Protecting Information Privacy in the Electronic Society

Sabrina De Capitani di Vimercati, Sara Foresti, and Pierangela Samarati

Dipartimento di Tecnologie dell'Informazione
Università degli Studi di Milano
Via Bramante 65 - 26013 Crema, Italy
firstname.lastname@unimi.it

Abstract. The privacy of users, the confidentiality of organizations, and the protection of huge collections of sensitive information, possibly related to data that might be released publicly or semi-publicly for various purposes, are essential requirements for the today's Electronic Society. In this chapter, we discuss the main privacy concerns that arise when releasing information to third parties. In particular, we focus on the data publication and data outsourcing scenarios, illustrating the emerging trends in terms of privacy and data protection and identifying some research directions to be investigated.

1 Introduction

The proliferation of information collected by organizations in their daily activities and made public and semi-public available for different purposes (e.g., to study trends, make statistical inference, or share knowledge among organizations) together with the access to inexpensive, fast computers with large storage capacities make it possible to draw damaging inferences about sensitive information. When sharing and disseminating data, it is crucial to guarantee that the privacy of the organizations or individuals to whom the data refer is properly preserved. Often sensitive data (e.g., medical or financial data) are protected by simply removing explicit identifiers (e.g., name, address, and phone number) from the released data in the, incorrect, belief that in this way data become anonymous. Such a de-identification process however provides no guarantee of anonymity. For instance, released information may contain other data (e.g., birth date and ZIP code) that in combination can be linked to publicly available information to re-identify the apparently anonymous data.

The same privacy problem arises when data are outsourced to external servers that become responsible for their management and for dissemination to other parties. In this case, data are no more under the direct control of their owners and therefore data privacy as well as data integrity may be put at risk. Moreover, the server storing and managing the data is often supposed to be honest-but-curious. A honest-but-curious server is relied upon for correctly managing the data and for guaranteeing their availability, but it is not trusted to access the data content.

Research and development communities as well as regulatory bodies and final users have been devoting much attention to the privacy and data protection problems, both in data publication and outsourcing scenarios. Many of the techniques investigated for ensuring proper privacy and data protection aim at limiting the possibility of identifying users and at protecting sensitive information about users. The goal of this chapter is to analyze emerging solutions as well as research directions for guaranteeing privacy and data protection in the data publication and outsourcing scenarios. The remainder of this chapter is organized as follows. Section 2 introduces the problem of protecting privacy in the publication of microdata and illustrates different concepts proposed in the literature to prevent the disclosure of sensitive information. Section 3 presents some techniques that provide privacy of data outsourced to external parties, focusing on query execution and access control enforcement. Finally, Section 4 concludes the chapter.

2 Privacy in data publication

In the past data were principally released in tabular form (*macrodata*) and through statistical databases. Today, however, there is a growing need of using and releasing specific stored data (*microdata*). While macrodata report the results of precomputed statistics, microdata contain specific information. The advantage of microdata with respect to macrodata is that they may be used to perform any kind of analysis. The disadvantage is that their release may put at risk the privacy of the *respondents* (i.e., individuals or companies) to whom the released microdata refer. In the following, we first describe the main motivation for which the privacy of the respondents can be at risk and then illustrate some possible countermeasures.

2.1 Identity disclosure

Microdata contain a set of attributes relating to single respondents in a sample or in a population. Microdata can be represented as tables composed of tuples with values from a set of attributes. The attributes in a microdata table can be usually partitioned into the following four classes.

- *Identifiers*. Attributes that uniquely identify respondents (e.g., `SSN` and `Name`).
- *Quasi-Identifier* (QI). Set of attributes that can be linked to external data sets (e.g., `Race`, `DoB`, and `Sex`).
- *Confidential attributes*. Attributes that contain sensitive information (e.g., `Illness`).
- *Non confidential attributes*. Any other attribute in the microdata table.

