, D, H_{\text{round}}, k, \ell, m, n)\), i.e., to check if there is any privacy violation in the anonymized database \(D'\) obtained by generalizing \(D\) to \(Cut\) on \(H_{\text{round}}\) by Definition 4. It follows that, for any set \(X\) of decorated, generalized, non-sensitive items with \(\text{power}(X) \leq (m, n)\), we have to count the supports of \(X\) and \(X \cup \{s\}\), i.e., \(\text{sup}(X)\) and \(\text{sup}(X \cup \{s\})\) for \(s \in I_S\). Subsequently, we need to determine the set of transactions that support \(X\) and \(X \cup \{s\}\), namely, \(TS(X)\) and \(TS(X \cup \{s\})\). For example, \(TS(\{\text{Wine} +\})\) is \(\{t_1, t_5\}\) in \(D\) in Table 1, and \(TS(\{\text{Wine} +, \text{AdultToy}\})\) is \(\{t_1\}\). Thus, \(\text{sup}(\{\text{Wine} +\}) = 2\), and \(\text{sup}(\{\text{Wine} +, \text{AdultToy}\}) = 1\).

The big challenge is how to represent transaction sets in a scalable manner as we cannot afford to explode transactions by inserting generalized items. To address this challenge, we propose a hierarchy tagging method which simply attaches hierarchical tags to leaf items to encode the hierarchy information and can be used with various representations.

In this subsection, we show how to extend the array-based \(TS\) representation, \(TVLA^1\), with hierarchical tags. For brevity, we first discuss the special case with \(n = 0\), i.e., without negative items, where we simply omit the + sign. At the end, we will discuss the general case with \(n > 0\).

#### 5.3.1 Hierarchy Tagging Technique for \(TS\) Representation

\(TVLA^1\) consists of three parts: an item list (IL), linked queues (LQ), and transaction arrays. Each IL entry has three fields: an item-id, a support count, and a pointer. Each transaction is stored in an array. The same items in different transaction arrays are threaded together by an LQ which is attached to the IL entry with the same item.

We propose an extended TVLA, which encodes the hierarchy information by 1) attaching a hierarchical tag to each non-sensitive leaf item in a transaction array, which appears at the upper right corner of the item as shown in Fig.3 and stores the level number of the highest hierarchical ancestor of the item that has not been encoded by the other items in the transaction; 2) conceptually representing IL by the hierarchy tree. The extended TVLA avoids expanding the transaction arrays while the exact hierarchy information is preserved.

For example, Fig.3 shows the extended TVLA which compresses \(D\) in Table 1. The first array represents \(t_1\). The tag of Wine is 0 since its highest ancestor Entity is at level 0. Milk is tagged with 2 since its highest ancestor not yet encoded is Dairy (at level 2). Although Entity and Nutrient are also its ancestors, they are encoded by Wine. Yogurt is tagged with 3 since Milk tells that ancestors of Yogurt are all encoded by other items.

![Extended TVLA with hierarchical tags](image-url)
The IL entry (Beer, 2) holds the leaf item Beer and its support 2 in $D$, and an LQ starting from (Beer, 2) threads Beer in the 2nd and 5th array together. The IL entry (Liquor, 3) holds Liquor and its support 3 coming from Beer and Wine (the hierarchical descendants of Liquor) while no array stores Liquor.

5.3.2 Determine $TS$ by Pseudo Projection

The transaction set supporting the empty itemset is the database itself. For an itemset $X$ and its superset $Y = X \cup \{y\}$ with $y$ being an item in IL of $TS(X)$, we can derive $TS(Y)$ from $TS(X)$ in two steps.

Step 1. Delimit the transaction arrays of $TS(Y)$ by following the LQ starting from the IL entries in $TS(X)$ holding the hierarchical descendants of $y$ and by checking hierarchical tags to avoid repetitive inclusions of an array.

