Chapter 1

K-ANONYMOUS DATA MINING: A SURVEY

V. Ciriani, S. De Capitani di Vimercati, S. Foresti, and P. Samarati
DTI - Università degli Studi di Milano
26013 Crema - Italy
{ciriani, decapita, foresti, samarati}@dti.unimi.it

Abstract Data mining technology has attracted significant interest as a means of identifying patterns and trends from large collections of data. It is however evident that the collection and analysis of data that include personal information may violate the privacy of the individuals to whom information refers. Privacy protection in data mining is then becoming a crucial issue that has captured the attention of many researchers.

In this chapter, we first describe the concept of k-anonymity and illustrate different approaches for its enforcement. We then discuss how the privacy requirements characterized by k-anonymity can be violated in data mining and introduce possible approaches to ensure the satisfaction of k-anonymity in data mining.

Keywords: k-anonymity, data mining, privacy

1. Introduction

The amount of data being collected every day by private and public organizations is quickly increasing. In such a scenario, *data mining techniques* are becoming more and more important for assisting decision making processes and, more generally, to extract hidden knowledge from massive data collections in the form of patterns, models, and trends that hold in the data collections. While not explicitly containing the original actual data, data mining results could potentially be exploited to infer information - contained in the original data - and not intended for release, then potentially breaching the privacy of the parties to whom the

data refer. Effective application of data mining can take place only if proper guarantees are given that the privacy of the underlying data is not compromised. The concept of privacy preserving data mining has been proposed in response to these privacy concerns [6]. Privacy preserving data mining aims at providing a trade-off between sharing information for data mining analysis, on the one side, and protecting information to preserve the privacy of the involved parties on the other side. Several privacy preserving data mining approaches have been proposed, which usually protect data by modifying them to mask or erase the original sensitive data that should not be revealed [4, 6, 13]. These approaches typically are based on the concepts of: *loss of privacy*, measuring the capacity of estimating the original data from the modified data, and *loss of information*, measuring the loss of accuracy in the data. In general, the more the privacy of the respondents to which the data refer, the less accurate the result obtained by the miner and vice versa. The main goal of these approaches is therefore to provide a trade-off between privacy and accuracy. Other approaches to privacy preserving data mining exploit cryptographic techniques for preventing information leakage [20, 30]. The main problem of cryptography-based techniques is, however, that they are usually computationally expensive.

Privacy preserving data mining techniques clearly depend on the definition of privacy, which captures what information is sensitive in the original data and should therefore be protected from either direct or indirect (via inference) disclosure. In this chapter, we consider a specific aspect of privacy that has been receiving considerable attention recently, and that is captured by the notion of *k-anonymity* [11, 26, 27]. k-anonymity is a property that models the protection of released data against possible re-identification of the respondents to which the data refer. Intuitively, k-anonymity states that each release of data must be such that every combination of values of released attributes that are also externally available and therefore exploitable for linking can be indistinctly matched to at least k respondents. k-anonymous data mining has been recently introduced as an approach to ensuring privacy-preservation when releasing data mining results. Very few, preliminary, attempts have been presented looking at different aspects in guaranteeing k-anonymity in data mining. We discuss possible threats to k-anonymity posed by data mining and sketch possible approaches to their counteracting, also briefly illustrating some preliminary results existing in the current literature. After recalling the concept of k-anonymity (Section 2) and some proposals for its enforcement (Section 3), we discuss possible threats to k-anonymity to which data mining results are exposed (Section 4). We then illustrate (Section 5) possible approaches

combining k-anonymity and data mining, distinguishing them depending on whether k-anonymity is enforced directly on the private data (before mining) or on the mined data themselves (either as a post-mining sanitization process or by the mining process itself). For each of the two approaches (Section 6 and 7, respectively) we discuss possible ways to capture k-anonymity violations to the aim, on the one side, of defining when mined results respect k-anonymity of the original data and, on the other side, of identifing possible protection techniques for enforcing such a definition of privacy.

2. *k*-Anonymity

k-anonymity [11, 26, 27] is a property that captures the protection of released data against possible re-identification of the respondents to whom the released data refer. Consider a private table PT, where data have been de-identified by removing explicit identifiers (e.g., `SSN` and `Name`). However, values of other released attributes, such as `ZIP`, `Date_of_birth`, `Marital_status`, and `Sex` can also appear in some external tables jointly with the individual respondents' identities. If some combinations of values for these attributes are such that their occurrence is unique or rare, then parties observing the data can determine the identity of the respondent to which the data refer or reduce the uncertainty over a limited set of respondents. *k-anonymity demands that every tuple in the private table being released be indistinguishably related to no fewer than k respondents.* Since it seems impossible, or highly impractical and limiting, to make assumptions on which data are known to a potential attacker and can be used to (re-)identify respondents, k-anonymity takes a safe approach requiring that, in the released table itself, the respondents be indistinguishable (within a given set of individuals) with respect to the set of attributes, called *quasi-identifier*, that can be exploited for linking. In other words, k-anonymity requires that if a combination of values of quasi-identifying attributes appears in the table, then it appears with at least k occurrences.

To illustrate, consider a private table reporting, among other attributes, the marital status, the sex, the working hours of individuals, and whether they suffer from hypertension. Assume attributes `Marital_status`, `Sex`, and `Hours` are the attributes jointly constituting the quasi-identifier. Figure 1.1 is a simplified representation of the projection of the private table over the quasi-identifier. The representation has been simplified by collapsing tuples with the same quasi-identifying values into a single tuple. The numbers at the right hand side of the table report, for each tuple, the number of actual occurrences, also spec-

Marital_status	*Sex*	*Hours*	*#tuples (Hyp. values)*
divorced	M	35	2 (0Y, 2N)
divorced	M	40	17 (16Y, 1N)
divorced	F	35	2 (0Y, 2N)
married	M	35	10 (8Y, 2N)
married	F	50	9 (2Y, 7N)
single	M	40	26 (6Y, 20N)

Figure 1.1. Simplified representation of a private table

ifying how many of these occurrences have values `Y` and `N`, respectively, for attribute `Hypertension`. For simplicity, in the following we use such a simplified table as our table PT.

The private table PT in Figure 1.1 guarantees k-anonymity only for $k \leq 2$. In fact, the table has only two occurrences of divorced (fe)males working 35 hours. If such a situation is satisfied in a particular correlated external table as well, the uncertainty of the identity of such respondents can be reduced to two specific individuals. In other words, a data recipient can infer that any information appearing in the table for such divorced (fe)males working 35 hours, actually pertains to one of two specific individuals.

It is worth pointing out a simple but important observation (to which we will come back later in the chapter): if a tuple has k occurrences, then any of its sub-tuples must have at least k-occurrences. In other words, the existence of k occurrences of any sub-tuple is a necessary (not sufficient) condition for having k occurrences of a super-tuple. For instance, with reference to our example, k-anonymity over quasi-identifier {`Marital_status`, `Sex`, `Hours`} requires that each value of the individual attributes, as well as of any sub-tuple corresponding to a combination of them, appears with at least k occurrences. This observation will be exploited later in the chapter to assess the non satisfaction of a k-anonymity constraint for a table based on the fact that a sub-tuple of the quasi-identifier appears with less than k occurrences. Again with reference to our example, the observation that there are only two tuples referring to divorced females allows us to assert that the table will certainly not satisfy k-anonymity for $k > 2$ (since the two occurrences will remain at most two when adding attribute `Hours`).

Two main techniques have been proposed for enforcing k-anonymity on a private table: *generalization* and *suppression*, both enjoying the property of preserving the truthfulness of the data.

Generalization consists in replacing attribute values with a generalized version of them. Generalization is based on a domain generaliza-

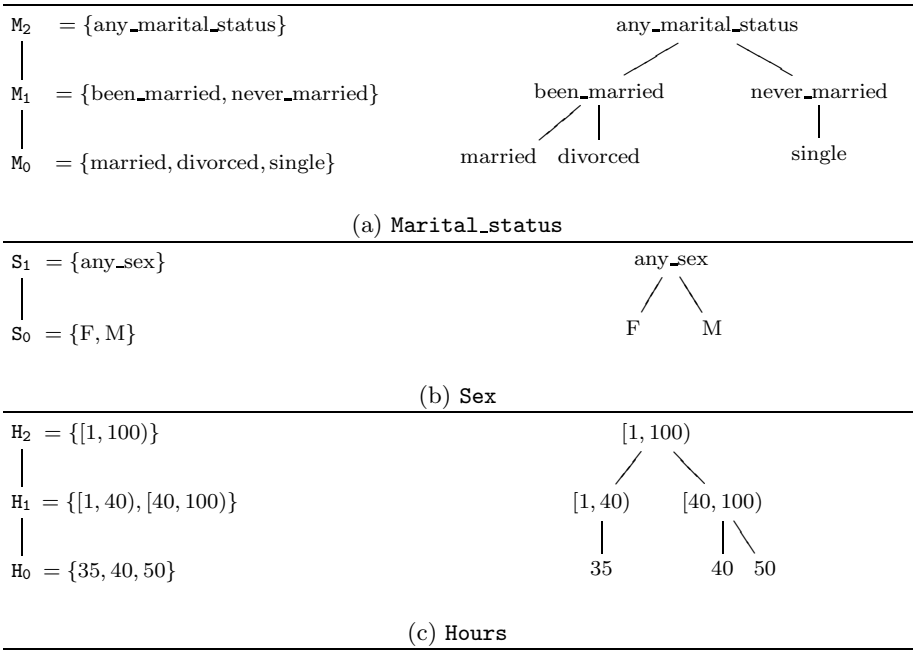

Figure 1.2. An example of domain and value generalization hierarchies

tion hierarchy and a corresponding value generalization hierarchy on the values in the domains. Typically, the domain generalization hierarchy is a total order and the corresponding value generalization hierarchy a tree, where the parent/child relationship represents the direct generalization/specialization relationship. Figure 1.2 illustrates an example of possible domain and value generalization hierarchies for the quasi-identifying attributes of our example.