To guarantee the anonymity of the respondents (i.e., to prevent identity disclosure), the identifiers are usually removed (or encrypted) from the released

Patients

SSN	Name	Race	DoB	Sex	Illness
		white	*64/10/02*	*F*	stomach ulcer
		white	64/10/25	M	stomach ulcer
		white	65/08/04	M	arrhythmia
		white	65/08/12	F	gastritis
		asian	64/07/15	M	aids
		asian	65/02/20	M	aids
		asian	64/07/11	M	flu
		asian	64/07/09	F	flu
		asian	65/02/23	F	hypertension
		black	72/04/30	M	flu

Fig. 1. An example of de-identified microdata table

Voter List

Name	Marital Status	Address	City	Race	YoB	Sex
....	....	....	....	....	....	....
....	....	....	....	....	....	....
Helen White	Married	1600 Madison Street	New York	*white*	*64*	*F*
....	....	....	....	....	....	....

Fig. 2. An example of non de-identified public available table

microdata table. This de-identification process is however not sufficient to guarantee anonymity. As a matter of fact, quasi-identifiers can be exploited for linking the released data with other publicly available data sets, thus possibly re-identifying (or reducing the uncertainty about) respondents [1, 2]. In the following, our analysis is focused on quasi-identifiers and confidential attributes, since we assume that any microdata table is de-identified before its release and that non confidential attributes do not pose privacy concerns.

Example 1. Consider microdata table Patients in Figure 1 containing information on the patients of a hospital, which has been de-identified by removing attributes SSN and Name, and the publicly available Voter List in Figure 2. Attributes QI ={Race, DoB, Sex} can be linked to the tuples in the Voter List in Figure 2, thus revealing the Name, Marital Status, Address, and City of respondents whose information is stored in the microdata table. In table Patients, for example, there is only one tuple referring to a white female born in 1964 (the first tuple). If this combination is unique in the external word as well, it reveals that this tuple refers to "*Helen White*, *Married*, *1600 Madison Street*, *New York*", who suffers from *stomach ulcer*.

Different protection techniques have been proposed (e.g., sampling, swapping values, adding noise) for protecting released microdata from improper disclosure [3]. These techniques obfuscate the data while maintaining some overall statistical properties of the resulting table. There are however some situations where there is the need of preserving truthful information within each tuple, thus making techniques that perturb data not applicable. k-anonymity has been proposed as an approach to protect respondents' identities while releasing truthful information [2]. In the following, we first present a brief overview of k-anonymity together with its enforcement via generalization and suppression and then describe other approaches that extend k-anonymity.

2.2 *k*-Anonymity

The concept of k-anonymity is based on the observation that often a re-identification happens whenever the released microdata table contains *high visibility records* or when some combination of values for the quasi-identifier is unique or rare in the real world. In these cases, the respondent could be easily identified using other publicly available information. To counteract this problem, k-anonymity tries to capture, on the microdata table to be released, one of the main requirements that has been followed by the statistical community and by agencies releasing the data: *the released data should be indistinguishably related to no less than a certain number of respondents.* To satisfy this requirement, it is however necessary to know, for each possible data recipient (or adversary), the available data (quasi-identifiers) that could be exploited for linking attacks. Since it seems impossible, or highly impractical and limiting, to have this knowledge, in [2] the author assumes that the microdata table has a single quasi-identifier including all attributes in the table that can be externally available, and contains at most one tuple for each respondent. k-anonymity then requires that *each quasi-identifier value in the released table to have at least k occurrences*. Note that this requirement represents a sufficient, but not a necessary, condition to guarantee that every tuple in the microdata table is indistinguishably related to no fewer than k respondents. For instance, considering QI $=\{$`Race`, `DoB`, `Sex`$\}$, the microdata table in Figure 1 is 1-anonymous since there are single occurrences of values over the quasi-identifier (e.g., "white, 64/10/02, F").