For example, consider deriving $TS(\{Dairy\})$ from $TS(\{\})$, i.e., the transaction set supporting the empty itemset whose TVLA is Fig.3. As Milk and Yogurt are the hierarchical descendants of Dairy, the 1st, 3rd, 4th, and 5th arrays comprise the transaction arrays of $TS(\{Dairy\})$. Although the 1st array appears both on LQs of Milk and Yogurt, the hierarchical tag of Yogurt on the 1st array says that Yogurt only represents itself, so when following LQ starting from the entry of Yogurt, the 1st array is ignored (repetition avoided).

Step 2. Build IL of $TS(Y)$ by correctly counting the support of each distinct item on the arrays delimited in Step 1. The supports of generalized items are correctly accumulated by examining hierarchical tags.

For example, the tag of Wine on the 1st array identifies at Step 1 is 0, which means that the 1st array supports Wine (at level 3) and its hierarchical ancestors, Liquor (at level 2), Nutrient (at level 1), and Entity (at level 0). So, we can correctly accumulate these items’ supports.

In the preceding discussion, we only consider the special case with $n = 0$ for brevity. For the general case with $n > 0$, we simply treat $i^+$ and $i^-$ as two different items, and the hierarchy tagging technique still applies. For example with $n > 0$, the tagged version of the first transaction is \{AdultToy, Beer$^2$, Wine$^0$, Milk$^2$, Yogurt$^3$, Jacket$^1$, Pants$^3$, Getta$^2$, Hose$^3$, Shoe$^3\}.

6 Experimental Evaluation

We perform comparative experiments to investigate two key points of this paper. The first key point is that assuming the adversary has unbounded knowledge as in [5-6] will introduce excessive distortion, which is investigated by comparing with the local generalization algorithm LG$^5$. The conclusion also applies to [6] in general. The second key point is that $(k, \ell)^{(m,n)}$-privacy provides sufficient privacy protection, and our near-optimal algorithm MOANA helps retain more data utility than heuristic algorithms and is efficient and scalable. This point is investigated by comparing with the global generalization algorithm AA$^7$ and the total suppression algorithm MM$^8$.

To ensure the fairness of comparative experiments, we employed the same benchmark databases POS and WV2$^{27}$ that were also used by [5-7] in their experiments. Notice that the search space of algorithms depends on the privacy parameters $m$ and $n$ as well as the maximum length of transactions and the number of distinct items in a database. Table 3 summarizes the characteristics of the databases, where $|D|$ is the size of a database, $|I|$ is the number of items, $\max|t|$ is the maximum length of transactions, and $\text{avg}|t|$ is the average.

Hierarchy trees were created by the procedure in [7], which randomly assigns domain items to the leaves of the tree. We report the average result based on three such random assignments. Due to the space limitation, we will mainly show the results on POS.

| Database | $|D|$   | $|I|$   | $\max|t|$ | $\text{avg}|t|$ |
|----------|--------|--------|----------|------------|
| POS      | 515597 | 1687   | 164      | 6.5        |
| WV2      | 77512  | 3340   | 161      | 5          |

We measure data utility in terms of usefulness in frequent pattern mining (defined in Subsection 6.2.2), and NCP$^{25}$ that defines the normalized information loss for generalizing database $D$ to $Cut$ as follows.

$$NCP(D, Cut) = \frac{\sum_{x \in Cut} O(D, x) \times NCP(x)}{\sum_{x \in Cut} O(D, x)},$$

$$NCP(x) = \begin{cases} 0, & \text{when } \#\text{leaves}(x) = 1, \\ \frac{\#\text{leaves}(x)}{\#\text{leaves}(\text{Root})}, & \text{otherwise}, \end{cases}$$

where $\#\text{leaves}(x)$ is the number of leaves in the subtree of the generalization taxonomy rooted at $x$, $\#\text{leaves}(\text{Root})$ is the total number of leaves of the taxonomy, and $O(D, x)$ is the total occurrences of all the descendants of $x$ in $D$. All experiments were conducted on a 2.6 GHz Intel Pentium IV PC with 1 GB RAM.