Generalization can be applied at the level of single cell (substituting the cell value with a generalized version of it) or at the level of attribute (generalizing all the cells in the corresponding column). It is easy to see how generalization can enforce k-anonymity: values that were different in the private table can be generalized to a same value, whose number of occurrences would be the sum of the number of occurrences of the values that have been generalized to it. The same reasoning extends to tuples. Figure 1.11(d) reports the result of a generalization over attribute `Sex` on the table in Figure 1.1, which resulted, in particular, in divorced people working 35 hours to be collapsed to the same tuple {`divorced`, `any_sex`, `35`}, with 4 occurrences. The table in Figure 1.11(d) satisfies k-anonymity for any $k \leq 4$ (since there are no less than 4 respondents for each combination of values of quasi-identifying attributes). Note that

Generalization	**Suppression**			
	Tuple	*Attribute*	*Cell*	*None*
Attribute	AG_TS	AG_AS ≡ AG_	AG_CS	AG_ ≡ AG_AS
Cell	CG_TS not applicable	CG_AS not applicable	CG_CS ≡ CG_	CG_ ≡ CG_CS
None	_TS	_AS	_CS	_ not interesting

Figure 1.3. Classification of k-anonymity techniques [11]

4-anonymity could be guaranteed also by only generalizing (to `any_sex`) the sex value of divorced people (males and females) working 35 hours while leaving the other tuples unaltered, since for all the other tuples not satisfying this condition there are already at least 4 occurrences in the private table. This cell generalization approach has the advantage of avoiding generalizing all values in a column when generalizing only a subset of them suffices to guarantee k-anonymity. It has, however, the disadvantage of not preserving the homogeneity of the values appearing in the same column.

Suppression consists in protecting sensitive information by removing it. Suppression, which can be applied at the level of single cell, entire tuple, or entire column, allows reducing the amount of generalization to be enforced to achieve k-anonymity. Intuitively, if a limited number of outliers would force a large amount of generalization to satisfy a k-anonymity constraint, then such outliers can be removed from the table thus allowing satisfaction of k-anonymity with less generalization (and therefore reducing the loss of information).

Figure 1.3 summarizes the different combinations of generalization and suppression at different granularity levels (including combinations where one of the two techniques is not adopted), which correspond to different approaches and solutions to the k-anonymity problem [11]. It is interesting to note that the application of generalization and suppression at the same granularity level is equivalent to the application of generalization only (AG_≡AG_AS and CG_≡CG_CS), since suppression can be modeled as a generalization to the top element in the value generalization hierarchy. Combinations CG_TS (cell generalization, tuple suppression) and CG_AS (cell generalization, attribute suppression) are not applicable since the application of generalization at the cell level implies the application of suppression at that level too.

3. Algorithms for Enforcing *k*-Anonymity

The application of generalization and suppression to a private table PT produces less precise (more general) and less complete (some values are suppressed) tables that provide protection of the respondents' identities. It is important to maintain under control, and minimize, the information loss (in terms of loss of precision and completeness) caused by generalization and suppression. Different definitions of minimality have been proposed in the literature and the problem of finding minimal k-anonymous tables, with attribute generalization and tuple suppression, has been proved to be computationally hard [2, 3, 22].

Within a given definition of minimality, more generalized tables, all ensuring minimal information loss, may exist. While existing approaches typically aim at returning any of such solutions, different criteria could be devised according to which a solution should be preferred over the others. This aspect is particularly important in data mining, where there is the need to maximize the usefulness of the data with respect to the goal of the data mining process (see Section 6). We now describe some algorithms proposed in literature for producing k-anonymous tables.

Samarati's Algorithms. The first algorithm for AG_TS (i.e., generalization over quasi-identifier attributes and tuple suppression) was proposed in conjunction with the definition of k-anonymity [26]. Since the algorithm operates on a set of attributes, the definition of domain generalization hierarchy is extended to refer to tuples of domains. The domain generalization hierarchy of a domain tuple is a lattice, where each vertex represents a generalized table that is obtained by generalizing the involved attributes according to the corresponding domain tuple and by suppressing a certain number of tuples to fulfill the k-anonymity constraint. Figure 1.4 illustrates an example of domain generalization hierarchy obtained by considering `Marital_status` and `Sex` as quasi-identifying attributes, that is, by considering the domain tuple $\langle \mathtt{M}_0, \mathtt{S}_0 \rangle$. Each path in the hierarchy corresponds to a generalization strategy according to which the original private table PT can be generalized. The main goal of the algorithm is to find a k-minimal generalization that suppresses less tuples. Therefore, given a threshold MaxSup specifying the maximum number of tuples that can be suppressed, the algorithm has to compute a generalization that satisfies k-anonymity within the MaxSup constraint. Since going up in the hierarchy the number of tuples that must be removed to guarantee k-anonymity decreases, the algorithm performs a binary search on the hierarchy. Let h be the height of the hierarchy. The algorithm first evaluates all the solutions at height

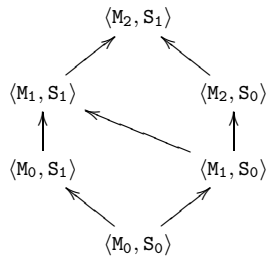

Figure 1.4. Generalization hierarchy for QI={Marital_status, Sex}

$\lfloor h/2 \rfloor$. If there is at least a k-anonymous table that satisfies the MaxSup threshold, the algorithm checks solutions at height $\lfloor h/4 \rfloor$; otherwise it evaluates solutions at height $\lfloor 3h/4 \rfloor$, and so on, until it finds the lowest height where there is a solution that satisfies the k-anonymity constraint. As an example, consider the private table in Figure 1.1 with QI={Marital_status, Sex}, the domain and value generalization hierarchies in Figure 1.2, and the generalization hierarchy in Figure 1.4. Suppose also that $k = 4$ and MaxSup= 1. The algorithm first evaluates solutions at height $\lfloor 3/2 \rfloor$, that is, $\langle \mathtt{M_0}, \mathtt{S_1} \rangle$ and $\langle \mathtt{M_1}, \mathtt{S_0} \rangle$. Since both the solutions are 4-anonymous within the MaxSup constraint, the algorithm evaluates solutions at height $\lfloor 3/4 \rfloor$, that is, $\langle \mathtt{M_0}, \mathtt{S_0} \rangle$. Solution $\langle \mathtt{M_0}, \mathtt{S_0} \rangle$ corresponds to the original table that is not 4-anonymous and violates the MaxSup constraint since 4-anonymity requires to suppress the two tuples $\langle$divorced, F$\rangle$. Consequently, the two solutions $\langle \mathtt{M_0}, \mathtt{S_1} \rangle$ and $\langle \mathtt{M_1}, \mathtt{S_0} \rangle$ are considered as minimal.

Bayardo-Agrawal's Algorithm. Bayardo and Agrawal [10] proposed another algorithm for AG_TS, called *k-Optimize*. Given a private table PT, and an ordered set QI=$\{A_1, \ldots, A_n\}$ of quasi-identifying attributes, *k-Optimize* assumes that each attribute $A_i \in$ QI is defined over a totally ordered domain D_i. An attribute generalization of A on D consists in partitioning D into a set of ordered intervals $\{I_1, \ldots, I_m\}$ such that $\bigcup_{i=1}^{m} I_i = D$ and $\forall v_i \in I_i$, $\forall v_j \in I_j$ if $i < j$, then $v_i < v_j$. The approach associates an integer, called *index*, with each interval in any domain of the quasi-identifying attributes. The index assignment reflects the total order relationship over intervals in the domains and among quasi-identifier attributes. As an example, consider the private table in Figure 1.1 where the quasi-identifying attributes are Marital_status and Sex. Suppose that the order between the quasi-identifying attributes is Marital_status followed by Sex, and the order among values inside each attribute domain is married, divorced,

Marital_status			Sex	
⟨[married]	[divorced]	[single]⟩	⟨[F]	[M]⟩
1	2	3	4	5

Figure 1.5. Index assignment to attributes `Marital_status` and `Sex`

`single` for `Marital_status`, and `F`, `M` for `Sex`. Figure 1.5 represents the index assignment obtained when no generalization is applied, that is, when each attribute value represents an interval.

A generalization is represented through the union of generalized sets for each attribute domain. Since the least value from each attribute domain must appear in any valid generalization for the attribute domain, it can be omitted. With respect to our example in Figure 1.5, the least values are 1 (`Marital_status=married`) and 4 (`Sex=F`). As an example, consider now the index list {3, 5}. After adding the least values, we obtain the generalizer sets {1,3} for attribute `Marital_status` and {4, 5} for attribute `Sex`, which in turn correspond to the following intervals of domain values: ⟨[married or divorced], [single]⟩ and ⟨[F], [M]⟩. The empty set { } represents the generalization where, for each domain, all values in the domain are generalized to the most general value. In our example, { } corresponds to index values {1} for `Marital_status` and {4} for `Sex`, which in turn correspond to ⟨[married or divorced or single]⟩ and ⟨[F or M]⟩ generalized domain values.

