k-Anonymity

V. Ciriani, S. De Capitani di Vimercati, S. Foresti, and P. Samarati

Università degli Studi di Milano, 26013 Crema, Italia
{ciriani, decapita, foresti, samarati}@dti.unimi.it

To protect respondents' identity when releasing microdata, data holders often remove or encrypt explicit identifiers, such as names and social security numbers. De-identifying data, however, provide no guarantee of anonymity. Released information often contains other data, such as race, birth date, sex, and ZIP code, that can be linked to publicly available information to re-identify respondents and to infer information that was not intended for release.

One of the emerging concept in microdata protection is *k-anonymity*, which has been recently proposed as a property that captures the protection of a microdata table with respect to possible re-identification of the respondents to which the data refer. k-anonymity demands that every tuple in the microdata table released be indistinguishably related to no fewer than k respondents. One of the interesting aspect of k-anonymity is its association with protection techniques that preserve the truthfulness of the data. In this chapter we discuss the concept of k-anonymity, from its original proposal illustrating its enforcement via generalization and suppression. We then survey and discuss research results on k-anonymity in particular with respect to algorithms for its enforcement. We also discuss different ways in which generalization and suppressions can be applied to satisfy k- anonymity and, based on them, introduce a taxonomy of k-anonymity solutions.

1 Introduction

Today's globally networked society places great demand on the dissemination and sharing of information, which is probably becoming the most important and demanded resource. While in the past released information was mostly in tabular and statistical form (*macrodata*), many situations call today for the release of specific data (*microdata*). Microdata, in contrast to macrodata reporting precomputed statistics, provide the convenience of allowing the final recipient to perform on them analysis as needed.

To protect the anonymity of the entities, called *respondents*, to which microdata undergoing public or semipublic release refer, data holders often remove or encrypt explicit identifiers such as names, addresses, and phone numbers. De-identifying data, however, provides no guarantee of anonymity. Released information often contains other data, such as race, birth date, sex, and ZIP code, which can be linked to publicly available information to re-identify (or restrict the uncertainty about) the data respondents, thus leaking information that was not intended for disclosure. The large amount of information easily accessible today, together with the increased computational power available to the attackers, make such linking attacks a serious problem. Indeed, the restricted access to information and its expensive processing, which represented a form of protection in the past, do not hold anymore. Information about us is collected every day, as we join associations or groups, shop for groceries, or execute most of our common daily activities [8, 10]; the amount of privately owned records that describe each citizen's finances, interests, and demographics is increasing every day. Information bureaus such as TRW, Equifax, and Trans Union hold the largest and most detailed databases on American consumers. Most municipalities sell population registers that include the identities of individuals along with basic demographics; examples include local census data, voter lists, city directories, and information from motor vehicle agencies, tax assessors, and real estate agencies. Typical data contained in these databases may include names, social security numbers, birth dates, addresses, telephone numbers, family status, and employment/salary histories. These data, which are often publicly distributed or sold, can be used for linking identities with de-identified information, thus allowing re-identification of respondents. This situation has raised particular concerns in the medical and financial fields, where microdata, which are increasingly released for circulation or research, can be or have been subject to abuses, compromising the privacy of individuals [4, 10, 35].

To illustrate the concept, consider the table in Fig. 1, which exemplifies medical data to be released. In this table, which we refer to as *Private Table* (PT), data have been de-identified by suppressing names and Social Security Numbers (SSNs) so not to explicitly disclose the identities of respondents. However, values of other released attributes, such as `Race`, `Date of birth`, `Sex`, `ZIP` and `Marital status` can also appear in some external table jointly with the individual identity, and can therefore allow them to be tracked. For instance, `ZIP`, `Date of birth`, `Sex`, and `Marital status` can be linked to the Voter List in Fig. 2 to reveal `Name`, `Address`, and `City`. In the private table, for example, there is only one divorced female (`F`) born on `64/04/12` and living in the `94142` area. This combination, if unique in the external world as well, uniquely identifies the corresponding tuple as pertaining to "`Sue J. Doe`, `900 Market Street`, `San Francisco`", thus revealing that she has reported `hypertension`. (Notice that the medical information is not assumed to be publicly associated with individuals, and the desired protection is to release the medical information in a way that the identities of individ-

SSN	Name	Race	Date of birth	Sex	ZIP	Marital status	Disease
		asian	64/04/12	F	94142	divorced	hypertension
		asian	64/09/13	F	94141	divorced	obesity
		asian	64/04/15	F	94139	married	chest pain
		asian	63/03/13	M	94139	married	obesity
		asian	63/03/18	M	94139	married	short breath
		black	64/09/27	F	94138	single	short breath
		black	64/09/27	F	94139	single	obesity
		white	64/09/27	F	94139	single	chest pain
		white	64/09/27	F	94141	widow	short breath

Fig. 1. De-identified private table (medical data)

Name	Address	City	ZIP	DOB	Sex	Status
.....	.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....	.....
Sue J. Doe	900 Market St.	San Francisco	*94142*	*64/04/12*	*F*	*divorced*
.....	.....	.....	.....	.....	.....	.....

Fig. 2. Non de-identified public available table

uals cannot be determined. However, the released characteristics for `Sue J. Doe` leads to determine which medical data among those released are hers.) While this example demonstrates an exact match, in some cases, linking allows one to detect a restricted set of individuals among whom there is the actual data respondent. To avoid the release of de-identified microdata still exposed to linking attacks, different microdata protection techniques can be applied (see chap. "Microdata Protection" for a survey of these different techniques). Among them, there are the commonly used approaches like sampling, swapping values, and adding noise to the data while maintaining some overall statistical properties of the resulting table. However, many uses require release and explicit management of microdata while needing *truthful* information within each tuple. This "data quality" requirement makes inappropriate those techniques that disturb data and therefore, although preserving statistical properties, compromise the correctness of single tuples. *k-anonymity*, together with its enforcement via *generalization* and *suppression*, has been therefore proposed as an approach to protect respondents' identities while releasing truthful information [26].

In this chap. we discuss k-anonymity, starting from its original proposal and surveying then the different algorithms proposed for its enforcement. Also, we will illustrate existing proposals enriching and refining the original definition of k-anonymity. The remainder of this chap. is organized as follows. Section 2 illustrates the basic concepts on k-anonymity and describes the

original k-anonymity definition with attribute generalization and tuple suppression. Section 3 introduces a taxonomy for classifying existing k-anonymity approaches. Section 4 and Sect. 5 describe the algorithms proposed in literature for producing k-anonymous tables. Section 6 briefly presents further studies based on the k-anonymity. Finally, Sect. 7 concludes the chapter.

2 *k*-Anonymity and *k*-Anonymous Tables

The concept of k-anonymity [27] tries to capture, on the private table PT to be released, one of the main requirements that has been followed by the statistical community and by agencies releasing the data, and according to which the released data should be indistinguishably related to no less than a certain number of respondents.

The set of attributes included in the private table, also externally available and therefore exploitable for linking, is called *quasi-identifier*. The requirement just stated is then translated in [26] in the k-anonymity requirement below, which states that every tuple released cannot be related to fewer than k respondents.

Definition 1 (k-anonymity requirement). *Each release of data must be such that every combination of values of quasi-identifiers can be indistinctly matched to at least k respondents.*

Since it seems impossible, or highly impractical and limiting, to make assumptions on the datasets available for linking to external attackers or curious data recipients, essentially k-anonymity takes a safe approach requiring that, in the released table itself, the respondents be indistinguishable (within a given set) with respect to the set of attributes. To guarantee the k-anonymity requirement, k-anonymity requires each quasi-identifier value in the released table to have at least k occurrences, as stated by the following definition.

Definition 2 (k-anonymity). *Let $T(A_1, \ldots, A_m)$ be a table, and QI be a quasi-identifier associated with it. T is said to satisfy* k-anonymity *with respect to QI iff each sequence of values in $T[QI]$ appears at least with k occurrences in $T[QI]$.*[1]

This definition is a sufficient condition for the k-anonymity requirement: a table satisfying Definition 2 for a given k clearly satisfies the k-anonymity requirement for such a k. If a set of attributes of external tables appears in the quasi-identifier associated with the private table PT, and the table satisfies Definition 2, the combination of the released data with the external data will never allow the recipient to associate each released tuple with less than k

[1] $T[QI]$ denotes the projection, maintaining duplicate tuples, of attributes QI in T.

respondents. For instance, with respect to the microdata table in Fig. 1 and the quasi-identifier {`Race, Date of birth, Sex, ZIP, Marital status`}, it easy to see that the table satisfies k-anonymity with $k = 1$ only, since there are single occurrences of values over the considered quasi-identified (e.g., the single occurrence "`asian`, `64/04/12`, `F`, `94142`, `divorced`").

The enforcement of k-anonymity requires the preliminary identification of the *quasi-identifier*. The quasi-identifier depends on the external information available to the recipient, as this determines her linking ability (not all possible external tables are available to every possible data recipient); and different quasi-identifiers can potentially exist for a given table. For the sake of simplicity, the original k-anonymity proposal [26] assumes that private table PT has a single quasi-identifier composed of all attributes in PT that can be externally available and contains at most one tuple for each respondent. Therefore, although the identification of the correct quasi-identifier for a private table can be a difficult task, it is assumed that the quasi-identifier has been properly recognized and defined. For instance, with respect to the microdata table in Fig. 1, a quasi-identifier can be the set of attributes {`Race, Date of birth, Sex, ZIP, Marital status`}.

2.1 Generalization and Suppression

Among the techniques proposed for providing anonymity in the release of microdata, the k-anonymity proposal focuses on two techniques in particular: *generalization* and *suppression*, which, unlike other existing techniques, such as scrambling or swapping, preserve the truthfulness of the information. We have already introduced generalization and suppression in chap. "Microdata Protection". We now illustrate here their specific definition and use in the context of k-anonymity.

Generalization consists in substituting the values of a given attribute with more general values. To this purpose, the notion of *domain* (i.e., the set of values that an attribute can assume) is extended to capture the generalization process by assuming the existence of a set of *generalized domains*. The set of original domains together with their generalizations is referred to as Dom. Each generalized domain contains generalized values and there exists a mapping between each domain and its generalizations. For instance, ZIP codes can be generalized by dropping, at each generalization step, the least significant digit; postal addresses can be generalized to the street (dropping the number), then to the city, to the county, to the state, and so on. This mapping is stated by means of a *generalization relationship* $\leq_D$. Given two domains D_i and $D_j \in$ Dom, $D_i \leq_D D_j$ states that values in domain D_j are generalizations of values in D_i. The generalization relationship $\leq_D$ defines a partial order on the set Dom of domains, and is required to satisfy the following conditions:

C1: $\forall D_i, D_j, D_z \in$ Dom:
$D_i \leq_D D_j, D_i \leq_D D_z \Rightarrow D_j \leq_D D_z \vee D_z \leq_D D_j$

C2: all maximal elements of Dom are singleton.