Among the different techniques used for providing anonymity in the release of microdata, the k-anonymity proposal [2] uses *generalization* and *suppression*, which have the advantage of preserving the truthfulness of the information. Generalization consists in replacing the values of an attribute with more general values (e.g., the data of birth can be substituted with the year of birth). This technique is based on a *generalization hierarchy*, where the most general value is at the root of the hierarchy and the leaves correspond to the most specific values. Figure 3 illustrates three examples of generalization hierarchies for attributes `Sex`, `Race`, and `DoB`. The generalization hierarchy specified for `DoB` generalizes the date of birth first to the year and month of birth, then to the year of birth, and then to $*$. The generalization hierarchies specified for `Sex` and `Race` generalize the corresponding specific values to $*$ and *person*, respectively. A generalization process therefore proceeds by replacing the values represented by the leaf nodes with one of their ancestor nodes at a higher level. Different generalized microdata tables can be built, depending on the amount of generalization applied on the considered attribute. The final effect of a generalization is that tuples in the original microdata table with different values for the quasi-identifier are generalized to the same value, thus becoming indistinguishable.

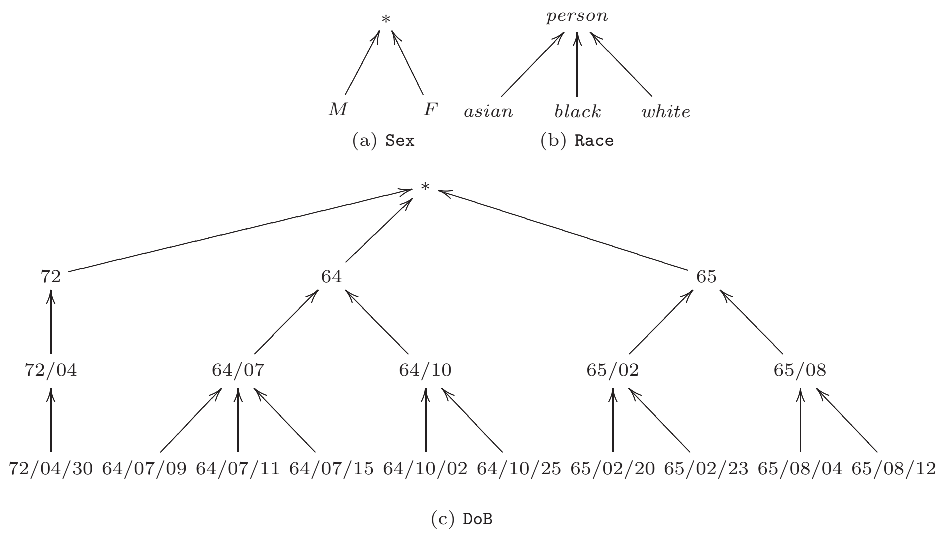

Fig. 3. Examples of generalization hierarchies

Suppression consists in removing information from the microdata table. The introduction of suppression can reduce the amount of generalization necessary to satisfy the k-anonymity constraint.

Generalization and suppression can be applied at different levels of granularity [4]. Generalization can be applied at the *attribute* level (i.e., a generalization step generalizes all the values of an attribute) or single *cells* (i.e., for a specific attribute, the table may contain values at different generalization levels). Suppression can be applied at the level of *tuple* (i.e., a suppression operation removes a whole tuple), *attribute* (i.e., a suppression operation obscures all the values of an attribute), or single *cells* (i.e., only the content of certain cells are removed). The possible combinations of the different choices for generalization and suppression (including also the choice of not applying one of the two techniques) result in different k-anonymity proposals and different algorithms for k-anonymity [4].

Example 2. Consider table PATIENTS in Figure 1 and suppose that $\mathsf{QI} = \{$`Race`, `DoB`, `Sex`$\}$. To guarantee 2-anonymity by applying only generalization at attribute level, it is necessary to generalize attribute `Race` to value *person* and attribute `DoB` to value $*$. Alternatively, since the last tuple in the table is an outlier, it is sufficient to perform one step of generalization on attribute `DoB` (i.e., the day of birth is removed), to generalize attribute `Sex` to $*$, and to suppress the last tuple to obtain the 2-anonymous table in Figure 4.