The k-Optimize algorithm builds a *set enumeration tree* over the set $\mathcal{I}$ of index values, which is a tree representing all possible subsets of $\mathcal{I}$, without duplications. The children of a node n correspond to the sets that can be formed by appending a single element of $\mathcal{I}$ to n, with the restriction that this single element must follow every element already in n according to the total order previously defined. Figure 1.6 illustrates an example of set enumeration tree over $\mathcal{I} = \{1, 2, 3\}$. Since each node in the tree represents how to generalize the original table PT, the visit of the set enumeration tree is equivalent to the evaluation of each possible solution to the k-anonymity problem. At each node n in the tree, the algorithm computes the cost (as determined by some cost metric) associated with the table that can be obtained by applying the generalization represented by n. This cost is then compared against the best cost found until that point. If the cost is lower than the best cost found until that point, it becomes the new best cost and node n is retained. Since a complete visit of the tree may however be impractical (the tree contains $2^{|\mathcal{I}|}$ nodes), k-Optimize proposes an heuristic pruning strategy. Intuitively, a node n can be pruned when the cost associated with its descendants cannot be optimal. To this purpose, the algorithm com-

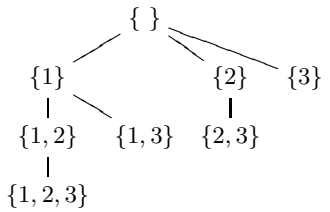

Figure 1.6. An example of set enumeration tree over set $\mathcal{I} = \{1, 2, 3\}$ of indexes

putes a lower bound on the cost that can be obtained by any node in the subtree rooted at n. If this lower bound is greater than the current best cost, node n is pruned. Note that k-Optimize can also be exploited as an heuristic algorithm, by stopping in advance the visit of the tree.

Incognito. Incognito, proposed by LeFevre, DeWitt and Ramakrishnan [18], is an algorithm for AG_TS based on the observation that k-anonymity with respect to any subset of QI is a necessary (not sufficient) condition for k-anonymity with respect to QI. Consequently, given a generalization hierarchy over QI, the generalizations that are not k-anonymous with respect to a subset QI$'$ of QI can be discarded along with all their descendants in the hierarchy.

Exploiting this observation, at each iteration i, for $i = 1, \ldots, |\,\mathsf{QI}\,|$, Incognito builds the generalization hierarchies for all subsets of the quasi-identifying attributes of size i. It then visits each node of the hierarchies discarding the generalizations that do not satisfy k-anonymity with respect to the considered set of attributes. Note that if a node of a generalization hierarchy satisfies k-anonymity, also all its generalizations satisfy k-anonymity and therefore they are not checked in the subsequent visits of the hierarchy. The algorithm then constructs the generalization hierarchies for all subsets of the quasi-identifying attributes of size $i + 1$ by considering only the generalizations of size i that satisfy the k-anonymity constraint at iteration i. Incognito terminates when the whole set of attributes in QI has been considered.

As an example, consider the table PT in Figure 1.1 and suppose that $\mathsf{QI} = \{\texttt{Marital\_status}, \texttt{Sex}\}$ and $k = 12$. The first iteration of Incognito finds that the original table is 12-anonymous with respect to $\texttt{M}_0$, and $\texttt{S}_1$. Note that since PT is 12-anonymous with respect to $\texttt{M}_0$, the table is 12-anonymous also with respect to $\texttt{M}_1$ and $\texttt{M}_2$ and therefore they are not checked. The algorithm then builds the generalization hierarchy on the $\langle \texttt{Marital\_status}, \texttt{Sex} \rangle$ pair by considering only the generalizations $\texttt{M}_0$, $\texttt{M}_1$, $\texttt{M}_2$ and $\texttt{S}_1$ that are 12-anonymous. The algorithm finds that the table is 12-anonymous with respect to $\langle \texttt{M}_0, \texttt{S}_1 \rangle$. Consequently, all

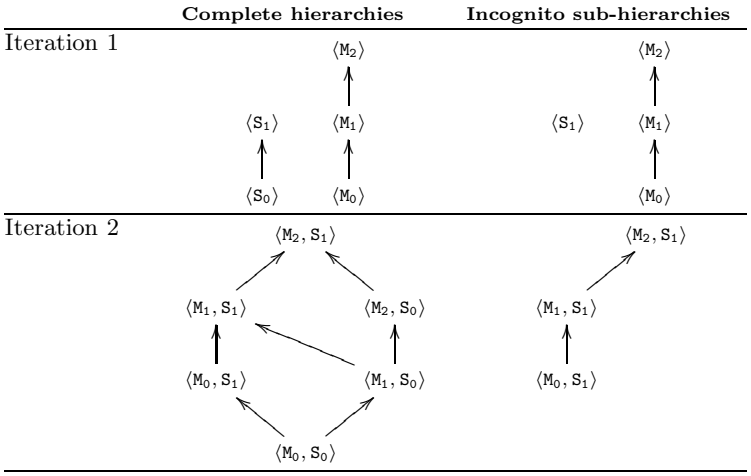

Figure 1.7. Sub-hierarchies computed by Incognito for the table in Figure 1.1

generalizations of $\langle \mathtt{M}_0, \mathtt{S}_1 \rangle$ (i.e., $\langle \mathtt{M}_1, \mathtt{S}_1 \rangle$ and $\langle \mathtt{M}_2, \mathtt{S}_1 \rangle$) are 12-anonymous and the search terminates. Figure 1.7 illustrates on the left-hand side the complete domain generalization hierarchies and on the right-hand side the sub-hierarchies computed by Incognito at each iteration.

Mondrian. The Mondrian algorithm, proposed by LeFevre, DeWitt and Ramakrishnan [19], is based on the *multidimensional global recoding* technique. A private table PT is represented as a set of points in a multidimensional space, where each attribute represents one dimension. To the aim of computing a k-anonymous table, the multidimensional space is partitioned in regions that have to contain at least k points. All points in a given region are then generalized to the same value for QI. Note that tuples in different regions can be generalized in different ways. It is proved that any multidimensional space partition contains at most $2d(k-1)+m$ points, where $d = |\mathsf{QI}|$ and m is the maximum number of tuples with the same quasi-identifier value in PT.

Since the computation of a multidimensional partitioning that minimizes information loss is a NP-hard problem, the authors propose a greedy algorithm that works as follows. Given a space region r, at each iteration the algorithm chooses a dimension d (if such a dimension exists) and splits the region at the median value x of d: all points such that $d > x$ will belong to one of the resulting regions, while all points with $d \leq x$ will belong to the other region. Note that this splitting operation is allowed only if there are more than k points within any region. The algorithm terminates when there are no more splitting op-

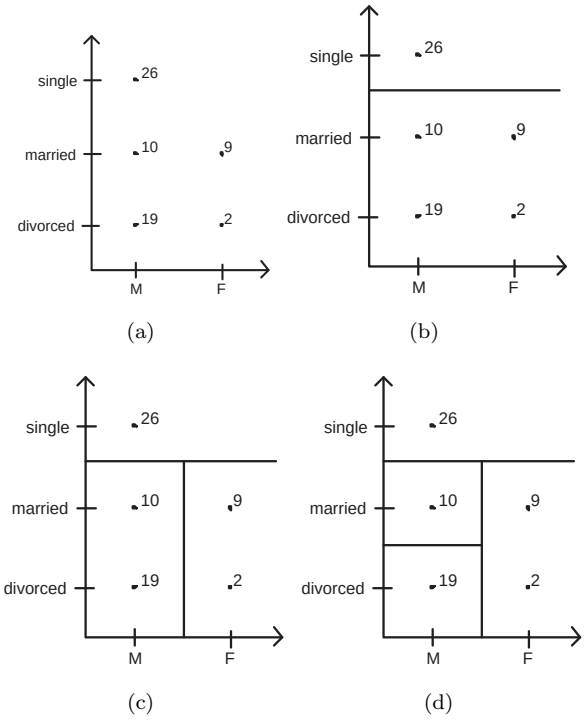

Figure 1.8. Spatial representation (a) and possible partitioning (b)-(d) of the table in Figure 1.1

erations allowed. The tuples within a given region are then generalized to a unique tuple of summary statistics for the considered region. For each quasi-identifying attribute, a summary statistic may simply be a static value (e.g., the average value) or the pair of maximum and minimum values for the attribute in the region. As an example, consider the private table PT in Figure 1.1 and suppose that QI = {`Marital_status`, `Sex`} and $k = 10$. Figure 1.8(a) illustrates the two dimensional representation of the table for the `Marital_status` and `Sex` quasi-identifying attributes, where the number associated with each point corresponds to the occurrences of the quasi-identifier value in PT. Suppose to perform a split operation on the `Marital_status` dimension. The resulting two regions illustrated in Figure 1.8(b) are 10-anonymous. The bottom region can be further partitioned along the `Sex` dimension, as represented in Figure 1.8(c). Another splitting operation along the `Marital_status` dimension can be performed on the region containing the points that

correspond to the quasi-identifying values ⟨`married`, `M`⟩ and ⟨`divorced`, `M`⟩. Figure 1.8(d) illustrates the final solution.

The experimental results [19] show that the Mondrian multidimensional method obtains good solutions for the k-anonymity problem, also compared with k-Optimize and Incognito.