Condition **C1** states that for each domain D_i, the set of domains generalization of D_i is totally ordered and, therefore, each D_i has at most *one* direct generalization domain D_j. It ensures determinism in the generalization process. Condition **C2** ensures that all values in each domain can always be generalized to a single value. The definition of a generalization relationship implies the existence, for each domain $D \in \mathsf{Dom}$, of a totally ordered hierarchy, called *domain generalization hierarchy*, denoted DGH_D.

A value generalization relationship, denoted $\leq_V$, can also be defined, which associates with each value in domain D_i a unique value in domain D_j, direct generalization of D_i. The value generalization relationship implies the existence, for each domain D, of a *value generalization hierarchy*, denoted VGH_D. It is easy to see that the value generalization hierarchy VGH_D is a *tree*, where the leaves are the values in D and the root (i.e., the most general value) is the value in the maximum element in DGH_D. Figure 3 illustrates an example of domain and value generalization hierarchies for domains: races ($\mathtt{R}_0$); sex ($\mathtt{S}_0$); a subset of the ZIP codes of San Francisco, USA ($\mathtt{Z}_0$); marital status ($\mathtt{M}_0$); and dates of birth ($\mathtt{D}_0$). The generalization relationship specified for ZIP codes generalizes a 5-digit ZIP code, first to a 4-digit ZIP code, and then to a 3-digit ZIP code. The other hierarchies are of immediate interpretation.

Since the approach in [26] works on sets of attributes, the generalization relationship and hierarchies are extended to refer to tuples composed of elements of Dom or of their values. Given a domain tuple $DT = \langle D_1, \ldots, D_n \rangle$ such that $D_i \in \mathsf{Dom}$, $i = 1, \ldots, n$, the domain generalization hierarchy of DT is $\mathsf{DGH}_{DT} = \mathsf{DGH}_{D_1} \times \ldots \times \mathsf{DGH}_{D_n}$, where the Cartesian product is ordered by imposing coordinate-wise order. Since each DGH_{D_i} is totally ordered, DGH_{DT} defines a lattice with DT as its minimal element and the tuple composed of the top of each $\mathsf{DGH}_{D_i}, i = 1, \ldots, n$ as its maximal element. Each path from DT to the unique maximal element of DGH_{DT} defines a possible alternative path, called *generalization strategy*, that can be followed when generalizing a quasi-identifier $QI = \{A_1, \ldots, A_n\}$ of attributes on domains $D_1, \ldots, D_n$. For instance, consider domains $\mathtt{R}_0$ (race) and $\mathtt{Z}_0$ (ZIP code) whose generalization hierarchies are illustrated in Fig. 3 (a) and (c). Figure 4 illustrates the domain generalization hierarchy of the domain tuple $\langle \mathtt{R}_0, \mathtt{Z}_0 \rangle$ together with the corresponding domain and value generalization strategies. There are three different generalization strategies, corresponding to the three paths from the bottom to the top element of lattice $\mathsf{DGH}_{\langle \mathtt{R}_0, \mathtt{Z}_0 \rangle}$. Intuitively, each node of the domain generalization hierarchy corresponds to a generalized table where the attributes in the quasi-identifier have been generalized according the corresponding domain tuple. Figure 5 illustrates all the possible generalized tables corresponding to the different nodes of the domain generalization hierarchy in Fig. 4.

Another method adopted in [26] to be applied in conjunction with generalization to obtain k-anonymity is *tuple suppression*. The intuition behind

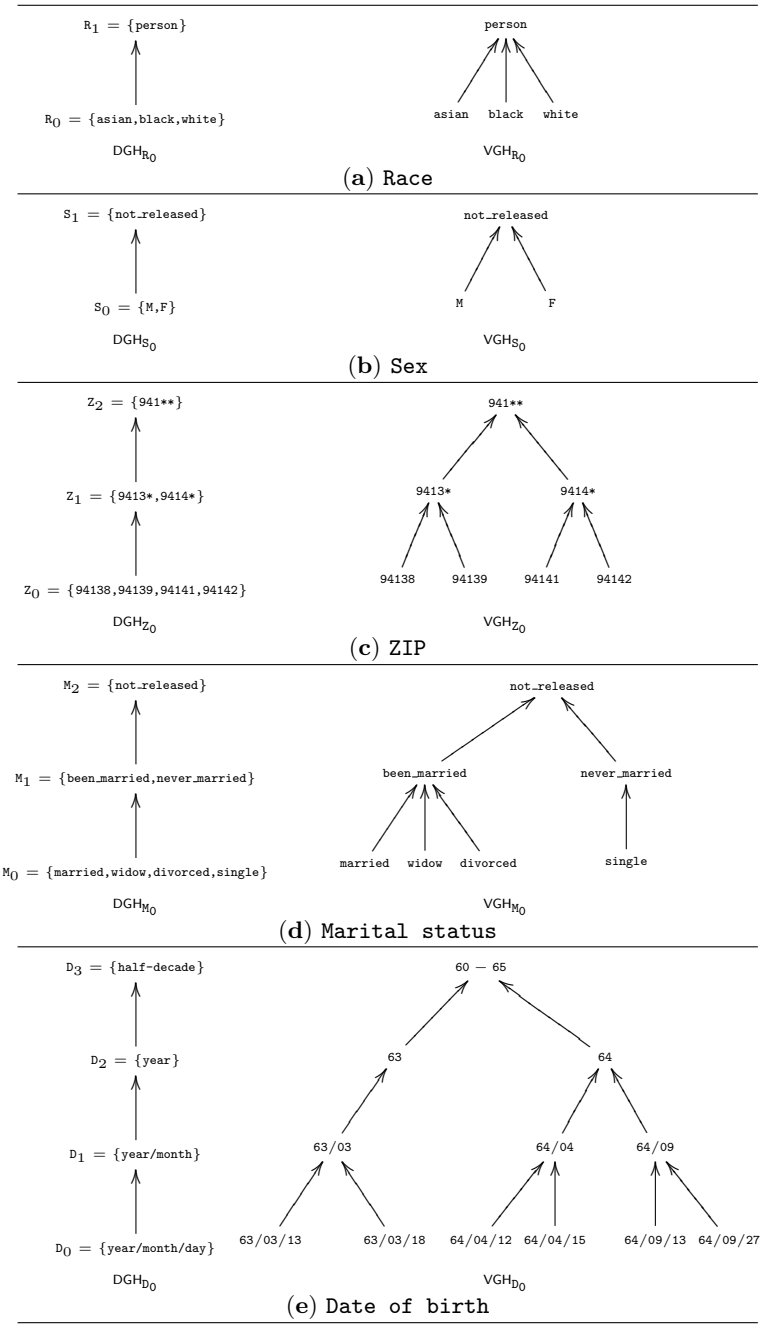

Fig. 3. Examples of generalization hierarchies

the introduction of suppression is that this additional method can reduce the amount of generalization necessary to satisfy the k-anonymity constraint. Suppression is therefore used to "moderate" the generalization process when a limited number of outliers (i.e., tuples with less than k occurrences) would force a great amount of generalization. For instance, consider the generalized tables in Fig. 5. The tuples in *italic* are those that would need to be suppressed in each generalized table to satisfy 2-anonymity without further generalization.

2.2 k-Minimal Generalization (with Suppression)

The application of generalization and suppression to a private table PT produces more general (less precise) and less complete (if some tuples are suppressed) tables that provide better protection of the respondents' identities. Generalized tables are then defined as follows.

Definition 3 (Generalized table - with suppression). *Let T_i and T_j be two tables defined on the same set of attributes. Table T_j is said to be a* generalization (with tuple suppression) *of table T_i, denoted $T_i \preceq T_j$, if:*

1. $|T_j| \leq |T_i|$;
2. *the domain $dom(A, T_j)$ of each attribute A in T_j is equal to, or a generalization of, the domain $dom(A, T_i)$ of attribute A in T_i;*
3. *it is possible to define an injective function associating each tuple t_j in T_j with a tuple t_i in T_i, such that the value of each attribute in t_j is equal to, or a generalization of, the value of the corresponding attribute in t_i.*

Given a private table PT, many tables obtained generalizing attributes and suppressing tuples in PT satisfy k-anonymity, but some of them are either too general or are obtained suppressing too much tuples. The goal is therefore to compute a table maintaining as much information as possible, under the k-anonymity constraint; in other words minimality of the solution should be guaranteed. The definition of *k-minimal generalization with suppression* is based on the concept of *distance vector*.

Definition 4 (Distance vector). *Let $T_i(A_1, \ldots, A_n)$ and $T_j(A_1, \ldots, A_n)$ be two tables such that $T_i \preceq T_j$. The distance vector of T_j from T_i is the vector $DV_{i,j} = [d_1, \ldots, d_n]$, where each d_z, $z = 1, \ldots, n$, is the length of the* unique *path between $dom(A_z, T_i)$ and $dom(A_z, T_j)$ in the domain generalization hierarchy DGH_{D_z}.*

It is possible to define a partial order relation between distance vectors, that is, $DV = [d_1, \ldots, d_n] \leq DV' = [d'_1, \ldots, d'_n]$ iff $d_i \leq d'_i$, $i = 1 \ldots n$. On the basis of the distance vector order relation, it is possible to build a *hierarchy of distance vectors*, which can be graphically represented as a lattice. Figure 6 illustrates the domain generalization hierarchy $\mathsf{DGH}_{\langle \mathtt{R}_0, \mathtt{Z}_0 \rangle}$ together with the corresponding hierarchy of distance vectors.

