Theory of Privacy and Anonymity

Valentina Ciriani, Sabrina De Capitani di Vimercati, Sara Foresti, Pierangela Samarati

University of Milan - 26013 Crema, Italy

{ciriani,decapita,foresti,samarati}@dti.unimi.it

Contents

1 Introduction

The increased power and interconnectivity of computer systems and the advances in memory sizes, disk storage capacity, and networking bandwidth allow data to be collected, stored, and analyzed in ways that were impossible in the past due to the restricted access to the data and the expensive processing (in both time and resources) of them. Huge data collections can be analyzed by powerful techniques (e.g., data mining techniques [12]) and sophisticated algorithms thus making possible *linking attacks* combining information available through different sources to infer information that was not intended for disclosure. For instance, by linking de-identified medical records (i.e., records where the explicit identifiers such as the Social Security Numbers have been removed) with other publicly available data or by looking at unique characteristics found in the released medical data, a data observer will most certainly be able to reduce the uncertainty about the identities of the users to whom the medical records refer, or -worse- to determine them exactly. This identity disclosure often implies leakage of sensitive information, for example, allowing data observers to infer the illness of patients. The need for *privacy* is therefore becoming an issue that most people are concerned about. Although there are many attempts to create a unified and simple definition of privacy, privacy by its own nature is a multifacet concept that may encompass several meaning, depending on different contexts.

In this chapter, we focus our attention on the technological aspect of privacy within today's global network infrastructure, where users interact with remote information sources for retrieving data or for using on-line services. In such a context, privacy involves the following three different but related concepts.

- *Privacy of the user*. It corresponds to the problem of protecting the relationship of users with a particular action and outcome. Since we focus on network infrastructures, we are interested in protecting the relationship between a user and the messages she sends, which can be, for example, queries for retrieving some information or requests for using a particular on-line service.

- *Privacy of the communication*. It corresponds to the classical problem of protecting the confidentiality of personal information when transmitted over the network (or with other forms of communication), and to the problem of protecting the privacy of a request to a service provider hiding the content of the request from every party, as well as from the service provider.

- *Privacy of the information*. It involves privacy policies as well as technologies for ensuring proper data protection. A basic requirement of a privacy policy is to establish both the responsibilities of the data holder with respect to data use and dissemination, and the rights of the user to whom the information refers to regulate such use and dissemination. Each user should be able to control further disclosure, view data collected about her and, possibly, make or require corrections.

We now discuss these three aspects more in details.

1.1 Privacy of the user

Privacy of the user concerns protecting the identities of the parties that communicate through a network to avoid possible attacks that have the main purpose to trace who is communicating with whom, or who is interacting with which server or searching for which data. This problem can be solved by providing techniques and protocols that guarantee an *anonymous communication* among different parties. In particular, anonymous communication is about the protection of the relationship between senders and the messages they send, called *sender anonymity*, between recipients and the messages they receive, called *recipient anonymity*, or both, meaning that anonymous senders send messages to anonymous recipients. In the literature, there are a number of approaches providing anonymous communication.

Mix networks, first proposed by Chaum [8], are a way of achieving anonymity on communication networks. Intuitively, a mix is a special node in a network that relays messages in such a way that an outside observer cannot link an outgoing message with an incoming message. Several mix nodes can also be chained to relay a message anonymously. Since their introduction, a large amount of research on mixnets has been performed and different mixnet topologies have been studied and compared [37].

Onion Routing [39] is another solution for ensuring anonymous connections. The main goal of Onion Routing is to guarantee that malicious users cannot determine the contents of messages flowing from a sender to a recipient, and that they cannot verify whether the sender and the recipient are communicating with each other. Onion Routing also provides sender anonymity while preserving the ability of the recipient to send a reply to the sender. With Onion Routing, a network includes a set of *onion routers*, which work as the ordinary routers, combined with mixing properties. When a sender wants to communicate anonymously with a particular recipient, an anonymous connection has to be setup. The sender therefore connects to a particular onion router that prepares an *onion*. An onion is a layered data structure including information about the route of the anonymous connection. In particular, the onion router randomly selects other onion routers and generates a message for each of such a router, providing it with symmetric keys for decrypting messages, and telling it which the next hop (an onion router or the final recipient) in the path will be. Note that anonymity is only provided from the first to the last onion router. The connections from the sender to the first onion router, and from the last onion router to the recipient are not anonymous.

Tor [17] allows users to communicate anonymously on the Internet. Tor is primarily used for anonymous TCP-based applications. Basically, messages are encrypted and then sent through a randomly chosen path of different servers in such a way that an observer cannot discover the source and the destination of the message.

Crowds [35] provides anonymity based on the definition of groups of users, called *crowds*, who collectively perform requests. In this way, each request is equally likely to originate from any user in the crowd. Each user is represented by a local *jondo* stored on her machine that receives a request from the user and removes from such a request all identifying information. A jondo acts as a web proxy that can forward both the user's and other users' requests to the end server, or to another jondo. In this way, since a jondo cannot tell whether or not a request has been initiated by the previous jondo (or the one before it, and so on), users maintain their anonymity within the crowd. The communications along a path of jondos is encrypted and such a path remains the same for the whole session, meaning that requests and replies follow the same path.

1.2 Privacy of the communication

Privacy of the communication is related to two aspects: *i)* protection of the confidentiality of personal information sent through a network; and *ii)* protection of the content of requests to prevent, for example, user profiling. The confidentiality of the personal information transmitted over the network can be ensured by adopting protocols (e.g., SSL) that encrypt it. The need for protecting also the request content arises because there are many real world scenarios where the request content could be misused by service providers. For instance, consider a medical database that contains information about known illnesses, symptoms, treatment options, specialists, and treatment costs. Suppose that a user submits a query to the medical database to collect information about a specific illness. Any other user, including the database administrator, who is able to observe such a query can then infer that the requestor, or a near relative/friend, might suffer from that specific illness.

The problem of protecting the request content is known as *Private Information Retrieval* (PIR) problem [10]. There are many different ways for formally defining the PIR problem. The first formulation assumes that a database can be modeled as a N-bit string and a user is interested in *privately retrieving* the i-th bit of the N bits stored at the server, meaning that the server does not know which is the bit the user is interested in. Starting from this formulation, many results and advancements have been obtained. In particular, depending on the assumptions about the computational power of the service provider, PIR proposals can be classified into two main classes: *theoretical* PIR and *computational* PIR. Theoretical PIR does not make any assumption about the computational power of the service provider and the privacy of the request must then be provided independently from the computational power of an attacker. Computational PIR assumes that a request is private if the service provider must solve a computationally intractable problem to break it. A naive solution to the theoretical PIR problem consists in completely downloading the database at the client side and performing the research on the local copy of the data. Such a solution however has a high communication cost, especially when the database contains a huge data collection. In [10] the authors prove that if there is only one copy of the database stored on one service provider, there is no solution to the theoretical PIR problem that is better than the solution corresponding to the download of the whole database. By contrast, if there are m copies of the data been stored at different non communicating servers, the theoretical PIR problem can be solved by submitting m independent requests (each single request does not provide any information about the bit on which the user is interested in) to the corresponding service providers. The m results are then combined by the user to obtain the answer to her request [5, 10, 24]. The computational PIR problem is typically solved by encrypting the request submitted to the service provider [7, 9, 27]. By exploiting properties of the encryption function, the server computes the encrypted result of the request, which can be decrypted only by the requestor.

Note that the PIR problem can also be seen as a *Secure Multi-Party Computation* (SMC) problem [19, 42]. The main goal of SMC is to allow a set of parties to compute the value of a function f on private inputs provided by the parties without revealing to them the private inputs of the other parties. The SMC problem is usually solved by modeling function f through a Boolean circuit, where gates correspond to protocols that require the collaboration among parties, since they know the input and share knowledge on the output. Other examples of SMC problems are represented by privacy-preserving statistical analysis [18] and privacy-preserving data mining [4, 31]. The SMC problem will be discussed in more details in Chapter **??**.

1.3 Privacy of the information

Privacy of the information is related to the definition of privacy policies expressing and combining different protection requirements, as well as the development of technologies for ensuring proper data protection. An important aspect of the data protection issue relates to the protection of the identities of the users to whom the data refer, meaning that the *anonymity* of the users must be guaranteed. Anonymity does not imply that no information at all is released, but requires that information released be non identifiable. More precisely, given a collection of personal information about a user, the privacy of the user is protected if the value of her personal information is kept private. Whenever a subset of the personal information can be used to identify the user, the anonymity of the user depends on keeping such a subset private. For instance, a recent study [23] shows that in 2000, 63% of the US population was uniquely identifiable by `gender`, `ZIP code`, and full `date of birth`. This means that while these individual pieces of information do not identify a user and therefore the user is anonymous according to the usual meaning of the term, the combined knowledge of `gender`, `ZIP code`, and full `date of birth` may result in the unique identification of a user. Note also that anonymity is always related to the identification of a user rather than the specification of that user. For instance, a user can be univocally identified through her SSN but in the absence of an information source that associates that SSN with a specific identity, the user is still anonymous. Ensuring proper anonymity protection then requires the investigation of the following different issues [13].

- *Identity disclosure protection.* Identity disclosure occurs whenever it is possible to re-identify a user, called *respondent*, from the released data. Techniques for limiting the possibility of re-identifying respondents should therefore be adopted.
- *Attribute disclosure protection.* Identity disclosure protection alone does not guarantee privacy of sensitive information because all the respondents in a group could have the same sensitive information. To overcome this issue, mechanisms that protect sensitive information about respondents should be adopted.
- *Inference channel protection.* Given the possibly enormous amount of data to be considered, and the possible inter-relationships between data, it is important that the security specifications and enforcement mechanisms provide automatic support for complex security requirements, such as those due to inference and data association channels [15].

Protection of anonymity is a key aspect in many different contexts, where the identities of the users (or organizations, associations, and so on) to whom the data refer have been removed, encrypted, or coded. The identity information removed or encoded to produce anonymous data includes names, telephone numbers, and Social Security Numbers. Although apparently anonymous, the de-identified data may contain other identifying information that uniquely or almost uniquely distinguishes the user [14, 20, 21], such as the `gender`, `ZIP code`, and full `date of birth` mentioned above. By linking such an information to publicly available databases (e.g., many countries provide access to public records to their citizens thus making available a broad range of personal information) associating them to the user's identity, a data recipient can determine to which user each piece of released information belongs, or restrict her uncertainty to a specific subset of users. To avoid this, specific data protection norms apply to data collected for a given purpose that state that such data can be further elaborated for historical, statistical or scientific purposes provided that "appropriate safeguards" are applied. In general, the "appropriate safeguards" depend on the method in which the data are released. In the past, data were principally released in the form of *macrodata*, which are statistics on users or organizations usually presented as two-dimensional tables, and through *statistical databases*, which are database whose users may retrieve only aggregate statistics [1]. Macrodata protection techniques are based on the *selective obfuscation of sensitive cells* [13]. Techniques for protecting statistical databases follow two main approaches [16]. The first approach restricts the statistical queries that can be made (e.g., queries that identify a small/large number of tuples) or the data that can be published. The second approach provides protection by returning to the user a modified result. The modification can be enforced directly on the stored data or at run time (i.e., when computing the result to be returned to the user). Many situations require today that the specific *microdata*, that is, data actually stored in the database and not an aggregation of them, are released. The advantage of releasing microdata instead of specific pre-computed statistics is an increased flexibility and availability of information for the users. At the same time, however, microdata, releasing more specific information, are subject to a greater risk of privacy breaches. Several techniques have been proposed in the literature to protect the disclosure of information that should be kept private [11, 13]. In this chapter, we focus on k-anonymity [36], an approach that, compared to the other commonly used approaches (e.g., sampling, swapping values, and adding noise to the data while maintaining some overall statistical properties of the resulting table), has the great advantage of protecting respondents' identities while releasing truthful information.

In the remainder of this chapter, we describe the k-anonymity concept (Section 2) and illustrate some proposals for its enforcement (Section 3 and Section 4). Note that k-anonymity protects identity disclosure, while remaining exposed to attribute disclosure [36]. We will see then how some researchers have just started proposing extensions to k-anonymity to protect also attribute disclosure [32] (Section 5).

2 k-Anonymity: the problem

Although k-anonymity is a concept that applies to any kind of data, for simplicity its formulation considers data represented by a *relational table*. Formally, let $\mathcal{A}$ be a set of attributes, $\mathcal{D}$ be a set of domains, and *dom*: $\mathcal{A} \rightarrow \mathcal{D}$ be a function that associates with each attribute $A \in \mathcal{A}$ a domain $D = dom(A) \in \mathcal{D}$, containing the set of values that A can assume. A *tuple* t over a set $\{A_1,\ldots,A_p\}$ of attributes is a function that associates with each attribute A_i a value $v \in dom(A_i)$, $i = 1, \ldots, p$.

Definition 2.1 (Relational table) *Let $\mathcal{A}$ be a set of attributes, $\mathcal{D}$ be a set of domains, and dom* $: \mathcal{A} \rightarrow \mathcal{D}$ *be a function associating each attribute with its domain. A* relational table T *over a finite set* $\{A_1,\ldots,A_p\} \subseteq \mathcal{A}$ *of attributes, denoted* $T(A_1,\ldots,A_p)$ *is a set of tuples over the set* $\{A_1,\ldots,A_p\}$ *of attributes.*

Notation $dom(A,T)$ denotes the domain of attribute A in T, $|T|$ denotes the number of tuples in T, and $t[A]$ represents the value v associated with attribute A in t. Similarly, $t[A_1,\ldots,A_k]$ denotes the subtuple of t containing the values of attributes $\{A_1,\ldots,A_k\}$. By extending this notation, $T[A_1,\ldots,A_k]$ represents the subtuples of T containing the values of attributes $\{A_1,\ldots,A_k\}$, that is, the projection of T over $\{A_1,\ldots,A_k\}$,

	SSN	Name	ZIP	MaritalStatus	Sex	Disease
1			22030	married	F	hypertension
2			22030	married	F	hypertension
3			22030	single	M	obesity
4			22032	single	M	HIV
5			22032	single	M	obesity
6			22032	divorced	F	hypertension
7			22045	divorced	M	obesity
8			22047	widow	M	HIV
9			22047	widow	M	HIV
10			22047	single	F	obesity

Figure 1: An example of private table PT

keeping duplicates.

In the remainder of this chapter, the relational table storing the data to be protected is called *private table*, denoted PT. Each tuple t in PT reports data referred to a specific *respondent* (usually an individual). For instance, Figure 1 illustrates an example of private table over attributes `SSN`, `Name`, `Zip`, `MaritalStatus`, `Sex`, and `Disease` containing ten tuples. Here, the identity information, that is, attributes `SSN` and `Name`, have been removed. The attributes (columns) of PT can be classified as follows.