Since the application of generalization and/or suppression techniques cause an *information loss* (i.e., the anonymized table contains less information than

Race	DoB	Sex	Illness
white	64/10	*	stomach ulcer
white	64/10	*	stomach ulcer
white	65/08	*	arrhythmia
white	65/08	*	gastritis
asian	64/07	*	aids
asian	64/07	*	flu
asian	64/07	*	flu
asian	65/02	*	aids
asian	65/02	*	hypertension

Fig. 4. An example of a 2-anonymous table

the original table), the approaches proposed try to minimize such an information loss. The problem of computing an *optimal k-anonymous table* (i.e., a table that does not generalize or suppress more than it is needed to reach the threshold k) adopting attribute level generalization and tuple level suppression has been widely studied in the literature and has been proved to be computationally hard [5–7]. As a consequence, both exact [2, 8, 9] and heuristic [10] algorithms have been proposed for its solution. Note that the computational complexity of such exact algorithms is exponential in the number of attributes composing the quasi-identifier. Therefore, if the quasi-identifier contains a lot of attributes, the exact algorithms are not applicable in practice.

2.3 Extensions of k-anonymity

k-anonymity is an important concept that has started a new line of research, as testified by many research proposals developed after its introduction in [2]. The main goal of k-anonymity is however preventing identity disclosure and then it suffers from *attribute disclosure*, meaning that a quasi-identifier can be exploited for inferring sensitive information about respondents. Two possible attacks that may violate the privacy of the respondents are the *homogeneity attack* and the *background knowledge attack*.

A homogeneity attack [2, 11] happens when all the tuples in the released table with the same value for the quasi-identifier have also the same value for the sensitive attribute. If an adversary knows that a target respondent is represented in the released table and also knows her quasi-identifier, then she can infer the sensitive value associated with the target respondent. As an example, consider the 2-anonymous table in Figure 4, where $\mathsf{QI} = \{\texttt{Race}, \texttt{DoB}, \texttt{Sex}\}$ and `Illness` is the sensitive attribute. Suppose now that *Alice* knows that *Helen* is a white female born in October 1964 and that she is represented in the 2-anonymous table. *Alice* can infer that *Helen* suffers from *stomach ulcer*.

The background knowledge attack happens when an adversary can exploit her prior knowledge to reduce the uncertainty about the value of the sensitive attribute of the target respondent. With reference to the previous example, suppose that *Alice* knows that *Carol* is a white female born in August 1965 and that she is represented in the 2-anonymous table. *Alice* can infer that *Carol*

Race	DoB	Sex	Illness
white	*	F	stomach ulcer
white	*	F	gastritis
white	*	M	stomach ulcer
white	*	M	arrhythmia
asian	*	M	aids
asian	*	M	aids
asian	*	M	flu
asian	*	F	flu
asian	*	F	hypertension

Fig. 5. An example of an anonymized table satisfying 2-diversity

suffers from either *arrhythmia* or *gastritis*. Suppose now that *Alice* also knows that *Carol* swims for two hours every day (background knowledge). Since a person that suffers from *arrhythmia* cannot swim for two hours a day (background knowledge), *Alice* can infer with certainty that *Carol* suffers from *gastritis*.

To prevent these attacks, in [11] the authors introduce the concept of ℓ-*diversity* that can be formalized as follows. Given an anonymous table, a q-block (i.e., a set of tuples in the table characterized by the same value for the quasi-identifier) satisfies ℓ-diversity if there are at least ℓ *well represented* values for the sensitive attribute in the q-block, where well-represented can be defined in different ways (e.g., there must be at least ℓ distinct values for the sensitive attribute in the q-block). An anonymous table satisfies ℓ-diversity if all its q-blocks are ℓ-diverse.

Example 3. Consider table PATIENTS in Figure 1, with QI ={Race, DoB, Sex}. Figure 5 illustrates a 2-diverse table, obtained by generalizing the values of attribute DoB to $*$ and suppressing the last tuple in PATIENTS.

