Protecting Respondents’ Identities in Microdata Release

Pierangela Samarati, Member, IEEE Computer Society

Abstract—Today’s globally networked society places great demand on the dissemination and sharing of information. While in the past released information was mostly in tabular and statistical form, many situations call today for the release of specific data (microdata). In order to protect the anonymity of the entities (called respondents) to which information refers, data holders often remove or encrypt explicit identifiers such as names, addresses, and phone numbers. Deidentifying data, however, provides no guarantee of anonymity. Released information often contains other data, such as race, birth date, sex, and ZIP code, that can be linked to publicly available information to reidentify respondents and inferring information that was not intended for disclosure. In this paper we address the problem of releasing microdata while safeguarding the anonymity of the respondents to which the data refer. The approach is based on the definition of $k$-anonymity. A table provides $k$-anonymity if attempts to link explicitly identifying information to its content map the information to at least $k$ entities. We illustrate how $k$-anonymity can be provided without compromising the integrity (or truthfulness) of the information released by using generalization and suppression techniques. We introduce the concept of minimal generalization that captures the property of the release process not to distort the data more than needed to achieve $k$-anonymity, and present an algorithm for the computation of such a generalization. We also discuss possible preference policies to choose among different minimal generalizations.

Index Terms—Privacy, data anonymity, disclosure control, microdata release, inference, record linkage, security, information protection.

1 INTRODUCTION

Information is today probably the most important and demanded resource. We live in an internetworked society that relies on the dissemination and sharing of information in the private as well as in the public and governmental sectors. Governmental, public, and private institutions are increasingly required to make their data electronically available [8], [18]. If in the past, this dissemination and sharing of information was mostly in statistical and tabular form, many situations require today that the specific stored data themselves, called microdata, be released. The advantage of releasing microdata instead of specific precomputed statistics is an increased flexibility and availability of information for the users. Microdata files produced by all Census Bureau demographic surveys and Federal agencies, such as the National Center for Education Statistics, Energy Information Administration, and Internal Revenue Services are today made available to purchasers and researchers. There are also databases maintained and released by the Department of Motor Vehicles (DMVs), Health Maintenance Organizations (HMOs), insurance companies, public offices, commercial organizations, and so on. To protect the privacy of the respondents (individuals, organizations, associations, business establishments, and so on) to which the data refer, released data are generally “sanitized” by removing all explicit identifiers such as names, addresses, and phone numbers. Although apparently anonymous, the deidentified data may contain other data, such as race, birth date, sex, and ZIP code, which uniquely or almost uniquely pertain to specific respondents (i.e., entities to which data refer) and make them stand out from others [13]. By linking these identifying characteristics to publicly available databases associating these characteristics to the respondent’s identity, the data recipients can determine to which respondent each piece of released data belongs, or restrict their uncertainty to a specific subset of individuals.

The large amount of information easily accessible today and the increased computational power available to the attackers make such linking attacks a serious problem. Information about us is collected every day, as we join associations or groups, shop for groceries, or execute most of our common daily activities [7], [8]. It has been estimated that, in the United States, there are currently about five billion privately owned records that describe each citizen’s finances, interests, and demographics. Information bureaus such as TRW, Equifax, and Trans Union hold the largest and most detailed databases on American consumers. Most municipalities sell population registers that include the identities of individuals along with basic demographics; examples include: local census data, voter lists, city directories, and information from motor vehicle agencies, tax assessors, and real estate agencies. In some states today, it is possible to get access to both the driver’s license and license plate files for a $25 fee. Typical data contained in these databases may include names, Social Security numbers, birth dates, addresses, telephone numbers, family status, and employment and salary histories. These data, which are often publicly distributed or sold, can be used for...
linking identities and deidentified information, thus allowing reidentification of the respondents.

The restricted access to information and expensive processing of it, in both time and resources, which represented a form of protection in the past, does not hold anymore. It is not difficult today for a data recipient to combine the “deidentified” microdata received with other publicly available data (e.g., voter registers). This situation has raised particular concerns in the medical and financial field, where microdata, which are increasingly being released for circulation or research, can be, or have been, subject to abuses compromising the privacy of individuals [2], [8], [12], [21].

To illustrate the problem, Fig. 1 exemplifies a table of medical data to be released. Data have been deidentified by suppressing names and Social Security numbers (SSNs) so not to disclose the identities of the individuals to whom the data refer. However, values of other released attributes, such as ZIP, DateOfBirth, Race, Sex, and MaritalStatus can also appear in some external table jointly with the individuals’ identities, and can therefore allow them to be tracked. As illustrated in Fig. 1, ZIP, DateOfBirth, and Sex can be linked to the Voter List to reveal the Name, Address, and City. For instance, in the Medical Data table there is only one female born on 9/15/61 and living in the 94142 area. This combination, if unique in the external world as well, uniquely identifies the corresponding bulleted tuple in the released data as pertaining to “Sue J. Carlson, 900 Market Street, San Francisco,” thus revealing that she has reported shortness of breath. (Notice that the medical information is not assumed to be publicly associated with the individuals, and the desired protection is to release the medical information in a way that the identities of the individuals cannot be determined. However, the released characteristics for Sue J. Carlson leads to determine which medical data among those released are hers.) While this example demonstrated an exact match, in some cases, linking can allow the identification of a restricted set of individuals to whom the released information could refer.

Several microdata disclosure protection techniques have been developed in the context of statistical databases, such as scrambling and swapping values and adding noise to the data while maintaining an overall statistical property of the result [1], [20]. However, many uses require release and explicit management of microdata while needing truthful information within each tuple. This “data quality” requirement makes inappropriate those techniques that disturb data and, therefore, although preserving statistical properties, compromise the correctness of the single pieces of information. Among the techniques proposed for providing anonymity in the release of microdata [13] we therefore, focus on two techniques in particular: generalization and suppression, which, unlike other existing techniques, such as scrambling or swapping, preserve the truthfulness of the information. Techniques for microdata release protection have been proposed and studied however, no formal model has been proposed for the specific use of generalization and suppression. Recently, two systems, namely, Mu-Argus [9] and Datafly [16], have been released which use suppression and generalization as techniques to provide disclosure control. However, again, no formal foundations or abstractions have

---

1. We will elaborate more on this in Section 7.
been provided for the techniques employed by both. Furthermore, approximations made by the systems can suffer from drawbacks, such as generalizing data more than is needed, like [16], or not providing adequate protection, like [9].

In this paper, we provide a formal foundation for the anonymity problem against linking and for the application of generalization and suppression techniques towards its solution. We introduce the definition of quasi-identifiers as attributes that can be exploited for linking and of k-anonymity as characterizing the degree of data protection with respect to inference by linking. We show how k-anonymity can be ensured in information release by generalizing and/or suppressing part of the data to be disclosed. Within this framework, we introduce the concepts of generalized table and of minimal generalization. Intuitively, a generalization is minimal if data are not generalized more than necessary to provide k-anonymity. We present an algorithm to compute a minimal generalization of a given table. We also introduce the concept of preferred generalization as a minimal generalization that satisfies defined preference criteria, and discuss possible preference criteria.

For simplicity and concreteness, we frame our work in the context of relational database systems. We note, however, that our approach does not depend on this assumption and can be applied to limit disclosure when information is represented with other data models.

The remainder of this paper is organized as follows: In Section 2, we introduce basic assumptions and definitions. In Section 3, we discuss generalization to provide anonymity. In Section 4, we introduce suppression and illustrate its application complementing generalization. In Section 5, we illustrate an approach to computing a minimal generalization and prove its correctness. In Section 6, we discuss some preference policies for choosing among different minimal generalizations. In Section 7, we present comparison with related work. Section 8 concludes the paper.

2 Assumptions and Preliminary Definitions

We assume the existence of a private table \( \mathcal{PT} \) to be anonymized for external release. The problem of releasing multiple tables with nondisjoint schemas can be easily reduced to the case considered by us by joining the tables. For the sake of simplicity, our discussion and examples refer to the privacy and reidentification of individuals, with the note that our approach is equally applicable to cases where information releases refer to different kinds of respondents (e.g., business establishments). In the following, we therefore use the terms individual and respondent interchangeably. Since removal of explicit identifiers is the first step to anonymization, we assume that all explicit identifiers (e.g., names, SSNs, and addresses) of respondents are either encrypted or suppressed, and we ignore them in the remainder of this paper. We use \( \mathcal{PT} \) to refer to the private table with explicit identifiers removed/encrypted.

Borrowing the terminology from [4], we call quasi-identifier a set of attributes in \( \mathcal{PT} \) that, in combination, can be linked with external information to reidentify the respondents to whom information refers. It is important to note that the set of attributes that constitute a quasi-identifier depends on the external information available to the recipient, as this determines the recipient’s linking ability (not all possible external tables are available to every possible data recipient). Different quasi-identifiers may therefore need to be considered for different recipients. In this paper, we consider the release with respect to a given data recipient. The consideration of different quasi-identifiers for different recipients requires the selective application of the approach for the specific quasi-identifiers depending on the data recipient to whom information has to be released. Similarly, we notice that more than one quasi-identifier may apply to each given data recipient. In particular, the number of quasi-identifiers to be considered for each data recipient depends on the different kinds of respondents whose identity is to be protected. In general, information in each table refers to one kind of respondent and, therefore, there is exactly one quasi-identifier for each possible data recipient. There may be however, cases where more quasi-identifiers need to be considered since data elements in the table to be released refer to more than one kind of respondent. As an example, consider a table where each tuple contains, in addition to information identifying the patient (like in Fig. 1), also information identifying the doctor, and both identities need to be protected. In the following, we assume the case of a single quasi-identifier. We also assume that \( \mathcal{PT} \) contains at most one tuple for each possible respondent.