- *Identifiers.* Attributes that uniquely identify a respondent. For instance, attribute `SSN` uniquely identifies the person with which it is associated.

- *Quasi-identifiers.* Attributes that, in combination, can be linked with external information, reducing the uncertainty over the identities of all or some of the respondents to whom information in PT refers. For instance, the set of attributes `ZIP`, `MaritalStatus`, and `Sex` may represent a quasi-identifier that can be exploited for linking PT with an external information source that associates `ZIP`, `MaritalStatus`, and `Sex` with `Name` and `Address`.

- *Confidential attributes.* Attributes that contain sensitive information. For instance, attribute `Disease` can be considered sensitive.

- *Non confidential attributes.* Attributes that do not fall into any of the categories above, that is, they do not identify respondents, cannot be exploited for linking, and do not contain sensitive information. For instance, attribute `FavoriteColor` is a non confidential attribute.

Although all the explicit identifiers (e.g., `SSN`) are removed from a private table PT that is public or semi-public released, the privacy of the respondents is at risk, since quasi-identifiers can be used to link each tuple in PT to a limited number of respondents. The main goal of k-anonymity is therefore to protect the released data against possible re-identification of the respondents to whom the data refer.

The k-anonymity requirement is formally stated by the following definition [36].

Definition 2.2 (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.*

The k-anonymity requirement implicitly assumes that the data owner knows how many respondents each released tuple matches. This information can be known precisely only by explicitly linking the released data with externally available data. Since it is not realistic to assume that the data owner knows the data in external tables, the definition of k-anonymity translates the k-anonymity requirement in terms of the released data themselves. k-anonymity requires each respondent to be indistinguishable with respect to at least other $k-1$ respondents in the released table, as stated by the following definition.

Definition 2.3 (k-anonymity) *Let $T(A_1, \ldots, A_p)$ 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 $T[\mathsf{QI}]$ appears at least with k occurrences in $T[\mathsf{QI}]$.*

The definition of k-anonymity represents a sufficient (but not a necessary) condition for the k-anonymity requirement. As a matter of fact, if a subset of QI appears in a publicly available table, the combination of the released k-anonymous table with the public table never allows an adversary to associate each released tuple with less than k respondents. For instance, the private table in Figure 1 is 1-anonymous w.r.t. $\mathsf{QI} = \{\texttt{ZIP}, \texttt{MaritalStatus}, \texttt{Sex}\}$, since there are unique combinations of values of the quasi-identifying attributes (e.g., $\langle$22030, single, M$\rangle$). However, the same table is 2-anonymous with respect to $\mathsf{QI} = \{\texttt{MaritalStatus}\}$. To correctly enforce k-anonymity, it is then necessary to identify the quasi-identifiers. The identification of quasi-identifiers depends on which data are known to a potential adversary. Since different adversaries may have different knowledge, and it seems highly improbable to exactly know which data are available to each adversary, the original k-anonymity proposal [36] assumes to define a unique quasi-identifier for each private table, including all attributes that are possibly externally available.

2.1 Generalization and suppression

Although different data disclosure protection techniques have been developed (e.g., scrambling, swapping values, and adding noise), k-anonymity is typically enforced by combining two non-perturbative protecting techniques, called *generalization* and *suppression*, which have the advantage of preserving the truthfulness of the information, in contrast to other techniques that compromise the correctness of the information [13].

Generalization. Generalization consists in substituting the values in a column (or cell) of $\mathsf{PT}[\mathsf{QI}]$ with more general values. Each attribute A in PT is initially associated with a *ground domain* $D = dom(A, \mathsf{PT})$. The generalization maps each value $v \in D$ to a more general value $v' \in D'$, where D' is a *generalized domain* for D. Given a ground domain D, we distinguish two classes of generalizations depending on how the generalized domain D' is defined.

- *Hierarchy-based generalization.* This technique is based on the definition of a *generalization hierarchy* for each attribute in QI, where the most general value is at the root of the hierarchy and the leaves correspond to the most specific values (i.e., to the values in the ground domain for the attribute). A hierarchy-based generalization maps the values represented by the leaf vertices with one of their ancestor vertices at a higher level (see Section 3). As an example, attribute `ZIP` can be generalized by suppressing, at each generalization step, the right most digit.

- *Recoding-based generalization.* This technique is based on the *recoding into intervals* protection method [13] and typically assumes a total order relationship among values in the considered domain. The ground domain of each attribute in QI is partitioned into possibly disjoint intervals and each interval is associated with a label (e.g., the extreme interval values). A recoding-based generalization maps the values in the ground domain with the intervals they belong to (see Section 4).

It is interesting to note that one of the main differences between hierarchy-based and recoding-based generalizations is that while hierarchies need to be pre-defined, intervals for the recoding are usually computed at runtime during the generalization process. Accordingly, the k-anonymity algorithms adopting a hierarchy-based generalization will receive as an input the generalization hierarchies associated with attributes in QI, while the k-anonymity algorithms adopting a recoding-based generalization will have the additional task of defining the intervals.

Suppression. Suppression consists in removing from the private table a cell, a column, a tuple or a set thereof. The intuition behind the introduction of suppression is that the combined use of generalization and suppression can reduce the amount of generalization necessary to satisfy the k-anonymity constraint. As a matter of fact, when a limited number of outliers (i.e., tuples with less than k occurrences) would force a great amount of generalization, their suppression allows the release of a less general (more precise) table. For instance, the suppression of tuples 3, 6, 7, and 10 of the private table in Figure 1 satisfies 2-anonymity with respect to $\mathsf{QI} = \{\texttt{ZIP}, \texttt{MaritalStatus}, \texttt{Sex}\}$ without any generalization.

	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

Figure 2: Classification of k-anonymity techniques

2.2 Classification of k-anonymity techniques

Generalization and suppression can be applied at different granularity levels, which corresponds to different approaches and solutions to the k-anonymity problem, and introduce a taxonomy of k-anonymity solutions [11]. Such a taxonomy is orthogonal to the type of generalization (i.e., hierarchy- or recoding-based) adopted.

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 individual cells; as a result a generalized table may contain, for a specific column, values at different generalization levels.

  Generalizing at the cell level has the advantage of allowing the release of more specific values (since 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.

Figure 2 summarizes the different combinations of generalization and suppression at all possible granularity levels (including combinations where one of the two techniques is not adopted). 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 of all domain values to a unique value. 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 k-Anonymity with hierarchy-based generalization

The problem of k-anonymizing a private table by exploiting generalization and suppression has been widely studied and a number of approaches have been proposed. In this section we focus on those solutions that adopt a hierarchy-based generalization (together with suppression) to achieve k-anonymity. Before discussing such solutions, we first formally define how generalization can be performed by adopting a pre-defined hierarchy and we then discuss the problem complexity.

For each ground domain $D \in \mathcal{D}$, we assume the existence of a set of generalized domains. The set of all ground and generalized domains is denoted Dom. The relationship between a ground domain and domains generalization of it is formally defined as follows.

Definition 3.1 (Domain generalization relationship) *Let* Dom *be a set of ground and generalized domains. A* domain generalization relationship*, denoted* $\leq_{\mathsf{D}}$*, is a partial order relation on* Dom *that satisfies the following conditions:*

C1: $\forall D_i, D_j, D_z \in \mathsf{Dom}$: $D_i \leq_{\mathsf{D}} D_j, D_i \leq_{\mathsf{D}} D_z \Rightarrow D_j \leq_{\mathsf{D}} D_z \vee D_z \leq_{\mathsf{D}} D_j$

C2: *all maximal elements of* Dom *are singleton.*

Condition **C1** states that, for each domain D_i, the set of its generalized domains is totally ordered and each D_i has at most *one* direct generalized domain D_j. This condition 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 the domain generalization relationship implies the existence, for each domain $D \in \mathsf{Dom}$, of a totally ordered hierarchy, called *domain generalization hierarchy* and denoted DGH_D. Each DGH_D can be graphically represented as a chain of vertices, where the top element corresponds to the singleton generalized domain, and the bottom element corresponds to D.

Analogously, a *value generalization relationship*, denoted $\leq_{\mathsf{V}}$, can also be defined that associates with each value $v_i \in D_i$ a unique value $v_j \in D_j$, where D_j is the direct generalization of D_i. The definition of the value generalization relationship implies the existence, for each domain $D \in \mathsf{Dom}$, of a partially ordered hierarchy,

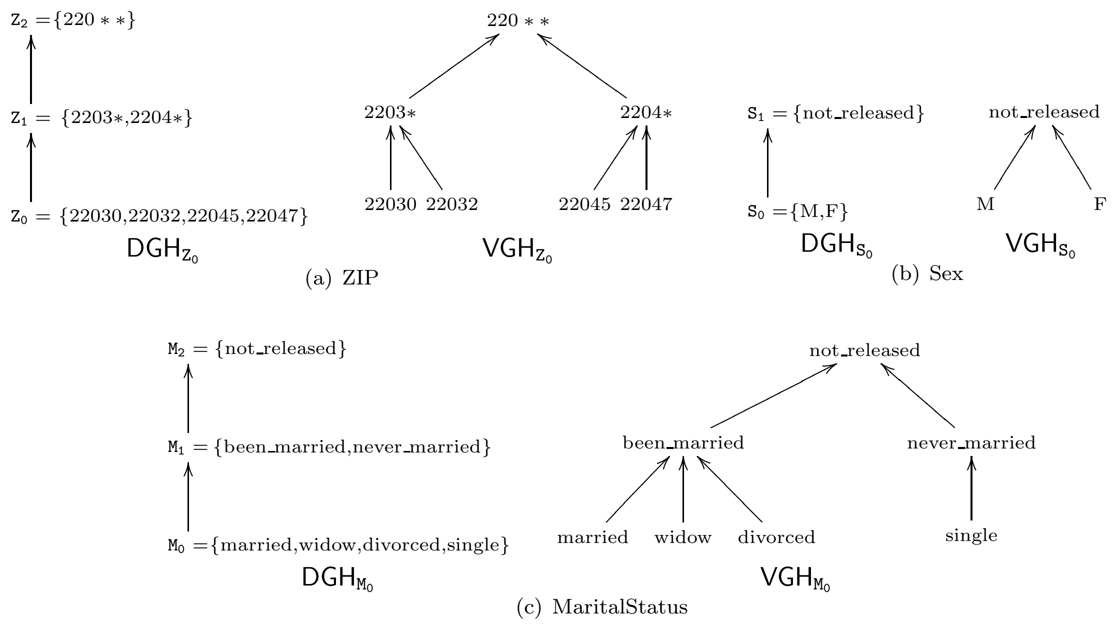
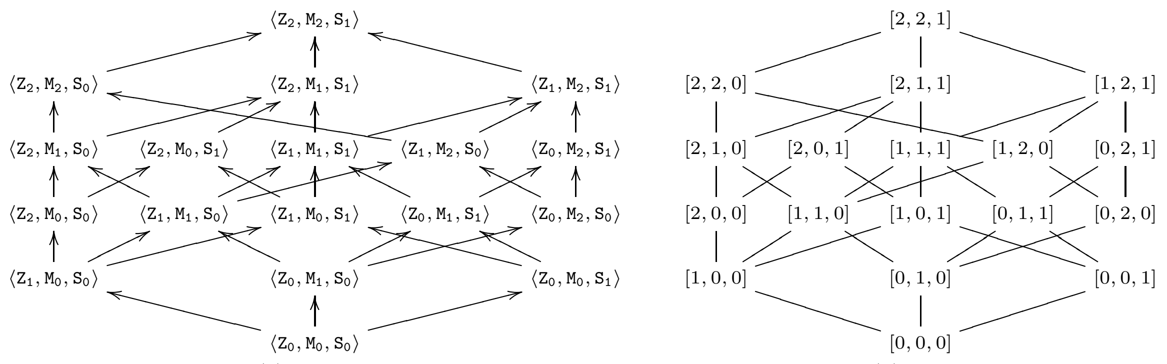


Figure 3: Examples of domain and value generalization hierarchies

called *value generalization hierarchy* and denoted VGH_D. Each VGH_D can be graphically represented as a tree, where the root element corresponds to the unique value in the top domain in DGH_D, and the leaves correspond to the values in D. Figure 3 shows an example of domain and value generalization hierarchies for attributes `Zip`, `Sex`, and `MaritalStatus`. Attribute `ZIP`, with domain $\mathtt{Z}_0$={22030, 22032, 22045, 22047}, is generalized by suppressing, at each step, the right-most digit. Attribute `Sex`, with domain $\mathtt{S}_0$={M, F}, is generalized to the not_released value (domain $\mathtt{S}_1$). Attribute `MaritalStatus`, with domain $\mathtt{M}_0$={single, married, divorced, widow}, is first generalized to the been_married and never_married values (domain $\mathtt{M}_1$), and then to the not_released value (domain $\mathtt{M}_2$).