Approximation Algorithms. Since the majority of the exact algorithms proposed in literature have computational time exponential in the number of the attributes composing the quasi-identifier, approximation algorithms have been also proposed. Approximation algorithms for _CS and CG_ have been presented, both for general and specific values of k (e.g., 1.5-approximation[1] for 2-anonymity, and 2-approximation for 3-anonymity [3]).

The first approximation algorithm for _CS was proposed by Meyerson and Williams [22] and guarantees a $O(k \log(k))$-approximation. The best-known approximation algorithm for _CS is described in [2] and guarantees a $O(k)$-approximate solution. The algorithm constructs a complete weighted graph from the original private table PT. Each vertex in the graph corresponds to a tuple in PT, and the edges are weighted with the number of different attribute values between the two tuples represented by extreme vertices. The algorithm then constructs, starting from the graph, a forest composed of trees containing at least k vertices, which represents the clustering for k-anonymization. Some cells in the vertices are suppressed to obtain that all the tuples in the same tree have the same quasi-identifier value. The cost of a vertex is evaluated as the number of cells suppressed, and the cost of a tree is the sum of the weights of its vertices. The cost of the final solution is equal to the sum of the costs of its trees. In constructing the forest, the algorithm limits the maximum number of vertices in a tree to be $3k-3$. Partitions with more than $3k-3$ elements are decomposed, without increasing the total solution cost. The construction of trees with no more than $3k-3$ vertices guarantees a $O(k)$-approximate solution.

An approximation algorithm for CG_ is described in [3] as a direct extension of the approximation algorithm for _CS presented in [2]. For taking into account the generalization hierarchies, each edge has a weight that is computed as follows. Given two tuples i and j and an attribute a, the generalization cost $h_{i,j}(a)$ associated with a is the lowest level of the value generalization hierarchy of a such that tuples i and j have the same generalized value for a. The weight $w(e)$ of the edge $e = (i, j)$ is

[1] In a minimization framework, a p-approximation algorithm guarantees that the cost C of its solution is such that $C/C^* \leq p$, where C^* is the cost of an optimal solution [17].

therefore $w(e) = \Sigma_a h_{i,j}(a)/l_a$, where l_a is the number of levels in the value generalization hierarchy of a. The solution of this algorithm is guaranteed to be a $O(k)$-approximation.

Besides algorithms that compute k-anonymized tables for any value of k, ad-hoc algorithms for specific values of k have also been proposed. For instance, to find better results for Boolean attributes, in the case where $k = 2$ or $k = 3$, an ad-hoc approach has been provided in [3]. The algorithm for $k = 2$ exploits the minimum-weight $[1, 2]$-factor built on the graph constructed for the 2-anonymity. The $[1, 2]$-factor for graph G is a spanning subgraph of G built using only vertices with no more than 2 outgoing edges. Such a subgraph is a vertex-disjoint collection of edges and pairs of adjacent vertices and can be computed in polynomial time. Each component in the subgraph is treated as a cluster, and a 2-anonymized table is obtained by suppressing each cell, for which the vectors in the cluster differ in value. This procedure is a 1.5-approximation algorithm. The approximation algorithm for $k = 3$ is similar and guarantees a 2-approximation solution.

4. *k*-Anonymity Threats from Data Mining

Data mining techniques allow the extraction of information from large collections of data. Data mined information, even if not explicitly including the original data, is built on them and can therefore allow inferences on original data to be withdrawn, possibly putting privacy constraints imposed on the original data at risk. This observation holds also for k-anonymity. The desire to ensure k-anonymity of the data in the collection may therefore require to impose restrictions on the possible output of the data mining process. In this section, we discuss possible threats to k-anonymity that can arise from performing mining on a collection of data maintained in a private table PT subject to k-anonymity constraints.

We discuss the problems for the two main classes of data mining techniques, namely *association rule* mining and *classification* mining.

4.1 Association Rules

The classical association rule mining operates on a set of transactions, each composed of a set of items, and produce association rules of the form $X \rightarrow Y$, where X and Y are sets of items. Intuitively, rule $X \rightarrow Y$ expresses the fact that transactions that contain items X tend to also contain items Y. Each rule has a *support* and a *confidence*, in the form of percentage. The support expresses the percentage of transactions that contain both X and Y, while the confidence expresses the percentage

of transactions, among those containing X, that also contain Y. Since the goal is to find common patterns, typically only those rules that have support and confidence greater than some predefined thresholds are considered of interest [5, 28, 31].

Translating association rule mining over a private table PT on which k-anonymity should be enforced, we consider the values appearing in the table as items, and the tuples reporting respondents' information as transactions. For simplicity, we assume here that the domains of the attributes are disjoint. Also, we assume support and confidence to be expressed in absolute values (in contrast to percentage). The reason for this assumption, which is consistent with the approaches in the literature, is that k-anonymity itself is expressed in terms of absolute numbers. Note, however, that this does not imply that the release itself will be made in terms of absolute values.

Association rule mining over a private table PT allows then the extraction of rules expressing combination of values common to different respondents. For instance, with reference to the private table in Figure 1.1, rule {`divorced`} $\rightarrow$ {`M`} with support 19, and confidence $\frac{19}{21}$ states that 19 tuples in the table refer to divorced males, and among the 21 tuples referring to divorced people 19 of them are male. If the quasi-identifier of table PT contains both attributes `Marital_status` and `Sex`, it is easy to see that such a rule violates any k-anonymity for $k > 19$, since it reflects the existence of 19 respondents who are divorced male (being `Marital_status` and `Sex` included in the quasi-identifier, this implies that no more than 19 indistinguishable tuples can exist for divorced male respondents). Less trivially, the rule above violates also k-anonymity for any $k > 2$, since it reflects the existence of 2 respondents who are divorced and not male; again, being `Marital_status` and `Sex` included in the quasi-identifier, this implies that no more than 2 indistinguishable tuples can exist for non male divorced respondents.

4.2 Classification Mining

In classification mining, a set of database tuples, acting as a training sample, are analyzed to produce a model of the data that can be used as a predictive classification method for classifying new data into classes. Goal of the classification process is to build a model that can be used to further classify tuples being inserted and that represents a descriptive understanding of the table content [25].

One of the most popular classification mining techniques is represented by *decision trees*, defined as follows. Each internal node of a decision tree is associated with an attribute on which the classification

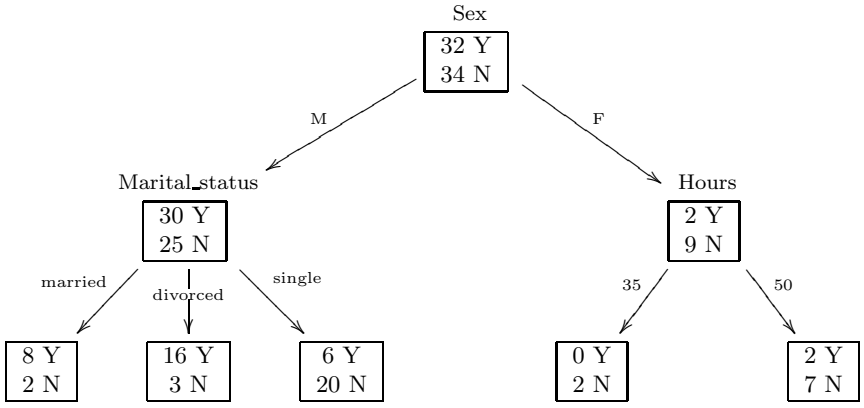

Figure 1.9. An example of decision tree

is defined (excluding the classifying attributes, which in our example is `Hypertension`). Each outgoing edge is associated with a split condition representing how the data in the training sample are partitioned at that tree node. The form of a split condition depends on the type of the attribute. For instance, for a numerical attribute A, the split condition may be of the form $A \leq v$, where v is a possible value for A. Each node contains information about the number of samples at that node and how they are distributed among the different class values.

As an example, the private table PT in Figure 1.1 can be used as a learning set to build a decision tree for predicting if people are likely to suffer from hypertension problems, based on their marital status, if they are male, and on their working hours, if they are female. A possible decision tree for such a case performing the classification based on some values appearing in quasi-identifier attributes is illustrates in Figure 1.9. The quasi-identifier attributes correspond to internal (splitting) nodes in the tree, edges are labeled with (a subset of) attribute values instead of reporting the complete split condition, and nodes simply contain the number of respondents classified by the node values, distinguishing between people suffering (`Y`) and not suffering (`N`) of hypertension.

While the decision tree does not directly release the data of the private table, it indeed allows inferences on them. For instance, Figure 1.9 reports the existence of 2 females working 35 hours (node reachable from path $\langle$`F`,`35`$\rangle$). Again, since `Sex` and `Hours` belong to the quasi-identifier, this information reflects the existence of no more than two respondents for such occurrences of values, thus violating k-anonymity for any $k > 2$. Like for association rules, threats can also be possible by combining classifications given by different nodes along the same path. For instance,

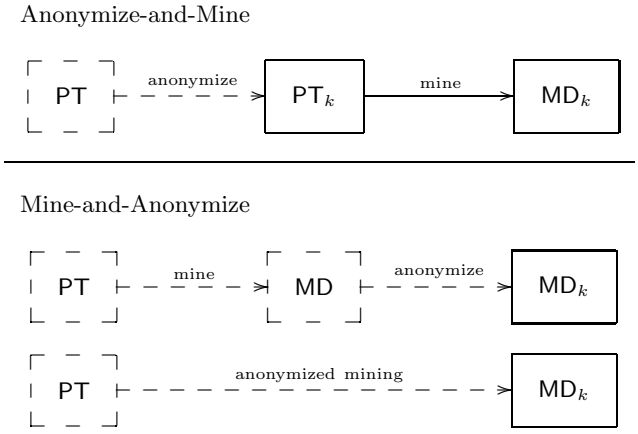

Figure 1.10. Different approaches for combining k-anonymity and data mining

considering the decision tree in Figure 1.9, the combined release of the nodes reachable from paths $\langle \mathtt{F} \rangle$ (with 11 occurrences) and $\langle \mathtt{F}, 50 \rangle$ (with 9 occurrences) allows to infer that there are 2 female respondents in PT who do not work 50 hours per week.