Given a table \( T(A_1, \ldots, A_n) \), a set of attributes

\[
\{A_1, \ldots, A_j\} \subseteq \{A_1, \ldots, A_n\},
\]

and a tuple \( t \in T, t[A_1, \ldots, A_j] \) denotes the sequence of the values of \( A_1, \ldots, A_j \) in \( t \), \( T[A_1, \ldots, A_j] \) denotes the projection, maintaining duplicate tuples, of attributes \( A_1, \ldots, A_j \) in \( T \). Also, \( |T| \) denotes \( T \)'s cardinality, that is, the number of tuples in \( T \).

Our goal is to allow the release of information in the private table while ensuring the anonymity of the information respondents. Guaranteeing complete anonymity is obviously an impossible task. A reasonable approach consists in giving some measure of the anonymity protection. We do this by introducing the concept of k-anonymity. A data release is said to satisfy k-anonymity if every tuple released cannot be related to fewer than \( k \) respondents, where \( k \) is a positive integer set by the data holder, possibly as the result of a negotiation with other parties [2], [8]. We can then specify the requirement on the protection of information released against linking attacks in terms of a k-anonymity constraint on the data release as follows.

Definition 2.1 (k-anonymity requirement). Each release of data must be such that every combination of values of quasi-identifiers can be indistinctly matched to at least \( k \) individuals.

Satisfaction of the k-anonymity requirement requires knowing how many individuals each released tuple matches. This information can be known precisely only by

2. Note that requiring each released tuple to match with at least \( k \) individuals is different from requiring each piece of released information to match with at least \( k \) individuals. For instance, with reference to the above mentioned example, if all the \( k \) tuples matching a set of \( k \) individuals have exactly the same value for attribute HealthProblem, the health problem is inevitably disclosed.
explicitly linking the released data with externally available data. With reference to the example in Fig. 1, the data holder should link the data in the deidentified medical table with the voter list as well as with any other table externally available that may contain information that can be exploited for linking. (Note the use of the term “link” instead of “join,” to denote that this evaluation may be more complex than a simple join due to the fact that information in external tables may be expressed in a form different from that of private table PT.) This is obviously an impossible task for the data holder. Although we can assume that the data holder knows which attributes may appear in external tables and possibly be available to recipients and, therefore, which sets of attributes are quasi-identifiers, the specific values of data in external tables cannot be assumed. The key to satisfying the $k$-anonymity requirement is therefore, to translate the requirement in terms of the released data themselves. In order to do that, we require the following assumption to hold.

**Assumption 2.1.** All attributes in table PT which are to be released and which are externally available in combination (i.e., appearing together in an external table or in possible joins between external tables) to a data recipient are defined in a quasi-identifier associated with PT.

Although this is not a trivial assumption, its enforcement is indeed possible and is today considered in many practical releases of data [17]. Information contained in external tables is of public knowledge and, often, has been distributed by the same data holders that need to safeguard anonymity in the release of private data. Besides, knowing which attributes can be used for linking is a basic requirement for protection. At the worst, pessimistic approaches can be taken, including in a quasi-identifier some attributes where there is the possibility that they may be, or become, available to the recipient. In this paper, we therefore assume that quasi-identifiers have been properly recognized and defined.

We can now introduce the definition of $k$-anonymity for a table as follows:

**Definition 2.2 ($k$-anonymity).** Let $T(A_1, \ldots, A_n)$ be a table and $QI$ be a quasi-identifier associated with it. $T$ is said to satisfy $k$-anonymity wrt $QI$ if each sequence of values in $T[QI]$ appears at least with $k$ occurrences in $T[QI]$.

Under Assumption 2.1, and under the hypothesis that the privately stored table contains at most one tuple for each respondent to which a quasi-identifier refers, $k$-anonymity of a released table clearly represents a sufficient condition for the satisfaction of the $k$-anonymity requirement. In other words, a table satisfying Definition 2.2 for a given $k$ satisfies the $k$-anonymity requirement for such a $k$. Intuitively, if each set of attributes part of external tables appears in a quasi-identifier associated with the table, and the table satisfies $k$-anonymity, then the combination of released data with external data will never allow the recipient to associate each released tuple with less than $k$ individuals. Consider a quasi-identifier $QI$, if Definition 2.2 is satisfied, each tuple in $PT[QI]$ has at least $k$ occurrences. Since the population of the private table is a subset of the population of the outside world, there will be at least $k$ individuals in the outside world matching these values. Moreover, since all attributes available outside in combination are included in $QI$, no additional attributes can be joined to $QI$ to reduce the cardinality of such a set. Also, no subset of $QI$ can compromise $k$-anonymity: If a quasi-identifier $QI$ has at least $k$ occurrences for each sequence of values, any subset of attributes in it will appear with $k' \geq k$ occurrences, that is, it will refer to $k' \geq k$ individuals.

To illustrate, consider the situation exemplified in Fig. 1, but assume that the released data contain two occurrences of the sequence "white,09/15/61,female, 94142,window". Then, at least two individuals matching such sequence will exist in the outside world represented by the voter list and it will not be possible for the data recipient to determine which of the two medical records associated with these values of the quasi-identifier belongs to which of the two individuals. Since $k$-anonymity with $k = 2$ was provided in the release, each released medical record could indistinctly belong to at least two individuals.

Given the assumption and definitions above and given a private table PT to be released and a quasi identifier QI, we focus on the problem of producing a version of PT that satisfies $k$-anonymity wrt $QI$.

### 3 Generalizing Data

Our first approach to providing $k$-anonymity is based on the definition and use of generalization relationships between domains and between values that attributes can assume.

#### 3.1 Generalization Relationships

In relational database systems, a domain (e.g., integer, string, date) is associated with each attribute to indicate the set of values that the attribute can assume. We refer to these domains as *ground*. We then extend the notion of domain to capture the *generalization* process by also assuming the existence of a set of (generalized) domains containing generalized values and of a mapping between each domain and domains generalization of it. For instance, ZIP codes can be generalized by dropping, at each generalization step, the least significant (rightmost) digit; postal addresses can be generalized to the street (dropping the number), then to the city, to the county, to the state, and so on. This mapping is stated by means of a generalization relationship $\leq_D$. Given two domains $D_i$ and $D_j$, relationship $D_i \leq_D D_j$ describes the fact that values in domain $D_i$ are generalization of values in domain $D_j$. Generalization relationship $\leq_D$ defines a partial order on the set Dom of (ground and generalized) domains, and is required to satisfy the following conditions:

1. $\forall D_i, D_j, D_k \in \text{Dom} : D_i \leq_D D_j, D_i \leq_D D_k \implies D_j \leq_D D_k \lor D_k \leq_D D_j$.

2. All maximal elements of Dom are singleton.

3. A universal relation combining external tables can be imagined [19].
Condition 1 states that for each domain \( D_i \), the set of domains generalization of \( D_i \) is totally ordered and, therefore, each \( D_i \) has at most one direct generalized domain. This condition ensures determinism in the generalization process. Condition 2 ensures that all values in each domain can be eventually generalized to a single value. The definition of the generalization relationship implies the existence, for each domain \( D \in \text{Dom} \), of a totally ordered hierarchy, called domain generalization hierarchy, \( \text{DGH} \).

A value generalization relationship, partial order \( \leq_v \), is also defined that associates with each value in domain \( D_i \) a unique value in domain \( D_j \) direct generalization of \( D_i \). The value generalization relationship implies the existence, for each domain \( D \), of a value generalization hierarchy \( \text{VGH} \). It is easy to observe that each \( \text{VGH} \) defines a tree whose leaves are values of \( D \) and whose root is the value in the domain that is the maximal element in \( \text{DGH} \).

Example 3.1. Fig. 2 illustrates an example of domain and value generalization hierarchies for domains: \( Z_0 \), representing a subset of the ZIP codes of San Francisco, CA; \( R_0 \), representing races; \( M_0 \), representing marital status; and \( S_0 \), representing sex. The generalization relationship specified for ZIP codes generalizes a 5-digit ZIP code, first to a 4-digit ZIP code, and then to a 3-digit ZIP code. The other hierarchies in the figure are of immediate interpretation.

In the remainder of this paper, we will often refer to a domain or value generalization hierarchy in terms of the graph representing all and only the direct generalizations between the elements in it (implied generalizations do not appear as arcs in the graph). Consequently, we will use the term hierarchy indiscriminately, to denote either a partially ordered set or the graph representing it. We will explicitly refer to the ordered set or to the graph when necessary.