Since most k-anonymity approaches work on sets of attributes, the definition of domain generalization hierarchy is extended to tuples of domains. A domain tuple $DT = \langle D_1, \ldots, D_n \rangle$ is an ordered set of domains composed through the Cartesian product to impose coordinate-wise order among domains. Since each domain $D_i \in DT$ is characterized by a total order domain generalization hierarchy DGH_{D_i}, domain tuple DT is characterized by a domain generalization hierarchy DGH_{DT} defined as DGH_{DT}=$\mathsf{DGH}_{D_1} \times \ldots \times \mathsf{DGH}_{D_n}$. DGH_{DT} is a lattice where the bottom element is DT and the top element is the tuple composed of all top elements in DGH_{D_i}, $i = 1, \ldots, n$. Each path from the bottom to the top element in DGH_{DT} is called *generalization strategy* and represents a possible strategy for generalizing quasi-identifier QI= $\{A_1, \ldots, A_n\}$, where $dom(A_i) = D_i$, $i = 1, \ldots, n$. Figure 4(a) illustrates the domain generalization hierarchy built on the domain tuple $\langle \mathtt{Z}_0, \mathtt{M}_0, \mathtt{S}_0 \rangle$ and obtained through the combination of the domain generalization hierarchies


$\langle Z_2, M_2, S_1\rangle$
$\langle Z_2, M_2, S_0\rangle$
$\langle Z_2, M_1, S_1\rangle$
$\langle Z_1, M_2, S_1\rangle$
$\langle Z_2, M_1, S_0\rangle$
$\langle Z_2, M_0, S_1\rangle$
$\langle Z_1, M_1, S_1\rangle$
$\langle Z_1, M_2, S_0\rangle$
$\langle Z_0, M_2, S_1\rangle$
$\langle Z_2, M_0, S_0\rangle$
$\langle Z_1, M_1, S_0\rangle$
$\langle Z_1, M_0, S_1\rangle$
$\langle Z_0, M_1, S_1\rangle$
$\langle Z_0, M_2, S_0\rangle$
$\langle Z_1, M_0, S_0\rangle$
$\langle Z_0, M_1, S_0\rangle$
$\langle Z_0, M_0, S_1\rangle$
$\langle Z_0, M_0, S_0\rangle$
[2, 2, 1]
[2, 2, 0]
[2, 1, 1]
[1, 2, 1]
[2, 1, 0]
[2, 0, 1]
[1, 1, 1]
[1, 2, 0]
[0, 2, 1]
[2, 0, 0]
[1, 1, 0]
[1, 0, 1]
[0, 1, 1]
[0, 2, 0]
[1, 0, 0]
[0, 1, 0]
[0, 0, 1]
[0, 0, 0]

(a) DGH_{DT} (b) VL_{DT}

Figure 4: Domain generalization hierarchy (a) and distance vector hierarchy (b) for DT=$\langle Z_0, M_0, S_0\rangle$

ZIP	**MaritalStatus**	**Sex**
220 ∗ ∗	not_released	not_released
220 ∗ ∗	not_released	not_released
220 ∗ ∗	not_released	not_released
220 ∗ ∗	not_released	not_released
220 ∗ ∗	not_released	not_released
220 ∗ ∗	not_released	not_released
220 ∗ ∗	not_released	not_released
220 ∗ ∗	not_released	not_released
220 ∗ ∗	not_released	not_released
220 ∗ ∗	not_released	not_released

(a) $\langle Z_2, M_2, S_1\rangle$

ZIP	**MaritalStatus**	**Sex**
220 ∗ ∗	been_married	F
220 ∗ ∗	been_married	F
220 ∗ ∗	never_married	M
220 ∗ ∗	never_married	M
220 ∗ ∗	never_married	M
220 ∗ ∗	been_married	F
220 ∗ ∗	been_married	M
220 ∗ ∗	been_married	M
220 ∗ ∗	been_married	M

(b) $\langle Z_2, M_1, S_0\rangle$

ZIP	**MaritalStatus**	**Sex**
2203∗	been_married	F
2203∗	been_married	F
2203∗	never_married	M
2203∗	never_married	M
2203∗	never_married	M
2203∗	been_married	F
2204∗	been_married	M
2204∗	been_married	M
2204∗	been_married	M

(c) $\langle Z_1, M_1, S_0\rangle$

Figure 5: Examples of generalized tables

DGH_{Z_0}, DGH_{M_0}, DGH_{S_0} illustrated in Figure 3.

Given a private table PT and its quasi-identifier QI, the application of generalization and suppression produces a *generalized table T* containing less information (more general values and less tuples) than PT, formally defined as follows.

Definition 3.2 (Generalized table) *Let $T_i(A_1,\ldots,A_n)$ and $T_j(A_1,\ldots,A_n)$ 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 $T_i \preceq T_j$, iff:*

1. *$|T_j| \leq |T_i|$;*

2. *$\forall A \in \{A_1, \ldots, A_n\}$: $dom(A, T_i) \leq_{\mathsf{D}} dom(A, T_j)$;*

3. *it is possible to define an injective mapping associating each tuple $t_j \in T_j$ with a tuple $t_i \in T_i$, such that $t_i[A] \leq_{\mathsf{V}} t_j[A]$, for all $A \in \{A_1, \ldots, A_n\}$.*

Given a private table PT and its quasi-identifier QI, there may exist different generalized tables that satisfy k-anonymity. Among all possible generalized tables, we are interested in a table that minimizes information loss, meaning that is k-anonymous and does not remove, through generalization and suppression, more information than necessary. As an example, consider the private table in Figure 1 with $\mathsf{QI} = \{\texttt{Zip}, \texttt{MaritalStatus}, \texttt{Sex}\}$ and suppose that $k = 3$. The generalized table in Figure 5(a) corresponding to the top element $\langle \texttt{Z}_2, \texttt{M}_2, \texttt{S}_1 \rangle$ in DGH_{DT}, which is composed of ten identical tuples (all the cells in each column have been generalized to the same value), is 3-anonymous but it removes more information than necessary. In fact, the generalized table corresponding to $\langle \texttt{Z}_2, \texttt{M}_1, \texttt{S}_0 \rangle$ in Figure 5(b) is 3-anonymous and it is more specific than the table corresponding to $\langle \texttt{Z}_2, \texttt{M}_2, \texttt{S}_1 \rangle$, since it contains more specific values for $\texttt{MaritalStatus}$ and $\texttt{Sex}$ attributes.

The goal of k-anonymity is to compute a generalized k-anonymous table, while maintaining as much information as possible. This concept is captured by the definition of *k-minimal generalization.* The formal definition of k-minimal generalization requires the introduction of the concept of *distance vector*.

Definition 3.3 (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 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}.*

The dominance relationship $\leq$ on integers is then extended on the set of distance vectors as follows. Given two distance vectors $DV = [d_1, \ldots, d_n]$ and $DV' = [d'_1, \ldots, d'_n]$, $DV \leq DV'$ iff $d_i \leq d'_i$, $i = 1, \ldots, n$. For each $\mathsf{PT}[\mathsf{QI}]$ defined on domain tuple DT, we can therefore define a partially ordered *hierarchy of distance vectors*, denoted VL_{DT}, containing the distance vectors of all the generalized tables of $\mathsf{PT}[\mathsf{QI}]$. Each VL_{DT} can be graphically represented as an isomorphic lattice to DGH_{DT}. The height of a distance vector DV in VL_{DT}, denoted **height**(DV,VL_{DT}), is equal to the sum of the elements in DV. The height of VL_{DT} corresponds to the height of its top element. As an example, Figure 4(b) illustrates the VL_{DT} lattice defined on $DT = \langle \texttt{Z}_0, \texttt{M}_0, \texttt{S}_0 \rangle$ and therefore isomorphic to the DGH in Figure 4(a).

Note that given an element in DGH_{DT} and the corresponding generalized table, Definition 3.2 allows any amount of suppression to achieve k-anonymity. However, we are interested in a table obtained by suppressing the minimum number of tuples necessary to achieve k-anonymity at a given level of generalization.

It is worth noting that, given a hierarchy of distance vectors VL_{DT}, the generalized tables corresponding to the distance vectors at the same height and ensuring minimality in suppression, may suppress a different number of tuples. Since the joint use of generalization and suppression permits to maintain as much information as possible in the released table, the question is whether it is better to generalize, loosing data precision, or to suppress, loosing completeness. The compromise proposed by Samarati [36] consists in establishing a threshold MaxSup to the maximum number of tuples that can be suppressed; within this threshold, generalization decides minimality. Given this threshold on the number of tuples that can be suppressed, a *k-minimal generalization* is defined as follows.

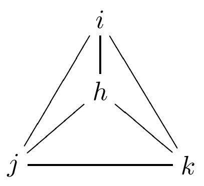
Definition 3.4 (k-minimal generalization) *Let T_i and T_j be two tables such that $T_i \preceq T_j$, and let* MaxSup *be the specified threshold of acceptable suppression. T_j is said to be a k-*minimal generalization *of table T_i iff:*

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 $\Rightarrow \neg(DV_{i,z} < DV_{i,j})$.*

This definition states that a generalization T_j of a table T_i is k-minimal if it satisfies k-anonymity (condition 1), it does not suppress more tuples than MaxSup (condition 2), and there does not exist another generalization of T_i satisfying these conditions and characterized by a distance vector lower than that of T_i (condition 3). As an example, consider the private table in Figure 1 with QI = {`Zip`, `MaritalStatus`, `Sex`} and suppose that $k = 3$ and MaxSup=2. The generalized table in Figure 5(c) is a k-minimal generalization for PT. As a matter of fact, it is 3-anonymous (condition 1), it suppresses less than 2 tuples (condition 2), and the generalized tables characterized by distance vectors lower than $[1, 1, 0]$ do not satisfy these two conditions: the generalized table corresponding to $[1, 0, 0]$ requires to suppress at least 7 tuples; the generalized tables corresponding to $[0, 1, 0]$ and $[0, 0, 0]$ require to suppress all the tuples.

Given a private table PT, there may exist more than one k-minimal generalization since DGH_{DT} is a lattice and two solutions may be non-comparable. Furthermore, the definition of k-minimal generalization only captures the concept that the least amount of generalization and the minimal required suppression to achieve k-anonymity is applied. Different *preference criteria* can be applied in choosing a preferred minimal generalization, among which [36]:

- *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);



T

	i	j	k	h
(i,j)	1	1	0	0
(i,k)	1	0	1	0
(j,k)	0	1	1	0
(i,h)	1	0	0	1
(j,h)	0	1	0	1
(k,h)	0	0	1	1

$\Longrightarrow$

T_A (T anonymized)

	i	j	k	h
(i,j)	$*$	$*$	$*$	0
(i,k)	$*$	$*$	$*$	0
(j,k)	$*$	$*$	$*$	0
(i,h)	$*$	$*$	$*$	1
(j,h)	$*$	$*$	$*$	1
(k,h)	$*$	$*$	$*$	1

Figure 6: An example of construction of T from a graph, as described in the proof of Theorem 3.1

- *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.

3.1 Problem complexity

All the models with hierarchies investigated in the literature (**AG_TS**, **AG_**, **CG_**, and **_CS**), as well as **_AS**, are NP-hard. The complexity results of all these models with hierarchies derive from the NP-hardness of **_CS** and **_AS**. To formally prove the NP-hardness of such problems, and without loss of generality, each table T can be seen as a matrix of m rows (tuples) and n columns (attributes). Each row x_i, $i = 1, \ldots, m$, is a n-dimensional string defined on an alphabet Σ, where each $c \in \Sigma$ corresponds to an attribute value. For instance, the projection over QI of the private table in Figure 1 is a matrix with 10 rows and each row is a string of three characters of the alphabet Σ= {22030, 22032, 22045, 22047, married, single, divorced, widow, F,M}.

The NP-hardness of **_AS** has been proved for $|\Sigma| \geq 2$ [33], while the NP-hardness of the **_CS** problem for $|\Sigma| \geq 3$ has been proved with a reduction from the "Edge Partition into Triangles" problem [3], which is an improvement on the NP-hardness proved for **_AS** when the size of alphabet Σ is equal to the number n of attributes. 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 that can be suppressed. Nevertheless, the computational complexity of the general **AG_TS** model (where the number of tuples that can be suppressed is no more a constant value given as input, but it should be minimized as part of the solution), is still an open issue. Note that the decisional versions of **_AS**, **_CS**, **AG_**, **AG_TS**, and **CG_** are obviously in NP [3].

We now give the proof of NP-hardness for **_CS** and we briefly describe the result for **_AS**.

T'

		i	j	k	h	i	j	k	h	i	j	k	h
001	(i,j)	1	2	0	0	1	2	0	0	2	1	0	0
010	(i,k)	1	0	2	0	2	0	1	0	1	0	2	0
011	(j,k)	0	1	2	0	0	2	1	0	0	2	1	0
100	(i,h)	2	0	0	1	1	0	0	2	1	0	0	2
101	(j,h)	0	2	0	1	0	1	0	2	0	2	0	1
110	(k,h)	0	0	2	1	0	0	2	1	0	0	1	2

$\Longrightarrow$

T'_A (T' anonymized)

	i	j	k	h	i	j	k	h	i	j	k	h
(i,j)	$*$	$*$	$*$	0	$*$	$*$	$*$	0	$*$	$*$	$*$	0
(i,k)	$*$	$*$	$*$	0	$*$	$*$	$*$	0	$*$	$*$	$*$	0
(j,k)	$*$	$*$	$*$	0	$*$	$*$	$*$	0	$*$	$*$	$*$	0
(i,h)	$*$	$*$	$*$	1	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$
(j,h)	$*$	$*$	$*$	1	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$
(k,h)	$*$	$*$	$*$	1	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$

Figure 7: An example of construction of T' from the graph in Figure 6, as described in the proof of Theorem 3.1

Theorem 3.1 *The* **_CS** *3-anonymity problem is NP-hard even for a ternary alphabet.*

Proof. The proof [3] is a reduction from the NP-hard problem of "Edge partition into triangles" [26], which can be formulated as follows: *given a graph $G = (V, E)$ with $|E| = 3m$ for some integer m, can the edges of G be partitioned into m triangles with disjoint edges?*

To facilitate the description of the reduction from "Edge partition into triangles", the proof will first describe the simpler reduction from "Edge partition into triangles and 4-stars" and then it will describe a reduction from "Edge partition into triangles". The overall proof therefore consists in two main steps. First, we show that, given a graph $G = (V, E)$ with $|E| = 3m$, we can construct a table T such that an optimal 3-anonymous solution for T costs at most $9m$ (i.e., T is obtained by suppressing at most $9m$ cells from PT) if and only if G can be partitioned into a collection of m disjoint triangles and 4-stars (a 4-star is a graph with 3 edges and 4 vertices, having one degree 3 and the others degree 1). Second, we show how to define a table T' from G such that an optimal 3-anonymous solution for T' costs at most $9m\lceil\log_2(3m+1)\rceil$ if and only if G can be partitioned into a collection of m disjoint triangles. In the following, we use character $*$ to denote suppressed cells.

Edge partition into triangles and 4-stars. Given a graph $G = (V, E)$ with $|E| = 3m$ and $|V| = n$, we can construct a table T that contains $3m$ rows (one for each edge) and n attributes (one for each vertex). The row corresponding to edge (i, j) has 1 in positions corresponding to attributes i and j; 0 otherwise.

Suppose that G can be partitioned into a collection of m disjoint triangles and 4-stars. Consider first a triangle with vertices i, j, and k. By suppressing the cells in the rows corresponding to edges (i, j), (i, k), and (j, k) and columns i, j, and k, we get 3 identical rows containing 3 $*$s each and 0 anywhere else. Consider now a 4-star with vertices i, j, k, and h and edges (i, h), (j, h), and (k, h). By suppressing the cells in the rows corresponding to edges (i, h), (j, h), and (k, h) and columns i, j, and k, we get 3 identical rows containing 3 $*$s each, with a single 1 (corresponding to attribute h) and 0 anywhere else. Since for each triangle and for each 4-star we obtain three identical generalized tuples by suppressing 9 cells, table T is 3-anonymous of cost $9m$. As a simple example, consider the tables in Figure 6, where the first three rows correspond to a triangle and the last three rows to a 4-star. The anonymized table on the right side shows the suppressed cells.