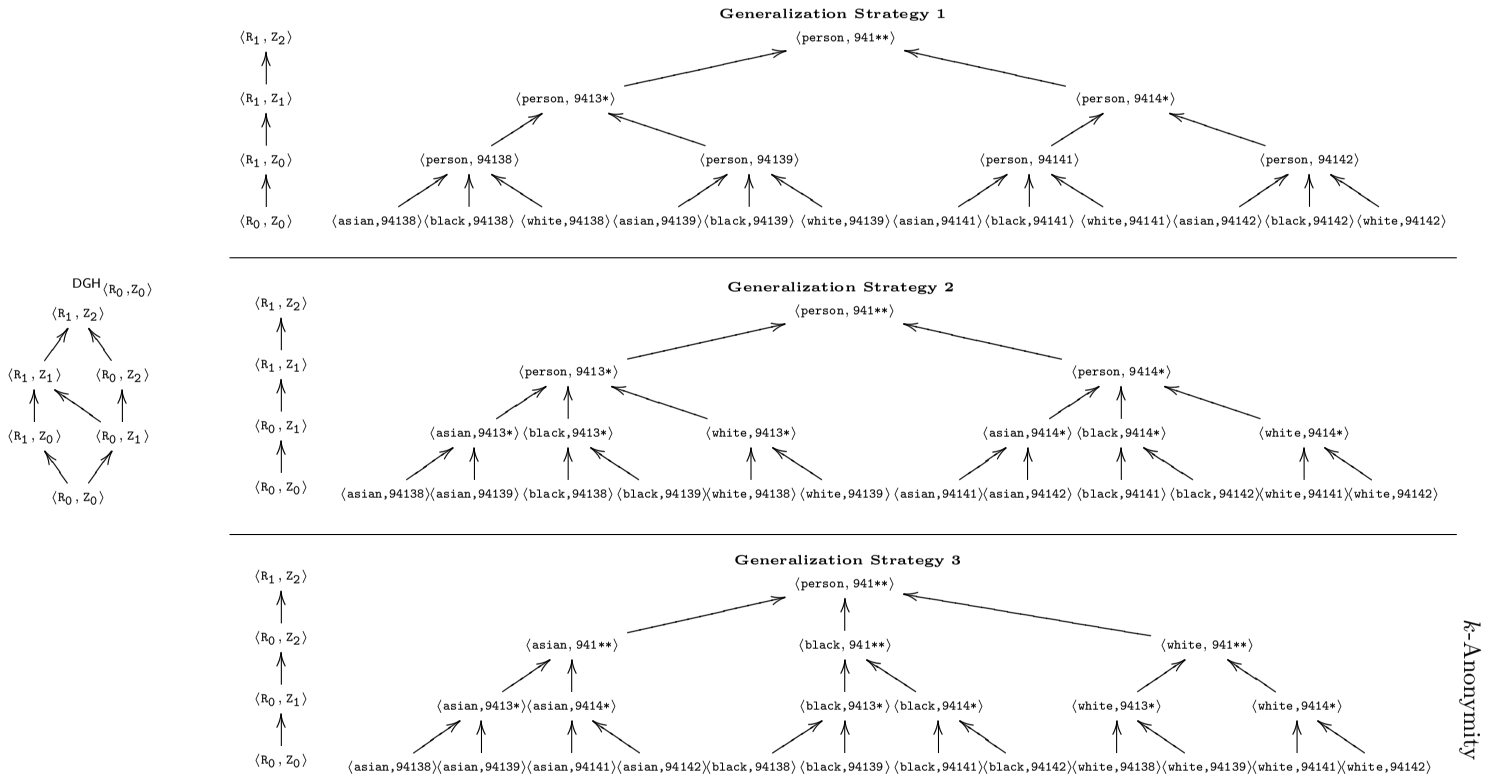

Fig. 4. Hierarchy $\mathsf{DGH}_{\langle R_0, Z_0\rangle}$ and corresponding Domain and Value Generalization Strategies

Race:R_0	**ZIP:**Z_0
asian	*94142*
asian	*94141*
asian	94139
asian	94139
asian	94139
black	*94138*
black	*94139*
white	*94139*
white	*94141*

(**a**)

Race:R_1	**ZIP:**Z_0
person	*94142*
person	94141
person	94139
person	94139
person	94139
person	*94138*
person	94139
person	94139
person	94141

(**b**)

Race:R_0	**ZIP:**Z_1
asian	9414*
asian	9414*
asian	9413*
asian	9413*
asian	9413*
black	9413*
black	9413*
white	*9413**
white	*9414**

(**c**)

Race:R_1	**ZIP:**Z_1
person	9414*
person	9414*
person	9413*
person	9413*
person	9413*
person	9413*
person	9413*
person	9413*
person	9414*

(**d**)

Race:R_0	**ZIP:**Z_2
asian	941**
asian	941**
asian	941**
asian	941**
asian	941**
black	941**
black	941**
white	941**
white	941**

(**e**)

Race:R_1	**ZIP:**Z_2
person	941**
person	941**
person	941**
person	941**
person	941**
person	941**
person	941**
person	941**
person	941**

(**f**)

Fig. 5. An example of a private table PT (**a**) and its generalizations

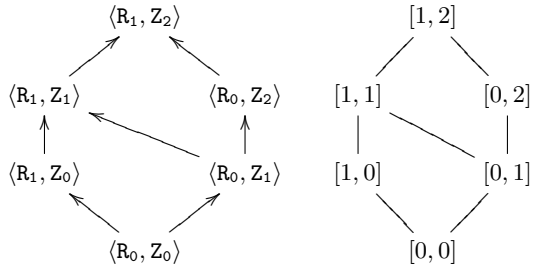

Fig. 6. Hierarchy $\mathsf{DGH}_{\langle R_0, Z_0\rangle}$ and corresponding hierarchy of distance vectors

Note that like for generalization, it is possible to adopt different suppression solutions for guaranteeing k-anonymity without removing more tuples than necessary (i.e., ensuring minimality of the suppression), at a given level of generalization. The joint use of generalization and suppression helps in maintaining as much information as possible in the process of k-anonymization. The question is whether it is better to generalize, loosing data precision, or to suppress, loosing completeness. Samarati in [26] assumes that the data holder

Race:R_0	**ZIP:**Z_0
asian	94142
asian	94141
asian	94139
asian	94139
asian	94139
black	94138
black	94139
white	94139
white	94141

PT

Race:R_1	**ZIP:**Z_0
person	94141
person	94139
person	94139
person	94139
person	94139
person	94139
person	94141

$GT_{[1,0]}$

Race:R_0	**ZIP:**Z_1
asian	9414*
asian	9414*
asian	9413*
asian	9413*
asian	9413*
black	9413*
black	9413*

$GT_{[0,1]}$

Fig. 7. A private table PT and its 2-minimal generalizations, assuming MaxSup=2

establishes a threshold, denoted MaxSup, specifying the maximum number of tuples that can be suppressed. The concept of *k-minimal generalization with suppression* is then formally defined as follows.

Definition 5 (k-minimal generalization - with suppression). *Let T_i and T_j be two tables such that $T_i \preceq T_j$, and let* MaxSup *be the specified threshold of acceptable suppression. T_j is said to be a* k-minimal *generalization of table T_i iff:*

1. *T_j satisfies k-anonymity enforcing minimal required suppression, that is, T_j satisfies k-anonymity and $\forall T_z : T_i \preceq T_z, DV_{i,z} = DV_{i,j}$, T_z satisfies k-anonymity $\Rightarrow |T_j| \geq |T_z|$*
2. *$|T_i| - |T_j| \leq$ MaxSup*
3. *$\forall T_z : T_i \preceq T_z$ and T_z satisfies conditions 1 and 2 $\Rightarrow \neg(DV_{i,z} < DV_{i,j})$.*

Intuitively, this definition states that a generalization T_j is k-minimal iff it satisfies k-anonymity, it does not enforce more suppression than it is allowed ($|T_i| - |T_j| \leq$ MaxSup), and there does not exist another generalization satisfying these conditions with a distance vector smaller than that of T_j.

Consider the private table in Fig. 1 and suppose that MaxSup = 2, QI = {Race, ZIP}, and $k = 2$. There are two k-minimal generalizations with suppression for it, namely $GT_{[0,1]}$ and $GT_{[1,0]}$ (see Fig. 7). These two tables are obtained from the tables in Figs. 5(b)-(c) by removing the outlier tuples, which are those written in italic. Note that $GT_{[1,1]}$, $GT_{[0,2]}$, and $GT_{[1,2]}$ (corresponding to tables in Figs. 5(d)-(e)-(f), respectively) are not k-minimal, since they do not satisfy condition 3 in Definition 5. $GT_{[0,0]}$ (corresponding to the table in Fig. 5(a)), which contains the original values with the italic tuples removed is not a k-minimal generalization with suppression as it does not satisfy condition 2 in Definition 5.

A private table may have more than one minimal generalization satisfying a k-anonymity constraint for a suppression threshold (e.g., in the previous example there are two minimal generalizations, $GT_{[1,0]}$ and $GT_{[0,1]}$). This is

completely legitimate, since the definition of "minimal" only captures the concept that the least amount of generalization and suppression necessary to achieve k-anonymity is enforced. Different *preference criteria* can be applied in choosing a preferred minimal generalization, among which [26]:

- *minimum absolute distance* prefers the generalization(s) with the smallest absolute distance, that is, with the smallest total number of generalization steps (regardless of the hierarchies on which they have been taken);
- *minimum relative distance* prefers the generalization(s) with the smallest relative distance, that is, that minimizes the total number of relative steps (a step is made relative by dividing it over the height of the domain hierarchy to which it refers);
- *maximum distribution* prefers the generalization(s) with the greatest number of distinct tuples;
- *minimum suppression* prefers the generalization(s) that suppresses less tuples, that is, the one with the greatest cardinality.

3 Classification of k-Anonymity Techniques

The original k-anonymity proposal just illustrated [26] considers the application of generalization at the attribute (column) level and suppression at the tuple (row) level. However, both generalization and suppression can also be applied, and have been investigated, at a finer granularity level. Before proceeding illustrating the different approaches to provide k-anonymity, we discuss the different ways in which generalization and suppression can be applied, and introduce the different models for k-anonymity.

	Suppression			
Generalization	*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

Fig. 8. Classification of k-anonymity techniques

Generalization can be applied at the level of:

- *Attribute (AG)*: generalization is performed at the level of column; a generalization step generalizes all the values in the column.
- *Cell (CG)*: generalization is performed on single cells; as a result a generalized table may contain, for a specific column, values at different generalization levels. For instance, in the `Date of birth` column

some cells can report the specific day (no generalization), others the month (one step of generalization), others the year (two steps of generalization), and so on. Generalizing at the cell level has the advantage of allowing the release of more specific values (as generalization can be confined to specific cells rather than hitting whole columns). However, besides a higher complexity of the problem, a possible drawback in the application of generalization at the cell level is the complication arising from the management of values at different generalization levels within the same column.

Suppression can be applied at the level of:

- *Tuple (TS)*: suppression is performed at the level of row; a suppression operation removes a whole tuple.
- *Attribute (AS)*: suppression is performed at the level of column, a suppression operation obscures all the values of a column.
- *Cell (CS)*: suppression is performed at the level of single cells; as a result a k-anonymized table may wipe out only certain cells of a given tuple/attribute.

The possible combinations of the different choices for generalization and suppression (including also the choice of not applying one of the two techniques) result in different models for k-anonymity, which can represent a taxonomy for classifying the different k-anonymity proposals. Different models bear different complexity and define in different ways the concept of minimality of the solutions.