Since we will be dealing with sets of attributes, it is useful to visualize the generalization relationship and hierarchies in terms of tuples composed of elements of \( \text{Dom} \) or of their values. Given a domain tuple \( DT = (D_1, \ldots, D_n) \) such that \( D_i \in \text{Dom}, i = 1, \ldots, n \), we define the domain generalization hierarchy of \( DT \) as \( \text{DGH}_{DT} = \text{DGH}_{D_1} \times \ldots \times \text{DGH}_{D_n} \), where the Cartesian product is ordered by imposing coordinate-wise order [5]. Since each \( \text{DGH}_{D_i} \) is totally ordered, \( \text{DGH}_{DT} \) defines a lattice with \( DT \) as its minimal element and the tuple composed of the top of each \( \text{DGH}_{D_i}, i = 1, \ldots, n \) as its maximal element. The generalization hierarchy of a domain tuple \( DT \) defines the different ways in which \( DT \) can be generalized: Each path from \( DT \) to the unique maximal element of \( \text{DGH}_{DT} \) in the graph describing \( \text{DGH}_{DT} \) defines a possible alternative path that can be followed when generalizing a quasi-identifier \( QI = \{A_1, \ldots, A_n\} \) of attributes on domains \( D_1, \ldots, D_n \). We refer to the set of nodes in each of such paths together with the generalization relationship between them as a generalization strategy for \( \text{DGH}_{DT} \). In correspondence with each generalization strategy of a domain tuple, there is a value generalization strategy describing the generalization at the value level.

Example 3.2. Consider domains \( R_0 \) (race) and \( Z_0 \) (ZIP code) whose generalization hierarchies are illustrated in Fig. 2. Fig. 3 illustrates the domain generalization hierarchy of the domain tuple \( (R_0, Z_0) \) together with the corresponding domain and value generalization strategies. There are three different generalization strategies, corresponding to the three paths from the bottom to the top element of lattice \( \text{DGH}_{(R_0,Z_0)} \).

3.2 Generalized Table and Minimal Generalization

Given a private table \( PT \), our first approach to provide \( k \)-anonymity consists in generalizing the values stored in the table. Intuitively, attribute values stored in the private table can be substituted, upon release, with generalized values. Since multiple values can map to a single generalized value, generalization may decrease the number of distinct tuples, thereby possibly increasing the size of the clusters containing tuples with the same values. We perform generalization at the attribute level. Generalizing an attribute means substituting its values with corresponding values from a more general domain. Generalization at the attribute level ensures that all values of an attribute belong to the same domain. However, as a result of the generalization process, the domain of an attribute can change. Note that since the domain of an attribute can change and since generalized values can be used in place of more specific ones, it is important that all the domains in a generalization hierarchy be compatible. Compatibility can be ensured by using the same storage representation form for
Definition 3.1 (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. $T_j$ is said to be a generalization of $T_i$, written $T_i \preceq T_j$, iff:

1. $|T_i| = |T_j|$. 
2. $\forall A_z \in \{A_1, \ldots, A_n\} : \text{dom}(A_z, T_i) \preceq \text{dom}(A_z, T_j)$. 
3. It is possible to define a bijective mapping between $T_i$ and $T_j$ that associates each tuple $t_i \in T_i$ with a tuple $t_j \in T_j$ such that $t_i[A_z] \preceq v t_j[A_z]$ for all $A_z \in \{A_1, \ldots, A_n\}$.

Definition 3.1 states that table $T_j$ is a generalization of table $T_i$, defined on the same set of attributes, iff:

1. $T_i$ and $T_j$ have the same number of tuples; 
2. The domain of each attribute $A_z$ in $T_j$ is equal to, or is a generalization of, the domain of $A_z$ in $T_i$; and 
3. It is possible to define a bijective mapping associating each tuple $t_i$ in $T_i$ with a tuple $t_j$ in $T_j$ such that the value of each attribute $A_z$ in $t_j$ is equal to, or is a generalization of, the value of $A_z$ in $t_i$.

Example 3.3. Consider table $PT$ illustrated in Fig. 4 and the domain and value generalization hierarchies for $R_0$ and $Z_0$ illustrated in Fig. 2. Assume $QI = \{\text{Race}, \text{ZIP}\}$ to be a quasi-identifier. The remaining five tables in Fig. 4 are all possible generalized tables for $PT$. For the clarity of the
example, each table reports the domain for each attribute in the table. With respect to $k$-anonymity, $\text{GT}_{[0,1]}$ satisfies $k$-anonymity for $k = 1, 2; \text{GT}_{[1,0]}$ satisfies $k$-anonymity for $k = 1, 2, 3; \text{GT}_{[0,2]}$ satisfies $k$-anonymity for $k = 1, \ldots, 4, \text{GT}_{[1,1]}$ satisfies $k$-anonymity for $k = 1, \ldots, 6$, and $\text{GT}_{[1,2]}$ satisfies $k$-anonymity for $k = 1, \ldots, 12$.

Given a table $T$, different possible generalizations exist. Not all generalizations, however, can be considered equally satisfactory. For instance, the trivial generalization bringing each attribute to the highest possible level of generalization, thus collapsing all tuples in $T$ to the same list of values, provides $k$-anonymity at the price of a strong generalization of the data. Such extreme generalization is not needed if a more specific table (i.e., containing more specific values) exists which satisfies $k$-anonymity. This concept is captured by the definition of $k$-minimal generalization. To introduce it we first introduce the notion of distance vector.

**Definition 3.2 (Distance vector).** Let $T_1(A_1, \ldots, A_n)$ and $T_2(A_1, \ldots, A_n)$ be two tables such that $T_1 \preceq T_2$. The distance vector of $T_1$ from $T_2$ is the vector $DV_{i,j} = [d_1, \ldots, d_n]$, where each $d_i, i = 1, \ldots, n$, is the length of the unique path between $D_i = \text{dom}(A_i, T_1)$ and $dom(A_i, T_2)$ in the domain generalization hierarchy $\text{DHG}_{D_i}$.

**Example 3.4.** Consider table $\text{PT}$ and its generalizations illustrated in Fig. 4. The distance vectors between $\text{PT}$ and each of its generalized tables is the vector appearing as a subscript of the table.

We extend the dominance relationship $\preceq$ on integers to distance vectors by requiring coordinate-wise ordering as follows: Given two distance vectors $DV = [d_1, \ldots, d_n]$ and $DV' = [d'_1, \ldots, d'_n]$, $DV \leq DV'$ if $d_i \leq d'_i$ for all $i = 1, \ldots, n$. Moreover, $DV < DV'$ if $DV \leq DV'$ and $DV \neq DV'$.

A generalization hierarchy for a domain tuple can be seen as a hierarchy (lattice) on the corresponding distance vectors. For instance, Fig. 5 illustrates the lattice representing the dominance relationship between the distance vectors corresponding to the possible generalizations of $(R_0, Z_0)$. In the following we denote with $\text{VL}_{\text{DT}}$ the hierarchy (lattice) of distance vectors corresponding to generalization hierarchy $\text{DHG}_{\text{DT}}$.

We can now introduce the definition of $k$-minimal generalization.

**Definition 3.3 (k-minimal generalization).** Let $T_1(A_1, \ldots, A_n)$ and $T_2(A_1, \ldots, A_n)$ be two tables such that $T_1 \preceq T_2$. $T_1$ is said to be a $k$-minimal generalization of $T_2$ if:

1. $T_1$ satisfies $k$-anonymity (Definition 2.2).
2. $\forall T_i : T_i \preceq T_2, T_i$ satisfies $k$-anonymity $\Rightarrow \neg(DV_{i,z} \leq DV_{i,j}).$

Intuitively, a generalization $T_1(A_1, \ldots, A_n)$ is $k$-minimal if there does not exist another generalization $T_2(A_1, \ldots, A_n)$ satisfying $k$-anonymity and whose domain tuple is dominated by $T_1$ in the domain generalization hierarchy of $\langle \text{dom}(A_1, T_1), \ldots, \text{dom}(A_n, T_1) \rangle$ (or, equivalently, in the corresponding lattice of distance vectors). If this were the case, $T_1$ would itself be a generalization for $T_2$. Note also that a table $T_i$ is the minimal generalization of itself for all $k$ such that $T_i$ satisfies $k$-anonymity.
Example 3.5. Consider table PT and its generalized tables illustrated in Fig. 4. For \( k = 2 \) two \( k \)-minimal generalizations exist, namely, \( GT_{[1]} \) and \( GT_{[0.1]} \). Among the other generalizations satisfying the \( k \)-anonymity requirement, \( GT_{[0.2]} \) is not minimal since it is a generalization of \( GT_{[0.1]} \); \( GT_{[1.1]} \) cannot be minimal since it is a generalization of both \( GT_{[1.0]} \) and \( GT_{[0.1]} \); \( GT_{[1.2]} \) is not minimal since it is a generalization of all of them. Also, there are only two \( k \)-minimal generalized tables for \( k = 3 \), which are \( GT_{[0.0]} \) and \( GT_{[0.2]} \).

Note that since \( k \)-anonymity requires the existence of \( k \) occurrences for each sequence of values only for attributes in the quasi-identifier, for every minimal generalization \( T_j \) of \( T_i \), \( \text{dom}(A_j, T_j) = \text{dom}(A_i, T_i) \) (or, equivalently, \( D V_{ij}[d_j] = 0 \)) for all attributes \( A_i \) that do not belong to the quasi-identifier. In other words, the generalization process operates only on attributes in the considered quasi-identifier.

4 SUPPRESSING DATA

In Section 3, we discussed how, given a private table PT, a generalized table can be produced which releases a more general version of the data in PT that satisfies a \( k \)-anonymity constraint. Generalization has the advantage of allowing the release of all the single tuples in the table, although in a more general form. Here, we illustrate a complementary approach to providing \( k \)-anonymity, which is suppression. Suppressing means to remove data from the table so that they are not released. Like generalization, suppression has been proposed in the context of micro and macrodata release, and is often used in the context of statistical databases [3], [13].