Suppose to have a 3-anonymous table T_A of cost at most $9m$ that is a generalization of T. We want to show that the graph G corresponding to T can be partitioned into a collection of m disjoint triangles and 4-stars. Since G is a simple graph and therefore there are no edges with the same end vertices, any 3 rows in the corresponding table T are distinct and differ in at least 3 positions; otherwise there would be multiple edges. This implies that each modified row in the anonymized table T_A contains at least 3 $*$s, as explained above, and the overall number of $*$s in T_A is at least $9m$. From the previous hypothesis, we have that the cost of T_A is exactly $9m$, and each row of T_A contains 3 $*$s. Therefore T_A contains only clusters of 3 identical rows. In fact, any tuple in the anonymized table T_A belonging to a group of more than 3 equal tuples would contain at least 4 $*$s. The corresponding graph contains then only edges composing either triangles or 4-stars. In fact, while each modified row in a triangle has 3 $*$s and 0 elsewhere (see Figure 6), each modified tuple in a 4-star contains 3 $*$s, a single 1, and 0 elsewhere (see Figure 6). No other configuration is possible. The overall solution corresponds to a partition of edges into triangles and 4-stars.
We now show the reduction from the "Edge partition into triangles" problem.
Edge partition into triangles. To prove that **_CS** 3-anonymity is reduced from "Edge partition into triangles", the informal idea is to make 4-stars costing more $*$s then triangles. A slightly different construction of the table from a graph $G = (V, E)$ with $|E| = 3m$ and $|V| = n$ is then required. Let $t = \lceil \log_2(3m + 1) \rceil$. We define a new table T' where each row has t blocks of n columns. Consider an arbitrary ordering of the edges in E and express the rank of an edge $e = (v_1, v_2)$ in binary notation $b_1 b_2 \dots b_t$. In the tuple corresponding to edge e, each block has 0 in all columns but the columns corresponding to the vertices v_1 and v_2, for which two configurations are possible: $conf_0$ has a 1 in column v_1 and a 2 in column v_2, and $conf_1$ has a 2 in column v_1 and a 1 in column v_2. The l-th block in the row corresponding to e has configuration $conf_{b_l}$. For instance, with respect to the graph shown in Figure 6, the edges are ranked from 1 (001 in binary notation) to 6 (110 in binary notation). Figure 7 illustrates table T' corresponding to the graph. We now show that the cost of the optimal 3-anonymity solution of T' is at most $9mt$ if and only if E can be partitioned into m disjoint triangles.

Suppose that E can be partitioned into m disjoint triangles. As previously discussed, every triangle in such a partition corresponds to a cluster with $3t$ $*$s in each tuple. Thus, we get a 3-anonymity solution of cost $9mt$.

Suppose to have a 3-anonymity solution of cost at most $9mt$. Again, any 3 tuples differ in at least $3t$ columns and the cost of any solution is at least $9mt$. Hence, the solution cost is exactly $9mt$ and each modified row has exactly $3t$ $*$s. Thus, each cluster has exactly 3 equal rows. We now show, by contradiction, that the corresponding edges form a triangle and cannot be a 4-star. Suppose, on the contrary, that the 3 rows form a 4-star. Let h be the common vertex and consider the integer $\{1, 2\}$ assigned by each of the 3 edges to h in $conf_0$. Two of the three edges must have assigned the same digit to h. However, since these two edges differ in rank, there must exist at least one block where they have a differen configuration (and therefore, a different digit in the column corresponding to h). Thus, the rows corresponding to the 3 edges contain an additional $*$ corresponding to column h in addition to the $3t$ $*$s corresponding to the remaining 3 columns (e.g., see the last 3 tuples of Figure 7). This contradicts the fact that each row has exactly $3t$ $*$s. Therefore, E can be partitioned into m disjoint triangles. ■

The corresponding theorem for **_AS** has been proved with a proof similar to the one that was proposed for **_CS** [33]. The result is the following.

Theorem 3.2 *The* **_AS** *problem for* $k > 2$ *is* NP*-hard for any alphabet* Σ *such that* $|\Sigma| \geq 2$.

The proof is a reduction from the "k-dimensional perfect matching" problem: *given a* k*-hypergraph* $G = (V, E)$ *is there a subset* S *of* G *with* $|V|/k$ *hyperedges such that each vertex of* G *is contained in exactly one hyperedge of* S*?*

By orchestrating Theorems 3.1 and 3.2 and the observations given in this section, we have the following result.

Corollary 3.1 *Problems* **_AS**, **_CS**, **AG_**, **AG_TS**, *and* **CG_** *are* NP*-hard for* $k > 2$, *and for any alphabet* Σ *such that* $|\Sigma| > 2$.

3.2 Algorithms for k-anonymity

Since the hierarchy-based generalization has been proposed first, the solutions based on such a type of hierarchy have been studied in more depth than the ones based on the recoding-based generalization. Here, we present two exact algorithms, both belonging to the **AG_TS** class, and both aimed at finding a k-minimal solution for a given PT [29, 36]. We also briefly describe the most important approximation algorithms proposed for k-anonymizing a private table.

Besides the two exact algorithms described in the following, Sweeney [38] proposed an exact algorithm (**AG_TS** class) that exhaustively examines all potential generalizations (with suppression) for identifying a minimal one satisfying the k-anonymity requirement. This approach is however impractical for large datasets.

3.2.1 Samarati's Algorithm

The first algorithm introduced for k-anonymizing a private table PT was proposed along with the definition of k-anonymity [36]. This algorithm follows the minimum absolute distance criterion (see Section 3) to determine a k-minimal generalization of PT.

Given a private table PT such that $|\mathsf{PT}| \geq k$,[1] the algorithm restricts its analysis to $\mathsf{PT}[\mathsf{QI}]$, where $\mathsf{QI}=\{A_1,\dots,A_n\}$ is defined on the domain tuple $DT = \langle D_1, \dots, D_n\rangle$ (i.e., $dom(A_i, \mathsf{PT}) = D_i$, $i = 1, \dots, n$).

Given a domain generalization hierarchy DGH_{DT}, each path from the bottom to the top element in DGH_{DT} represents a generalization strategy that is characterized by a locally minimal generalization defined as follows.

Definition 3.5 (Locally minimal generalization) *Let* DGH_{DT} *be a domain generalization hierarchy, k be the anonymity threshold, and* MaxSup *be the suppression threshold. Any path in* DGH_{DT} *is characterized by a* locally minimal generalization*, which is the lowest generalization in the path that satisfies k-anonymity and suppresses a number of tuples lower than* MaxSup.

A locally minimal generalization represents the k-anonymous generalization that maintains the most information with respect to a given generalization strategy.

Example 3.1 *Consider the table in Figure 1 and the domain generalization and distance vector hierarchies in Figure 4. If $k = 3$ and* $\mathsf{MaxSup} = 2$*, along path* $\langle Z_0, M_0, S_0\rangle \rightarrow \langle Z_1, M_0, S_0\rangle \rightarrow \langle Z_2, M_0, S_0\rangle \rightarrow \langle Z_2, M_1, S_0\rangle \rightarrow \langle Z_2, M_2, S_0\rangle \rightarrow \langle Z_2, M_2, S_1\rangle$*, the generalized table corresponding to* $\langle Z_2, M_1, S_0\rangle$ *(see Figure 5(b)) is a locally minimal generalization.*

The definition of locally minimal generalization can be exploited to compute k-minimal generalizations, on the basis of the following theorem.

Theorem 3.3 [36] *Let* $T_i(A_1,\dots,A_n)=\mathsf{PT}[\mathsf{QI}]$ *be a table to be generalized and let* $DT = \langle D_1, \dots, D_n\rangle$ *($D_z = dom(A_z, T_i)$, $z = 1, \dots, n$) be the domain tuple of its attributes. Every k-minimal generalization of T_i is a locally minimal generalization for some strategy in* DGH_{DT}.

Note that a locally minimal generalization may not correspond to a k-minimal generalization, since a generalization may be locally minimal along one path but not along another one. With respect to Example 3.1, the locally minimal generalization $\langle \mathtt{Z}_2, \mathtt{M}_1, \mathtt{S}_0\rangle$ is not a k-minimal generalization, since the generalized table corresponding to $\langle \mathtt{Z}_1, \mathtt{M}_1, \mathtt{S}_0\rangle$ (see Figure 5(c)) is k-anonymous, suppresses a number of tuples lower than MaxSup and contains more specific values for the `Zip` attribute. Exploiting Theorem 3.3, a naive method to compute a k-minimal generalization consists in finding all locally minimal generalizations by visiting all the paths in DGH_{DT}, starting from the bottom and stopping at the first generalization that both satisfies k-anonymity and suppresses a number of tuples lower than MaxSup. By discarding the locally minimal generalizations that are dominated by other locally minimal generalizations, only k-minimal generalizations of

[1]Note that if $1 \leq |\mathsf{PT}| < k$, a k-anonymous version of PT does not exist.

Algorithm 3.1 (Samarati's Algorithm)

```
INPUT
$T$ = PT[QI]: private table
$k$: anonymity requirement
MaxSup: suppression threshold
$\forall A \in$ QI, DGH$_A$: domain generalization hierarchies

OUTPUT
$sol$: distance vector corresponding to the $k$-minimal generalization of PT[QI]

MAIN
Let VL$_{DT}$ be the distance vector hierarchy
/* matrix initialization phase */
$Outlier := \emptyset$ /* set of outlier values */
order($T$) /* order the tuples in the table */
$V := \emptyset$ /* set of distinct tuples */
$counter[t_1] := 1$ /* first tuple */
for $i$:=2...$|T|$ do
    if $t_i \neq t_{(i-1)}$ then /* if the $i$-th is different from the $(i-1)$th tuple */
      $V := V \cup \{t_i\}$
      $counter[t_i] := 1$
      if $counter[t_{(i-1)}] < k$ then $Outlier := Outlier \cup \{t_{(i-1)}\}$
    else $counter[t_i] := counter[t_i] + 1$
for $j$:=1...$|V|$ do
    for $i$:=1...$|Outlier|$ do
        VT[$i$,$j$] := distance($v_j$,$outlier_i$)
/* binary search phase */
$low := 0$
$high :=$ height($\top$,VL$_{DT}$)
$sol := \top$
while $low<high$ do
        $h := \lfloor \frac{low+high}{2} \rfloor$
        $Vectors := \{vec|$height($vec$,VL$_{DT}$)$=h\}$
        $reach\_k := false$
        while $Vectors \neq \emptyset \land reach\_k \neq true$ do
                let $vec$ be a vector in $Vectors$
                $Vectors := Vectors - \{vec\}$
                if Satisfy($T$,$vec$,VT,$|Outlier|$,$|V|$) then
                    $sol := vec$
                    $reach\_k := true$
        if $reach\_k = true$ then $high := h$
        else $low := h+1$
return($sol$)

SATISFY($T$,$vec$,VT,$rows$,$columns$)
$req\_sup := 0$
for $i$:=1...$rows$ do
    $c := 0$
    for $j$:=1...$columns$ do
        if VT[$i$,$j$]$\leq vec$ then
            $c := c + counter[v_j]$
    if $c<k$ then $req\_sup := req\_sup + c$
if $req\_sup<$MaxSup then return($true$)
else return($false$)
```

Figure 8: Algorithm that computes a k-minimal generalization [36]

PT are maintained. However, since the number of paths in DGH$_{DT}$ may be very high, such a naive strategy is not viable. The key idea exploited by Algorithm 3.1 in Figure 8 to reduce the computational time is that the number of tuples that need to be suppressed to satisfy k-anonymity decreases while going up in a path.

Theorem 3.4 [36] *Let T_i=PT[QI] be a table to be generalized and T_j and T_z be two of its generalizations (i.e., $T_i \preceq T_j$ and $T_i \preceq T_z$) enforcing minimal required suppression. Then, $DV_{i,j} < DV_{i,z} \Rightarrow |T_j| \leq |T_z|$.*

From Theorem 3.4, it follows that given two generalized tables T_j and T_z of PT such that $DV_{i,j} < DV_{i,z}$, if T_j is k-anonymous and suppresses a number of tuples lower than MaxSup, also T_z is k-anonymous and suppresses a number of tuples lower than MaxSup. Furthermore, if there is no a solution that guarantees

		1	2	3	4	5	6	7
		t_1/t_2	t_3	t_4/t_5	t_6	t_7	t_8/t_9	t_{10}
1	t_1/t_2	[0,0,0]	[0,2,1]	[1,2,1]	[1,1,0]	[2,1,1]	[2,1,1]	[2,2,0]
2	t_3	[0,2,1]	[0,0,0]	[1,0,0]	[1,2,1]	[2,2,0]	[2,2,0]	[2,0,1]
3	t_4/t_5	[1,2,1]	[1,0,0]	[0,0,0]	[0,2,1]	[2,2,0]	[2,2,0]	[2,0,1]
4	t_6	[1,1,0]	[1,2,1]	[0,2,1]	[0,0,0]	[2,0,1]	[2,1,1]	[2,2,0]
5	t_7	[2,1,1]	[2,2,0]	[2,2,0]	[2,0,1]	[0,0,0]	[1,1,0]	[1,2,1]
6	t_8/t_9	[2,1,1]	[2,2,0]	[2,2,0]	[2,1,1]	[1,1,0]	[0,0,0]	[0,2,1]
7	t_{10}	[2,2,0]	[2,0,1]	[2,0,1]	[2,2,0]	[1,2,1]	[0,2,1]	[0,0,0]

Figure 9: An example of the VT matrix computed with respect to the table in Figure 1

k-anonymity suppressing a number of tuples lower than MaxSup at height h, there cannot exist a solution at height $h' < h$ that guarantees it. Such a consequence is exploited by the Algorithm 3.1 to compute a k-minimal generalization for PT[QI], by performing a binary search on the heights of distance vectors.

The algorithm takes as input the projection T over quasi-identifying attributes of the private table PT, the anonymity constraint k, the suppression threshold MaxSup, and the domain generalization hierarchies for the attributes composing the quasi-identifier. During the initialization phase, the algorithm first builds VL_{DT} (i.e., the distance vector lattice based on the domain generalization hierarchies) and a matrix VT, with a row for each distinct tuple t in T with less than k occurrences and a column for each distinct tuple in T. Each entry $\mathsf{VT}[x, y]$ of the matrix contains the distance vector $[d_1, \ldots, d_n]$ between tuples $x = \langle v_1 \ldots v_n\rangle$ and $y = \langle v'_1 \ldots v'_n\rangle$, where d_i, $i = 1, \ldots, n$, is the distance from the domain of v_i and v'_i to the domain to which they generalize to the same value. The binary search phase visits the lattice VL_{DT} as follows. Variable *high* is first initialized to the height of VL_{DT} and all the generalized tables corresponding to distance vectors at height $\lfloor \frac{high}{2} \rfloor$ are first evaluated. If there is at least one generalized table satisfying k-anonymity and suppressing a number of tuples lower than MaxSup, the algorithm evaluates the generalized tables at height $\lfloor \frac{high}{4} \rfloor$, otherwise those at height $\lfloor \frac{3high}{4} \rfloor$, and so on. The algorithm stops when it reaches the lowest height in VL_{DT} where there is at least a generalized table satisfying k-anonymity and the MaxSup constraint.