5. *k*-Anonymity in Data Mining

Section 4 has illustrated how data mining results can compromise the k-anonymity of a private table, even if the table itself is not released. Since proper privacy guarantees are a must for enabling information sharing, it is then important to devise solutions ensuring that data mining does not open the door to possible privacy violations. With particular reference to k-anonymity, we must ensure that k-anonymity for the original table PT be not violated.

There are two possible approaches to guarantee k-anonymity in data mining.

- *Anonymize-and-Mine*: anonymize the private table PT and perform mining on its k-anonymous version.
- *Mine-and-Anonymize*: perform mining on the private table PT and anonymize the result. This approach can be performed by executing the two steps independently or in combination.

Figure 1.10 provides a graphical illustration of these approaches, reporting, for the Mine-and-Anonymize approach, the two different cases: one step or two steps. In the figure, boxes represent data, while arcs represent processes producing data from data. The different data boxes

are: PT, the private table; PT_k, an anonymized version of PT; MD, a result of a data mining process (without any consideration of k-anonymity constraints); and MD_k, a result of a data mining process that respects the k-anonymity constraint for the private table PT. Dashed lines for boxes and arcs denote data and processes, respectively, reserved to the data holder, while continuous lines denote data and processes that can be viewed and executed by other parties (as their visibility and execution does not violate the k-anonymity for PT).

Let us then discuss the two approaches more in details and their trade-offs between applicability and efficiency of the process on the one side, and utility of data on the other side.

Anonymize-and-Mine (AM) This approach consists in applying a k-anonymity algorithm on the original private table PT and releasing then a table PT_k that is a k-anonymized version of PT. Data mining is performed, by the data holder or even external parties, on PT_k. The advantage of such an approach is that it allows the decoupling of data protection from mining, giving a double benefit. First, it guarantees that data mining is safe: since data mining is executed on PT_k (and not on PT), by definition the data mining results cannot violate k-anonymity for PT. Second, it allows data mining to be executed by others than the data holder, enabling different data mining processes and different uses of the data. This is convenient, for example, when the data holder may not know a priori how the recipient may analyze and classify the data. Moreover, the recipient may have application-specific data mining algorithms and she may want to directly define parameters (e.g., accuracy and interpretability) and decide the mining method only after examining the data. On the other hand, the possible disadvantages of performing mining on anonymized data is that mining operates on less specialized and complete data, therefore usefulness and significance of the mining results can be compromised. Since classical k-anonymity approaches aim at satisfying k-anonymity minimizing information loss (i.e., minimizing the amount of generalization and suppression adopted), a k-anonymity algorithm may produce a result that is not suited for mining purposes. As a result, classical k-anonymity algorithms may hide information that is highly useful for data mining purposes. Particular care must then be taken in the k-anonymization process to ensure maximal utility of the k-anonymous table PT_k with respect to the goals of the data mining process that has to be executed. In particular, the aim of k-anonymity algorithms operating on data intended for data mining should not be the mere minimization of information loss, but

the optimization of a measure suitable for data mining purposes. A further limitation of the Anonymize-and-Mine approach is that it is not applicable when the input data can be accessed only once (e.g., when the data source is a stream). Also, it may be overall less efficient, since the anonymization process may be quite expensive with respect to the mining one, especially in case of sparse and large databases [1]. Therefore, performing k-anonymity before data mining is likely to be more expensive than doing the contrary.

Mine-and-Anonymize (MA) This approach consists in mining original non-k-anonymous data, performing data mining on the original table PT, and then applying an anonymization process on the data mining result. Data mining can then be performed by the data holder only, and only the sanitized data mining results (MD_k) are released to other parties. The definition of k-anonymity must then be adapted to the output of the data mining phase. Intuitively, no inference should be possible on the mined data allowing violating k-anonymity for the original table PT. This does not mean that the table PT must be k-anonymous, but that if it was not, it should not be known and the effect of its non being k-anonymous be not visible in the mined results. In the Mine-and-Anonymize approach, k-anonymity constraints can be taken into consideration after data mining is complete (*two-step* Mine-and-Anonymize) or within the mining process itself (*one-step* Mine-and-Anonymize). In *two-step* Mine-and-Anonymize the result needs to be sanitized removing from MD all data that would compromise k-anonymity for PT. In *one-step* Mine-and-Anonymize the data mining algorithm needs to be modified so to ensure that only results that would not compromise k-anonymity for PT are computed (MD_k). The two possible implementations (one step vs two steps) provide different trade-offs between applicability and efficiency: *two-step* Mine-and-Anonymize does not require any modification to the mining process and therefore can use any data mining tool available (provided that results are then anonymized); *one-step* Mine-and-Anonymize requires instead to redesign data mining algorithms and tools to directly enforce k-anonymity, combining the two steps can however result in a more efficient process giving then performance advantages. Summarizing, the main drawback of Mine-and-Anonymize is that it requires mining to be executed only by the data holder (or parties authorized to access the private table PT). This may therefore impact applicability. The main advantages are efficiency of the mining process and quality of the results: performing min-

ing before, or together with, anonymization can in fact result more efficient and allow to keep data distortion under control to the goal of maximizing the usefulness of the data.

6. Anonymize-and-Mine

The main objective of classical k-anonymity techniques is the minimization of information loss. Since a private table may have more than one minimal k-anonymous generalization, different preference criteria can be applied in choosing a minimal generalization, such as minimum absolute distance, minimum relative distance, maximum distribution, or minimum suppression [26]. In fact, the strategies behind heuristics for k-anonymization can be typically based on preference criteria or even user policies (e.g., the discourage of the generalization of some given attributes).

In the context of data mining, the main goal is retaining useful information for data mining, while determining a k-anonymization that protects the respondents against linking attacks. However, it is necessary to define k-anonymity algorithms that guarantee data usefulness for subsequent mining operations. A possible solution to this problem is the use of existing k-anonymizing algorithms, choosing the maximization of the usefulness of the data for classification as a preference criteria.

Recently, two approaches that anonymize data before mining have been presented for classification (e.g., decision trees): a top-down [16] and a bottom-up [29] technique. These two techniques aim at releasing a k-anonymous table $T(A_1, \ldots, A_m, \mathit{class})$ for modeling classification of attribute *class* considering the quasi-identifier $\mathsf{QI} = \{A_1, \ldots, A_m\}$. k-anonymity is achieved with cell generalization and cell suppression (CG_), that is, different cells of the same attribute may have values belonging to different generalized domains. The aim of *preserving anonymity for classification* is then to satisfy the k-anonymity constraint while preserving the classification structure in the data.

The top-down approach starts from a table containing the most general values for all attributes and tries to *refine* (i.e., specialize) some values. For instance, the table in Figure 1.11(a) represents a completely generalized table for the table in Figure 1.1. The bottom-up approach starts from a private table and tries to generalize the attributes until the k-anonymity constraint is satisfied.

In the top-down technique a refinement is performed only if it has some suitable properties for guaranteeing both anonymity and good classification. For this purpose, a selection criterion is described for guiding the top-down refinement process to heuristically maximize the classifi-

cation goal. The refinement has two opposite effects: it increases the information of the table for classification and it decreases its anonymity. The algorithm is guided by the functions *InfoGain(v)* and *AnonyLoss(v)* measuring the information gain and the anonymity loss, respectively, where v is the attribute value (cell) candidate for refinement. A good candidate v is such that *InfoGain(v)* is large, and *AnonyLoss(v)* is small. Thus, the selection criterion for choosing the candidate v to be refined maximizes function $Score(v) = \frac{InfoGain(v)}{AnonyLoss(v)+1}$. Function *Score(v)* is computed for each value v of the attributes in the table. The value with the highest score is then specialized to its children in the value generalization hierarchy.

An attribute value v, candidate for specialization, is considered useful to obtain a good classification if the frequencies of the class values are not uniformly distributed for the specialized values of v. The entropy of a value in a table measures the dominance of the majority: the more dominating the majority value in the class is, the smaller the entropy is. *InfoGain(v)* then measures the reduction of entropy after refining v (for a formal definition of *InfoGain(v)* see [16]). A good candidate is a value v that reduces the entropy of the table. For instance, with reference to the private table in Figure 1.1 and its generalized version in Figure 1.11(a), *InfoGain*(`any_marital_status`) is high since for `been_married` we have 14 N and 26 Y, with a difference of 12, and for `never_married` we have 20 N and 6 Y, with a difference of 14 (see Figure 1.11(b)). On the contrary, *InfoGain*($[1, 100)$) is low since for $[0, 40)$ we have 8 Y and 6 N, with a difference of 2, and for $[40, 100)$ we have 24 Y and 28 N, with a difference of 2. Thus `Marital_status` is more useful for classification than `Hours`.