Here, we illustrate its particular application in complementing generalization to provide \( k \)-anonymity.

We apply suppression at the tuple level, that is, a tuple can be suppressed only in its entirety. Suppression is used to “moderate” the generalization process when a limited number of outliers (i.e., tuples with less than \( k \) occurrences) would force a great amount of generalization. To clarify, consider the table illustrated in Fig. 1, whose projection on quasi-identifier \( QI = \{ \text{Race}, \text{DateOfBirth}, \text{Sex}, \text{ZIP}, \text{MaritalStatus} \} \) is reported in Fig. 6 and suppose \( k \)-anonymity with \( k = 2 \) is to be provided. Suppose also that attribute \( \text{DateOfBirth} \) has a domain date with the following generalizations: from the specific date (\( \text{mm/dd/yyyy} \)) to the month (\( \text{mm/yyyy} \)) to the year (\( \text{yyyy} \)) to a 5-year interval (e.g., [60–64]) to a 10-year interval (e.g., [60, 69]) and so on.4 It is easy to see that because of the last tuple in the table, for the \( k \)-anonymity requirement to be satisfied, we need two steps of generalization on \( \text{DateOfBirth} \), one step of generalization on \( \text{ZIPCode} \), one step of generalization on \( \text{MaritalStatus} \), and either one further step on \( \text{Sex} \), \( \text{ZIPCode} \), and \( \text{MaritalStatus} \), or, alternatively, on \( \text{Race} \) and \( \text{DateOfBirth} \). The two possible minimal generalizations are illustrated in Fig. 6. It can be easily seen that had the last tuple not been present, \( k \)-anonymity for \( k = 2 \) could have been simply achieved by one step of generalization on attribute \( \text{DateOfBirth} \), as illustrated in Fig. 7. Suppressing the tuple therefore allows the satisfaction of the \( k \)-anonymity requirement with less generalization.

In illustrating how suppression interplays with generalization to provide \( k \)-anonymity, we begin by restating the definition of generalized table as follows:

Definition 4.1 (Generalized Table—With Suppression). 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. \( T_j \) is said to be a generalization of \( T_i \), written \( T_i \preceq T_j \), iff:

1. \( |T_j| \leq |T_i| \).
2. \( \forall A_j \in \{ A_1, \ldots, A_n \} : \text{dom}(A_j, T_i) \leq \text{dom}(A_j, T_j). \)
3. It is possible to define an injective mapping between \( T_i \) and \( T_j \) that associates each tuple \( t_i \in T_i \) with a tuple \( t_j \in T_j \) such that \( t_i[A_j] \leq t_j[A_j] \) for all \( A_j \in \{ A_1, \ldots, A_n \} \).

The definition above differs from Definition 3.1 since it allows tuples appearing in \( T_i \) not to have any corresponding generalized tuple in \( T_j \). Intuitively, tuples in \( T_i \) not having any corresponding tuple in \( T_j \) are tuples that have been suppressed.

Definition 4.1 allows any amount of suppression in a generalized table. Obviously, we are not interested in tables that suppress more tuples than necessary to achieve \( k \)-anonymity at a given level of generalization. This is captured by the following definition.

Definition 4.2 (Minimal required suppression). Let \( T_i \) be a table and \( T_j \) a generalization of \( T_i \) satisfying \( k \)-anonymity. \( T_j \) is said to enforce minimal required suppression iff

\[
\forall T_z : T_z \preceq T_j, D V_{iz} = D V_{ij},
\]

\( T_z \) satisfies \( k \)-anonymity \( \Rightarrow |T_j| \geq |T_z| \).

Definition 4.2 states that a table \( T_j \) enforces minimal required suppression wrt a \( k \)-anonymity requirement if there does not exist another generalization \( T_z \) with the same distance vector as \( T_j \) that satisfies the \( k \)-anonymity requirement by suppressing less tuples.

Example 4.1. Consider table PT and its generalizations illustrated in Fig. 8. The tuples written in bold and marked with double lines in each table are the tuples that must be suppressed to achieve \( k \)-anonymity of 2. Suppression of any set of tuples not including them all would not reach the required anonymity. Suppression of any superset would be unnecessary (not satisfying minimal required suppression).

With suppression, more generalized tables with the same distance vector and satisfying a given \( k \)-anonymity requirement may exist. However, as the lemma below states, there is a unique generalization \( T_j \) among them which enforces minimal required suppression. Any other generalization providing \( k \)-anonymity with the same distance vector would suppress more tuples, and more precisely a proper superset of the tuples suppressed in \( T_j \). (Condition \( |T_j| \geq |T_z| \) in Definition 4.2 can equivalently be
expressed as $T_{ij} \geq T_i$. Intuitively, the generalization enforcing minimal required suppression at a given distance vector can be obtained by first applying the generalization corresponding to the distance vector and then removing all and only the tuples that appear with fewer than $k$ occurrences.
Lemma 4.1. Let $T_i(A_1, \ldots, A_n)$ be a table, $D_i$ be the domain of $A_i$, and $h_i$ be the height of domain generalization hierarchy $DGH_{D_i}, i = 1, \ldots, n$. For all distance vectors $DV_i = [0, \ldots, 0]$ and integers $k, 0 < k \leq |T_i|$: 1) There exists one and only one generalized table $T_j$, $T_i \preceq T_j$, $DV_{i;j} = DV_i$, satisfying $k$-anonymity by enforcing minimal required suppression; 2) $\forall T_z, T_i \preceq T_z, T_z$ satisfies $k$-anonymity, $DV_{z;j} = DV_{z;i} : T_i \neq T_j \Rightarrow T_j \supset T_z$.

Proof. Consider table $T_i$ and a generalization $T_j$ of it, which satisfies $k$-anonymity and enforces minimal required suppression. Note that such a $T_j$, possibly empty, always exists. Consider a table $T_j$, generalization of $T_i$, that provides $k$-anonymity such that $T_i \neq T_j$ and $DV_{i;j} = DV_{i;i}$. Let $T = T_i \cap T_j$ be the set of tuples that are in $T_i$, and are not in $T_j$. Table $T_j \cup T$ is a table with distance vector equal to $DV_{i;j}$ and $DV_{i;i}$. Since both $T_j$ and $T_z$ satisfy $k$-anonymity, also $T_j \cup T$ trivially does. Moreover, $|T_j \cup T| \geq |T_j|$. Since $T_j$ enforces minimal required suppression, $|T_j| \geq |T_j \cup T|$. Hence, $|T_j| = |T_j \cup T|$ that, since $T \cap T_j = \emptyset$, implies $T = \emptyset$ that, together with $T_i \neq T_j$, implies $T_j \supset T_z$ which proves the lemma. □

Given Lemma 4.1, in the remainder of the paper we restrict our attention to generalizations enforcing minimal required suppression and when referring to the generalization satisfying a $k$-anonymity constraint at a given distance vector, we will intend the unique generalization with that distance vector that satisfies the $k$-anonymity constraint enforcing minimal required suppression. To illustrate, with reference to table $PT$ in Fig. 8 and with respect to $k$-anonymity with $k = 2$, we refer to its generalizations where tuples in bold have been suppressed, as illustrated in Fig. 9.

Generalization and suppression are two different approaches to obtaining, from a given table, a table that satisfies $k$-anonymity. The two approaches clearly produce the best results when jointly applied. For instance, we have already observed how, with respect to the table in Fig. 7 and the $k$-anonymity requirement with $k = 2$, generalization alone is unsatisfactory (see Fig. 6). Suppression alone, on the other side, would also behave badly as it would require suppression of all the tuples in the table. Joint application of the two techniques allows, instead, the release of a table like the one in Fig. 7, where the last tuple has been suppressed. The question is therefore, whether it is better to generalize, at the cost of less precision in the data, or to suppress, at the cost of losing completeness. From observations of real-life applications and requirements, we assume the following: We consider an acceptable suppression threshold $\text{MaxSup}$ is specified stating the maximum number of suppressed tuples that is considered acceptable. Within this acceptable threshold, suppression is considered preferable to generalization (in other words, it is better to suppress more tuples than to enforce more generalization). The reason for this is...
that suppression affects single tuples, whereas general-
ization modifies all the values associated with an attribute, 
thus affecting all the tuples in the table. Tables that enforce 
suppression beyond MaxSup are considered unacceptable.

Given these assumptions, we can now restate the def-
ition of k-minimal generalization taking suppression 
into consideration.

Definition 4.3 (k-minimal generalization—without sup-
presion). Let Tj and Tj be two tables such that Tj ≤ Tj, 
and let MaxSup be the specified threshold of acceptable suppression. 
Tj is said to be a k-minimal generalization of a table Tj iff:
1. Tj satisfies k-anonymity enforcing minimal required 
suppression (Definitions 2.2 and 4.2).
2. |Tj| − |Tj| ≤ MaxSup.
3. ∀Tj : T ≤ Tj and Tj satisfies Conditions 1 and 2 
   ⇒ ¬(DVj ≤ DVj). 

Intuitively, generalization Tj is k-minimal iff it satisfies k-
anonymity, it does not enforce more suppression than it is 
allowed, and there does not exist another generalization 
satisfying these conditions with a distance vector smaller 
than that of Tj. (Note that from Lemma 4.1, we do not need to 
consider tables with the same distance vector as Tj.)