To check whether or not at a given height h there is a table that satisfies k-anonymity and the MaxSup suppression threshold, for each distance vector *vec* at height h the algorithm calls function **Satisfy**. Function **Satisfy** computes the number of tuples *req_sup* that need to be suppressed for achieving k-anonymity in table T_{vec} corresponding to the *vec* distance vector. For each row x in VT, **Satisfy** computes the sum c of the occurrences of tuples y (column of the matrix) such that $\mathsf{VT}[x, y] \leq vec$. As a matter of fact, all tuples such that $\mathsf{VT}[x, y] \leq vec$ will be generalized to the same value in T_{vec} as x. Therefore, if $c < k$, c is added to *req_sup* since the considered tuples will be outliers for T_{vec} and will be therefore suppressed to satisfy k-anonymity. If *req_sup* is lower than MaxSup, T_{vec} satisfies k-anonymity and the MaxSup threshold and **Satisfy** returns *true*; otherwise it returns *false*.

Example 3.2 *Consider the private table in Figure 1 with* QI $=$ {`Zip`, `MaritalStatus`, `Sex`} *and the corresponding hierarchies in Figure 4. Also, suppose that* $k = 3$, *and* MaxSup*=2. The algorithm builds matrix* VT *illustrated in Figure 9, which is composed of seven rows (all quasi-identifying values appear less than 3 times in* PT*) and 7 columns since there are 7 distinct values for* QI *in* PT *(*$t_1 = t_2$*,* $t_4 = t_5$*, and* $t_8 = t_9$*).*

By exploiting the hierarchies in Figure 4, the algorithm computes the distance vectors between pairs of tuples. For instance, VT[1,4]=$[1, 1, 0]$*, which is the distance vector of the generalized table at which* t_1=t_2=⟨*22030, married, F*⟩ *and* t_6=⟨*22032, divorced, F*⟩ *are generalized to the same value* ⟨*2203*, been_married, F*⟩*.*

Once VT *has been computed, the algorithm starts with the binary search phase and evaluates first the generalizations at height* $\lfloor 5/2 \rfloor = 2$*, that is, the tables corresponding to distance vectors* $[2, 0, 0]$*,* $[1, 1, 0]$*,* $[1, 0, 1]$*,* $[0, 1, 1]$*, and* $[0, 2, 0]$*. The table corresponding to* $[1, 1, 0]$ *satisfies* 3*-anonymity suppressing only one tuple. It is easy to see that for each row but the last one of the* VT *matrix in Figure 9, there are* 3 *tuples that are generalized to the same tuple. The algorithm then evaluates tables at height* $\lfloor 5/4 \rfloor = 1$*, that is, the tables corresponding to distance vectors* $[1, 0, 0]$*,* $[0, 1, 0]$*, and* $[0, 0, 1]$*. Since none of these tables is* 3*-anonymous and suppresses a number of tuples lower than* MaxSup*, the solution returned by the algorithm is* $[1, 1, 0]$*, which corresponds to* $\langle Z_1, M_1, S_0 \rangle$*.*

The time complexity of the algorithm in Figure 8 is exponential in the number (n) of attributes in QI.

First, the algorithm builds the domain generalization hierarchy for QI (VL_{DT}) from the given domain generalization hierarchies of the individual attributes in QI. This step requires a complexity of $O(|\mathsf{DGH}|)$ in time. The algorithm then sorts the tuples in the table with a standard sorting algorithm in polynomial time. The third step initializes the matrix VT, and it requires a time proportional to its dimension ($O(m^2 \cdot n)$). The last step computes a minimal generalization with a binary search in the lattice that verifies the anonymity of the generalized tables corresponding to each node of the lattice considered. Note that, even if this is a binary search, the lattice contains, in the middle level, a huge number of possible generalizations. For instance, even if we assume that QI contains only attributes with domain generalization hierarchies of height 2, the number of possible generalizations in the middle level of the corresponding DGH is $\binom{n}{n/2} \in O(2^{n/2})$. Therefore, in the worst case, the time complexity of the binary search phase is upperbounded by $|\mathsf{DGH}| \cdot (n \cdot m)$, where $|\mathsf{DGH}| = n \cdot \prod_{i=1}^{n} h_i$ with h_i the height of the domain generalization hierarchy of the single attribute A_i in QI. The overall time complexity of the algorithm is then $O((n \cdot m) \cdot (m + |\mathsf{DGH}|))$, where $|\mathsf{DGH}| \in O(n \cdot (h_{max})^n)$ and $h_{max} = max\{h_1, \ldots, h_n\}$.

3.2.2 Incognito

The *Incognito* algorithm [29] has been proposed for computing a k-minimal generalization by using a domain generalization hierarchy DGH_{DT} on the domain tuple of the quasi-identifier, where the vertices in DGH_{DT} are only the vertices that correspond to k-anonymous generalized tables.

Since the computational cost of the k-anonymization algorithms is mainly due to the verification of the k-anonymity condition on a great number of generalizations of the private table PT, Incognito reduces the number of tables for which this verification is necessary by exploiting the observation that if a table T with quasi-identifier QI is k-anonymous, T is k-anonymous with respect to any quasi-identifiers $Q \subset \mathsf{QI}$.

Definition 3.6 (Subset property) *Let $T(A_1,\ldots,A_n)$ be a table and $\mathsf{QI}=\{A_1,\ldots,A_m\}$ be its quasi-identifier. If T is k-anonymous with respect to QI, T is k-anonymous with respect to any $Q \subseteq \mathsf{QI}$.*

This subset property represents a necessary (not sufficient) condition for k-anonymity. As a matter of fact, a table T can be k-anonymous with respect to QI only if T is k-anonymous with respect to any subset of QI. Incognito then excludes a priori the generalizations in DGH_{DT} that cannot be k-anonymous with respect to QI. To this purpose, Incognito iteratively builds all the domain generalization hierarchies on subsets of QI, excluding at each step the vertices representing generalizations that do not satisfy k-anonymity (see Figure 10).

At the first iteration, for each $A_i \in \mathsf{QI}$, Incognito builds DGH_{D_i} with $D_i = dom(A_i, \mathsf{PT})$. For each DGH_{D_i}, Incognito then follows a bottom-up breadth first search on the domain generalization hierarchy. If a table T_v corresponding to vertex v in DGH_{D_i} satisfies k-anonymity, Incognito marks true v and all vertices v', such that $v \leq v'$, without explicitly computing the generalizations corresponding to v' to check if they are k-anonymous.[2]

At the second iteration, for each subset $\{A_i,A_j\} \subseteq \mathsf{QI}$, Incognito builds $\mathsf{DGH}_{\langle D_i,D_j \rangle}$ with $D_i = dom(A_i, \mathsf{PT})$ and $D_j = dom(A_j, \mathsf{PT})$. The domain generalization hierarchy $\mathsf{DGH}_{\langle D_i,D_j \rangle}$ is built by combining (through function **Compose**) DGH_{D_i} and DGH_{D_j}. For the subset property the **Compose** function removes from $\mathsf{DGH}_{\langle D_i,D_j \rangle}$ all those vertices that represent domain tuples containing generalized domains for D_i (or D_j) marked false in DGH_{D_i} (or DGH_{D_j}).

At iteration i, the algorithm builds the domain generalization hierarchies on subsets of QI composed of i attributes, using the vertices marked true in the domain generalization hierarchies computed at iteration $i-1$. It terminates at iteration $i = |\mathsf{QI}|$, when it evaluates the domain generalization hierarchy for the whole quasi-identifier.

[2]Incognito applies the monotonicity property of k-anonymity stating that if a table T is k-anonymous all its generalized tables are k-anonymous.

Algorithm 3.2 (Incognito Algorithm)

```
INPUT
$T$ = PT[QI]: private table where QI = $\{A_1, \ldots, A_n\}$ and $DT = \langle dom(A_1,\mathsf{PT}),\ldots,dom(A_n,\mathsf{PT})\rangle$
$k$: anonymity requirement
$\forall D \in DT$, DGH$_D$: domain generalization hierarchies where for all $v \in$ DGH$_D$, mark($v$) is set false

OUTPUT
DGH$_D$: restricted version of DGH$_{DT}$

MAIN
for $i$=1...$n$ do
    $\mathcal{D}^i := \{D|D\subseteq DT \wedge |D| = i\}$ /* all subsets of $i$ elements in QI */
    for each $D\in \mathcal{D}^i$ do
        if $i \neq 1$ then DGH$_D$ := Compose($D$)
        for $h$=0...height($\top$) do /* $\top$ represents the root node in DGH$_D$ */
            /* height($v$) denotes the length of the path from the bottom element in DGH$_D$ to $v$ */
            $\mathcal{V}^h := \{v|v \in$ DGH$_D \wedge$ height$(v)=h \wedge$ mark$(v)=false\}$
            for each $v \in \mathcal{V}^h$ do
                let $T_v$ be the generalized table corresponding to vertex $v$
                if Satisfy($T_v$,$k$) then
                  for each $v'|v' \in$ DGH$_D \wedge v \leq v'$ do mark($v'$):=$true$
return(DGH$_D$)

COMPOSE($D$)
DGH$_D$ := ($V$,$E$)
$i$ := 1
for each $D_i \in \{D' \subset D$:$|D'| = |D| - 1\}$ do
    $V_i := \{v|v \in$ DGH$_{D_i} \wedge$ mark$(v)=true\}$
    $i := i + 1$
$V := V_1 \times \ldots \times V_i$
for each $v \in V$ do mark($v$) := $false$
$E := \{(v_i,v_j)|v_i,v_j\in V,v_i \leq v_j,\ \nexists v_z \in V,v_i \leq v_z \wedge v_z \leq v_j\}$
return(DGH$_D$)
```

Figure 10: Algorithm that computes reduced generalization hierarchies [29]

Example 3.3 *Consider the private table in Figure 1 with* QI = {`ZIP`, `MaritalStatus`, `Sex`} *and assume that* $k = 3$ *and* MaxSup=*2. Figure 11 illustrates, on the left-hand side, the complete domain generalization hierarchies for all the subsets of* QI*, and on the right-hand side, the sub-hierarchies computed by Incognito at each iteration.*

Iteration 1.

- DGH$_{\langle Z_0\rangle}$. *Vertices* $\langle Z_0\rangle$, $\langle Z_1\rangle$, *and* $\langle Z_2\rangle$ *are marked true, since table* $T_{\langle Z_0\rangle}$ *satisfies* 3*-anonymity by suppressing a number of tuples lower than* MaxSup.
- DGH$_{\langle M_0\rangle}$. *Vertex* $\langle M_0\rangle$ *is marked false, since to satisfy* 3*-anonymity, in table* $T_{\langle M_0\rangle}$ *we need to suppress more than* 3 *tuples. Vertex* $\langle M_1\rangle$ *and vertex* $\langle M_2\rangle$ *are marked true since table* $T_{\langle M_1\rangle}$ *satisfies* 3*-anonymity by suppressing number of tuples lower than* MaxSup.
- DGH$_{\langle S_0\rangle}$. *Vertices* $\langle S_0\rangle$ *and* $\langle S_1\rangle$ *are marked true, since table* $T_{\langle S_0\rangle}$ *satisfies 3-anonymity by suppressing a number of tuples lower than* MaxSup.

Iteration 2.

- DGH$_{\langle Z_0,M_0\rangle}$. *Since* $\langle M_0\rangle$ *has been marked false in the previous iteration, this hierarchy does not include vertices* $\langle Z_0, M_0\rangle$, $\langle Z_1, M_0\rangle$, *and* $\langle Z_2, M_0\rangle$. *Vertex* $\langle Z_0, M_1\rangle$ *is marked false, since* $T_{\langle Z_0,M_1\rangle}$ *satisfies*

3-anonymity only if more than 3 tuples are suppressed. Vertices $\langle Z_0, M_2\rangle$, $\langle Z_1, M_2\rangle$, $\langle Z_2, M_2\rangle$, $\langle Z_1, M_1\rangle$, and $\langle Z_2, M_1\rangle$ are instead marked true, since tables $T_{\langle Z_0,M_2\rangle}$ and $T_{\langle Z_1,M_1\rangle}$ satisfy 3-anonymity by suppressing a number of tuples lower than MaxSup.

- $\mathsf{DGH}_{\langle Z_0,S_0\rangle}$. *Vertex $\langle Z_0, S_0\rangle$ is marked false since $T_{\langle Z_0,S_0\rangle}$ satisfies 3-anonymity only if more than* MaxSup *tuples re suppressed. Vertices $\langle Z_0, S_1\rangle$, $\langle Z_1, S_1\rangle$, $\langle Z_2, S_1\rangle$, $\langle Z_1, S_0\rangle$, and $\langle Z_2, S_0\rangle$ are marked true, since tables $T_{\langle Z_0,S_1\rangle}$ and $T_{\langle Z_1,S_0\rangle}$ satisfy 3-anonymity by suppressing a number of tuples lower than* MaxSup.

- $\mathsf{DGH}_{\langle M_0,S_0\rangle}$. *Since $\langle M_0\rangle$ has been marked false in the previous iteration, this hierarchy does not include vertices $\langle M_0, S_0\rangle$ and $\langle M_0, S_1\rangle$. All the other vertices in the hierarchy are marked true, since table $T_{\langle M_1,S_0\rangle}$ satisfies 3-anonymity by suppressing a number of tuples lower than* MaxSup.

Iteration 3.

- $\mathsf{DGH}_{\langle Z_0,M_0,S_0\rangle}$. *Since $\mathsf{DGH}_{\langle Z_0,M_0\rangle}$ does not contain vertices $\langle Z_0, M_0\rangle$, $\langle Z_1, M_0\rangle$, and $\langle Z_2, M_0\rangle$ and vertex $\langle Z_0, M_1\rangle$ has been marked false, this hierarchy does not contain vertices $\langle Z_0, M_0, S_0\rangle$, $\langle Z_1, M_0, S_0\rangle$, $\langle Z_2, M_0, S_0\rangle$, $\langle Z_0, M_0, S_1\rangle$, $\langle Z_1, M_0, S_1\rangle$, $\langle Z_2, M_0, S_1\rangle$, $\langle Z_0, M_1, S_0\rangle$, and $\langle Z_0, M_1, S_1\rangle$. Analogously, since vertex $\langle Z_0, S_0\rangle$ has been marked false in $\mathsf{DGH}_{\langle Z_0,S_0\rangle}$, this hierarchy does not contain vertex $\langle Z_0, M_2, S_0\rangle$. Vertices $\langle Z_1, M_1, S_0\rangle$, $\langle Z_1, M_1, S_1\rangle$, $\langle Z_1, M_2, S_0\rangle$, $\langle Z_1, M_2, S_1\rangle$, $\langle Z_2, M_1, S_0\rangle$, $\langle Z_2, M_1, S_1\rangle$, $\langle Z_2, M_2, S_0\rangle$, and $\langle Z_2, M_2, S_1\rangle$ are marked true, since table $T_{\langle Z_1,M_1,S_0\rangle}$ satisfies 3-anonymity by suppressing a number of tuples lower than* MaxSup. *Analogously, vertex $\langle Z_0, M_2, S_1\rangle$ is marked true, since table $T_{\langle Z_0,M_2,S_1\rangle}$ satisfies 3-anonymity by suppressing a number of tuples lower than* MaxSup.