Let us define the anonymity degree of a table as the maximum k for which the table is k-anonymous. The loss of anonymity, defined as *AnonyLoss(v)*, is the difference between the degrees of anonymity of the table before and after refining v. For instance, the degrees of the tables in Figures 1.11(b) and 1.11(c) are 26 (tuples containing: `never_married`, `any_sex`, `[1,100)`) and 19 (tuples containing: `married`, `any_sex`, `[1,100)`), respectively. Since the table in Figure 1.11(c) is obtained by refining the value `been_married` of the table in Figure 1.11(b), *AnonyLoss*(`been_married`) is 7.

The algorithm terminates when any further refinement would violate the k-anonymity constraint.

EXAMPLE 1.1 *Consider the private table in Figure 1.1, and the value generalization hierarchies in Figure 1.2. Let us suppose* QI = {`Marital_status`, `Sex`, `Hours`} *and* $k = 4$. *The algorithm starts from*

Marital_status	*Sex*	*Hours*	*#tuples (Hyp. values)*
any_marital_status	any_sex	[1,100)	66 (32Y, 34N)

(a) Step 1: the most general table

Marital_status	*Sex*	*Hours*	*#tuples (Hyp. values)*
been_married	any_sex	[1,100)	40 (26Y, 14N)
never_married	any_sex	[1,100)	26 (6Y, 20N)

(b) Step 2

Marital_status	*Sex*	*Hours*	*#tuples (Hyp. values)*
divorced	any_sex	[1,100)	21 (16Y, 5N)
married	any_sex	[1,100)	19 (10Y, 9N)
never_married	any_sex	[1,100)	26 (6Y, 20N)

(c) Step 3

Marital_status	*Sex*	*Hours*	*#tuples (Hyp. values)*
divorced	any_sex	35	4 (0Y, 4N)
divorced	any_sex	40	17 (16Y, 1N)
married	any_sex	35	10 (8Y, 2N)
married	any_sex	50	9 (2Y, 7N)
single	any_sex	40	26 (6Y, 20N)

(d) Final table (after 7 steps)

Figure 1.11. An example of top-down anonymization for the private table in Figure 1.1

the most generalized table in Figure 1.11(a), and computes the scores: Score(`any_marital_status`)*, Score*(`any_sex`)*, and Score*([1, 100))*.*

Since the maximum score corresponds to value `any_marital_status`*, this value is refined, producing the table in Figure 1.11(b). The remaining tables computed by the algorithm are shown in Figures 1.11(c), and 1.11(d). Figure 1.11(d) illustrates the final table since the only possible refinement (*`any_sex` *to* `M` *and* `F`*) violates 4-anonymity. Note that the final table is 4-anonymous with respect to* QI = {`Marital_status`, `Sex`, `Hours`}*.*

The bottom-up approach is the dual of the top-down approach. Starting from the private table, the objective of the bottom-up approach is to generalize the values in the table to determine a k-anonymous table preserving good qualities for classification and minimizing information loss.

The effect of generalization is thus measured by a function involving anonymity gain (instead of anonymity loss) and information loss.

Note that, since these methods compute a minimal k-anonymous table suitable for classification with respect to *class* and QI, the computed table PT_k is optimized only if classification is performed using the entire set QI. Otherwise, the obtained table PT_k could be too general. For instance, consider the table in Figure 1.1, the table in Figure 1.11(d) is a 4-anonymization for it considering QI = {`Marital_status, Sex, Hours`}. If classification is to be done with respect to a subset QI′ = {`Marital_status, Sex`} of QI, such a table would be too general. As a matter of fact, a 4-anonymization for PT with respect to QI′ can be obtained from PT by simply generalizing `divorced` and `married` to `been_married`. This latter generalization would generalize only 40 cells, instead of the 66 cells (`M` and `F` to `any_sex`) generalized in the table in Figure 1.11(d).

7. Mine-and-Anonymize

The Mine-and-Anonymize approach performs mining on the original table PT. Anonymity constraints must therefore be enforced with respect to the mined results to be returned. Regardless of whether the approach is executed in one or two steps (see Section 5), the problem to be solved is to translate k-anonymity constraints for PT over the mined results. Intuitively, the mined results should not allow anybody to infer the existence of sets of quasi-identifier values that have less than k occurrences in the private table PT. Let us then discuss what this implies for association rules and for decision trees.

7.1 Enforcing k-Anonymity on Association Rules

To discuss k-anonymity for association rules it is useful to distinguish the two different phases of association rule mining:

1 find all combinations of items whose support (i.e., the number of joint occurrences in the records) is greater than a minimum threshold σ (frequent itemsets mining);

2 use the frequent itemsets to generate the desired rules.

The consideration of these two phases conveniently allows expressing k-anonymity constraints with respect to observable itemsets instead of association rules. Intuitively, k-anonymity for PT is satisfied if the observable itemsets do not allow inferring (the existence of) sets of quasi-identifier values that have less than k occurrences in the private table.

Itemset	*Support*
{∅}	66
{M}	55
{M, 40}	43
{single, M, 40}	26
{divorced}	21
{divorced, M}	19
{married}	19

Figure 1.12. Frequent itemsets extracted from the table in Figure 1.1

It is trivial to see that any itemset X that includes only values on quasi-identifier attributes and with a support lower than k is clearly *unsafe*. In fact, the information given by the itemset corresponds to stating that there are less than k respondents with occurrences of values as in X, thus violating k-anonymity. Besides trivial itemsets such as this, also the combination of itemsets with support greater than or equal to k can breach k-anonymity.

As an example, consider the private table in Figure 1.1, where the quasi-identifier is {`Marital_status`, `Sex`, `Hours`} and suppose 3-anonymity must be guaranteed. All itemsets with support lower than 3 clearly violate the constraint. For instance, itemset {`divorced`, `F`} with support 2, which holds in the table, cannot be released. Figure 1.12 illustrates some examples of itemsets with support greater than or equal to 19 (assuming lower supports are not of interest). While one may think that releasing these itemsets guarantees any k-anonymity for $k \leq 19$, it is not so. Indeed, the combination of the two itemsets {`divorced`, `M`}, with support 19, and {`divorced`}, with support 21, clearly violates it. In fact, from their combination we can infer the existence of two tuples in the private table for which the condition '`Marital_status` = `divorced` $\wedge$ $\neg$(`Sex` = `M`)' is satisfied. Being `Marital_status` and `Sex` included in the quasi-identifier, this implies that no more than 2 indistinguishable tuples can exist for divorced non male respondents, thus violating k-anonymity for $k > 2$. In particular, since `Sex` can assume only two values, the two itemsets above imply the existence of (not released) itemset {`divorced`, `F`} with support 2. Note that, although both itemsets ({`divorced`}, 21) and ({`divorced`, `M`}, 19) cannot be released, there is no reason to suppress both, since each of them individually taken is safe.

The consideration of inferences such as those, and of possible solutions for suppressing itemsets to block the inferences while maximizing the utility of the released information, bring some resembling with

Marital_status	*Sex*	*Hours*	*#tuples*
been_married	M	[1-40)	12
been_married	M	[40-100)	17
been_married	F	[1-40)	2
been_married	F	[40-100)	9
never_married	M	[40-100)	26

(a) PT

A	*B*	*C*	*#tuples*
1	1	0	12
1	1	1	17
1	0	0	2
1	0	1	9
0	1	1	26

(b) T

Figure 1.13. An example of binary table

the primary and secondary suppression operations in statistical data release [12]. It is also important to note that suppression is not the only option that can be applied to sanitize a set of itemsets so that no unsafe inferences violating k-anonymity are possible. Alternative approaches can be investigated, including adapting classical statistical protection strategies [12, 14]. For instance, itemsets can be combined, essentially providing a result that is equivalent to operating on *generalized* (in contrast to specific) data. Another possible approach consists in introducing *noise* in the result, for example, *modifying the support* of itemsets in such a way that their combination never allows inferring itemsets (or patterns of them) with support lower than the specified k.

A first investigation of translating the k-anonymity property of a private table on itemsets has been carried out in [7–9] with reference to private tables where all attributes are defined on binary domains. The identification of unsafe itemsets bases on the concept of *pattern*, which is a boolean formula of items, and on the following observation. Let X and $X \cup \{A_i\}$ be two itemsets. The support of pattern $X \wedge \neg A_i$ can be obtained by subtracting the support of itemset $X \cup \{A_i\}$ from the support of X. By generalizing this observation, we can conclude that given two itemsets $X = \{A_{x_1} \dots A_{x_n}\}$ and $Y = \{A_{x_1} \dots A_{x_n}, A_{y_1} \dots A_{y_m}\}$, with $X \subset Y$, the support of pattern $A_{x_1} \wedge \dots \wedge A_{x_n} \wedge \neg A_{y_1} \wedge \dots \wedge \neg A_{y_m}$ (i.e., the number of tuples in the table containing X but not $Y - X$) can be inferred from the support of X, Y, and all itemsets Z such that $X \subset Z \subset Y$. This observation allows stating that a set of itemsets satisfies k-anonymity only if all itemsets, as well as the patterns derivable from them, have support greater than or equal to k.