Example 4.2. Consider private table PT illustrated in Fig. 8, 
the k-anonymity requirement with k = 2, and the conse-
quent generalizations providing k-anonymity illustrated in 
Fig. 9. Depending on the acceptable suppression threshold, 
the following generalizations are considered minimal:

- MaxSup = 0: GT[1,1] (GT[0,0], GT[1,0], GT[0,1]), and GT[0,2] 
suppress more tuples than it is allowed, GT[2,2] is 
not minimal because of GT[1,1];
- MaxSup = 1: GT[1,0] and GT[0,2] (GT[0,0] and GT[1,1]) 
suppress more tuples than it is allowed, GT[1,1] is 
not minimal because of GT[1,0] and GT[0,1];
- MaxSup = 2.3: GT[1,0] and GT[1,1] (GT[0,0]) suppress 
mores tuples than it is allowed, GT[0,2] is 
not minimal because of GT[1,0], GT[1,1] and GT[0,2];
- MaxSup ≥ 4: GT[0,0] (all the other generalizations 
   are not minimal because of GT[0,0]).

A question to be addressed is what is actually to be 
suppressed, when suppression is applied. In the discussion 
above we talked about suppression of a tuple meaning that 
all the elements of attributes in the quasi-identifier (and not 
a proper subset of them) are suppressed. It is then to see 
whether the corresponding elements not belonging to the 
 quasi-identifier (HealthProblem in the example of Fig. 1) 
should be suppressed also or should be released. If they are 
suppressed, then no information on the stored data is 
conveyed to the recipient, although the number of 
suppressed tuples could be safely released to allow 
consideration of the fact that some tuples have been 
suppressed in aggregated data or statistics that the recipient 
may wish to derive. We observe that such a complete 
blanking of information is not necessary when the null 
 quasi-identifier values could refer to more than k entities in 
the universe of discourse. In other words, it is not necessary 
when there are more than k individuals in the external 
tables to whom the tuples whose quasi-identifier elements 
have been suppressed could refer, situation this that 
appears very likely.

5 Computing a k-minimal Generalization

We have defined the concept of k-minimal generalization 
corresponding to a given private table. Here, we illustrate 
an approach to computing such a generalization.

In the following, we restrict our attention to private 
tables with cardinality at least equal to the k-anonymity 
requirement to be satisfied (i.e., |PT| ≥ k). Trivially, the 
problem would not make sense otherwise since if |PT| < k 
then no generalization of PT (but the empty table) exists 
which satisfies k-anonymity. Condition |PT| ≥ k is therefore 
a necessary condition for k-anonymity. Since the maximal 
value of each domain generalization hierarchy is single-
lon (Condition 2 in Section 3) the condition is also sufficient. 
Also, given that the k-anonymity property is required only 
for attributes in quasi-identifiers, we consider only attri-
butes in the considered quasi-identifier. More precisely, 
instead of considering the whole table PT to be generalized, 
we consider its projection PT[QI], keeping duplicates, on 
the attributes of the considered quasi-identifier QI.

In Section 3, we illustrated the concepts of generalization 
hierarchy and strategies for a domain tuple. Given a table 
T = (A1, . . . , An) = PT[QI], the corresponding domain gen-
eralization hierarchy on DT = (dom(A1), . . . , dom(An), T) 
pictures all the possible generalizations and their relation-
ships. Each path (strategy) in it defines a different way in 
which the generalization process can be enforced. With 
respect to a strategy, we can define the concept of locally 
 minimal generalization as the generalization that is 
minimal, i.e., lowest in the hierarchy, among those 
satisfying k-anonymity, with respect to the set of general-
izations in the strategy. As the following theorem states, 
each k-minimal generalization is locally minimal with 
respect to some strategy.

Theorem 5.1 Let T = (A1, . . . , An) = PT[QI] be a table to be 
generalized and let DT = (D1, . . . , Dn), where

Dz = dom(Az, Tz), z = 1, . . . , n,

be the domain tuple of the corresponding ground domains. 
Every k-minimal generalization of T is a locally minimal 
generalization for some strategy of DGHyr.

Proof. By contradiction. Suppose Tj is k-minimal but is not 
locally minimal with respect to any strategy. Then, there 
exists a strategy that contains Tj and also contains a 
generalization Tj that satisfies k-anonymity by suppress-
sing no more tuples than what is allowed and such that 
Tj ≤ Tj ≤ Tj, Tj ̸= Tj. Hence, Tj satisfies Conditions 1 and 
2 of Definition 4.3. Moreover, since Tj ≤ Tj and Tj ̸= Tj, 
then DVj < DVj. Hence, Tj cannot be minimal, which 
contradicts the assumption.

It is important to note that since strategies are not disjoint, 
the converse is not necessarily true, that is, a generalization 
locally minimal with respect to a strategy might not be a
k-minimal generalization. Consider Example 4.2, where the different strategies are illustrated in Fig. 3, and assume MaxSup = 1. GT\textsubscript{1,1} is locally minimal with respect to the second strategy. However it is not minimal because of GT\textsubscript{1,0} (appearing in the first strategy).

From Theorem 5.1, a k-minimal preferred generalization can be naively computed by following each generalization strategy from the domain tuple to the maximal element of the hierarchy and stopping, in each strategy, at the first generalization which satisfies k-anonymity within the allowed suppression threshold. The non-k-minimal generalizations can then be discarded, and the preferred generalization chosen. The fact that the local minimal might not necessarily be a minimal solution implies the need to search all the strategies. This process is clearly much too costly, given the high number of strategies that should be followed. The number of different strategies for a domain tuple $DT = \langle D_1, \ldots, D_n \rangle$ is $(h_1 + \cdots + h_n)^n / h_1 \cdot \cdots \cdot h_n$, where each $h_i$ is the length of the path from $D_i$ to the top domain in DGH\textsubscript{D}.

The key to cut down such a computation lies in the observation that the number of tuples (outliers) that need to be removed to satisfy a k-anonymity requirement can only decrease going up in a strategy. Hence, the cardinality of the table enforcing minimal required suppression to satisfy a k-anonymity constraint can only increase going up in a strategy. This observation is formalized by the following theorem.

**Theorem 5.2.** Let $T_T = PT[QI]$ be a table to be generalized and $T_j$ and $T_z$ with $T_i \preceq T_j$ and $T_i \preceq T_z$ be two of its generalizations enforcing minimal required suppression. Then, $DV_{i,j} < DV_{i,z} \Rightarrow |T_j| \leq |T_z|$. 

**Proof.** We suppose the lemma does not hold and derive a contradiction. Suppose $DV_{i,j} < DV_{i,z}$ and $|T_j| > |T_z|$. Then, there exists a set $T \subset T_j$ of tuples that do not appear (at the more general domain tuple) in $T_j$. Let $T'$ be the generalization of all the tuples in $T$ at the domain tuple of $T_j$. Consider $T' = T_j \cup T$. Since tuples in $T'$ have not been suppressed in $T_j$, they do not represent outliers in $T_j$. Hence, $T'$ cannot represent outliers in $T_j$. Then, $T'$ is a table at the same distance vector as $T_j$ which provides k-anonymity and such that $T' \preceq T_j$, which implies $|T'| > |T_j|$, contradicting the assumption that $T_j$ enforces minimal required suppression. \(\square\)

Directly from the theorem above, observing that $|T_j| \leq |T_z|$ implies $(|T_j| - |T_z|) \geq (|T_j| - |T_z|)$, we can state the following:

**Corollary 5.1.** Let $T_T = PT[QI]$ be a table to be generalized and let $T_j$ and $T_z$, with $T_i \preceq T_j$ and $T_i \preceq T_z$, be two of its generalizations satisfying k-anonymity by enforcing minimal required suppression such that $DV_{i,j} < DV_{i,z}$. If $|T_j| - |T_z| \leq \text{MaxSup}$ then, $|T_j| - |T_z| \leq \text{MaxSup}$. Analogously, if $|T_j| - |T_z| > \text{MaxSup}$ then, $|T_j| - |T_z| > \text{MaxSup}$. 

Hence, if a table $T_j$ with distance vector $DV_{i,z}$ cannot provide k-anonymity by suppressing a number of tuples lower than MaxSup, then also all tables $T_j$ such that $DV_{i,j} < DV_{i,z}$ cannot.

In the following, we make use of the information on the heights of distance vectors. The height of a distance vector $DV$ in a distance vector lattice VL, denoted by $\text{height}(DV, VL)$, is the length of the paths from $DV$ to the minimal element of VL. The following lemma states the relationships between distance vectors and heights.

**Lemma 5.1.** Let $VL = (DV, \leq)$ be a lattice of distance vectors. 

\[\forall DV_{i,j}, DV_{i,z} \in DV : DV_{i,j} < DV_{i,z} \Rightarrow \text{height}(DV_{i,j}, VL) < \text{height}(DV_{i,z}, VL).\]

**Proof.** Trivially, since $DV_{i,j} < DV_{i,z}$, $DV_{i,z}$ dominates $DV_{i,j}$ in $VL$. Then, there is a path from $DV_{i,z}$ to the bottom element of the lattice passing by $DV_{i,j}$. Hence, the height of $DV_{i,z}$ is at least equal to the sum of $\text{height}(DV_{i,j}, VL)$ and the length of the path from $DV_{i,z}$ to $DV_{i,j}$. \(\square\)

Note that since the minimal element is the zero vector and each edge corresponds to incrementing of 1 exactly one element in a distance vector, the height of a vector in the lattice is the sum of the elements in it, that is, $\text{height}(d_1, \ldots, d_n, VL) = \sum_{i=1}^{n} d_i$.