ℓ-diversity is however not immune to attacks. In [12] the authors show that ℓ-diverse tables are vulnerable to the *skewness attack* and *similarity attack*. In particular, a skewness attack happens when the distribution of the values for the sensitive attribute in a q-block is different from the distribution of the values for the attribute in the population (e.g., a value that is rare in the population is frequent in the q-block). If an attacker knows that a respondent is represented in the released table and that she belongs to a specific q-block, she can reduce her uncertainty about the value of the sensitive attribute for the target respondent. As an example, consider the 2-diverse table in Figure 5 and suppose that *Alice* knows that *John* is an asian male born in 1964 and that he is represented in the table. *Alice* can infer that *John* is one of the three respondents in the third q-block. Since the q-block has 2 out of 3 tuples suffering from *aids*, *Alice* can infer that *John* has 67% probability of suffering from *aids*, against the 1% of the considered population. *Similarity attack* happens when the values assumed by the sensitive attribute within a q-block are semantically similar. If an attacker knows that a respondent is represented in the table and that it belongs to a specific q-block, the privacy of the target respondent is put at risk. As an example, consider the 2-diverse table in Figure 5 and suppose that *Alice* knows that

Helen is a white female born in 1964 and that she is represented in the table. *Alice* can infer that *Helen* is one of the two respondents in the first q-block and then that she suffers from a stomach disease. In [12] the authors introduce the concept of t-*closeness* to counteract these attacks. A q-block satisfies t-closeness if the relative distance between the distribution of the values assumed by the sensitive attribute in the q-block and the distribution of the values assumed by the sensitive attribute is lower than t. A table satisfies t-closeness if all the q-blocks in the table satisfy t-closeness. The main problem is how to compute the distance between two probabilistic distributions. The authors propose the use of the Earth Mover's distance (EMD).

In addition to the concept of ℓ-diversity and t-closeness, other models have been also proposed (e.g., [13, 14]). Like for k-anonymity and its previously discussed variations, these models however do not take into consideration the *external knowledge* that an adversary may have and that can be used for inferring sensitive information about individuals with high confidence. On this problem, some enhanced variations of k-anonymity can be found in the literature (e.g., [15–18]). Typically these proposals address the problem of *positive inference* that happens whenever an adversary can infer that a respondent *has* a given sensitive value or a value within a restricted set (which is contrast to the *negative inference* that happens when an adversary can infer that a respondent *does not have* a given sensitive value). These proposals describe different modeling of the external knowledge and represent a first important attempt towards the development of efficacy and efficient anonymization solutions in the presence of external knowledge.

2.4 Open Issues

The problem of preserving the privacy of the respondents in data publication has been widely studied. However, there are still different open issues that need to be further investigated.

- *Protection against utility measures*. Research is needed to develop measures to allow users to assess, besides the protection offered by the data, the utility of the released data. Clearly, utility may be different depending on the data recipients and the use intended for the information. Approaches should be therefore devised that maximize information utility with respect to intended uses, while properly guaranteeing privacy.
- *Merging of different tables and views*. The original k-anonymity proposal as well as its variations are based on the assumption that there is a single table whose content has to be released and that the table contains at most one tuple for each respondent. Work is needed to release these two assumptions. In particular, the problem of releasing different tables providing anonymity needs to be investigated.
- *External knowledge*. k-anonymity do not model external knowledge that can be further exploited for inference and expose the data to identity or attribute disclosure. Although some proposals have started the analysis of this

problem, work is still needed to model external knowledge and take it into account in the process of computing the anonymized table.

– *Extensions and enrichment of the constraints and protection requirements.* k-anonymity and its variations capture only the defense against identity and attribute disclosure attacks. However, when we consider a data publication scenario, the protection requirements that may need to be enforced can be more sophisticated. For instance, we may need the possibility of specifying that given associations among data are sensitive. Along this line of research, some recent proposals have introduced the idea of using fragmentation for enforcing sensitive associations [19–21].