6.1 Evaluation by Comparing with AA $(\ell = 1, n = 0)$

We investigate if our algorithm MOANA can significantly reduce information loss compared to the heuristic algorithm AA$^7$, and how MOANA performs in terms of efficiency. Since AA can only enforce $(k, \ell)^{(m,n)}$-privacy with $\ell = 1$ and $n = 0$, this is the basic setting. Other parameters are set as follows: $k$ varies from 5 to 200 with 5 being the default, $m$ varies...
from 1 to 11 with 3 being the default for POS and 6 for WV2, which is an experiment parameter setting that is more general than and compatible to the setting in [7] that yields results that can provide more insights and can be verified with results in [7].

Fig. 4(a) shows the information loss on POS with the varying k. MOANA has 1/4 to 1/2 less information loss than AA. Fig. 4(b) shows the information loss on POS with the varying m. When m is small the gap in information loss between AA and MOANA is not significant, but when m is large (≥ 3), AA incurs 1/2 more information loss than MOANA. We observed the similar results on WV2 as shown in Figs. 4(c) and 4(d), although the gain by MOANA compared to AA is less striking, which is due to the dataset characteristics. WV2 has a shorter average transaction size and hence less distinct itemsets subject to privacy attacks, and thus needs less generalization while WV2 also has a taller taxonomy and hence more good generalization solutions. Therefore, even a heuristic solution incurs little information loss, and thus the gain by a (near) optimal solution is less striking. In general, a near-optimal solution by MOANA has a significant gain in data utility relative to a heuristic solution by AA.

Fig. 4(e) shows the runtime on POS with the varying k. MOANA is twice to 4 times faster than AA. Fig. 4(f) shows the runtime on POS with varying m. MOANA is also twice to 4 times faster than AA. We also observed the similar results on WV2 as shown in Figs. 4(g) and 4(h). Surprisingly, our near-optimal algorithm MOANA is more efficient than the heuristic algorithm AA, which is partly due to our hierarchy tagging technique is quite scalable and hence helps improve efficiency. When m is large (> 9), the generalization cut remains almost the same, so the runtime for a single round of MOANA to derive a generalization cut based on the previous round is minor. Therefore, MOANA can work with any m.

6.2 Evaluation by Comparing with LG (ℓ = 1, \(m > 0\))

The notion of LG\(^{[5]}\) is equivalent to the extreme case of \((k, \ell)(m,n)\)-privacy with \(m + n = |I_N|\) (i.e., the size of the domain of non-sensitive items) and \(\ell = 1\) (identity disclosure only). With such an extreme case, there is no doubt that LG has less information loss than MOANA in terms of NCP since LG employs the multi-dimensional partitioning (local recoding)\(^{[11]}\) while MOANA employs the full subtree generalization (global recoding)\(^{[24]}\).

Nevertheless, it is interesting to compare LG and MOANA with \(m\) and \(n\) varying in a moderate range for the following reasons. First, such a comparison in terms of NCP helps investigate if our \((k, \ell)(m,n)\)-privacy notion retains enough data utility while providing sufficient privacy protection. Second, such a comparison in terms of frequent pattern mining helps understand the expense coming with LG, that is, the loss of the domain exclusiveness.

6.2.1 Comparing with LG in Terms of NCP and Runtime

For simplicity, we set the same value for \(m\) and \(n\), i.e., \(m = n\), which varies from 1 to 30 with 20 being the default. And \(k\) varies from 5 to 200 with 5 being the default, which yields results that are verifiable with results in [5].

Fig. 5(a) shows the information loss on POS with the varying \(k\). The information loss incurred by LG is up...
Jun-Qiang Liu: Publishing Set-Valued Data Against Realistic Adversaries

Fig. 5. Comparison with LG on POS ($\ell = 1, n > 0$). (a) NCP vs $k$ ($m = n = 20$, POS). (b) NCP vs $m = n$ ($k = 5$, POS). (c) Runtime vs $k$ ($m = n = 20$, POS). (d) Runtime vs $m = n$ ($k = 5$, POS). (e) #Patterns vs $k$ ($m = n = 20$, POS). (f) Error vs $k$ ($m = n = 20$, POS). (g) #Patterns vs $m = n$ ($k = 5$, POS). (h) Error vs $m = n$ ($k = 5$, POS). (i) #Patterns vs minsup ($m = n = 20$, $k = 5$). (j) Error vs minsup ($m = n = 20$, $k = 5$).