A first attempt to introduce a taxonomy for classifying k-anonymity approaches has been described in [20], where the authors distinguish between the application of suppression and generalization at the cell or attribute level. Our taxonomy refines and completes this classification. Below we discuss the different models resulting from our classification, characterize them, and classify existing approaches accordingly. We refer to each model with a pair (separated by _), where the first element describes the level of generalization (AG, CG, or none) and the second element describes the level of suppression(TS, AS, CS, or none). Table in Fig. 8 summarizes these models.

AG_TS Generalization is applied at the level of attribute (column) and suppression at the level of tuple (row). This is the assumption considered in the original model [26], as well as in most of the subsequent approaches providing efficient algorithms for solving the k-anonymity problem [5, 18, 20, 29, 33], since it enjoys a tradeoff between the computational complexity and the quality of the anonymized table.

AG_AS Both generalization and suppression are applied at the level of column. No specific approach has investigated this model. It must also be noted that if attribute generalization is applied, attribute suppression is not needed; since suppressing an attribute (i.e., not releasing any of its values) to reach k-anonymity can equivalently be modeled via a generalization of all the attribute values to the maximal element in the value

hierarchy. This model is then equivalent to model **AG_** (attribute generalization, no suppression). Note that this observation holds assuming that attribute suppression removes only the values and not the column itself (this assumption seems reasonable since removal of the column is not needed for k-anonymity).

AG_CS Generalization is applied at the level of column, while suppression at the level of cell. It allows to reduce the effect of suppression, at the price however of a higher complexity of the problem. No specific investigation of this model has been performed with reference to k-anonymity. We note, however, that this approach has been investigated in earlier work by the μ-argus [11, 16, 17] and Datafly [28] software, which applied the same principles behind k-anonymity, but without guarantees on the minimality of the solutions (which is instead a basic principle behind k-anonymity).

AG_ Generalization is applied at the level of column, suppression is not considered. As noted above, it is equivalent to model **AG_AS**. Note also that both, **AG_AS** and **AG_**, are subsumed by model **AG_TS**, which reduces to them in the case where the suppression threshold MaxSup is set to zero.

CG_CS Both generalization and suppression are applied at the cell level. Then, for a given attribute we can have values at different levels of generalization. By observations similar to those illustrated for **AG_AS**, this model is equivalent to **CG_** (cell generalization, no suppression). Indeed, suppression of a cell can be equivalently modeled as the generalization of the cell at the maximal element of the value hierarchy.

CG_ Generalization is applied at the level of cell, suppression is not considered [3]. As just noted, it is equivalent to **CG_CS**.

_TS Suppression is applied at the tuple level, generalization is not allowed. No approach has investigated this model, which however can be modeled as a reduction of **AG_TS** to the case where all the generalization hierarchies have height zero (i.e., no hierarchy is defined). It is interesting to note that in this case the computational complexity of the problem of finding a k-anonymous table becomes polynomial (as solving it requires simply to delete from the original table all the outliers), and the minimal solution is unique. The application of tuple suppression alone has however limited applicability.

AS Suppression is applied at the attribute level, generalization is not allowed. No explicit approach has investigated this model. We note, however, that it can be modeled as a reduction of **AG** where all the generalization hierarchies have height of 1.

CS Suppression is applied at the cell level, generalization is not allowed [2, 24]. Again, it can be modeled as a reduction of **AG** where all the generalization hierarchies have height of 1.

In addition to these models, we have the obvious uninteresting combination _ (no generalization, no suppression) and two models, which are not applicable, namely: **CG_TS** (cell generalization, tuple suppression) and **CG_AS** (cell

generalization, attribute suppression). The reason for their non applicability is that since generalizing a value at the maximum element in the value hierarchy is equivalent to suppressing it, supporting generalization at the fine grain of cell clearly implies the ability of enforcing suppression at that level too.

Note that, because of the equivalence relationships pointed out in the discussion above, there are essentially seven possible models. For equivalent models, in the following we use **AG_** to indistinguishably refer to **AG_** and **AG_AS**, and **CG_** to indistinguishably refer to **CG_** and **CG_CS**. Fig. 9 illustrates an example of a private table (Fig. 9(a)) and a possible 2-anonymized version of it according to these different models.

Among these seven models: **_TS** is, as noted, straightforward and not that interesting; **AG_CS** has not been formally investigated; while for **_AS** only the complexity has been studied but no solution has been proposed. By contrast, **AG_TS**, **AG_**, **CG_**, and **_CS** have been extensively studied and algorithms for their enforcement have been proposed; we will then illustrate them in the remainder of the chapter.

Before illustrating the different proposals, it is interesting to note the complexity of the problem. All the models investigated in the literature (**AG_TS**, **AG_**, **CG_**, and **_CS**), as well as **_AS**, are NP-hard. NP-hardness has been proved for **_CS** and **_AS** [2, 3, 24]. Suppose that the private table consists of n m-dimensional vectors (i.e., tuples) $x_1, \ldots, x_n \in \Sigma^m$, where Σ is an alphabet. The NP-hardness of **_AS** has been proved in [24] for $|\Sigma| \geq 2$, by a reduction from the "k-dimensional Perfect Matching" problem. Furthermore, the NP-hardness of the **_CS** problem for $|\Sigma| \geq 3$ has been proved in [3] with a reduction from the NP-hard problem of "Edge Partition into Triangles". The last result is an improvement upon the NP-hardness prove in [24] for the **_CS** problem, which requires an alphabet of size n.

NP-hardness of **_CS** and **_AS** clearly implies NP-hardness of **CG_** and **AG_**, respectively. This implication holds since suppression can be considered as a special case of generalization where all hierarchies have height of 1. Note also that NP-hardness of **AG_** implies NP-hardness of **AG_TS**, where, as in the existing proposals, tuple suppression is regulated with the specification of a maximum number of tuples (MaxSup) that can be suppressed.

It is interesting to note that, instead the decisional versions of **AS_**, **CS_**, **AG_**, **AG_TS**, and **CG_** are in NP [3].

4 Algorithms for AG_TS and AG_

The problem of finding minimal k-anonymous tables, with attribute generalization and tuple suppression, is computationally hard. Consistently with this, the majority of the exact algorithms proposed in literature have computational time exponential in the number of the attributes composing the quasi-identifier. However, when the number $|QI|$ of attributes in the quasi-identifier is small compared with the number n of tuples in the private table

Race	DOB	Sex	ZIP
asian	64/04/12	F	94142
asian	64/09/13	F	94141
asian	64/04/15	F	94139
asian	63/03/13	M	94139
asian	63/03/18	M	94139
black	64/09/27	F	94138
black	64/09/27	F	94139
white	64/09/27	F	94139
white	64/09/27	F	94141

(**a**) PT

Race	DOB	Sex	ZIP
asian	64/04	F	941**
asian	64/04	F	941**
asian	63/03	M	941**
asian	63/03	M	941**
black	64/09	F	941**
black	64/09	F	941**
white	64/09	F	941**
white	64/09	F	941**

(**b**) **AG_TS**

Race	DOB	Sex	ZIP
asian	*	F	*
asian	*	F	*
asian	*	F	*
asian	63/03	M	9413*
asian	63/03	M	9413*
black	64/09	F	9413*
black	64/09	F	9413*
white	64/09	F	*
white	64/09	F	*

(**c**) **AG_CS**

Race	DOB	Sex	ZIP
asian	64	F	941**
asian	64	F	941**
asian	64	F	941**
asian	63	M	941**
asian	63	M	941**
black	64	F	941**
black	64	F	941**
white	64	F	941**
white	64	F	941**

(**d**) **AG_≡AG_AS**

Race	DOB	Sex	ZIP
asian	64	F	941**
asian	64	F	941**
asian	64	F	941**
asian	63/03	M	94139
asian	63/03	M	94139
black	64/09/27	F	9413*
black	64/09/27	F	9413*
white	64/09/27	F	941**
white	64/09/27	F	941**

(**e**) **CG_≡CG_CS**

Race	DOB	Sex	ZIP

(**f**) **_TS**

Race	DOB	Sex	ZIP
asian	*	F	*
asian	*	F	*
asian	*	F	*
asian	*	M	*
asian	*	M	*
black	*	F	*
black	*	F	*
white	*	F	*
white	*	F	*

(**g**) **_AS**

Race	DOB	Sex	ZIP
asian	*	F	*
asian	*	F	*
asian	*	F	*
asian	*	M	94139
asian	*	M	94139
*	64/09/27	F	*
*	64/09/27	F	94139
*	64/09/27	F	94139
*	64/09/27	F	*

(**h**) **_CS**

Fig. 9. A private table (**a**) and some 2-anonymized version of according to different models

Algorithm	Model	Algorithm's type	Time complexity
Samarati [26]	AG_TS	Exact	exponential in $\|QI\|$
Sweeney [29]	AG_TS	Exact	exponential in $\|QI\|$
Bayardo-Agrawal [5]	AG_TS	Exact	exponential in $\|QI\|$
LeFevre-et-al. [20]	AG_TS	Exact	exponential in $\|QI\|$
Aggarwal-et-al. [2]	_CS	$O(k)$-Approximation	$O(kn^2)$
Meyerson-Williams [24][2]	_CS	$O(k \log k)$-Approximation	$O(n^{2k})$
Aggarwal-et-al. [3]	CG_	$O(k)$-Approximation	$O(kn^2)$
Iyengar [18]	AG_TS	Heuristic	limited number of iterations
Winkler [33]	AG_TS	Heuristic	limited number of iterations
Fung-Wang-Yu [12]	AG_	Heuristic	limited number of iterations

Fig. 10. Some approaches to k-anonymity (n is the number of tuples in PT)

PT, these exact algorithms with attribute generalization and tuple suppression are practical. In particular, when $|QI| \in O(\log n)$, these exact algorithms have computational time polynomial in the number of tuples of PT, provided that the threshold on the number of suppressed tuples (MaxSup) is constant in value.

Recently many exact algorithms for producing k-anonymous tables through attribute generalization and tuple suppression have been proposed [5, 20, 26, 29]. Samarati [26] presented an algorithm that exploits a binary search on the domain generalization hierarchy to avoid an exhaustive visit of the whole generalization space. Bayardo and Agrawal [5] presented an optimal algorithm that starts from a fully generalized table (with all tuples equal) and specializes the dataset in a minimal k-anonymous table, exploiting ad-hoc pruning techniques. Finally, LeFevre, DeWitt, and Ramakrishnan [20] described an algorithm that uses a bottom-up technique and a priori computation. Sweeney [29] proposed an algorithm that exhaustively examines all potential generalizations for identifying a minimal one satisfying the k-anonymity requirement. This latter approach is clearly impractical for large datasets, and we will therefore not discuss it further. We will now describe these approaches in more details.