3 Data Outsourcing

The evolution of computer technology promises to offer inexpensive storage that allows the collection and distribution of huge amount of (possibly sensitive) information. Organizations have then to add data storage (and skilled administrative personnel) at a high rate to mange such a amount of information. An alternative solution to the problem, which is becoming increasingly popular as it saves costs and provides service benefits, it is represented by *data outsourcing*. Data are stored together with application front-ends at external servers who take full responsibility of their management. Typically, in such a scenario the external server is relied upon for ensuring high availability of the outsourced data while cannot always be trusted with respect to the confidentiality of data content (i.e., the server may be *honest-but-curious*). Since traditional access control techniques cannot prevent the server itself from making unauthorized accesses to the outsourced data, data are encrypted. In the following, we assume that the outsourced data are stored within a relational database and encryption is applied at the tuple level via a symmetric encryption function (the problems and solutions discussed however apply to any data model). The resulting scenario and the parties involved are illustrated in Figure 6: *data owner* is an organization (or a person) that outsources her data to make them available for controlled external release; *user* is a person that submits access requests; *client* is the user's front-end, which is in charge of translating access requests formulated by the user in equivalent requests operating on the outsourced data; *server* is the external party that stores and manages the data on behalf of the data owner.

Different issues need to be carefully investigated to allow the effective and widespread use of outsourcing functionalities in a secure and private way [22]. In this section, we focus on two of such issues: *query execution* and *access control enforcement*.

3.1 Query execution

Since the outsourced data are encrypted, the server cannot directly execute queries submitted by a user on them; the server can only return the whole encrypted data to the client, thus leaving to it the burden of executing queries. Solutions have been therefore proposed for enabling external servers to (partially)

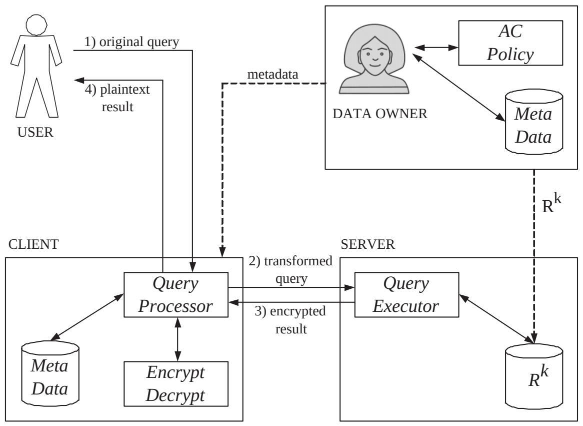

Fig. 6. Data outsourcing scenario

PATIENTS^k

<u>tid</u>	**etuple**	I_R	I_D	I_S	I_I
1	ksjie89sdd	β	ε	λ	μ
2	hh382npzs8	β	ε	κ	μ
3	8793ndsodn	β	δ	κ	η
4	md0324js90	β	δ	λ	θ
5	iweor04we9	α	γ	κ	η
6	jd93209eny	α	γ	κ	η
7	hi293jkfkd	α	γ	κ	θ
8	ndhw83nd81	α	γ	λ	θ
9	jsiw923j9s	α	γ	λ	μ
10	i39jxcisas	β	ε	κ	θ

Fig. 7. An example of an encrypted relation

execute queries [23–25]. These solutions are based on the definition of additional *indexing information* stored together with the encrypted data. Each relation r_i over relational schema $R_i(A_{i_1}, \ldots, A_{i_n})$ in the plaintext database is mapped onto an encrypted relation r_i^k over relational schema $R_i^k(\underline{tid}, etuple, I_{i_1}, \ldots, I_{i_n})$ in the encrypted database. Attribute *tid* is a numerical attribute added to the encrypted relation and acting as a primary key; *etuple* is the attribute containing the encrypted tuple; I_{i_j} is the index associated with the j-th attribute A_{i_j} in R_i.

Example 4. Consider relation PATIENTS in Figure 1. By assuming that only attributes `Race`, `DoB`, `Sex`, and `Illness` are involved in queries and therefore that there is the need to define an index for them, a possible encrypted relation is illustrated in Figure 7. Here, index values are conventionally represented with Greek letters. Note that the number of tuples in the encrypted relation is equal to the number of tuples in the corresponding plaintext relation.