As an example, consider the private table PT in Figure 1.13(a), where all attributes can assume two distinct values. This table can be transformed into the binary table T in Figure 1.13(b), where A corresponds to '`Marital_status = been_married`', B corresponds to '`Sex = M`', and C corresponds to '`Hours = [40,100)`'. Figure 1.14 reports the lattice

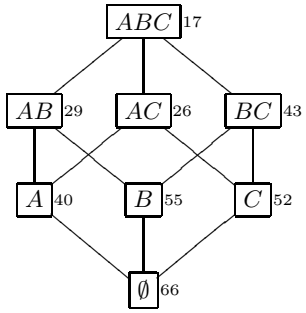

Figure 1.14. Itemsets extracted from the table in Figure 1.13(b)

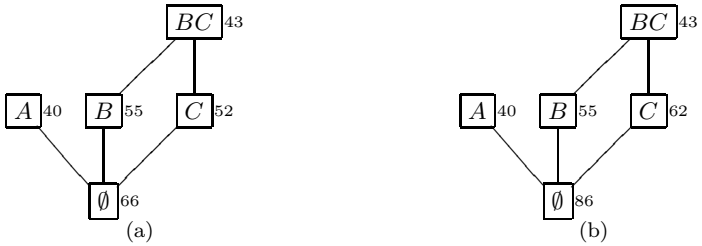

Figure 1.15. Itemsets with support at least equal to 40 (a) and corresponding anonymized itemsets (b)

of all itemsets derivable from T together with their support. Assume that all itemsets with support greater than or equal to the threshold $\sigma = 40$, represented in Figure 1.15(a), are of interest, and that $k = 10$. The itemsets in Figure 1.15(a) present two inference channels. The first inference is obtained through itemsets $X_1 = \{C\}$ with support 52, and $Y_1 = \{BC\}$ with support 43. According to the observation previously mentioned, since $X_1 \subset Y_1$, we can infer that pattern $C \wedge \neg B$ has support $52 - 43 = 9$. The second inference channel is obtained through itemsets $X_2 = \{\emptyset\}$ with support 66, $Y_2 = \{BC\}$ with support 43, and all itemsets Z such that $X_2 \subset Z \subset Y_2$, that is, itemsets $\{B\}$ with support 55, and $\{C\}$ with support 52. The support of pattern $\neg B \wedge \neg C$ can then be obtained by applying again the observation previously mentioned. Indeed, from $\{BC\}$ and $\{B\}$ we infer pattern $B \wedge \neg C$ with support $55 - 43 = 12$, and from $\{BC\}$ and $\{C\}$ we infer pattern $\neg B \wedge C$ with support $52 - 43 = 9$. Since the support of itemset $\{\emptyset\}$ corresponds to the total number of tuples in the binary table, the support of $\neg B \wedge \neg C$ is computed by subtracting the support of $B \wedge \neg C$ (12), $\neg B \wedge C$ (9), and $B \wedge C$ (43) from the support of $\{\emptyset\}$, that is, $66 - 12 - 9 - 43 = 2$. The

result is that release of the itemsets in Figure 1.15(a) would not satisfy k-anonymity for any $k > 2$.

In [9] the authors present an algorithm for detecting inference channels that is based on a classical data mining solution for concisely representing all frequent itemsets (closed itemsets [24]) and on the definition of maximal inference channels. In the same work, the authors propose to block possible inference channels violating k-anonymity by modifying the support of involved itemsets. In particular, an inference channel due to a pair of itemsets $X = \{A_{x_1} \dots A_{x_n}\}$ and $Y = \{A_{x_1} \dots A_{x_n}, A_{y_1} \dots A_{y_m}\}$ is blocked by increasing the support of X by k. In addition, to avoid contradictions among the released itemsets, also the support of all subsets of X is increased by k. For instance, with respect to the previous two inference channels, since k is equal to 10, the support of itemset $\{C\}$ is increased by 10 and the support of $\{\emptyset\}$ is increased by 20, because $\{\emptyset\}$ is involved in the two channels. Figure 1.15(b) illustrates the resulting anonymized itemsets. Another possible strategy for blocking channels consists in decreasing the support of the involved itemsets to zero. Note that this corresponds basically to removing some tuples in the original table.

7.2 Enforcing *k*-Anonymity on Decision Trees

Like for association rules, a decision tree satisfies k-anonymity for the private table PT from which the tree has been built if no information in the tree allows inferring quasi-identifier values that have less than k occurrences in the private table PT. Again, like for association rules, k-anonymity breaches can be caused by individual pieces of information or by combination of apparently anonymous information. In the following, we briefly discuss the problem distinguishing two cases depending on whether the decision tree reports frequencies information for the internal nodes also or for the leaves only.

Let us first consider the case where the tree reports frequencies information for all the nodes in the tree. An example of such a tree is reported in Figure 1.9. With a reasoning similar to that followed for itemsets, given a k, all nodes with a number of occurrences lower than k are *unsafe* as they breach k-anonymity. For instance, the fourth leaf (reachable through path $\langle$`F,35`$\rangle$) is unsafe for any k-anonymity higher than 2. Again, with a reasoning similar to that followed for itemsets, also combinations of nodes that allow inferring patterns of tuples containing quasi-identifying attributes with a number of occurrences lower than k breach k-anonymity for the given k. For instance, nodes corresponding to paths $\langle$`F`$\rangle$ and to $\langle$`F,50`$\rangle$, which taken individually would appear to

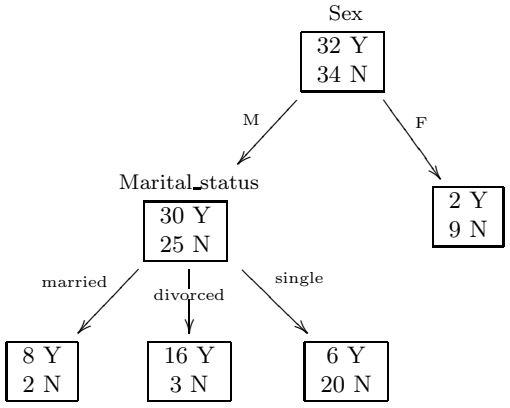

Figure 1.16. 3-anonymous version of the tree of Figure 1.9

satisfy any k-anonymity constraint for $k \leq 9$, considered in combination would violate any k-anonymity for $k > 2$ since their combination allows inferring that there are no more than two tuples in the table referring to females working a number of hours different from 50. It is interesting to draw a relationship between decision trees and itemsets. In particular, any node in the tree corresponds to an itemset dictated by the path to reach the node. For instance, with reference to the tree in Figure 1.9, the nodes correspond to itemsets: {}, {`M`}, {`M,married`}, {`M,divorced`}, {`M,single`}, {`F`}, {`F,35`}, {`F,40`}, {`F,50`}, where the support of each itemset is the sum of the `Y`s and `N`s in the corresponding node. This observation can be exploited for translating approaches for sanitizing itemsets for the sanitization of decision trees (or viceversa). With respect to blocking inference channels, different approaches can be used to anonymize decision trees, including suppression of unsafe nodes as well as other nodes as needed to block combinations breaching anonymity (secondary suppression). To illustrate, suppose that 3-anonymity is to be guaranteed. Figure 1.16 reports a 3-anonymized version of the tree in Figure 1.9. Here, besides suppressing node $\langle$`F,35`$\rangle$, its sibling $\langle$`F,50`$\rangle$ has been suppressed to block the inference channel described above.

Let us now consider the case where the tree reports frequencies information only for the leaf nodes. Again, there is an analogy with the itemset problem with the additional consideration that, in this case, itemsets are such that none of them is a subset of another one. It is therefore quite interesting to note that the set of patterns of tuples identified by the tree nodes directly corresponds to a generalized version of

the private table PT, where some values are suppressed (CG_). This property derives from the fact that, in this case, every tuple in PT satisfies exactly one pattern (path to a leaf). To illustrate, consider the decision tree in Figure 1.17, obtained from the tree in Figure 1.9 by suppressing occurrences in non-leaf nodes. Each leaf in the tree corresponds to a generalized tuple reporting the value given by the path (for attributes appearing in the path). The number of occurrences of such a generalized tuple is reported in the leaf. If a quasi-identifier attribute does not appear along the path, then its value is set to $*$. As a particular case, if every path in the tree contains all the quasi-identifier attributes and puts conditions on specific values, the generalization coincides with the private table PT. For instance, Figure 1.18 reports the table containing tuple patterns that can be derived from the tree in Figure 1.17, and which corresponds to a generalization of the original private table PT in Figure 1.1. The relationship between trees and generalized tables is very important as it allows us to express the protection enjoyed of a decision tree in terms of the generalized table corresponding to it, with the advantage of possibly exploiting classical k-anonymization approaches referred to the private table. In particular, this observation allows us to identify as unsafe *all and only* those nodes corresponding to tuples whose number of occurrences is lower than k. In other words, in this case (unlike for the case where frequencies of internal nodes values are reported) there is no risk that combination of nodes, each with occurrences higher than or equal to k, can breach k-anonymity.

Again, different strategies can be applied to protect decision trees in this case, including exploiting the correspondence just withdrawn, translating on the tree the generalization and suppression operations that could be executed on the private table. To illustrate, consider the tree in Figure 1.17, the corresponding generalized table is in Figure 1.18, which clearly violates any k-anonymity for $k > 2$. Figure 1.19 illustrates a sanitized version of the tree for guaranteeing 11-anonymity obtained by suppressing the splitting node `Hours` and combining nodes ⟨`M,married`⟩ and ⟨`M,divorced`⟩ into a single node. Note how the two operations have a correspondence with reference to the starting table in Figure 1.18 with an attribute generalization over `Hours` and a cell generalization over `Marital_status`, respectively. Figure 1.20 illustrates the table corresponding to the tree in Figure 1.19.