4.1 Samarati's Algorithm

The first algorithm for guaranteeing k-anonymity was proposed in conjunction with the definition of k-anonymity in [26]. The algorithm exploits both generalization and tuple suppression over quasi-identifier attributes and computes a k-minimal solution according to the minimum absolute distance preference criteria (see Sect. 2). Since the k-anonymity definition is based on a quasi-identifier, the algorithm works only on this set of attributes and on tables with more than k tuples (this last constraint being clearly a necessary condition for a table to satisfy k-anonymity).

[2] Meyerson and Williams have also described in [24] a $O(k \log |QI|)$-approximation algorithm with polynomial time complexity ($O(|QI|n^3)$) for the **_CS** model.

As described in Sect. 2, given a domain generalization hierarchy, there are different paths from the bottom element of the hierarchy and the hierarchy's root. Each path corresponds to a different strategy according to which the original private table PT can be generalized (see, for instance, Fig. 4). Along each path there is exactly one *locally minimal* generalization, that is, a table satisfying k-anonymity and maintaining as much information as possible. The locally minimal generalization is the lowest node in the path satisfying k-anonymity. Each k-minimal generalization is locally minimal with respect to a path; the converse is not true, that is, a locally minimal generalization with respect to a given path might not be a k-minimal generalization. A naive approach to compute a k-minimal generalization would then consist in following each generalization strategy (path) in the domain generalization hierarchy stopping the process at the first generalization that satisfies k-anonymity, within the MaxSup constraint. Once all paths have been evaluated, at least one of the locally minimal generalizations is also a k-minimal generalization with suppression and can be chosen according to the preference criteria mentioned in Sect. 2. Given the high number of paths that should be followed, this naive approach is not practically applicable.

The key idea exploited in [26] to cut down the computation is the observation that going up in the hierarchy the number of tuples that must be removed to guarantee k-anonymity decreases. Each node in the domain generalization hierarchy is associated with a number, called *height*, which is equal to the sum of the elements in the corresponding distance vector. The height of a distance vector DV in a distance vector lattice VL is denoted by $height(DV, \mathsf{VL})$. The observation above ensures that if there is no solution that guarantees k-anonymity suppressing less than MaxSup tuples at height h, there cannot exist a solution, with height lower than h that guarantees it. This property is exploited by using a binary search approach on the lattice of distance vectors corresponding to the domain generalization hierarchy of the domains of the quasi-identifier. Consider lattice VL of height $h = height(\top, \mathsf{VL})$, where $\top$ is the top element of the lattice. First, the vectors at height $\lfloor \frac{h}{2} \rfloor$ are evaluated. If there is a vector that satisfies k-anonymity within the suppression threshold established at height $\lfloor \frac{h}{2} \rfloor$, then the vectors at height $\lfloor \frac{h}{4} \rfloor$ are evaluated, otherwise those at height $\lfloor \frac{3h}{4} \rfloor$, and so on, until the algorithm reaches the lowest height for which there is a distance vector that satisfies k-anonymity by suppressing no more tuples than MaxSup. As an example, consider the microdata table in Fig. 1, and assume $QI = \{\texttt{Race}, \texttt{ZIP}\}$, where the domain generalization hierarchy, of height 3, is as illustrated in Fig. 6. Suppose also that $k = 2$ and MaxSup $= 2$. The algorithm starts by evaluating the generalizations at height $\lfloor 3/2 \rfloor = 1$, since there is a solution at level 1 (actually both $\mathsf{GT}_{[1,0]}$ and $\mathsf{GT}_{[0,1]}$ are solutions), the algorithm proceeds by evaluating generalizations at level $\lfloor 3/4 \rfloor = 0$. Table $\mathsf{GT}_{[0,0]}$ suppresses more than 2 tuples for 2-anonymity, so it is not a solution. The (local) minimal solutions are then $\mathsf{GT}_{[1,0]}$ and $\mathsf{GT}_{[0,1]}$.

Although this approach is simple, it requires the computation of all the generalized tables. To avoid such a computation, the concept of distance vector between tuples is introduced and exploited. Let T be a table and $x, y \in T$ be two tuples such that $x = \langle v'_1, \ldots, v'_n \rangle$ and $y = \langle v''_1, \ldots, v''_n \rangle$ where v'_i and v''_i are values in domain D_i, for $i = 1 \ldots, n$. The *distance vector* between x and y is the vector $V_{x,y} = [d_1, \ldots, d_n]$ where d_i is the (equal) length of the two paths from v'_i and v''_i to their closest common ancestor in the value generalization hierarchy VGH_{D_i} (or, in other words, the distance from the domain of v'_i and v''_i to the domain at which they generalize to the same value v_i). For instance, with reference to the PT illustrated in Fig. 1 and the hierarchies in Fig. 3, the distance vector between ⟨`asian,94139`⟩ and ⟨`black,94139`⟩ is [1,0], at which they both generalize to ⟨`person,94139`⟩.

Intuitively, the distance vector $V_{x,y}$ between two tuples x and y in table T_i is the distance vector $DV_{i,j}$ between T_i and the table T_j, with $T_i \preceq T_j$ where the domains of the attributes in T_j are the most specific domains for which x and y generalize to the same tuple t. By looking at the distance vectors between the tuples in a table we can determine whether a generalization at a given vector satisfies k-anonymity by suppressing less than MaxSup tuples without computing the generalization. More precisely, we can determine, for each distance vector DV, the minimum required suppression for the k-anonymity constraint to be satisfied by the generalization corresponding to DV. The approach works as follows. Let $T_i = \mathsf{PT}[QI]$ be the table to be considered. For each distinct tuple $x \in T_i$ determine $count(x, T_i)$ as the number of occurrences of x in T_i. Build a matrix VT with a row for each of the different outliers (i.e., tuples with less than k occurrences) and a column for each different tuple in the table. Entry $\mathsf{VT}[x, y]$ contains the distance vector between tuples x and y, that is, $\mathsf{VT}[x, y] = V_{x,y}$. (Note that the table is symmetric so only half on it actually needs to be computed.) Now, let *vec* be the distance vector of a generalization to consider as a potential solution. For each row x, compute C_x as the sum of the occurrences $count(y, T_i)$ of tuples y (column of the matrix) such that $\mathsf{VT}[x, y] \leq vec$. These are tuples that at generalization *vec* would generalize to the same tuple as x, and the sum of their occurrences is the size of the resulting cluster. Determine then *req_sup* as the sum of the occurrences of all the outlier tuples x (row of the matrix) such that C_x so computed is smaller than k, that is, *req_sup*$= \sum_{x|C_x<k} count(x, T_i)$. Intuitively, *req_sup* is the number of tuples that would still be outliers in the generalization corresponding to distance vector *vec*, and which would therefore need to be removed for the k-anonymity requirement to be satisfied. Hence, if *req_sup* $\leq$ MaxSup the generalization with distance vector *vec* satisfies k-anonymity by suppressing less tuples than the threshold allowed. Otherwise it does not. Figure 11 illustrates a vector matrix VT for the table of Fig. 7. As an example, consider the generalized table $\mathsf{GT}_{[1,0]}$ and suppose that MaxSup $= 2$ and $k = 2$. It is easy to see that $\mathsf{GT}_{[1,0]}$ satisfies 2-anonymity and that the outlier tuples are t_1 and t_6.

	t_1	t_2	$t_3/t_4/t_5$	t_6	t_7	t_8	t_9
t_1	$[0,0]$	$[0,1]$	$[0,2]$	$[1,2]$	$[1,2]$	$[1,2]$	$[1,1]$
t_2	$[0,1]$	$[0,0]$	$[0,2]$	$[1,2]$	$[1,2]$	$[1,2]$	$[1,0]$
t_6	$[1,2]$	$[1,2]$	$[1,1]$	$[0,0]$	$[0,1]$	$[1,1]$	$[1,2]$
t_7	$[1,2]$	$[1,2]$	$[1,0]$	$[0,1]$	$[0,0]$	$[1,0]$	$[1,2]$
t_8	$[1,2]$	$[1,2]$	$[1,0]$	$[1,1]$	$[1,0]$	$[0,0]$	$[0,2]$
t_9	$[1,1]$	$[1,0]$	$[1,2]$	$[1,2]$	$[1,2]$	$[0,2]$	$[0,0]$

Fig. 11. Distance vectors between tuples of table PT in Fig. 7

Race			ZIP			
⟨[asian]	[black]	[white]⟩	⟨[94138]	[94139]	[94141]	[94142]⟩
1	2	3	4	5	6	7

Fig. 12. Index assignment to attributes Race and ZIP

4.2 Bayardo-Agrawal's Algorithm

Bayardo and Agrawal [5] propose an interesting algorithm for **AG_TS**, called *k-Optimize*, which often obtains good solutions with a reduced computational time. According to this approach, an attribute generalization for an attribute A with an ordered domain D consists in a partitioning of the attribute domain into intervals such that each possible value in the domain appears in some interval and each value in a given interval I precedes any value in the intervals following I. As an example, consider attribute Race on domain $D_1 = \{$asian, black, white$\}$ where the values in D_1 are ordered according to a lexicographic order, and attribute ZIP on domain $D_2 = \{94138, 94139, 94141, 94142\}$ where the values follow a numeric order. For instance, domain D_1 can be partitioned into three intervals, namely [asian], [black], and [white], and domain D_2 can be partitioned into four intervals, namely [94138], [94139], [94141], and [94142]. The approach then assumes an order among quasi-identifier attributes and associates an integer, called *index*, with each each interval in any domain of the quasi-identifier attributes. The index assignment reflects the total order relationship over intervals in the domains and among quasi-identifier attributes. For instance, consider the quasi-identifier attributes Race and Zip and suppose that Race precedes ZIP. Figure 12 illustrates the value ordering and the corresponding index values. As it is visible from this fig., the index values associated with the intervals of domain D_1 of attribute Race are lower than the index values associated with the intervals of domain D_2 of attribute ZIP since we assume that Race precedes ZIP. Moreover, within each domain the index assignment reflects the total order among intervals. More formally, the indexes associated with the intervals of domain D_i of attribute A_i are lower than the indexes associated with intervals of domain D_j of attribute A_j, if attribute A_i precedes A_j in the order relationship. Moreover, indexes associ-

ated with each interval I of domain D_i follow the same order as intervals in D_i.