By using the height of the vectors, we can apply the reasoning above across strategies. Let $T$ be the maximal element of $VL$. For each height $h$, $0 \leq h \leq \text{height}(T, VL)$, if there is no vector at height $h$ corresponding to a generalization that satisfies Conditions 1 and 2 of Definition 4.3, then there cannot be any vector corresponding to a generalization which satisfies these conditions at height lower than $h$. We exploit this property by searching for a vector producing a k-minimal generalization through the application of a binary search approach on the lattice $VL$ of distance vectors corresponding to the domain generalization hierarchy of the domains of the quasi-identifier attributes. Consider lattice $VL$ of height $h = \text{height}(T, VL)$. First, the vectors at height $\lfloor \frac{h}{2} \rfloor$ are evaluated. If there is a vector that satisfies k-anonymity within the suppression threshold established at height $\lfloor \frac{h}{2} \rfloor$, then the vectors at height $\lfloor \frac{h}{4} \rfloor$ are evaluated, otherwise those at height $\lfloor \frac{3h}{4} \rfloor$, and so on, until we reach the lowest height for which there is a distance vector that satisfies k-anonymity by suppressing no more tuples than MaxSup. The process, whose interpretation is straightforward, is reported as algorithm Find_vector in Fig. 10. The call to function satisfies returns true if the generalization enforcing minimal required suppression at distance vector vec satisfies k-anonymity by suppressing no more tuples than MaxSup. It returns false otherwise.

The following theorem states the correctness of the process.

**Theorem 5.3.** Let $T_T = PT[QI]$ be a table to be generalized, $k \leq |PT|$ the k-anonymity constraint, and MaxSup the maximum suppression threshold allowed. 1) Find_vector always terminates by returning a vector sol. 2) Table GT\textsubscript{sol} generalization of $T_T$ at distance vector sol providing k-anonymity by enforcing minimal required suppression, is a k-minimal solution according to Definition 4.3.

**Proof.** At every iteration of the cycle in Step 2, $low \leq \text{try} < hight$. At the end of the cycle, either $low$ is set to $\text{try} + 1$, or $hight$ is set to $\text{try}$. Hence, condition

5. All such paths have the same length.
Find_vector

INPUT: Table $T_i = PT[Q_I]$ to be generalized, anonymity requirement $k$, suppression threshold $\text{MaxSup}$, lattice $\mathcal{VL}_{DT}$ of the distance vectors corresponding to the domain generalization hierarchy $\mathcal{DG}_{DT}$, where $DT$ is the tuples of the domains of the quasi-identifier attributes.

OUTPUT: The distance vector $sol$ of a generalized table $GT_{sol}$ that is a $k$-minimal generalization of $PT[Q_I]$ according to Definition 4.3.

METHOD: Executes a binary search on $\mathcal{VL}_{DT}$ based on height of vectors in $\mathcal{VL}_{DT}$.

1. $low := 0$; $high := \text{height}(T, \mathcal{VL}_{DT})$; $sol := T$
2. while $low < high$
   2.1 $try := \lfloor \frac{low + high}{2} \rfloor$
   2.2 $Vectors := \{ \text{vec} | \text{height(vec,} \mathcal{VL}_{DT}) = try \}$
   2.3 $reach.k := \text{false}$
   2.4 while $Vectors \neq \emptyset$ and $reach.k \neq \text{true}$ do
      Select and remove a vector $vec$ from $Vectors$
      if satisfies$(vec.k, T_i, \text{MaxSup})$ then $sol := vec$; $reach.k := \text{true}$
   2.5 if $reach.k = \text{true}$ then $high := try$ else $low := try + 1$
3. Return $sol$

Fig. 10. An algorithm for computing a vector corresponding to a $k$-minimal generalization.

$low < high$ of the while statement will eventually evaluate false. Variable $sol$ is initialized to the top of the vector lattice, which, by the assumption of singleton maximal domains and $|PT| \geq k$ trivially satisfies $k$-anonymity with no suppression. The value of $sol$ is modified in Step 2.4 to be a vector corresponding to a generalized table $T_j$ which suppresses fewer tuples than $\text{MaxSup}$. Hence, the algorithm always returns a vector that satisfies $k$-anonymity by suppressing fewer tuples than $\text{MaxSup}$. Suppose then that the algorithm returns a vector $sol = DV_{i,j}$ corresponding to a generalization which is not $k$-minimal according to Definition 4.3. Since $DV_{i,j}$ does not correspond to a minimal solution, a vector $DV_{i,z} < DV_{i,j}$ exists corresponding to a generalized table that satisfies Conditions 1 and 2 of Definition 4.3. By Lemma 5.1, height

$$(DV_{i,z}, \mathcal{VL}_{DT}) < \text{height}(DV_{i,j}, \mathcal{VL}_{DT}).$$

Since the cycle terminated, condition $low < high$ evaluated false. high is the last (lowest) level, among those examined, at which a solution $sol$ has been found. When the algorithm terminates,

$low \geq high = \text{height}(sol = DV_{i,j}, \mathcal{VL}_{DT}) > \text{height}(DV_{i,z}, \mathcal{VL}_{DT}).$

Hence, $low$ must have been modified in some cycle of the search to a value $l + 1 > \text{height}(DV_{i,z}, \mathcal{VL}_{DT})$. Then, after the evaluation of such cycle $reach.k$ evaluated false meaning no vector was found at height $l$ corresponding to a generalization which satisfied $k$-anonymity by suppressing fewer tuples than $\text{MaxSup}$. Let $DV_{i,j}$ be a vector with height $l$ and such that $DV_{i,j} > DV_{i,z}$ (since each strategy is a path from the bottom to the top of the lattice such a vector always exists). Let $T_i$ ($T_j$ respectively) be the generalized table with vector $DV_{i,j}$ ($DV_{i,z}$ respectively) satisfying $k$-anonymity by enforcing minimal required suppression. Hence, $|T_i| - |T_j| > \text{MaxSup}$, which by Theorem 5.2 (see, also, Corollary 5.1) implies $|T_i| - |T_j| > \text{MaxSup}$. Hence, $T_z$ also suppresses more tuples than $\text{MaxSup}$ and cannot be a solution, which leads us to a contradiction.

Function satisfies can be executed computing the generalization and evaluating the size of the resulting clusters of tuples. We briefly sketch here an approach to execute satisfies without the need of such a computation. The approach makes use of the concept of distance vector between tuples. Let $T$ be a table and $x, y \in T$ be two tuples such that $x = (v'_1, \ldots, v'_n)$ and $y = (v''_1, \ldots, v''_n)$, where $v'_i$ and $v''_i$ are values in domain $D_i$ for $i = 1, \ldots, n$. The distance vector between $x$ and $y$ is the vector $V_{x,y} = [d_1, \ldots, d_n]$, where $d_i$ is the (equal) length of the two paths from $v'_i$ and $v''_i$ to their closest common ancestor in the value generalization hierarchy $\mathcal{VGH}_{DT}$ (or, in other words, the distance from the domain of $v'_i$ and $v''_i$ to the domain at which they generalize to the same value $v_i$). For instance, with reference to the $PT$ illustrated in Fig. 4 and the hierarchies in Fig. 3, the distance vector between (Asian, 94139) and (Black, 94139) is $[1,0]$, at which they both generalize to (person, 94139). Intuitively, the distance vector $V_{x,y}$ between two tuples $x$ and $y$ in table $T_i$ is the distance vector $DV_{i,j}$ between $T_i$ and the table $T_j$, with $T_i \preceq T_j$, where the domains of the attribute in $T_j$ are the most specific domains for which $x$ and $y$ generalize to the same tuple $t$. By looking at the distance vectors between the tuples in a table, we can determine whether a generalization at a given vector satisfies $k$-anonymity by suppressing fewer tuples than $\text{MaxSup}$ without computing the generalization. More precisely, we can determine, for each distance vector $DV$, the minimum required suppression for the $k$-anonymity constraint to be satisfied by the generalization corresponding to $DV$. The approach works as follows: Let $T_i = PT[Q_I]$ be the table to be considered. For each distinct tuple $x \in T_i$ determine count$(x, T_i)$, as the
number of occurrences of \( x \) in \( T \). Build a matrix \( VT \) with a row for each of the different outliers (i.e., tuples with less than \( k \) occurrences) and a column for each different tuple in the table. Entry \( VT[x, y] \) contains the distance vector tuples \( x \) and \( y \), that is, \( VT[x, y] = V_{x,y} \). (Note that the table is symmetric so only half on it actually needs to be computed.) Now, let \( vec \) be the distance vector of a generalization to consider as a potential solution. For each row \( x \), compute \( C_x \) as the sum of the occurrences \( count(y, T_i) \) of tuples \( y \) (column of the matrix) such that \( VT[x, y] \leq vec \). These are tuples that at generalization \( vec \) would generalize to the same tuple as \( x \), and the sum of their occurrences is the size of the resulting cluster. Determine then \( req_{sup} \) as the sum of the occurrences of all the outlier tuples \( x \) (row of the matrix) such that \( C_x \) so computed is smaller than \( k \), that is,

\[
req_{sup} = \sum_{x | C_x < k} count(x, T_i).
\]

Intuitively, \( req_{sup} \) is the number of tuples that would still be outliers in the generalization corresponding to distance vector \( vec \), and which would therefore need to be removed for the \( k \)-anonymity requirement to be satisfied. Hence, if \( req_{sup} \leq MaxSup \), the generalization with distance vector \( vec \) satisfies \( k \)-anonymity by suppressing less tuples than the threshold allowed. Otherwise, it does not.