The problem of sanitizing decision trees has been studied in the literature by Friedman et al. [15, 16], who proposed a method for directly building a k-anonymous decision tree from a private table PT. The proposed algorithm is basically an improvement of the classical decision tree building algorithm, combining mining and anonymization in a single pro-

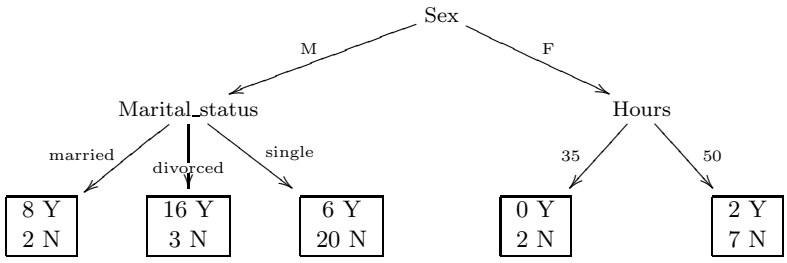

Figure 1.17. Suppression of occurrences in non-leaf nodes in the tree in Figure 1.9

Marital_status	*Sex*	*Hours*	*#tuples (Hyp. values)*
divorced	M	∗	19 (16Y, 3N)
∗	F	35	2 (0Y, 2N)
married	M	∗	10 (8Y, 2N)
∗	F	50	9 (2Y, 7N)
single	M	∗	26 (6Y, 20N)

Figure 1.18. Table inferred from the decision tree in Figure 1.17

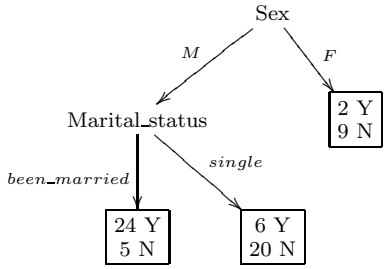

Figure 1.19. 11-anonymous version of the tree in Figure 1.17

cess. At initialization time, the decision tree is composed of a unique root node, representing all the tuples in PT. At each step, the algorithm inserts a new splitting node in the tree, by choosing the attribute in the quasi-identifier that is more useful for classification purposes, and updates the tree accordingly. If the tree obtained is non-k-anonymous, then the node insertion is rolled back. The algorithm stops when no node can be inserted without violating k-anonymity, or when the classification obtained is considered satisfactory.

Marital_status	*Sex*	*Hours*	*#tuples (Hyp. values)*
been_married	M	*	29 (24Y, 5N)
*	F	*	11 (2Y, 9N)
single	M	*	26 (6Y, 20N)

Figure 1.20. Table inferred from the decision tree in Figure 1.19

8. Conclusions

A main challenge in data mining is to enable the legitimate usage and sharing of mined information while at the same time guaranteeing proper protection of the original sensitive data. In this chapter, we have discussed how k-anonymity can be combined with data mining for protecting the identity of the respondents to whom the data being mined refer. We have described the possible threats to k-anonymity that can arise from performing mining on a collection of data and characterized two main approaches to combine k-anonymity in data mining. We have also discussed different methods that can be used for detecting k-anonymity violations and consequently eliminate them in association rule mining and classification mining.

k-anonymous data mining is however a recent research area and many issues are still to be investigated such as: the combination of k-anonymity with other possible data mining techniques; the investigation of new approaches for detecting and blocking k-anonymity violations; and the extension of current approaches to protect the released data mining results against attribute, in contrast to identity, disclosure [21].

Acknowledgments

This work was supported in part by the European Union under contract IST-2002-507591, by the Italian Ministry of Research Fund for Basic Research (FIRB) under project "RBNE05FKZ2", and by the Italian MIUR under project 2006099978.

References

[1] Charu C. Aggarwal. On k-anonymity and the curse of dimensionality. In *Proc. of the 31th VLDB Conference*, Trondheim, Norway, September 2005.

[2] Gagan Aggarwal, Tomas Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, and An Zhu. Anonymizing tables. In *Proc. of the 10th International Conference on Database Theory (ICDT'05)*, Edinburgh, Scotland, January 2005.

[3] Gagan Aggarwal, Tomas Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, and An Zhu. Approximation algorithms for k-anonymity. *Journal of Privacy Technology*, November 2005.

[4] Dakshi Agrawal and Charu C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proc. of the 20th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, Santa Barbara, California, June 2001.

[5] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th VLDB Conference*, Santiago, Chile, September 1994.

[6] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *Proc. of the ACM SIGMOD Conference on Management of Data*, Dallas, Texas, May 2000.

[7] Maurizio Atzori, Francesco Bonchi, Fosca Giannotti, and Dino Pedreschi. Blocking anonymity threats raised by frequent itemset mining. In *Proc. of the 5th IEEE International Conference on Data Mining (ICDM 2005)*, Houston, Texas, November 2005.

[8] Maurizio Atzori, Francesco Bonchi, Fosca Giannotti, and Dino Pedreschi. k-anonymous patterns. In *Proc. of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, Porto, Portugal, October 2005.

[9] Maurizio Atzori, Francesco Bonchi, Fosca Giannotti, and Dino Pedreschi. Anonymity preserving pattern discovery. *VLDB Journal*, November 2006.

[10] Roberto J. Bayardo and Rakesh Agrawal. Data privacy through optimal k-anonymization. In *Proc. of the International Conference on Data Engineering (ICDE'05)*, Tokyo, Japan, April 2005.

[11] Valentina Ciriani, Sabrina De Capitani di Vimercati, Sara Foresti, and Pierangela Samarati. k-anonymity. In T. Yu and S. Jajodia, editors, *Security in Decentralized Data Management*. Springer, Berlin Heidelberg, 2007.

[12] Valentina Ciriani, Sabrina De Capitani di Vimercati, Sara Foresti, and Pierangela Samarati. Microdata protection. In T. Yu and S. Jajodia, editors, *Security in Decentralized Data Management*. Springer, Berlin Heidelberg, 2007.

[13] Alexandre Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, and Johannes Gehrke. Privacy preserving mining of association rules. In *Proc. of the 8th ACM SIGKDD International Conference on*

Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada, July 2002.

[14] Federal Committee on Statistical Methodology. Statistical policy working paper 22, May 1994. Report on Statistical Disclosure Limitation Methodology.

[15] Arik Friedman, Assaf Schuster, and Ran Wolff. Providing k-anonymity in data mining. *VLDB Journal*. Forthcoming.

[16] Benjamin C.M. Fung, Ke Wang, and Philip S. Yu. Anonymizing classification data for privacy preservation. *IEEE Transactions on Knowledge and Data Engineering*, 19(5):711–725, May 2007.

[17] Michael R. Garey and David S. Johnson *Computers and Intractability*. W. H. Freeman & Co., New York, NY, USA, 1979.

[18] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Incognito: efficient full-domain k-anonymity. In *Proc. of the ACM SIGMOD Conference on Management of Data*, Baltimore, Maryland, June 2005.

[19] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Mondrian multidimensional k-anonymity. In *Proc. of the International Conference on Data Engineering (ICDE'06)*, Atlanta, Georgia, April 2006.

[20] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. *Journal of Cryptology*, 15(3):177–206, June 2002.

[21] Ashwin Machanavajjhala, Johannes Gehrke, and Daniel Kifer. ℓ-density: Privacy beyond k-anonymity. In *Proc. of the International Conference on Data Engineering (ICDE'06)*, Atlanta, Georgia, April 2006.

[22] Adam Meyerson and Ryan Williams On the complexity of optimal k-anonymity. In *Proc. of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, Paris, France, June 2004.

[23] Hyoungmin Park and Kyuseok Shim. Approximate algorithms for k-anonymity. In *Proc. of the ACM SIGMOD Conference on Management of Data*, Beijing, China, June 2007.

[24] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *Proc. of the 7th International Conference on Database Theory (ICDT '99)*, Jerusalem, Israel, January 1999.

[25] Rajeev Rastogi and Kyuseok Shim. PUBLIC: A decision tree classifier that integrates building and pruning. In *Proc. of the 24th VLDB Conference*, New York, September 1998.

[26] Pierangela Samarati. Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, November 2001.

[27] Pierangela Samarati and Latanya Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *Proc. of the 17th ACM-SIGMOD-SIGACT-SIGART Symposium on the Principles of Database Systems*, page 188, Seattle, WA, 1998.

[28] Ramakrishnan Srikant and Rakesh Agrawal. Mining generalized association rules. In *Proc. of the 21th VLDB Conference*, Zurich, Switzerland, September 1995.

[29] Ke Wang, Philip S. Yu, and Sourav Chakraborty. Bottom-up generalization: A data mining solution to privacy protection. In *Proc. of the 4th IEEE International Conference on Data Mining (ICDM 2004)*, Brighton, UK, November 2004.

[30] Zhiqiang Yang, Sheng Zhong, and Rebecca N. Wright. Privacy-preserving classification of customer data without loss of accuracy. In *Proc. of the 5th SIAM International Conference on Data Mining*, Newport Beach, California, April 2005.

[31] Mohammed J. Zaki and Ching-Jui Hsiao. Charm: An efficient algorithm for closed itemset mining. In *Proc. of the 2nd SIAM International Conference on Data Mining*, Arlington, Virginia, April 2002.