Incognito iteratively builds the domain hierarchies of increasing dimension until the hierarchy DGH_{DT} containing the entire QI is built. While in most cases some hierarchies are not built entirely, since some vertices are removed at iteration i due to the results obtained at iteration $i-1$, in the worst case all the hierarchies are completely built. Let Dom be the set of all ground and generalized domains for the quasi-identifier QI, n be the number of attributes composing QI, and m be the number of tuples in PT. We can note that each subset of Dom appears *exactly* in one of the domain generalization hierarchies on subsets of QI, therefore the cost of all the hierarchies computed in the worst case by Incognito is upperbounded by $(n \cdot 2^{\mathsf{Dom}})$, where n is the upperbound of the cost of each single vertex (since each vertex has at most one component for each attribute in QI) and 2^{Dom} is the number of possible vertices. Since Incognito, in the worst case, tests if table PT, generalized according to each subset of Dom, is k-anonymous and suppresses a number of tuples lower than MaxSup, its complexity is in $O((n \cdot m) \cdot n \cdot 2^{\mathsf{Dom}})$.

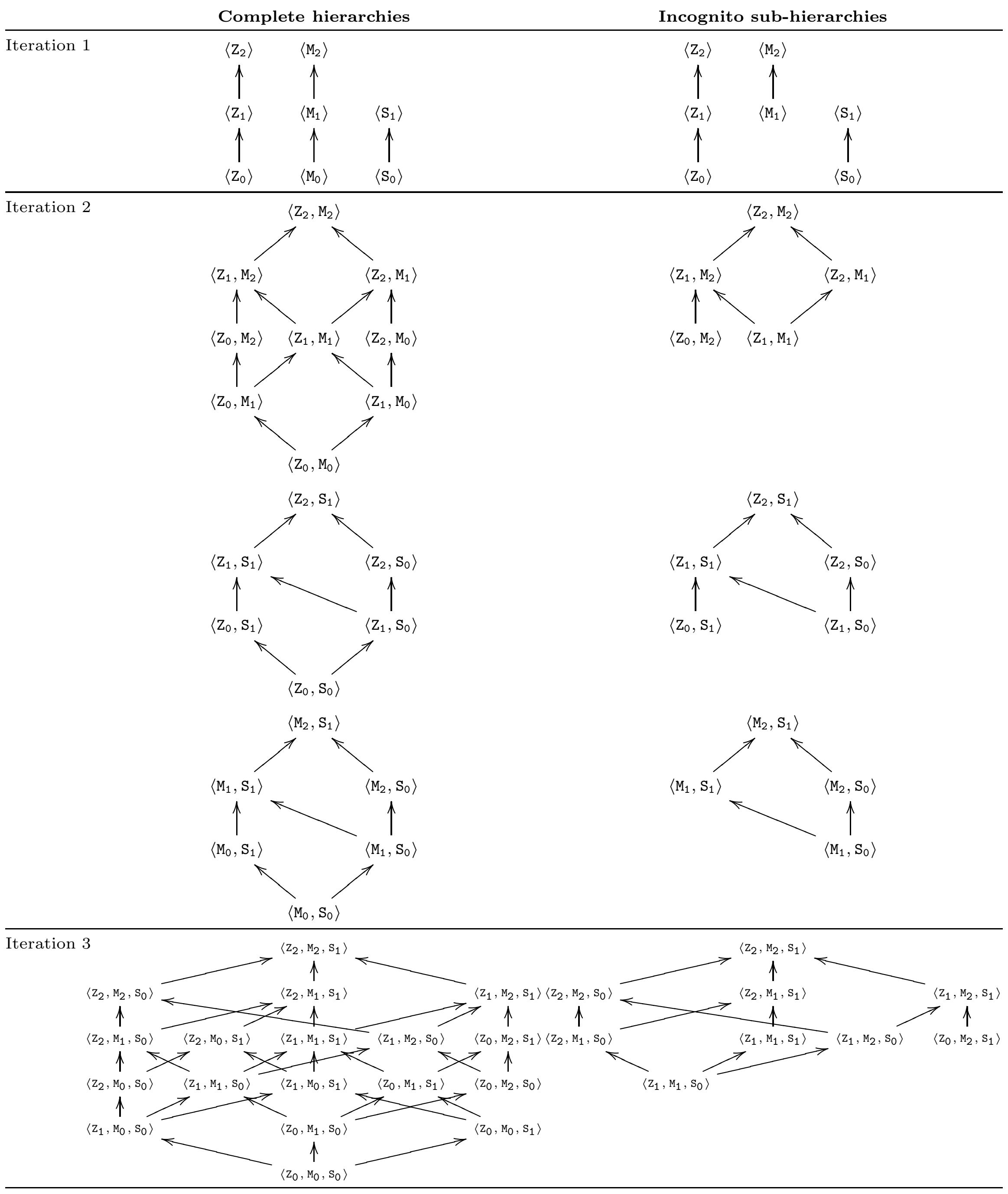


Figure 11: Sub-hierarchies computed by Incognito for the table in Figure 1

3.2.3 Approximation algorithms

Exact algorithms 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 can be exponential in the number of cells in the table.

The first approximation algorithm for **_CS** was proposed by Meyerson and Williams [33] and guarantees a $O(k \log(k))$-approximation. The best-known polynomial approximation algorithm for **_CS** guarantees a $O(k)$-approximate solution [2]. Such an $O(k)$-approximate 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 each, which represent the groups of tuples of at least k elements that are generalized to the same value for k-anonymization. Some cells in the vertices are suppressed to guarantee that all the tuples in the same tree have the same quasi-identifier value (i.e., to achieve k-anonymity). The cost of a vertex is evaluated as the number of cells suppressed, and the cost of a tree is the sum of the costs 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. With the construction of trees with no more than $3k - 3$ vertices, the authors prove that their solution is a $O(k)$-approximation.

An approximation algorithm for **CG_** is described in [3] as a direct extension of the approximation algorithm for **_CS** [2]. For taking into account the generalization hierarchies, each edge has a weight that is computed as follows. Given two tuples t_i and t_j and an attribute A, the generalization cost $h_{t_i,t_j}(A)$ associated with A is the lowest level of the value generalization hierarchy $\mathsf{VGH}_{dom(A,\mathsf{PT})}$ such that tuples t_i and t_j have the same generalized value for A. The weight $w(e)$ of the edge $e = (t_i, t_j)$ is therefore $w(e) = \Sigma_A h_{t_i,t_j}(A)/l_A$, where l_A is the number of levels in $\mathsf{VGH}_{dom(A,\mathsf{PT})}$. The solution of this algorithm is guaranteed to be a $O(k)$-approximation.

Recently, Park and Shim [34] described an algorithm for **_CS** with $O(\log k)$-approximation ratio, but the overall time complexity is $O((nm)^{2k})$.

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 cases where $k = 2$ or $k = 3$, an ad-hoc approach has been provided [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 a 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 with recoding-based generalization

The algorithms described in the previous section assume that the generalizations follow a predefined hierarchy. However, also a recoding-based generalization can be adopted for k-anonymity [6, 25, 28] that exploits an order relationship among values in the attribute domains. The algorithms adopting a recoding-based generalization are therefore not forced to follow a pre-defined strategy for generalizing values.

Definition 4.1 (Recoding function) *Let $D=\{v_1, \ldots, v_x\}$ be a domain on which a total order relationship is defined. A* recoding *function $\rho : D \rightarrow 2^{|D|}$ for D partitions the domain in a set of (possibly disjoint) intervals $I_1, \ldots, I_y$ $(y \leq x)$, such that $\bigcup_{i=1}^{y} I_i = D$ and $\forall v_i \in I_a, \forall v_j \in I_b$, with $a \neq b$, $v_i \leq v_j$ iff $a < b$.*

As an example, consider attribute `MaritalStatus` and the $\mathsf{VGH}_{\mathtt{M_0}}$ in Figure 3. A possible order among the values in `M0` is {married, widow, divorced, single}. A recoding function can partition `M0` into two intervals I_1=[married, widow] and I_2 = [divorced, single].

Recoding-based generalization associates with each interval I_j a value v_j representing the values of the ground domain belonging to I_j. Any value in PT that belongs to I_j is then generalized to v_j.

The main advantage of a recoding-based generalization strategy is that it is possible to analyze solutions that would not be investigated adopting a hierarchy-based generalization, and this might improve the quality of the final solution. However, since recoding-based generalization strategies are based on the total order among attribute values, they are influenced by such an ordering, which may be difficult to define when the attribute domain does not have a characterizing ordering relationship. Furthermore, when the domain is categorical,[3] it might be difficult to represent the values obtained by the generalization process, since the set of values that need to be indistinguishable in the released table might not be similar. In some cases, the only way to describe an interval of values is the direct enumeration of its elements. For instance, the two subsets [married, widow] and [divorced, single] are described by enumerating their elements, while the set [married, widow, divorced] can be described as been_married.

4.1 Algorithms for k-anonymity

We now describe two k-anonymization algorithms adopting recoding-based generalization: k-Optimize and Mondrian multidimensional k-anonymity.

[3]An attribute is categorical if it can assume a limited and specified set of values on which arithmetic operations cannot be defined. For instance, attribute `MaritalStatus` is categorical.

ZIP				MaritalStatus				Sex	
⟨[22030]	[22032]	[22045]	[22047] ⟩	⟨[married]	[widow]	[divorced]	[single]⟩	⟨[M]	[F]⟩
1	2	3	4	5	6	7	8	9	10

Figure 12: Index assignment to attributes `ZIP`, `MaritalStatus`, and `Sex`

4.1.1 k-Optimize

Bayardo and Agrawal [6] propose an **AG_TS** algorithm called *k-Optimize*, based on a recoding-based generalization defined on the quasi-identifying attributes. k-Optimize determines a solution minimizing a predefined cost function that measures the loss of information due to generalization and suppression.

k-Optimize assumes the existence of a total order relationship among the attributes composing QI, and a total order relationship on each of the domains of the quasi-identifying attributes. It uses these total order relationships to associate an integer value, called *index*, with each interval in any domain of the quasi-identifier attributes. Note that, at initialization time, any value in any domain of QI represents an interval itself. The index assignment reflects the total order relationship among quasi-identifying attributes and within their domains.

Example 4.1 *Consider the private table in Figure 1 with* QI $= \{$`ZIP`*,* `MaritalStatus`*,* `Sex`$\}$ *and suppose that* `ZIP` *precedes* `MaritalStatus` *that, in turn, precedes* `Sex`*. Suppose also that the values within the domain of each of these attributes follow the same order as the leaves in the hierarchies in Figure 3. Figure 12 illustrates the index values assigned assuming that each value represents an interval. As it is visible from the figure, due to the order among attributes, the index values associated with the intervals of the* `Zip` *domain are lower than the index values associated with the intervals of the domains of* `MaritalStatus` *and* `Sex`*. Furthermore, within each domain the assignment of the index values follow the total order among intervals.*

Given a set of indexes $\mathcal{I}$, a generalization is represented by the union of individual index values. As an example, with reference to Figure 12, the union of index values 1 and 2 means that the `ZIP` values 22030 and 22032 have been generalized to the same indistinguishable value. Since only contiguous index values can be unioned for generalization purposes, their union is represented by the least index value. This implies that given a set $\mathcal{I}$ of index values, if an index value I does not appear in $\mathcal{I}$, the index I has been generalized to the nearest value appearing in $\mathcal{I}$ and lower than I. For instance, with respect to the set of indexes in Figure 12, the set $\mathcal{I} = \{1, 3, 5, 8, 9, 10\}$ implicitly indicates that index 2 has been generalized to the same value as 1, 4 to the same value as 3, and 6 and 7 to the same value as 5. Since the least value in any attribute domain will certainly appear in any generalization, it can be omitted in the representation of $\mathcal{I}$, assuming that it always implicitly belongs to the generalized domain. Consequently, the set $\mathcal{I} = \{1, 3, 5, 8, 9, 10\}$ can be represented as $\mathcal{I} = \{3, 8, 10\}$. This set of index values represents the following

Algorithm 4.1 (k-Optimize Algorithm)

```
INPUT
$\mathcal{I}$: set of index values corresponding to the original domains of PT[QI]
$k$: anonymity requirement

OUTPUT
$\mathcal{I}'$: set of index values representing the $k$-anonymous generalized table

MAIN
$root$ := { }
$best.cost := \infty$
$best.sol := null$
Optimize($root$,$best$)

OPTIMIZE($node$,$best$)
if Satisfy($node$) then
  $cost$ := Cost($node$)
  if $cost \leq best.cost$ then
    $best.cost := cost$
    $best.sol := node$
  for each $i \in \{idx | idx \in \mathcal{I} \wedge (\forall j \in node, idx \geq j \vee node = \emptyset)\}$ do
     $child := node \cup \{i\}$
     $lb$ := LowerBound($child$)
     if $lb \leq best.cost$ then
       $best$ := Optimize($child$,$best$)
     else
       Prune($root$,$node$) /* prune nodes having $node$ as a subset */
return($best$)
```

Figure 13: k-Optimize algorithm adopting a pre-order traversal strategy [6]

interval values: {[1,2],[3,4]}, corresponding to {[22030,22032],[22045,22047]}; {[5,6,7],[8]}, corresponding to {[married,widow,divorced],[single]}; and {[9],[10]}, corresponding to {[M],[F]}. Note that the empty set { } represents the most general anonymization. For instance, with reference to Example 4.1, { } corresponds to the generalizations {1} for attribute ZIP, {5} for attribute MaritalStatus, and {9} for attribute Sex, which in turn correspond to the generalized values ZIP: {[22030, 22032, 22045, 22047]}; MaritalStatus: {[married,widow,divorced,single]}; and Sex: {[M,F]}.

Each generalized table that can be obtained from PT adopting a recoding-based generalization can be represented as a subset of the indexes $\mathcal{I}$ associated with the original domains of quasi-identifying attributes. Each of these solutions is associated with a cost, reflecting the information loss caused by the chosen generalization and by the tuples suppressed to achieve k-anonymity.