Note that the information on the distance vectors between tuples in \( PT[QI] \) can also be exploited to restrict the set of vectors to be considered by algorithm \( \text{Find\_vector} \). It is easy to see, in fact, that all the nonnull vectors which are minimal among the vectors in a row \( x \) with \( count(x, PT) > MaxSup \) represent lower bounds for solutions not suppressing \( x \) and then no solution including tuple \( x \) can exist below the lowest height of such vectors. Moreover, the least upper bound of the vectors in the table represents an upper bound for the solution (any vector above corresponds to a solution that is non minimal). Also, the distance vector of any solution satisfying Definition 4.3 will have, for each attribute \( A_i \), a distance \( d_i \) that appears for \( A_i \) in some entries of the matrix. More precisely, if \( [d_1, \ldots, d_n] \) represents a potential solution, for some value of \( k \) and MaxSup, then for each \( i = 1, \ldots, n \) there must exist a distance vector \( [d'_1, \ldots, d'_n] \) between an outlier and some tuple in the table with \( d'_i = d_i \).

**Example 5.1.** Consider the table \( PT[QI] \) of Fig. 6, whose table of distance vectors is illustrated in Fig. 11, and assume \( MaxSup = 0 \). By looking at the last row of the table in Fig. 11, we note that any solution for \( PT \) which does not suppress tuple \( t_{11} \) will have to dominate either \([0,2,1,2,2]\) or \([1,3,0,1,1]\), and that no solution not suppressing \( t_{11} \) can be found below height 6. Also, the least upper bound of the vectors in the table is \([1,3,2,2,2]\). Hence, any solution will be dominated by it and no minimal solution can be found above level 10.

6 Preferences

In the previous section we illustrated an algorithm to retrieve a \( k \)-minimal generalization. It is clear from Section 4 that a table may have more than one minimal generalization satisfying a \( k \)-anonymity constraint for a suppression threshold. This is completely legitimate since the definition of “minimal” only captures the concept that the least amount of generalization and suppression necessary to achieve \( k \)-anonymity is enforced. However, multiple solutions may exist which satisfy this condition. The algorithm illustrated in the previous section returns a \( k \)-minimal generalization with the lowest height among all those existing. Although this may be considered a generally acceptable preference criteria, other preference criteria may exist depending on subjective measures and preferences of the data recipient. For instance, depending on the use of the released data, it may be preferable to generalize some attributes instead of others. We outline here some simple preference criteria that can be applied in choosing a preferred minimal generalization. To do that, we first introduce the notion of relative distance between attributes and absolute and relative distance between tables.

Let \( T_i(A_1, \ldots, A_n) \) be a table and \( T_j(A_1, \ldots, A_n) \) be one of its generalizations with distance vector \( DV_{i,j} = [d_1, \ldots, d_n] \). We refer to \( d_z \) as the absolute distance of attribute \( A_z \) in the two tables. The relative distance \( rd_z \) of each attribute \( A_z \) is obtained by dividing this absolute distance over the total height \( h_z \) of the domain generalization hierarchy of \( dom(A_z, T_i) \), i.e., \( rd_z = \frac{d_z}{h_z} \). Hence, we define the absolute (respectively) distance of \( T_j \) from \( T_i \), written \( \text{Absdist}_{i,j} \) (\( \text{Reldist}_{i,j} \) respectively), as the sum of the absolute (relative respectively) distance for each attribute. Formally, \( \text{Absdist}_{i,j} = \sum_{z=1}^{n} d_z \), and \( \text{Reldist}_{i,j} = \sum_{z=1}^{n} \frac{d_z}{h_z} \).
Given those distance measures, we can outline the following basic preference policies:

**Minimum absolute distance** prefers the generalization(s) that has 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) that has the smallest relative distance, that is, that minimizes the total number of relative steps. A step is made relative by dividing it over the height of the domain hierarchy to which it refers.

**Maximum distribution** prefers the generalization(s) that contains the greatest number of distinct tuples.

**Minimum suppression** prefers the generalization(s) that suppresses less tuples, i.e., has the greatest cardinality.

**Example 6.1.** Consider Example 4.2. Suppose MaxSup = 1. Minimal generalizations are \( GT_{[1,0]} \) and \( GT_{[0,2]} \). Under minimum absolute distance, \( GT_{[1,0]} \) is preferred. Under minimum relative distance, maximum distribution, and minimum suppression policies, the two generalizations are equally preferable. Suppose MaxSup = 2. Minimal generalizations are \( GT_{[1,0]} \) and \( GT_{[0,1]} \). Under the minimum absolute distance policy, the two generalizations are equally preferable. Under the minimum suppression policy, \( GT_{[1,0]} \) is preferred. Under the minimum relative distance and the maximum distribution policies, \( GT_{[0,1]} \) is preferred.

The list above is obviously not complete and additional preference policies can be thought of, which may be useful in certain releases. Other possible policies may include: preferring the generalization with the highest absolute relative distance peak; or that generalizes the minimum number of attributes (regardless of the number of steps); or the maximum number of attributes (thus distributing the generalization requirements over a large set). Preference criteria can also be specified in terms of any set of constraints on distance vectors or tables. For instance, explicit preferences can be specified stating that it is better to generalize certain attributes (because not useful in the specific data release) instead of others. The specific preference policy to use, of course, depends on the specific use for the released data. Examination of an exhaustive set of possible policies is outside the scope of this paper. The specific preference policy to be applied to a given data release may be specified by the requester at the time of access, by the data holder, or by both of them jointly.

It is easy to see that since the absolute distance of a domain tuple is exactly the height of the corresponding distance vector, the algorithm reported in Fig. 10 returns the solution at the minimal absolute distance. Application of specific preference policies, which cannot exploit properties of the hierarchy, may require the computation of all the \( k \)-minimal solutions to retrieve the one to be preferred. The algorithm for retrieving all the solutions can be obtained by modifying the one presented in Fig. 10 to work on vectors instead of heights (executing a binary search from the maximal to the minimal element of \( VL_{\mathcal{DT}} \) considering all the vectors at a given height and starting a new binary search for each of them as new high or low element depending on whether it evaluates true or false on satisfies).

Before closing this section, we observe that in a similar way, the values of \( k \) and MaxSup can also be subject to some discretionary preference criteria. In particular, \( k \) may depend, besides on the recipient’s linking ability, on negotiation between the involved parties, that is, data holder, recipient, and, possibly, respondents to whom information refer or laws regulating the protection of their privacy. Within a defined anonymity requirement \( k \), the suppression threshold can be specified as a preference expressed by the data recipient. Like preference criteria, the suppression threshold does not compromise anonymity and, therefore, recipients could require the application of the measure that best suits their needs with respect to the trade-offs between generalization and suppression.

Generalization and suppression, by providing protection, imply some loss on information being released. Generalization implies a loss of accuracy, as information released is less precise. Suppression implies a loss of completeness, as some information is not released. Data quality measures can then be applied evaluating a release with respect to loss of accuracy and completeness. For instance, accuracy could be measured as the ratio between the depth of the generalization over the total height of the generalization hierarchy of the corresponding domain tuple (or equivalently, 1 minus the relative height of the generalization). Completeness could be measured as the number of released tuples over the total number of tuples in the original table. Since, given a specified \( k \), the more the suppression allowed the less the generalization needed, the greater the accuracy the lower the completeness and vice versa. Different suppression thresholds could therefore be applied depending on data quality criteria set by the data recipients.

### 7 Related Work

The problem of protecting against the ability of data recipients to determine sensitive information from other information released to them has been considerably studied in the framework of statistical databases [1], [6]. However, most attention has been devoted to the protection of inference in aggregate statistics and tabular data in contrast to microdata. As a consequence, while a good set of methodologies exist for controlling macrodata release, “many decisions for the disclosure limitation of microdata are based only on precedents and judgement calls” [13]. Moreover, most approaches to protect the vulnerability of microdata from linking attacks use technique of data disturbance, which, although safeguarding specific statistical properties compromise the correctness, or truthfulness, of each specific piece of data (tuple in relational terms) [13]. Their use therefore limits the usefulness of the released data. The generalization and suppression techniques used in this paper preserve instead information truthfulness: The data released are always correct, although they may be less precise in case of generalization. Even suppression, although hiding some data, still preserves information truthfulness. In fact, the recipient is informed that some tuples and how many of them, have been suppressed and,
therefore, are not released. Such an information can then be taken into consideration by the recipient when processing the released data for producing statistics, in such a way that these statistics will not result distorted.

The use of the generalization and suppression as techniques to protect microdata release is not a novelty [13]. In particular, approaches like recoding (e.g., releasing the month and year of birth instead of the complete birth date), rounding (e.g., rounding incomes to the nearest one thousand dollars), and bottom- and top-coding (e.g., simply releasing the information that a value is smaller or higher than a given amount instead of releasing the specific value) can be represented in terms of generalization. Although generalization and suppression have already been proposed and investigated (often in the combination with other techniques), no formal framework has been provided for their use.