A generalization is then represented through the union of the individual index values for each attribute. The least value in an attribute domain can be omitted since it will certainly appear in the generalizations for that domain. For instance, with respect to the total order of the value domains in Fig. 12, notation {6} identifies a generalization, where the generalizations are {1} for attribute `Race` and {4, 6} for attribute `ZIP`. These, in turn, represent the following value intervals: `Race`: ⟨[asian or black or white]⟩; `ZIP`: ⟨[94138 or 94139], [94141 or 94142]⟩. Note that the empty set { } represents the most general anonymization. For instance, with respect to our example, { } corresponds to the generalizations {1} for attribute `Race` and {4} for attribute `ZIP`, which in turn correspond to the generalized values `Race`: ⟨[asian or black or white]⟩; `ZIP`: ⟨[94138 or 94139 or 94141 or 94142]⟩.

k-Optimize builds a *set enumeration tree* over the set I of index values. The root node of the tree is the empty set. The children of a node n will enumerate those sets that can be formed by appending a single element of 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 13 illustrates an example of set enumeration tree over $I = \{1, 2, 3\}$. The consideration of a tree guarantees the existence of a unique path between the root and each node. The visit of the set enumeration tree using a standard traversal strategy is equivalent to the evaluation of each possible solution to the k-anonymity problem. At each node n in the tree the cost of the generalization strategy represented by n is computed and compared against the best cost found until that point; if lower it becomes the new best cost. This approach however is not practical because the number of nodes in the tree is $2^{|I|}$; therefore [5] proposes heuristics and pruning strategies. In particular, k-Optimize prunes a node n when it can determine that none of its descendants could be optimal. According to a given cost function, k-Optimize computes a lower bound on the cost that can be obtained by any node in the sub-tree rooted at n. The subtree can be pruned if the computed lower bound is higher than the best cost found by the algorithm until that point. Note that when a subtree is pruned also additional nodes can be removed from the tree. For instance, consider the set enumeration tree in Fig. 13 and suppose that node {1, 3} can be pruned. This means that a solution that contains index values 1 and 3 is not optimal and therefore also node {1, 2, 3} can be pruned.

k-Optimize can always compute the best solution in the space of the generalization strategies. Since the algorithm tries to improve the solution at each visited node evaluating the corresponding generalization strategy, it is possible to fix a maximum computational time, and obtain a good, but not optimal, solution.

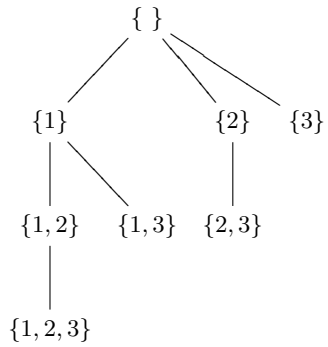

Fig. 13. An example of set enumeration tree over set $I = \{1, 2, 3\}$ of indexes

4.3 Incognito

LeFevre, DeWitt and Ramakrishnan [20] propose an efficient algorithm for computing k-minimal generalization, called *Incognito*, which takes advantage of a bottom-up aggregation along dimensional hierarchies and a priori aggregate computation.

The key idea of Incognito is that if a table T with quasi-identifier QI of m attributes is k-anonymous, T is k-anonymous with respect to any quasi-identifiers QI', where $QI' \subset QI$. In other words, the k-anonymity with respect to a proper subset of QI is a necessary (not sufficient) condition for the k-anonymity with respect to QI. Exploiting this observation, Incognito excludes in advance some generalizations from the hierarchy in a priori computation.

The strategy followed by Incognito is a bottom-up breadth-first search on the domain generalization hierarchy. The algorithm generates all the possible minimal k-anonymous tables for a given private table PT. First (iteration 1), it checks k-anonymity for each single attribute in QI, discarding those generalizations that do not satisfy k-anonymity for the single attribute. Then, it combines the remaining generalizations in pairs performing the same control on pairs of attributes (iteration 2); then in triples (iteration 3), and so on, until the whole set of attributes in QI is considered (iteration $|QI|$). More precisely, for each combination, Incognito checks the satisfaction of the k-anonymity constraint with a bottom-up approach; when a generalization satisfies k-anonymity, all its direct generalizations also certainly satisfy k-anonymity and therefore they are no more considered. It is important to note that at iteration i, Incognito considers all the combinations of i attributes together, by considering only the generalizations that satisfied the k-anonymity constraint at iteration $i - 1$.

As an example, consider table PT in Fig. 1 and suppose that the quasi-identifier is QI={`Race`, `Sex`, `Marital status`}, and assume $k = 2$. At iteration 1, Incognito checks 2-anonymity on each single attribute, and finds that $\mathtt{M_0}$ does not satisfy 2-anonymity. At iteration 2, Incognito checks 2-

anonymity on all the possible pairs of attributes, that is, ⟨Race, Sex⟩, ⟨Race, Marital status⟩, and ⟨Sex, Marital status⟩. In particular, Incognito has to first check the 2-anonymity with respect to the lowest tuples that can be formed with the single attributes generated at iteration 1 (i.e., ⟨R_0, S_0⟩, ⟨R_0, M_1⟩, and ⟨S_0, M_1⟩). It is easy to see that the microdata table in Fig. 1 is 2-anonymous with respect to ⟨R_0, S_0⟩ and ⟨S_0, M_1⟩ but is not 2-anonymous with respect to ⟨R_0, M_1⟩ because, for example, there is only one occurrence of ⟨white, been_married⟩. Incognito therefore proceeds by checking generalizations ⟨R_0, M_2⟩ and ⟨R_1, M_1⟩. These generalizations satisfy 2-anonymity and then Incognito can start iteration 3. Due to the previous iterations, Incognito has to first check generalizations ⟨R_0, S_0, M_2⟩, and ⟨R_1, S_0, M_1⟩. Since these two generalizations satisfy the 2-anonymity property, the algorithm terminates. Figure 14 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 (i.e., from which the generalizations, which are a priori known not to satisfy k-anonymity, have been discarded).

4.4 Heuristic Algorithms

The algorithms presented so far find exact solutions for the k-anonymity problem. Since k-anonymity is a NP-hard problem, all these algorithms have complexity exponential in the size of the quasi-identifier. Alternative approaches have proposed the application of heuristic algorithms. The algorithm proposed by Iyengar [18] is based on genetic algorithms and solves the k-anonymity problem using an incomplete stochastic search method. The method does not assure the quality of the solution proposed, but experimental results show the validity of the approach. Winkler [34] proposes a method based on simulated annealing for finding locally minimal solutions, which requires high computational time and does not assure the quality of the solution.

Fung, Wang and Yu [12] present a top-down heuristic to make a table to be released k-anomymous. The approach applies to both continuous and categorical attributes. The top-down algorithm starts from the most general solution, and iteratively specializes some values of the current solution until the k-anonymity requirement is violated. Each step of specialization increases the information and decreases the anonymity. Therefore, at each iteration, the heuristic selects a "good" specialization guided by a goodness metric. The metric takes into account both the "information gain" and the "anonymity loss".

Due to heuristic nature of these approaches, no bounds on efficiency and goodness of the solutions can be given; however experimental results can be used to assess the quality of the solution retrieved.

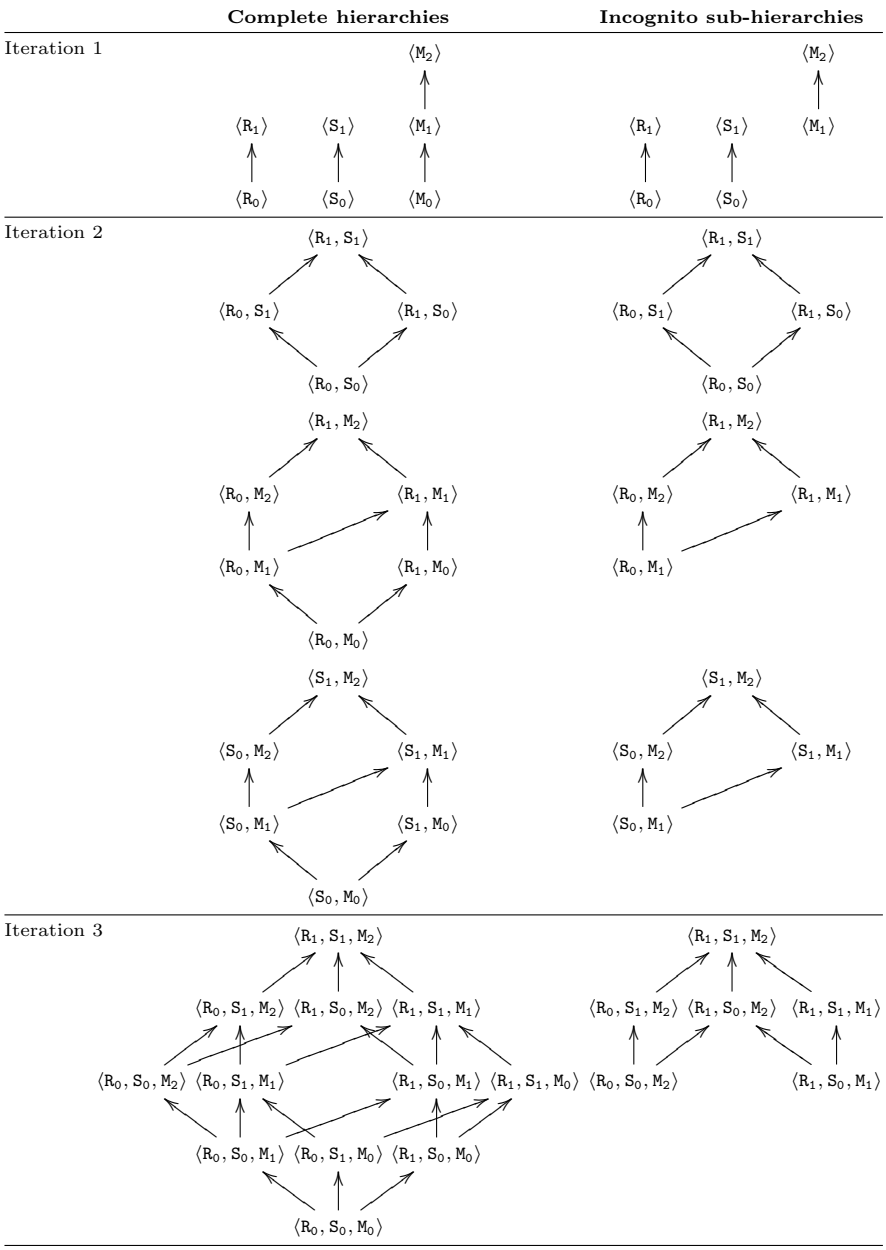

Fig. 14. Sub-hierarchies computed by Incognito for the table in Fig. 1

5 Algorithms for _CS and CG_ Models

The exact algorithms just illustrated for solving the k-anonymity problem for **AG_TS** and **AG_** are, due to the complexity of the problem, exponential in the size of the quasi-identifier. The exact algorithms for models **_CS** and **CG_** can be much more expensive, since the computational time could be exponential in the number of tuples in the table.