To compute the optimal generalized table, k-Optimize builds a *set enumeration tree* over the set $\mathcal{I}$ of index values, where each node corresponds to a generalized table of the original private table PT. The root node of the tree is the empty set. The children of a node *node* enumerate the sets that can be formed by appending a single element of $\mathcal{I}$ to *node*, with the restriction that this single element must follow every element already in *node*, according to the given total order. Figure 14 illustrates a set enumeration tree over $\mathcal{I} = \{2, 3, 4\}$ (e.g., this tree could represent the set enumeration tree for attribute ZIP). 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

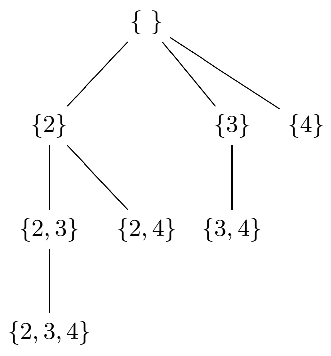
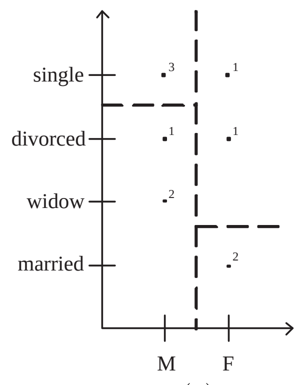


Figure 14: An example of set enumeration tree over set $\mathcal{I} = \{2, 3, 4\}$ of indexes

to the k-anonymity problem.

Algorithm k-Optimize (see Figure 13) visits the set enumeration tree built on the basis of the given private table PT, keeping track of the best generalization (variable *best*) found at each step. For each visited node *node* in the tree that satisfies the k-anonymity constraint (function **Satisfy**), it computes (function **Cost**) the cost *cost* of the generalization strategy represented by *node*. If the cost of the solution associated with *node* is *lower* than *best.cost*, that is, the lowest cost found during the visit at that point, *node* becomes the new locally best solution and its cost becomes the comparison term for the following visited nodes. Since the number of nodes in the tree is $2^{|\mathcal{I}|}$, this approach is not practical and the authors have therefore proposed heuristics and pruning strategies to reduce the computational costs of the k-Optimize algorithm [6]. The idea is that if none of the nodes in the subtree rooted at a child *child* of *node* can have a cost lower than the locally optimal solution computed before visiting *child*, k-Optimize prunes the subtree rooted at *child* without visiting the solutions it contains. To this purpose, for each visited node *node*, k-Optimize computes the lower bound of the cost function for the subtree rooted at each of its children *child* (function **LowerBound**), that is, the lowest possible cost that a solution in the subtree can have. The lower bound for a subtree *child* is computed by noting that the solutions in the subtree rooted at *child* need to suppress a greater number of tuples than *node* to guarantee k-anonymity, since they represent more specific solutions. If this lower bound is higher than the locally optimal solution, the subtree rooted at *child* is pruned. Note that when a subtree is pruned also additional nodes can be removed from the tree (procedure **Prune**). For instance, consider the set enumeration tree in Figure 14 and suppose that node {2, 4} is pruned. This means that all the solutions that contain index values 2 and 4 are not optimal, therefore also node {2, 3, 4} 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 set a maximum computational time, and obtain a good, but non optimal, solution.



(a)

MaritalStatus	**Sex**
married	F
married	F
single	M
single	M
single	M
[widow,divorced,single]	F
[married,widow,divorced]	M
[married,widow,divorced]	M
[married,widow,divorced]	M
[widow,divorced,single]	F

(b)

Figure 15: An example of strict multidimensional partitioning (a) and corresponding generalized table (b)

4.1.2 Mondrian multidimensional algorithm

LeFevre et al. [28] propose a $\mathbf{CG}_{-}$ algorithm for anonymizing a private table PT based on *multidimensional global recoding*, by extending the single-dimension recoding-based generalization to the multi-dimensional case.

On the basis of the order of values defined for all the domains of attributes composing QI, a multidimensional generalization function defines a set of *multidimensional regions*. These regions correspond to the intervals defined in the single-dimensional scenario. Let QI=$\{A_1, \ldots, A_n\}$ be a quasi-identifier and $\mathcal{D}$=$\{D_1, \ldots, D_n\}$ be the set of ground domains of the attributes $A_1, \ldots, A_n$. Each domain in $\mathcal{D}$ is a dimension for the multidimensional space, where PT[QI] can be represented as a set of points: each tuple $t \in$ PT[QI] represents a point in the multidimensional space defined by $\mathcal{D}$ and its coordinates are the values assumed by the quasi-identifying attributes in t.

A multidimensional region within such a space is represented by two tuples p=(p_1, ..., p_n) and v=(v_1, ..., v_n) such that $p_i \leq v_i$, $i = 1, \ldots, n$. A tuple t=(t_1, ..., t_n) belongs to the region represented by $\langle p, v \rangle$ if $p_i \leq t_i \leq v_i$, $i = 1, \ldots, n$. A multidimensional region can therefore be seen as a n-dimensional rectangular, whose edges are parallel to the axis. As an example, consider the private table in Figure 1 and suppose that QI=\{`MaritalStatus`, `Sex`\}. Figure 15(a) provides a graphical representation of the multidimensional space determined by $\mathcal{D}$=\{$\texttt{M}_0$, $\texttt{S}_0$\}, where the domain of attribute `Sex` is represented on the x-axis and the domain of attribute `MaritalStatus` is represented on the y-axis. Regions are delimited by lines parallel to the axis and points are associated with the number of occurrences of the corresponding quasi-identifier values in PT[QI]. A *strict multidimensional partitioning* is a set of multidimensional regions that cover the whole space defined on $\mathcal{D}$.

Definition 4.2 (Strict multidimensional partitioning) *Let* QI *be a quasi-identifier and* $\mathcal{D}$ =

$\{D_1, \ldots, D_n\}$ *be the set of domains for the attributes composing it. A* strict multidimensional partitioning *is a set of* non-overlapping *multidimensional regions covering the space defined by* $\mathcal{D}$*.*

A strict multidimensional partitioning of the space defined by $\mathcal{D}$ represents the generalized table T obtained by making all the tuples belonging to the same multidimensional region indistinguishable, that is, by generalizing the tuples in the same region to the same generalized tuple. For instance, Figure 15(b) represents a possible generalized table corresponding to the regions in Figure 15(a), where the generalized values of attributes `MaritalStatus` and `Sex` are obtained by listing all values that belong to a given region.

A strict multidimensional partitioning for a private table PT is k-anonymous if any multidimensional region in the space defined by $\mathcal{D}$ contains either zero or at least k tuples of PT. The problem of computing the *optimal* (i.e., the one that minimizes information loss) strict k-anonymous multidimensional partitioning is NP-hard, since its decisional version can be reduced from the well known partition problem [28]. It is important to note here that this result is not implied by the NP-hardness of the hierarchy-based k-anonymity problem demonstrated in [33], since the two problems are different and cannot be reduced one to the other.

Given the set of points along with the number of their occurrences induced by private table PT on the multidimensional space defined by $\mathcal{D}$, a *multidimensional cut* for such a set of points is an axis-parallel cut that produces two disjoint sets of points. A multidimensional cut perpendicular to dimension D_i, corresponding to attribute A_i in QI, is performed at a given value v of D_i. Any tuple t such that $t[A_i] \leq v$ will belong to one partition, while any tuple t such that $t[A_i] > v$ will belong to the other one. A multidimensional cut is *allowable* with respect to a given k-anonymity constraint if and only if the resulting regions contain a set of points representing at least k tuples of the original table (i.e., the sum of the occurrences of the points in the region obtained is at least k). The recursive application of allowable multidimensional cuts on the original set of points representing PT generates a multidimensional strict partitioning.

Definition 4.3 (Minimal strict multidimensional partitioning) *A strict multidimensional partitioning, composed of regions* $R_1, \ldots, R_r$*, is* minimal *if there does not exist an allowable multidimensional cut for any of the sets of points in its regions.*

For instance, for $k = 2$, the strict multidimensional partitioning in Figure 15(a) is minimal, since none of its regions can be further split obtaining two regions containing at least 2 tuples each.

The maximum number of points contained in any region R_i in a minimal strict multidimensional partitioning is $2n(k-1)+o$, where $n = |\mathsf{QI}|$ is the number of dimensions (i.e., attributes composing the quasi-identifier), k is the k-anonymity constraint, and o is the maximum number of occurrences of a quasi-identifier value in PT (i.e., the maximum number of copies of the same point in the multidimensional space), as formally proved in [28].

Algorithm 4.2 (Mondrian Algorithm)

```
INPUT
$partition$=$\{p|p = \langle p_1, \dots, p_n\rangle \wedge p_i \in D_i, i = 1, \dots, n\rangle\}$: set of points that represent private table PT[QI]
$k$: anonymity requirement

OUTPUT
$T$: $k$-anonymized table for PT[QI]

MAIN
Anonymize($partition$)

ANONYMIZE($partition$)
if(no allowable multidimensional cut for $partition$) then
  return Generalize($partition$) /* generalize all the tuples in partition to the same value */
else
  $dim$ := ChooseDimension($partition$) /* choose the dimension for the cut */
  $fs$ := frequency_set($partition$, $dim$) /* frequency of the values in the domain of $dim$ */
  $split\_val$ := find_median($fs$) /* find the split point as the median for $dim$ wrt $fs$ */
  $lhs$ := $\{p \in partition | p_{dim} \leq split\_val\}$ /* first partition created */
  $rhs$ := $\{p \in partition | p_{dim} > split\_val\}$ /* second partition created */
  return Anonymize($rhs$)$\cup$Anonymize($lhs$) /* recursive call */
```

Figure 16: Mondrian multidimensional k-anonymity algorithm [28]

The algorithm proposed in [28], and illustrated in Figure 16, is a greedy solution to the minimal multidimensional k-anonymity problem. The algorithm receives in input the set *partition* of points together with their occurrences that represent a private table PT, and the k value. The algorithm cuts the given n-dimensional space in two non overlapping spaces and is recursively applied to each resulting space, until the minimal strict multidimensional partitioning is reached. At each recursive call, the algorithm chooses the dimension with the widest (normalized) range of values. It then splits the region by performing a strict multidimensional cut perpendicular to the chosen dimension, adopting as a split value the median of *partition* projected on the chosen dimension. If the cut is allowable, the algorithm is recursively called on each of the two regions obtained; otherwise the cut is not performed. As an example, consider the private table in Figure 1 and suppose that QI=$\{$`MaritalStatus`, `Sex`$\}$. Figure 17 represents an example of step by step execution of the algorithm in Figure 16, where $k = 2$. The algorithm first chooses the `Sex` dimension and performs a cut at value 'M', obtaining the regions in Figure 17(b). The algorithm is then recursively called on the region characterized by `Sex`=M. This call causes a cut on attribute `MaritalStatus` at value 'single'. The two resulting regions (Figure 17(c)) cannot be further partitioned and, therefore the algorithm executes the recursive call on the region characterized by `Sex`=F. This call causes a cut on attribute `MaritalStatus` at value 'widow'. The two resulting regions cannot be further partitioned. The result obtained by the algorithm is then represented in Figure 17(d).

This algorithm is greedy, since it chooses the dimension to exploit for the next cut on the basis of the local properties of the multidimensional region. It represents a $O(n)$-approximation algorithm for the k-anonymity problem (with recoding-based generalization and adopting strict multidimensional cuts), supposing o/k to be a constant ratio. The time complexity for the given algorithm is $O((n \cdot m) \cdot log(n \cdot m))$, when $(n \cdot m)$ is

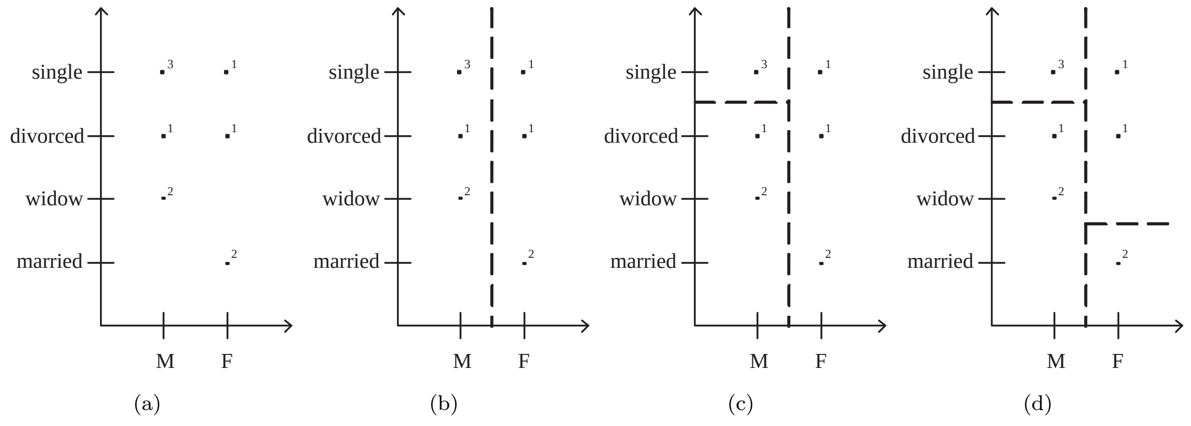


Figure 17: Spatial representation (a) and possible partitioning (b)-(d) of the table in Figure 1

the number of cells composing the original private table.

Besides strict multidimensional partitioning, also *relaxed* multidimensional partitioning can be adopted for the k-anonymity problem. The main difference of relaxed multidimensional partitioning with respect to strict multidimensional partitioning is that the regions covering the multidimensional space defined by $\mathcal{D}$ can be potentially overlapping. When adopting relaxed multidimensional partitioning, each tuple in PT is generalized following the values of one of the regions to which the point corresponding to the tuple belongs. Therefore, duplicate values in PT can be generalized to different values in the released table. The Mondrian algorithm proposed can be adapted to operate also with relaxed multidimensional partitioning. In this case, it computes a 2-approximate solution of the problem.

5 Attribute disclosure protection

Even if k-anonymity represents a solution to the problem of identity disclosure, since it protects respondents from attacks aimed at reducing the uncertainty about their identities, it does not protect from attribute disclosure. As a matter of fact, attribute disclosure is possible even on tables protected against identity disclosure.

Machanavajjhala, Gehrke, and Kifer define two possible attacks to k-anonymous tables: *homogeneity attack* (already noted in [36]) and *background knowledge attack* [32]. 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 homogeneity assumptions, if an attacker knows the quasi-identifier value of a respondent and knows that this respondent belongs to the population represented in the table, the attacker can infer which is the value for the sensitive attribute for the known respondent,

ZIP	**MaritalStatus**	**Sex**	**Disease**
2203∗	been_married	F	hypertension
2203∗	been_married	F	hypertension
2203∗	never_married	M	obesity
2203∗	never_married	M	HIV
2203∗	never_married	M	obesity
2203∗	been_married	F	hypertension
2204∗	been_married	M	obesity
2204∗	been_married	M	HIV
2204∗	been_married	M	HIV

Figure 18: An example of 3-anonymous table

since all the tuples with the given quasi-identifier value have the same value for the sensitive attribute. For instance, consider the 3-anonymous table in Figure 18, where `Disease` is a sensitive attribute. If `Alice` knows that her friend `Hellen` is a married female living in an area with ZIP code 22030, she can infer that `Hellen` suffers from `hypertension`, since all tuples with ⟨2203*,been_married,F⟩ as a quasi-identifier value are characterized by `Disease=‘hypertension’`.