Given a query Q defined on the original plaintext relational schema, the definition of indexes allows its partial execution at the server side, provided it is translated into an equivalent query operating on the encrypted data (see Figure 6). The query Q submitted by a user is then sent to the client (step 1), which translates Q into two queries: Q_s that operates on the encrypted database using indexes, and Q_c that operates on the result of Q_s. The client then passes query Q_s to the server (step 2), which evaluates the query and returns the result to the client (step 3). The client decrypts the relation obtained from the server, evaluates Q_c on the plaintext result of Q_s, and returns the result to the user (step 4). The process of transforming Q in Q_s and Q_c depends both on the indexing method adopted and on the kind of query Q. In the literature, different kinds of indexing techniques have been proposed (e.g., [24–29]). In [24], the authors propose a *bucket-based indexing*, where the domain of an attribute A is partitioned in a number of non-overlapping subsets of values containing contiguous values. Each partition is then associated with a unique value and the set of these values is the domain for index I associated with A. In [25] the authors propose a *hash-based index*, where index values are obtained by applying a hash function h to the plaintext values of the attribute on which the index is build. Both these two methods support equality queries (i.e., queries with conditions of the form $A = v$ or $A_i = A_j$) while do not easily support range queries (i.e., queries that retrieve all tuples where the value of an attribute is between a given range) since the index domain does not necessarily preserve the plaintext domain ordering. In [25] the authors present a B+-tree index that allows the evaluation of equality and range queries at the server side and of GROUP BY and ORDER BY SQL clauses. The idea is to outsource to the external sever an encrypted version of a B+-tree index build over a plaintext attribute of relational schema R. The encrypted B+-tree is iteratively queried by the client for retrieving the desired data. In [26] the authors present an order preserving encryption schema (OPES) to support equality and range queries as well as max, min, and count queries over encrypted data. In [27] the authors define an order preserving encryption with splitting and scaling (OPESS) schema, again supporting both equality and range queries. Other indexing techniques have been also proposed working, for example, on attributes with character/string domains [28] and supporting aggregation operators through privacy homomorphisms [29].

It is important to note that the definition of an index over an attribute must consider two conflicting requirements: on one hand, the indexing information should be related with the data well enough to provide for an effective query execution mechanism; on the other hand, the relationship between indexes and data should not open the door to inference and linking attacks. An evaluation of the inference exposure in encrypted databases enriched with indexing information has been presented in [30], where the authors show that if a limited number of indexes is provided, then an adversary cannot easily violate the confidentiality of the outsourced data.

3.2 Access Control Enforcement

Access control enforcement in the data outsourcing scenario cannot rely on traditional techniques (i.e., on the presence of a reference monitor), since the external

$$\begin{array}{c|cccccccccc|}
 & \mathbf{t_1} & \mathbf{t_2} & \mathbf{t_3} & \mathbf{t_4} & \mathbf{t_5} & \mathbf{t_6} & \mathbf{t_7} & \mathbf{t_8} & \mathbf{t_9} & \mathbf{t_{10}} \\
\hline
\mathbf{A} & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
\mathbf{B} & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\
\mathbf{C} & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
\mathbf{D} & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
\mathbf{E} & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
\mathbf{F} & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
\hline
\end{array}$$

Fig. 8. An example of access matrix

server is not trusted to enforce the policy defined by the data owner. Also, the data owner cannot directly enforce her policy, since this would require her involvement in all users' access requests for filtering the results according to the policy defined. A promising solution for this problem consists in integrating encryption and access control. The basic idea is to encrypt data using different keys; users can decrypt (and therefore access) all and only the data for which they know the encryption key. If access to the keys is limited to certain users of the system, different users are given different access privileges.

In [31] the authors illustrate an approach where an authorization policy defined by the data owner is translated into an *equivalent encryption policy*. The encryption policy allows all and only the accesses permitted by the corresponding authorization policy, defines the keys communicated to the users, the keys used for encrypting the resources, and a mechanism that the users can exploit to obtain the encryption key of the resources they are entitled to access. The authorization policy is represented through an access matrix $\mathcal{A}$ with a row for each user in the system and a column for each resource to be protected. Each cell $\mathcal{A}[u_i, r_j]$ may assume two values: 1, if u_i is allowed to access r_j; 0, otherwise. Given an access matrix $\mathcal{A}$ over sets $\mathcal{U}$ and $\mathcal{R}$ of users and resources, respectively, $acl(r_j)$ denotes the access control list of resource r_j (i.e., the set of users who can access r_j).