Approximation algorithms for **_CS** and **CG_** have been proposed, both for general and specific values of k (e.g., 1.5-approximation for 2-anonymity, and 2-approximation for 3-anonymity in [3]). 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 [13]. Both heuristics and approximation algorithms do not guarantee the minimality of their solution, but while we cannot perform any evaluation on the result of a heuristic, an approximation algorithm guarantees near-optimum solutions.

The first approximation algorithm for **_CS** was proposed by Meyerson and Williams [24]. They presented an algorithm for k-anonymity, which guarantees a $O(k \log(k))$-approximation. Two approximation algorithms for **_CS** and **CG_**, with unbounded value of k, are described in [2, 3] and guarantee a $O(k)$-approximation solution.

The best-known approximation algorithm for **_CS** is described in [2] and guarantees a $O(k)$-approximation solution. The algorithm constructs a complete weighted graph from the original private table PT. Each node in the graph corresponds to a tuple in PT, and the arcs are weighted with the number of different attribute values between the two tuples represented by extreme nodes. The algorithm then constructs, starting from the graph, a forest composed of trees containing at least k nodes, which represents the clustering for k-anonymization. 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, the cost of a tree instead is the sum of the weights of its arcs. The cost of the final solution is equal to the sum of the costs of its trees. On the contrary, the cost of a k-anonymity solution is the cost of the biggest partition in the final forest, since the presence of big clusters implies the unification of many respondents into a single one, which causes information loss. In constructing the forest, the algorithm attempts to limit the maximum number of nodes in a tree to be $3k-3$. Partitions with more than $3k-3$ elements are decomposed, without increasing the total solution cost.

An approximation algorithm for **CG_** is described in [3] as a direct generalization of the approximation algorithm for **_CS** presented in [2]. The main difference is that, in this version, the weights of edges depend on the lowest level of generalization, for each attribute, that makes the tuples in the extreme nodes equal.

To find out better results for Boolean attributes, in case $k = 2$ and $k = 3$, a different approach has been provided in [3]. Since **_CS** and **CG_** are equivalent when we use Boolean attributes, we consider here only k-anonymity with **_CS**.

The algorithm for $k = 2$ exploits the minimum-weight $[1, 2]$-factor built on the graph constructed for the 2-anonymity instance. The $[1, 2]$-factor for graph G is the spanning subgraph of G, built using only vertexes of degree 1 or 2 (i.e., with no more than 2 outgoing edges). Such a subgraph is a vertex-disjoint collection of edges and pairs of adjacent nodes and can be computed in polynomial time. Each component in the subgraph is treated as a cluster, and we can obtain a 2-anonymized table 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.

6 Further Studies on *k*-Anonymity

We now briefly survey some interesting studies based on the concept of k-anonymity.

Multidimensional k-Anonymity

The generalization procedures in the original k-anonymity proposal [26] assume a value generalization hierarchy, where each value has only a single generalization. For instance, with respect to the hierarchy in Fig. 3, a single step of generalization for ZIP code 94142 produces the unique value 9414∗. However, a generalization step could produce different generalized values. For instance, some possible generalized values corresponding to value 94142 are 9414∗ and 941∗2. If we assume that the generalization hierarchy is a graph instead of a tree, the generalization problem can even be harder. For solving this problem, LeFevre, DeWitt and Ramakrishnan propose a multidimensional model for k-anonymity [21, 22]. The authors show that the problem is still NP-hard and propose a greedy approximation algorithm for both numerical and categorical datasets. The time complexity of the algorithm proposed is $O(n \log n)$, where n is the number of tuples in the original table. The resulting k-anonymous table has a higher quality than the anonymized tables produced by other single-dimensional algorithms.

ℓ-Diversity

Although k-anonymity is a technique adopted to protect microdata respondents' privacy, it is vulnerable to some attacks that may lead to privacy breach. Machanavajjhala, Gehrke, and Kifer describe two possible attacks, namely *homegeneity attack* (already noted in [26]) and *background knowledge attack* [23]. Consider a k-anonymized table, where there is a sensitive attribute and suppose that all tuples with a specific value for the quasi-identifier have the same sensitive attribute value. Under these assumptions (homogeneity attack), if an attacker knows both the quasi-identifier value of an entity and

Race	DOB	Sex	ZIP	Disease
asian	64	F	941**	hypertension
asian	64	F	941**	obesity
asian	64	F	941**	chest pain
asian	63	M	941**	obesity
asian	63	M	941**	obesity
black	64	F	941**	short breath
black	64	F	941**	short breath
white	64	F	941**	chest pain
white	64	F	941**	short breath

Fig. 15. A 2-anonymous table according to the **AG**$_-$ model

knows that this entity is represented in the table, the attacker can infer the sensitive value associated with certainty. For instance, with respect to the 2-anonymous table in Fig. 15, if `Alice` knows that `Carol` is a `black female` and that her data are in the microdata table, she can infer that `Carol` suffers of `short breath`, as both the tuples having these values for the `Race` and `Sex` attributes are associated with the `short breath` value for the `Disease` attribute. The 2-anonymous table is therefore exposed to attribute linkage.

The background knowledge attack is instead based on a prior knowledge of the attacker of some additional external information. For instance, suppose that `Alice` knows that `Hellen` is a `white female`. `Alice` can then infer that `Hellen` suffers of `chest pain` or `short breath`. Suppose now that `Alice` knows that `Hellen` runs for two hours every day. Since a person that suffers of `short breath` cannot run for a long period, `Alice` can infer with probability equal to 1 that `Hellen` suffers of `chest pain`.

To avoid such attacks, Machanavajjhala, Gehrke, and Kifer introduce the notion of ℓ-*diversity* [23]. Given a private table PT and a generalization GT of PT, let q-block be a set of tuples in GT with the same quasi-identifier value. A q-block is said to be ℓ-*diverse* if it contains at least ℓ different values for the sensitive attribute. It is easy to see that with this additional constraint, the homogeneity attack is no more applicable because each q-block set has at least ℓ (≥ 2) distinct sensitive attribute values. Analogously, the background knowledge attack becomes more complicate as ℓ increases because the attacker needs more knowledge to individuate a unique value associable to a predefined entity. The algorithm proposed in [23] therefore generates k-anonymous tables with the ℓ-diversity property. The algorithm checks the ℓ-diversity property, which is a monotonic property with respect to the generalization hierarchies considered for k-anonymity purposes.

It is important to note that the proposed algorithm considers only one sensitive attribute at time. The consequence is that even if each sensitive attribute in GT satifies the ℓ-diversity property, the whole table GT may not respect the ℓ-diversity property because the combination of the background knowledge on two or more sensitive attributes may lead to privacy breaches.

Evaluation of k-Anonymity

Some recent papers evaluate the results of k-anonymization using data mining techniques [1, 12, 32]. In particular, Aggarwal [1] shows that, when the number of attributes in the quasi-identifier increases, the information loss of the resulting k-anonymized table may become very high. The intuition behind this result is that the probability that k tuples in the private table are "similar" (i.e., they correspond to the same tuple in the anonymized table with a reduced loss of information) is very low. The ability to identify minimal quasi-identifiers is therefore important.

Distributed Algorithms

Besides anonymizing data locally maintained by a data holder, it is also important to anonymize data distributed through different interconnected parties. To this purpose, recently some algorithms for distributed k-anonymity have been proposed [19, 31, 38].

Jiang and Clifton [19] suppose a microdata table to be vertically partitioned and stored at two different sites. The whole data can be reconstructed through a join on a common key. The authors propose a communication protocol allowing the two data holders to put together their data, obtaining a k-anonymized table. Basically, the two data holders agree on the tuples that should be generalized to the same quasi-identifier value before release. Once the two data holders agree on the strategy to adopt, they generalize their values following this common strategy. Wang, Fung and Dong [31] propose another approach for vertically partitioned tables, where the parties interact to individuate the best generalization strategy to adopt for k-anonymization.

Zhong, Yang and Wright [38] propose instead a distributed k-anonymity method for a horizontally partitioned table. The table has $m + n$ attributes, where m attributes form a quasi-identifier for the table and the remaining n attributes are the sensitive attributes. Also, there are N customers that own a single tuple in the table. To build a k-anonymous table, the authors propose two different solutions. In the first one, each customer encrypts her sensitive attributes using an encryption key. The decryption of these sensitive attributes can be done only if there are at least k tuples with equal value for the corresponding quasi-identifier. This technique corresponds to the application of tuple suppression because if there are less than k tuples with the same quasi-identifier value, their sensitive attributes cannot be decrypted.

The second solution adopts cell suppression on the original microdata table by applying Meyerson and William's algorithm [24] to the distributed scenario.

Name	Sex	Disease
Sue	F	hypertension
Claire	F	obesity
Ann	F	chest pain
John	M	obesity
David	M	obesity
Mary	F	short breath
Alice	F	short breath
Carol	F	chest pain
Kate	F	short breath

PT

Name	Sex
Sue	F
Claire	F
Ann	F
John	M
David	M
Mary	F
Alice	F
Carol	F
Kate	F

view v_1

Sex	Disease
F	hypertension
F	obesity
F	chest pain
F	short breath
M	obesity

view v_2

Fig. 16. A private table PT and two possible views on PT

k-Anonymity with Multiple Views

The individuals' privacy can be violated by inferring information through multiple views on a private table. This problem is known as *data association*. Data association refers to the possibility that two or more attributes are considered more sensitive when their values are associated than when either appears separately. For instance, consider the private table PT in Fig. 16, where `Name` is an identifier and `Disease` is a sensitive attribute whose association with `Name` must be protected. Attributes `Name` and `Disease` should therefore be released through different views. Fig. 16 illustrates two possible views, namely v_1 and v_2, of the original private table. By combining the information contained in these two views, it is however possible to reconstruct the associations between `Name` and `Disease` (e.g., `John` and `David` with `obesity`).

Yao, Wang and Jajodia [7] investigate this issue and use k-anonymity as a measure on information disclosure by a set of views with respect to a given data association that has to be protected. The authors first introduce the notion of *association cover* with respect to a set of views as follows. Suppose that the association between attributes `ID` and `P` must be protected, where `ID` is an identifier and `P` is a sensitive attribute. An association cover with respect to a set of views v is a set of pairs $\{\langle \texttt{id}, \texttt{p}_1\rangle, \ldots, \langle \texttt{id}, \texttt{p}_n\rangle\}$, where `id` is a fixed value of attribute `ID` and $\texttt{p}_i, i = 1, \ldots, n$, is a value that can be associated with `id` in the set of views v. For instance, $\{\langle$`Sue, hypertension`$\rangle, \langle$`Sue, obesity`$\rangle, \langle$`Sue, chest pain`$\rangle, \langle$`Sue, short breath`$\rangle\}$ is an association cover with respect to views v_1 and v_2 in Fig. 16. Given a set of views v and an integer k, v is said to violate k-anonymity if there exists an association cover w.r.t. v of size less than k. In our example, the association cover $\{\langle$`John, obesity`$\rangle\}$ and the association cover $\{\langle$`David, obesity`$\rangle\}$ violate k-anonymity for any $k > 1$. Intuitively, this means that if a set of views v does not violate k-anonymity for a specific user-defined k value, we can state that the association between attributes `ID` and `P` is protected.