The background knowledge attack is instead based on a-priori knowledge of the attacker of some additional external information. For instance, with reference to the 3-anonymous table in Figure 18, suppose that `Alice` knows that `Bob` is a single man living in 22032 area. `Alice` can then infer that `Bob` suffers of `obesity` or `HIV`. Suppose now that `Alice` knows that `Bob` is thin. `Alice` can infer with probability equal to 1 that `Bob` suffers of `HIV`.

To prevent homogeneity and background knowledge attacks, Machanavajjhala, Gehrke, and Kifer introduce the notion of *ℓ-diversity* [32].

Definition 5.1 (ℓ-diversity principle) *Let $T(A_1,\ldots,A_n,S)$ be a table,* $\mathsf{QI}=\{A_i,\ldots,A_n\}$ *be its quasi-identifier, ℓ be a user-defined threshold, and S be a sensitive attribute. A set of tuples in T having the same value for* QI*, called q-block, is said to be ℓ-*diverse *if it contains at least ℓ different values for S. T is said to be ℓ-*diverse *if all its q-blocks are ℓ-*diverse*.*

If a k-anonymous table is ℓ-diverse, the homogeneity attack is no more applicable, since each q-block 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 with a predefined respondent. For instance, if ℓ=2, only two of the three q-blocks in the table in Figure 18 are ℓ-diverse, while the third is not ℓ-diverse ($\ell = 1$).

Machanavajjhala, Gehrke, and Kifer [32] prove that ℓ-diversity satisfies the monotonicity property with respect to DGH_{DT}. Monotonicity means that if T_i guarantees ℓ-diversity, any T_j, such that $T_i \preceq T_j$ satisfies

ℓ-diversity. This implies that any algorithm originally thought for k-anonymity can also be used to achieve ℓ-diversity, by simply checking the ℓ-diversity property every time a table is tested for k-anonymity.

Note that the original definition of ℓ-diversity considers the presence of one sensitive attribute only and cannot be simply extended to sets of attributes, since ℓ-diversity may be violated even if each sensitive attribute separately satisfies it. If the private table contains more than one sensitive attribute, ℓ-diversity can be achieved by ensuring that the given table is ℓ-diverse with respect to each sensitive attribute S_i separately, running the algorithm considering, as a quasi-identifier, QI unioned with all sensitive attributes but S_i.

After the introduction of ℓ-diversity, the problem of attribute disclosure has received much attention, and different proposals have been studied [40, 41]. p-sensitive k-anonymity [40] is another approach very similar to ℓ-diversity that considers the case of table PT with more than one sensitive attribute.

(α, k)-anonymity [41] takes a different approach to the attribute disclosure problem. It supposes that not all the values in the domain of a sensitive attribute are equally sensitive. For instance, the obesity value for attribute `Disease` is not sensitive, while the HIV value is sensitive. To the aim of protecting the association of quasi-identifying values with sensitive values, (α, k)-anonymity imposes a different constraint to k-anonymous tables. If $dom(S, \mathsf{PT})$ has only one sensitive value s, every q-block must have a relative frequency of s not greater than α, meaning that the number of occurrences of s in the block divided by the cardinality of the block cannot be greater than α.

Definition 5.2 ((α, k)-Anonymity) *Let $T_i(A_1,\ldots,A_n,S)$ and $T_j(A_1,\ldots,A_n,S)$ be two tables with $\mathsf{QI}=\{A_1,\ldots,A_n\}$ such that $T_i[\mathsf{QI}]\preceq T_j[\mathsf{QI}]$, S be a sensitive attribute, s be the only sensitive value in $dom(S, T_i)$, and $0 < \alpha < 1$ be a user-defined threshold. T_j is α-deassociated with respect to QI and s if the relative frequency of s in every q-block is not greater than α. T_j is said to be* (α, k)-anonymous *if it satisfies both the k-anonymity and the α-deassociation constraints.*

For instance, consider the 3-anonymous table in Figure 18, and suppose that the unique sensitive value for `Disease` is HIV. Suppose also that $\alpha = 0.4$ and $k = 3$. The table in Figure 18 is not $(0.4, 3)$-anonymous because the last q-block does not satisfy the 0.4-deassociation constraint, since the sensitive value appears twice in a block of 3 tuples with a relative frequency of 0.67. The other two q-blocks are 0.4-deassociated.

The (α, k)-anonymization problem is NP-hard since, considering a binary alphabet, it can be reduced from the edge partition into the 4-cliques problem [41]. Since also (α, k)-anonymity is a monotonic property with respect to DGH_{DT}, the classical k-anonymity algorithms can be adopted to achieve it. However, a local recoding strategy that adopts a top-down approach (i.e., starting from a completely generalized table, it selectively specializes the table without violating (α, k)-anonymity constraint) can produce better

ZIP	**Marital status**	**Sex**	**Diabetes**	**Cholesterol**
2203∗	been_married	F	N	230
2203∗	been_married	F	N	220
2203∗	never_married	M	Y	250
2203∗	never_married	M	Y	260
2203∗	never_married	M	N	250
2203∗	been_married	F	N	275
2204∗	been_married	M	N	285
2204∗	been_married	M	Y	210
2204∗	been_married	M	N	190

Figure 19: An example of 3-anonymous and 2-diverse table

results [41].

Although ℓ-diversity protects data against attribute disclosure, this protection leaves space to attacks based on the distribution of values inside q-blocks. Li, Li, and Venkatasubramanian [30] show that ℓ-diversity suffers from two attacks: the *skewness attack* and the *similarity attack*. As an example of skewness attack, consider the 2-diverse table in Figure 19 and the binary sensitive attribute `Diabetes`, which is characterized by a skewed distribution. The table has a q-block having 2 out of 3 tuples with a positive value for `Diabetes` and only one tuples with a negative value for it. Even if the table satisfies 2-diversity, it is possible to infer that respondents in the given q-block have 67% probability of contracting diabetes, compared to the 30% of the whole population. Furthermore, if the values for the sensitive attribute S in a q-block are distinct but semantically similar, an external recipient can however infer important information on it by applying a similarity attack. For instance, with reference to the table in Figure 19, it is possible to infer that single males living in the 2203∗ area have the cholesterol value between 250 and 260, since all the tuples in this q-block have these values for the considered sensitive attribute.

To counteract these attacks, Li, Li, and Venkatasubramanian [30] introduce the t-closeness requirement, which is a stronger requirement than ℓ-diversity.

Definition 5.3 (t-Closeness Principle) *Let $T_i(A_1,\ldots,A_n,S)$ and $T_j(A_1,\ldots,A_n,S)$ be two tables such that $T_i[A_1,\ldots,A_n] \preceq T_j[A_1,\ldots,A_n]$, S be a sensitive attribute, and t be a user-defined threshold. A q-block in T_j satisfies* t-closeness *if the distance between the distribution of S in this q-block and the distribution of S in T_i is lower than t. T_j satisfies* t-closeness *if all its q-blocks satisfy* t-closeness.

By imposing that the distribution of sensitive values in the released table must be similar to the distribution in the private table, t-closeness helps in preventing both skewness and similarity attacks. As a matter of fact, all the sets of tuples with the same QI value in the released table have approximately the same sensitive value distribution as the whole original population. t-closeness is a difficult property to achieve since t-closeness requires to measure the distance between two distributions of values, either numerical or categorical. In [30], the authors propose to adopt the Earth Mover's Distance (EMD) measure. The advantage of this measure is that, as demonstrated in the paper, it can be easily integrated with the Incognito algorithm due to its generalization and subset properties, which imply monotonicity with respect to both the number of attributes and the generalization level chosen.

6 Conclusions

The management of privacy in today's global infrastructure is a complex issue, since it requires the combined application of solutions coming from technology (technical measures), legislation (law and public policy), ethics, and organizational/individual policies and practices. We focused on the technological aspect of privacy, which involves three different but related dimensions: privacy of the user, privacy of the communication, and privacy of the information (data protection). The chapter discussed the data protection dimension of privacy, which is more generally related to the collection, management, use, and protection of personal information. The data protection aspect requires the investigation of different problems including the problem of protecting the identities (anonymity) of the users to whom the data refer. This problem is becoming more and more difficult because of the increased information availability and ease of access as well as the increased computational power provided by today's technology. Many techniques have been developed for protecting data released publicly or semi-publicly from improper disclosure.

In this chapter, we presented a specific technique that has been receiving considerable attention recently, and that is captured by the notion of *k-anonymity*. We discussed the concept of k-anonymity and described two classes of algorithms for its enforcement, which differ by the generalization technique (i.e., hierarchy-based or recoding-based) adopted. We also discussed recent proposals for attribute disclosure protection, which are aimed at extending the k-anonymity algorithms with additional properties.

Acknowledgements

This work was supported in part by the EU, within the Seventh Framework Programme (FP7/2007-2013) under grant agreement n^{o} 216483, and by the Italian MIUR, within PRIN 2006, under project 2006099978.

References

[1] N. Adam and J. Wortman. Security-control methods for statistical databases: A comparative study. *ACM Computing Surveys*, 21(4):515–556, 1989.

[2] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. In *Proc. of the 10th International Conference on Database Theory (ICDT'05)*, Edinburgh, Scotland, January 2005.

[3] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Approximation algorithms for k-anonymity. *Journal of Privacy Technology*, November 2005.

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

[5] A. Ambainis. Upper bound on communication complexity of private information retrieval. In *Proc. of the 24th International Colloquium on Automata, Languages and Programming*, Bologna, Italy, July 1997.

[6] R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *Proc. of the 21st International Conference on Data Engineering (ICDE'05)*, Tokyo, Japan, April 2005.

[7] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In *Proc. of EUROCRYPT'99*, Prague, Czech Republic, May 1999.

[8] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.

[9] B. Chor and N. Gilboa. Computationally private information retrieval (extended abstract). In *Proc. of the 29th Annual ACM Symposium on Theory of Computing*, El Paso, Texas, USA, May 1997.

[10] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *Journal of ACM*, 45(6):965–981, April 1998.

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

[12] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, and P. Samarati. k-anonymous data mining: A survey. In C. C. Aggarwal and P. S. Yu, editors, *Privacy-Preserving Data Mining: Models and Algorithms.* Springer-Verlag, 2007.

[13] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, and P. Samarati. Microdata protection. In T. Yu and S. Jajodia, editors, *Security in Decentralized Data Management.* Springer, Berlin Heidelberg, 2007.

[14] L. Cox. A constructive procedure for unbiased controlled rounding. *Journal of the American Statistical Association*, 82(398):520–524, 1987.

[15] S. Dawson, S. De Capitani di Vimercati, P. Lincoln, and P. Samarati. Maximizing sharing of protected information. *Journal of Computer and System Sciences*, 64(3):496–541, May 2002.

[16] D. Denning. Inference controls. In *Cryptography and Data Security.* Addison-Wesley Publishing Company, 1982.

[17] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proc. of the 12th USENIX Security Symposium*, San Diego, California, USA, August 2004.

[18] W. Du and M. Atallah. Privacy-preserving cooperative statistical analysis. In *Proc. of the 17th Annual Computer Security Applications Conference*, New Orleans, Louisiana, USA, December 2001.

[19] W. Du and M. Atallah. Secure multi-party computation problems and their applications: a review and open problems. In *Proc. of the 2001 Workshop on New Security Paradigms*, Cloudcroft, New Mexico, September 2001.

[20] G. Duncan, S. Keller-McNulty, and S. Stokes. Disclosure risk vs. data utility: The R-U confidentiality map. Technical report, Los Alamos National Laboratory, 2001. LA-UR-01-6428.

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

[22] A. Gionis, A. Mazza, and T. Tassa. k-Anonymization revisited. In *Proc. of the International Conference on Data Engineering*, Cancun, Mexico, 2008.

[23] P. Golle. Revisiting the uniqueness of simple demographics in the us population. In *Proc. of the Workshop on Privacy in the Electronic Society*, Alexandria, Virginia, USA, October 2006.

[24] Y. Ishai and E. Kushilevitz. Improved upper bounds on information-theoretic private information retrieval (extended abstract). In *Proc. of the 31st annual ACM Symposium on Theory of Computing*, Atlanta, Georgia, USA, May 1999.

[25] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *Proc. of the 8th ACM SIGKDD Internationale Conference on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada, 2002.

[26] V. Kann. Maximum bounded h-matching is max snp-complete. *Information Processing Letters*, 49:309318, 1994.

[27] E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proc. of the 38th Annual Symposium on Foundations of Computer Science*, Miami Beach, Florida, USA, October 1997.

[28] K. LeFevre, D. DeWitt., and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *Proc. of the International Conference on Data Engineering (ICDE'06)*, Atlanta, Georgia, USA, April 2006.

[29] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *Proc. of ACM SIGMOD Conference on Management of Data*, Baltimore, Maryland, USA, June 2005.

[30] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and ℓ-diversity. In *Proc. of the 23nd International Conference on Data Engineering*, Istanbul, Turkey, April 2007.

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

[32] A. Machanavajjhala, J. Gehrke, and D. Kifer. ℓ-density: Privacy beyond k-anonymity. In *Proc. of the International Conference on Data Engineering (ICDE'06)*, Atlanta, Georgia, USA, April 2006.

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

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

[35] M. Reiter and A. Rubin. Anonymous Web transactions with crowds. *Communications of the ACM*, 42(2):32–48, February 1999.

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

[37] K. Sampigethaya and R. Poovendran. A survey on mix networks and their secure applications. *Proceedings of the IEEE*, 94(12):2142–2181, December 2006.

[38] L. Sweeney. Achieving k-anonynity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):571–588, 2002.

[39] P. Syverson, D. Goldschlag, and M. Reed. Anonymous connections and onion routing. In *Proc. of the 18th Annual Symposium on Security and Privacy*, Oakland, California, USA, May 1997.

[40] T. M. Truta and B. Vinay. Privacy protection: p-sensitive k-anonymity property. In *Proc. of the 22nd International Conference on Data Engineering Workshop (ICDEW'06)*, Atlanta, Georgia, USA, April 2006.

[41] R. C.-W. Wong, J. Li, A. W.-C. Fu, and K. Wang. (α,k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, Pennsylvania, USA, August 2006.

[42] A. Yao. Protocols for secure computations. In *Proc. of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, Chicago, Illinois, USA, November 1982.