Fig. 5(b) shows the information loss on POS with the varying $m$. The information loss by LG is greater than MOANA with $m = n < 10$, the same as MOANA with $10 \leq m = n \leq 20$, and smaller than MOANA with $m = n > 20$.

Figs. 5(c) and 5(d) show the runtime on POS with the varying $k$ and $m = n$ respectively. For all the cases, LG is much more efficient than MOANA since it employs the partition approach, which is made possible by handling set-valued data as a relational table, i.e., treating every distinct item as a binary attribute and taking all attributes (items) as the quasi-identifier.

Overall, LG incurs less information loss in terms of NCP with the same privacy parameter, which however comes with an expense as explained in Subsection 6.2.2. Nevertheless, MOANA can retain enough data utility while providing sufficient privacy protection, e.g., $(k, \ell)^{(m, n)}$-privacy with $m$ and $n \geq 20$.

### 6.2.2 Comparing with LG in Frequent Pattern Mining

We measure the usefulness of anonymized data in frequent pattern mining as follows. Given a support threshold $\text{minsup}$, let $OP$ be the set of generalized frequent patterns in the original data, and $AP$ be that in the anonymized data. First, we define $\#\text{Patterns} = |AP|$, which is an indicator of the amount of information that is retained. Second, we define an indicator of the accuracy of the retained information,

$$\text{Error} = \frac{\sum_{r \in OP \setminus AP} |\text{sup}(r, OP) - \text{sup}(r, AP)|}{\sum_{r \in AP \cap OP} \text{sup}(r, OP)},$$

where $\text{sup}(r, OP)$ and $\text{sup}(r, AP)$ are the supports of pattern $r$ in $OP$ and $AP$ respectively.

Figs. 5(e), 5(g) and 5(i) show #Patterns with the varying $k$, $m = n$, and $\text{minsup}$ respectively. The additional parameter $\text{minsup}$ varies between 1% and 10% with 1% being the default. Figs. 5(f), 5(h) and 5(j) show Error. The results show that MOANA retains more information than LG. More importantly, MOANA keeps the accurate support counts of patterns, i.e., the same as in the original data, while LG has an error rate around 40% for $\text{minsup} = 1\%$. The observations are as follows.

1) The seemingly low information loss by LG in terms of NCP comes with the loss of the domain exclusiveness, i.e., Beer and Liquor co-exist in the anonymized data. Therefore, the support counts of patterns are inaccurate as some Beer is generalized to Liquor while some is not.

2) With the increase of $\text{minsup}$, the error rate by LG decreases, which suggests that LG gets good
approximations for the patterns that have very high supports and hence that are at very high level of the generalization hierarchy. In other words, to be useful for frequent pattern mining, all Beer in published data will be generalized to Liquor or even Nutrient.

3) MOANA keeps the accurate supports since the domain exclusiveness is preserved. For the same sake, AA\textsuperscript{[7]} should too. As the executable of AA did not output the anonymized data, we could not evaluate AA in frequent pattern mining. But AA should preserve less frequent patterns as higher information loss suggests data are generalized to higher levels.

6.3 Comparing with MM and Evaluating General Case (\(\ell > 1\))

We evaluate our approach in preventing both identity disclosure and sensitive item disclosure (\(\ell > 1\)) with the POS dataset. We randomly selected a fixed number \(\beta\) (beta) of sensitive items from items with a support less than 50\% since a dataset containing a sensitive item with a support above 50\% does not satisfy the eligibility condition\textsuperscript{[28]} for \(\ell \geq 2\). \(\beta\) ranges from 20 to 100 with 60 being the default, which is similar to the setting in \[6\]. The default value of \(k\) is 5. And \(\theta\) (theta) stands for \(1/\ell\) with \(\theta = 20\%\) (\(\ell = 5\)) being the default.

6.3.1 Comparing with MM (\(\ell > 1, n = 0\))

As MM\textsuperscript{[8]} does not consider the knowledge about the absence of items, we run MOANA with \(n = 0\) and \(m\) ranging from 1 to 30, but MM can only run with \(m \leq 3\), so the default \(m\) is 3.