Yao, Wang and Jajodia show that the problem of stating whether a set of views violates k-anonymity is in general computationally hard (NP^{NP}-hard). In the case where no functional dependencies exist among the views, the problem becomes simpler, and a polynomial checking algorithm for its solution is described [7].

k-Anonymity with Micro-Aggregation

Domingo-Ferrer and Mateo-Sanz [9] propose the use of micro-aggregation (instead of generalization and suppression) to achieve k-anonymity.

Micro-aggregation requires to divide a microdata set in a number of clusters of at least k tuples each. For each attribute, the average value over each cluster is computed and used to replace each of the original averaged values. The optimal k-partition solution is a partition that maximizes the homogenity within each cluster; the higher the homogeneity within each cluster, the lower the information loss since the micro-aggregation replaces values in a cluster by the cluster *centroid*. The sum of squares is the traditional criterion to measure homogeneity in clustering. The problem of optimal micro-aggregation is related to the classical minimum sum-of-squares clustering that is a NP-hard problem [25]. Domingo-Ferrer and Mateo-Sanz therefore propose to determine an optimal solution by reducing the solution space. To this purpose, their approach consider only solutions with clusters of size between k and $2k$. The minimum size is fixed to k to achieve k-anonymity, while the maximum is set to $2k$ to minimize information loss.

k-Anonymity for Protecting Location Privacy

The k-anonymity property has been studied also for protecting location privacy [6, 14]. In the context of location-based services, Bettini, Wang and Jajodia [6] present a framework for evaluating the privacy of a user identity when location information is released. In this case, k-anonymity is guaranteed, not among a set of tuples of a database, but in a set of individuals that can send a message in the same spatio-temporal context.

k-Anonymity for Communication Protocols

k-anonymity has also been investigated to preserve privacy in communication protocols [15, 30, 36, 37], with the notion of *sender* (*receiver*, resp.) k-anonymity. A communication protocol is *sender k-anonymous* (*receiver k-anonymous*, resp.) if it guarantees that an attacker, who is trying to discover the sender (receiver, resp.) of a message, can just detect a set of k possible senders (receivers, resp.).

7 Conclusions

k-anonymity has recently been investigated as an interesting approach to protect microdata undergoing public or semi-public release from linking attacks. In this chap., we illustrated the original k-anonymity proposal and its enforcement via generalization and suppression as means to protect respondents' identities while releasing truthful information. We then discussed different ways in which generalization and suppression can be applied, thus defining a possible taxonomy for k-anonymity and discussed the main proposals existing in the literature for solving the k-anonymity problems in the different models. We have also illustrated further studies building on the k-anonymity concept to safeguard privacy.

8 Acknowledgments

This work was supported in part by the European Union within the PRIME Project in the FP6/IST Programme under contract IST-2002-507591 and by the Italian MIUR within the KIWI and MAPS projects.

References

1. Aggarwal C (2005). On k-anonymity and the curse of dimensionality. In Proc. of the 31st International Conference on Very Large Data Bases (VLDB'05), Trondheim, Norway.
2. Aggarwal G, Feder T, Kenthapadi K, Motwani R, Panigrahy R, Thomas D, Zhu A (2005). Anonymizing tables. In Proc. of the 10th International Conference on Database Theory (ICDT'05), pp. 246–258, Edinburgh, Scotland.
3. Aggarwal G, Feder T, Kenthapadi K, Motwani R, Panigrahy R, Thomas D, Zhu A (2005). Approximation algorithms for k-anonymity. Journal of Privacy Technology, paper number 20051120001.
4. Anderson R (1996). A security policy model for clinical information systems. In Proc. of the 1996 IEEE Symposium on Security and Privacy, pp. 30–43, Oakland, CA, USA.
5. Bayardo RJ, Agrawal R (2005). Data privacy through optimal k-anonymization. In Proc. of the 21st International Conference on Data Engineering (ICDE'05), pp. 217–228, Tokyo, Japan.
6. Bettini C, Wang XS, Jajodia S (2005). Protecting privacy against location-based personal identification. In Proc. of the Secure Data Management, Trondheim, Norway.
7. Jajodia S, Yao C, Wang XS (2005). Checking for k-anonymity violation by views. In Proc. of the 31st International Conference on Very Large Data Bases (VLDB'05), Trondheim, Norway.
8. Dobson J, Jajodia S, Olivier M, Samarati P, Thuraisingham B (1998). Privacy issues in www and data mining. In Proc. of the 12th IFIP WG11.3 Working Conference on Database Security, Chalkidiki, Greece. Panel notes.

9. Domingo-Ferrer J, Mateo-Sanz JM (2002). Practical data-oriented microaggregation for statistical disclosure control. IEEE Transactions on Knowledge and Data Engineering, 14(1):189–201.
10. Duncan GT, Jabine TB, de Wolf VA editors (1993). Private Lives and Public Policies. National Academy Press.
11. Franconi L, Polettini S (2004). Individual risk estimation in μ-ARGUS: a review. In Domingo-Ferrer J and Torra V, editors, Privacy in Statistical Databases, vol. 3050 of LNCS, pp. 262–372. Springer, Berlin Heidelberg.
12. Fung B, Wang K, Yu P (2005). Top-down specialization for information and privacy preservation. In Proc. of the 21st International Conference on Data Engineering (ICDE'05), Tokyo, Japan.
13. Garey M, Johnson D (1979). Computers and Intractability. Freeman and Company.
14. Gedik B, Liu L (2005). A customizable k-anonymity model for protecting location privacy. In Proc. of the 25th International Conference on Distributed Computing Systems (IEEE ICDCS), Columbus, Ohio, USA.
15. Hughes D, Shmatikov V (2004). Information hiding, anonymity and privacy: a modular approach. Journal of Computer Security, 12(1):3–36, 2004.
16. Hundepool A, Van deWetering A, Ramaswamy R, Franconi L, Capobianchi A, DeWolf PP, Domingo-Ferrer J, Torra V, Brand R, Giessing S (2003). μ-ARGUS version 3.2 software and user's manual. Statistics Netherlands. http://neon.vb.cbs.nl/casc.
17. Hundepool A, Willenborg L (1996). μ- and τ-ARGUS: software for statistical disclosure control. In Proc. of the 3rd International Seminar on Statistical Confidentiality, Bled.
18. Iyengar V (2002). Transforming data to satisfy privacy constraints. In Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 279–288, Edmonton, Alberta, Canada.
19. Jiang W, Clifton C (2005). Privacy-preserving distributed k-anonymity. In Proc. of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, Storrs, CT, USA.
20. LeFevre K, DeWitt DJ, Ramakrishnan R (2005). Incognito: Efficient full-domain k-anonymity. In Proc. of the 24th ACM SIGMOD International Conference on Management of Data, pp. 49–60, Baltimore, Maryland, USA.
21. LeFevre K, DeWitt DJ, Ramakrishnan R (2005). Multidimensional k-anonymity. Technical Report 1521, Department of Computer Sciences, University of Wisconsin, Madison, USA.
22. LeFevre K, DeWitt DJ, Ramakrishnan R (2006). Mondrian multidimensional k-anonymity. In Proc. of the International Conference on Data Engineering (ICDE'06), Atlanta, GA, USA.
23. Machanavajjhala A, Gehrke J, Kifer D (2006). ℓ-diversity: Privacy beyond k-anonymity. In Proc. of the International Conference on Data Engineering (ICDE'06), Atlanta, GA, USA.
24. Meyerson A, Williams R (2004). On the complexity of optimal k-anonymity. In Proc. of the 23rd ACM-SIGMOD-SIGACT-SIGART Symposium on the Principles of Database Systems, pp. 223–228, Paris, France.
25. Oganian A, Domingo-Ferrer J (2001). On the complexity of optimal microaggregation for statistical disclosure control. Statistical Journal of the United Nations Economic Comission for Europe, 18(4):345–354.

26. Samarati P (2001). Protecting respondents' identities in microdata release. IEEE Transactions on Knowledge and Data Engineering, 13(6):1010–1027.
27. Samarati P, Sweeney L (1998). 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, p. 188, Seattle, WA, USA.
28. Sweeney L (1997). Guaranteeing anonymity when sharing medical data, the Datafly system. In Journal of the American Medical Informatics Association, Washington, DC: Hanley & Belfus, Inc.
29. Sweeney L (2002). Achieving k-anonynity privacy protection using generalization and suppression. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, 10(5):571–588.
30. von Ahn L, Bortz A, Hopper NJ (2003). k-anonymous message transmission. In Proc. of the 10th ACM Conference on Computer and Communications Security, pp. 122–130, Washington, DC, USA.
31. Wang K, Fung B, Dong G (2005). Integrating private databases for data analysis. In Kantor P et al., editor, ISI 2005, vol. 3495 of LNCS, pp. 171–182. Springer-Verlag.
32. Wang K, Yu P, Chakraborty S (2004). Bottom-up generalization: a data mining solution to privacy protection. In Proc. of the 4th International Conference on Data Mining (ICDM'04), Brighton, UK.
33. Winkler WE (2002). Using simulated annealing for k-anonymity. Technical Report 7, U.S. Census Bureau.
34. Winkler WE (2004). Masking and re-identification methods for public-use microdata: Overview and research problems. In Domingo-Ferrer J, editor, Privacy in Statistical Databases 2004. Springer, New York.
35. Woodward B (1995). The computer-based patient record confidentiality. The New England Journal of Medicine, 333(21):1419–1422.
36. Xu S, Yung M (2004). k-anonymous secret handshakes with reusable credentials. In Proc. of the 11th ACM Conference on Computer and Communications Security, pp. 158–167, Washingtion, DC, USA.
37. Yao G, Feng D (2004). A new k-anonymous message transmission protocol. In Proc. of the 5th International Workshop on Information Security Applications,(WISA'04), pp. 388–399, Jeju Island, Korea.
38. Zhong S, Yang Z, and Wright R (2005). Privacy-enhancing k-anonymization of customer data. In Proc. of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'05), pp. 139–147, Baltimore, Maryland, USA.

Index