The work closest to ours is represented by two systems, called \( \mu \)-ARGUS [9] and Datafly [16], which have also investigated the specific use of generalization and suppression techniques to protect microdata release. The two systems, however, were completely lacking a formalization of the problem and its solutions, which allows us to reason about the correctness and quality of the resulting table. To our knowledge, this paper is the first paper providing that. In some sense therefore, our work and the cited systems represent different levels of looking at the problem and its solutions. We can however attempt some comparison with respect to how generalization and suppression techniques are enforced in these systems and in our model.

In both systems, generalization and suppression are used to derive, from a table to be released, a table where combinations of attribute values have, provided some allowed suppression, at least a minimum number (called \textit{binsize} in [9] and [16]) of occurrences. The way generalization and suppression techniques are enforced in these systems however presents some shortcomings.

In \( \mu \)-ARGUS the user specifies an overall binsize and specifies which attributes are sensitive (i.e., can constitute a quasi-identifier in our terms) by assigning a value between 0 and 3 to each attribute. 2- and 3-combinations across sensitive fields are then evaluated and combinations that have fewer occurrences than the specified binsize are subjected to generalization or suppression. The specified sensitivity values guide the choice of the 2, 3-combinations to be considered. The generalization/suppression process works by considering each sensitive attribute singularly and then 2- and 3-way combinations determined according to the attribute sensitivity. The responsibility of whether to generalize or suppress rests with the user and suppression is enforced at the cell level (for minimizing information loss). The approach proposed in \( \mu \)-ARGUS has several drawbacks. The major drawback consists in the fact that despite the binsize requirement, the resulting table may actually allow the recipient to link the information to fewer than binsize respondents. The main reason for this is that \( \mu \)-ARGUS only checks 2 and 3-way combinations and, therefore, characterizing combinations composed of more than three fields may not be detected. To illustrate, Fig. 12 reports an original table\(^6\) and a corresponding release from \( \mu \)-ARGUS, where \( k = 2 \), the quasi-identifier consists of all the attributes and the weighting of the attributes are as follows [17]: The \textit{SSN} attribute was tagged “most identifying;” the \textit{DOB}, \textit{Sex}, and \textit{ZIP} attributes were tagged “more identifying;” and the \textit{Race} attribute was tagged “identifying.”\(^7\) Looking at the resulting table: there is only one tuple with values \( \text{(white, 1964, male, 94138)} \). If this situation is true in the outside world (external table) as well, this individual could be uniquely recognized by the recipient. We can also notice the existence of only one tuple with attribute \textit{Sex} equal to \textit{female} and \textit{ZIP} code equal to 94139. Again, despite the suppression if this is true in the external data as well, also this individual can be reidentified by the recipient.

Datafly [16] also uses the notion of binsize as the minimum number of occurrences of values of an attribute (or combination of them). For each attribute in the table, the data holder specifies an anonymity level, which is a number between 0 and 1 used by Datafly to determine the binsize

\begin{table}
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
\textbf{SSN} & \textbf{Race} & \textbf{DOB} & \textbf{Sex} & \textbf{ZIP} \\
\hline
819181496 & black & 09/20/65 & male & 94141 \\
195925972 & black & 02/14/65 & male & 94141 \\
902750852 & black & 10/23/65 & female & 94138 \\
985820581 & black & 08/24/65 & female & 94138 \\
209559459 & black & 11/07/64 & female & 94138 \\
679392975 & black & 12/01/64 & female & 94138 \\
819419149 & white & 10/23/64 & male & 94138 \\
74201844 & white & 03/15/65 & female & 94139 \\
985302952 & white & 08/13/65 & male & 94139 \\
874593560 & white & 05/05/64 & male & 94139 \\
70382052 & white & 02/13/67 & male & 94138 \\
963963603 & white & 03/21/67 & male & 94138 \\
\hline
\end{tabular}
\caption{An example of application of \( \mu \)-ARGUS to produce an anonymized table [17].}
\end{table}

\(^6\) For the sake of simplicity, ZIP codes have been modified to refer to the values used in this paper.

\(^7\) We refer to [17] for details.
that the attribute must satisfy. Intuitively, 0 implies no requirement, 1 implies generalization to the maximum level, and the higher the anonymity value the higher the binsize. Each recipient is also assigned a profile composed of a value in the range \([0,1]\) for every attribute, describing the probability that the recipient could use the attribute for linking. The binsize to be actually enforced with reference to a specific data release is obtained by weighting the binsize previously specified with the recipient profile. The higher the value in the profile, the higher the binsize requirement. The specification of anonymity and user’s profiles allows flexibility in the anonymity requirement specification, although the actual requirement resulting from a specification may not always be clear. Moreover, control on combination of attributes is executed only for attributes for which a value of 1 is specified for the recipient, so we consider this case for comparison. Given a quasi-identifier and a binsize requirement \(k\) computed as discussed above, Datafly cycles on the following basic steps. If the number of outliers is smaller than a specified threshold of suppression, the outliers are removed and the process terminates. Otherwise, one step of generalization is performed on the attribute with the largest number of distinct values. This step is repeat until the tuples achieve the binsize requirement (i.e., \(k\)-anonymity) within the suppression threshold. With reference to our framework, Datafly therefore walks through a specific generalization strategy, determined at each step by looking at the occurrences of values in the different attributes, and then stopping at the local minimal solution. As we have already discussed such an approach does not guarantee minimality of the resulting solution. Datafly can therefore overgeneralize. To illustrate, consider table \(PT\), and the corresponding generalizations illustrated in Fig. 4. As stated in Example 3.5, for \(k = 3\) there are two minimal generalizations, namely, \(GT_{[1,0]}\) and \(GT_{[2]}\). We assume no suppression. Given \(PT\), Datafly first generalizes attribute \(ZIP\) because it has more distinct values (4 versus the 3 of \(Race\)), producing \(GT_{[0.01]}\). Because \(k\) is not reached Datafly, then generalizes \(Race\) which now has the greater number of distinct values. As a result, table \(GT_{[0.1]}\) is returned which, as already discussed is not minimal; there is no need to generalize both attributes in this case.

Other related work includes research addressing protection against inference attacks (e.g., [10], [11], [14]). Inference from linking, as addressed in our work, has similar concerns since both involve drawing conclusions from released data. However, we are more specifically concerned with the ability of the recipients to “link” released data to other data available to them.

8 Conclusions

We have addressed the problem of protecting privacy in information release and presented an approach to disclosing microdata such that the identities of the respondents cannot be recognized. The anonymity requirement is expressed by specifying a number \(k\) stating the required protection. Enforcing \(k\)-anonymity means ensuring that the information recipient will not be able, even when linking information to external data, to associate each released tuple with less than \(k\) individuals. We have illustrated how the \(k\)-anonymity requirement can be translated, through the concept of quasi-identifiers, in terms of a property on the released table. We have illustrated how \(k\)-anonymity can be enforced by using generalization and suppression techniques. We have introduced the concept of generalized table, minimal generalization, and minimal required suppression, capturing the property of a data release to enforce \(k\)-anonymity while generalizing and suppressing only what strictly necessary to satisfy the protection requirement. We have also illustrated an approach to computing such a generalization, and discussed possible preference policies to choose among different minimal generalizations. A prototype of the system is under implementation.

This work represents only a first step towards the definition of a complete framework for information disclosure control and opens space for future work. Future issues to be investigated include: the investigation of efficient algorithms to enforce the proposed techniques; the consideration of updates modifying the stored data; the consideration of multiple releases and consequent possibilities of collusions by multiple recipients or by the same recipients through multiple queries; the application of the techniques at the finer granularity level of cell; the investigation of additional techniques for providing \(k\)-anonymity; the development of a model for the specification of protection requirements of the data with respect to different possible classes of data recipients.

Acknowledgments

The work reported in this paper was performed while the author was with the Computer Science Laboratory of SRI International, Menlo Park, California, and was supported in part by DARPA/Rome Laboratory under grant F30602-96-C-0337, by the US National Science Foundation under grant ECS-94-22688. Preliminary versions of this work were done in collaboration with Latanya Sweeney, discussions with whom are acknowledged, who implemented the model and performed experimental results [15]. The author would like to thank Steven Dawson, Sabrina De Capitani di Vimercati, and Patrick Lincoln for comments that improved the presentation of the work.

References

SAMARATI: PROTECTING RESPONDENTS’ IDENTITIES IN MICRODATA RELEASE


Pierangela Samarati is a professor in the Information Technology Department at the University of Milan, Crema Italy. Her main research interests are in data and application security, information system security, access control models and systems, and information protection, in general. She has participated in several projects involving different aspects of information protection. On these topics, she has published more than 70 refereed technical papers in international journals and conferences. She is the coauthor of the book Database Security, Addison-Wesley, 1995. She has been a computer scientist at the Computer Science Laboratory at SRI, California. She has been a visiting researcher at the Computer Science Department of Stanford University, California, and at the ISSE Department of George Mason University, Virginia. She is currently serving as the coordinator of the Working Group on Security of the Italian Association for Information Processing, AICA (since 1993); as the Italian representative in the IFIP, International Federation for Information Processing, Technical Committee 11 on “EDP Security” (since 1993); as the chair of the IFIP Working Group 11.3 on Database and Applications Security (since 1998). She is a member of the steering committees of: the ACM-SIGSAC, Special Interest Group on Security, Audit, and Control, the ACM Conference on Information and Communications Security, (ACM-CCS), the International Conference on Information and Communications Security (ESORICS). She is a member of the IEEE Computer Society.

For more information on this or any computing topic, please visit our Digital Library at http://computer.org/publications/dlib.