Figs. 6(a)\textendash 6(c) show the information loss with the varying \(m\), \(\beta\), and \(\theta\) (1/\(\ell\)) respectively. The information loss by MM (\(n = 0\)) is 8\% with \(m \leq 2\) and over 30\% for the other cases. That by MOANA (\(n = 0\)) is all no more than 5\%. Figs. 6(d)\textendash 6(f) show the runtime. For \(m \leq 3\), MM (\(n = 0\)) is as efficient as MOANA (\(n = 0\)), but MM (\(n = 0\)) cannot run with \(m > 3\) because of scalability issue.

Figs. 6(g)\textendash 6(i) show \#Patterns where minsup = 1\%. For \(m \leq 2\), MM (\(n = 0\)) keeps more frequent patterns than MOANA (\(n = 0\)), but for \(m \geq 3\), MOANA (\(n = 0\)) keeps 34\% more. Figs. 6(j)\textendash 6(l) show Error. While MOANA (\(n = 0\)) always retains the exact support counts of frequent patterns, MM (\(n = 0\)) has an error rate around 40\% for \(m > 3\).

![Fig.6. Comparison with MM (n = 0) and general case (n > 0) on POS. (a) NCP vs m = n. (b) NCP vs beta. (c) NCP vs theta. (d) Runtime vs m = n. (e) Runtime vs beta. (f) Runtime vs theta. (g) #Patterns vs m = n. (h) #Patterns vs beta. (i) #Patterns vs theta. (j) Error vs m = n. (k) Error vs beta. (l) Error vs theta.](image-url)
The comparison with MM suggests that the total suppression approach is not good at handling sparse data in terms of a general purpose metric. MM also has a large Error since suppression reduces the supports of generalized items and generalized patterns although it does not affect the supports of a pattern consisting of only items at the leaf level of the hierarchy.

6.3.2 Evaluating the General Case ($\ell > 1, n > 0$)

We investigate if we can provide sufficient protection and retain enough data utility in enforcing the full requirement of $(k, \ell)^{(m,n)}$-privacy. For simplicity, we set $m = n$, and report the results in Fig.6 with the default $m = n = 5$.

In particular, Figs. 6(a)~6(c) also show the information loss by MOANA ($m = n$) with $m$ and $n$ varying from 1 to 30, with $\beta$ varying from 20 to 100, and with $\theta (1/\ell)$ varying from 5% to 30% respectively. Figs. 6(d)~6(f) show the runtime, Figs. 6(g)~6(i) $\# $Patterns, and Figs. 6(j)~6(l) Error.

The major findings are as follows. 1) MOANA ($m = n$) can provide sufficient protection (with $m$ and $n$ up to 30) while retaining enough data utility (with NCP $\leq 7.2\%$). 2) Interestingly, MOANA ($m = n$) runs much faster than MOANA ($n = 0$) with large $m$ and $n$ as shown in Fig.6(d), which suggests that enforcing a stringent privacy requirement is easier than a moderate privacy requirement in terms of computational overhead.

7 Conclusions

This paper proposes $(k, \ell)^{(m,n)}$-privacy for preventing attacks from a realistic adversary who has bounded knowledge about the presence and absence of items in the target transaction, and presents a near-optimal algorithm MOANA for anonymizing set-valued data. This notion provides sufficient privacy protection, and MOANA can retain enough data utility and is quite efficient and scalable.

References


Jun-Qiang Liu is a professor at the School of Information and Electronic Engineering in Zhejiang Gongshang University. He received his Ph.D., M.Sc. and B.Sc. from Simon Fraser University in Canada, Zhejiang University and Peking University in China respectively. His research interests include information security, privacy protection, data mining, management information systems, and software engineering. He has published over 50 papers in prestigious journals and international conferences, including ACM SIGKDD, IEEE ICDE, IEEE ICDM, Chinese Journal of Computers, Journal of Software, Journal of Computer Research and Development, and Chinese Journal of Electronics.