Example 5. Consider relation PATIENTS in Figure 1 and suppose that $\mathcal{U}$ $=\{A, \ldots, F\}$, and $\mathcal{R}$ includes all tuples of PATIENTS. Figure 8 illustrates an example of access matrix regulating accesses to the tuples of PATIENTS. According to this matrix, for example, $acl(t_1)=\{A,B,C,D\}$.

The solution proposed in [31] for generating an encryption policy is based on a *key derivation* method (e.g., [32–35]). A key derivation method allows the computation of a key k_i starting from the knowledge of another key k_j and a piece of publicly available information. This characteristic of key derivation methods can be used for creating an encryption policy such that only one key is released to each user and each resource is encrypted at most once. To this purpose, in [31] the authors define a *key derivation hierarchy* that exploits the hierarchy among sets of users induced by the partial order relationship based on set containment ($\subseteq$). Each vertex v in the hierarchy is associated with a key k and a public label l and represents a set of users. Each edge in the hierarchy between vertices

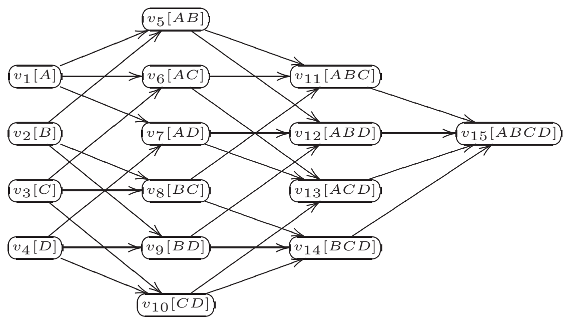

Fig. 9. An example of a key derivation hierarchy over $\{A, B, C, D\}$

v_i and v_j corresponds to a public token $t_{i,j}$ defined as $k_j \oplus h(k_i, l_j)$, where $\oplus$ is the bitwise *xor* operator, and h is a deterministic cryptographic function [33]. Token $t_{i,j}$ allows the computation of the k_j through the value of k_i and l_j (i.e., $k_j = t_{i,j} \oplus h(k_i, l_j)$). To correctly enforce the policy represented in $\mathcal{A}$, it is then sufficient to communicate to each $u \in \mathcal{U}$ the key of the vertex representing $\{u\}$ and to encrypt each resource r with the key of the vertex representing $acl(r)$. We note that users never share the same key, while resources with the same access control list are encrypted using the same encryption key.

Example 6. Consider the first four rows and the first four columns of the access matrix in Figure 8. Figure 9 illustrates the key derivation hierarchy induced by the set containment relationship defined on $\mathcal{U} = \{A, B, C, D\}$. We note that, for example, user A, who knows k_1, can derive all and only the keys k_i associated with vertices in the hierarchy representing a set of users including A (i.e., vertices $v_5, \ldots, v_7$, $v_{11}, \ldots, v_{13}$, and v_{15}). Therefore, the encryption policy in Figure 9 correctly enforces the considered access control policy.

It is easy to see that the key derivation hierarchy in Figure 9 implies the use of a great number of tokens, which in turn makes key derivation less efficient. In fact, the public tokens are stored at the server side to allow any user to access them when needed. Key derivation causes a series of interactions with the server and of search operations on the set of tokens. To limit the effort in key derivation, it is necessary to reduce the number of tokens by removing from the key derivation hierarchy the vertices and edges that are not necessary to enforce $\mathcal{A}$.

Although the problem of minimizing the number of edges (tokens) in a key derivation hierarchy is NP-hard, the authors in [31] proposes a heuristic algorithm that is based on two observations: *i)* the vertices needed for correctly enforcing an authorization policy are those representing the singleton sets of users and the acls of the resources to be protected; *ii)* when two or more vertices have more than two common direct ancestors, the insertion of a vertex representing the set of users corresponding to these ancestors reduce the total

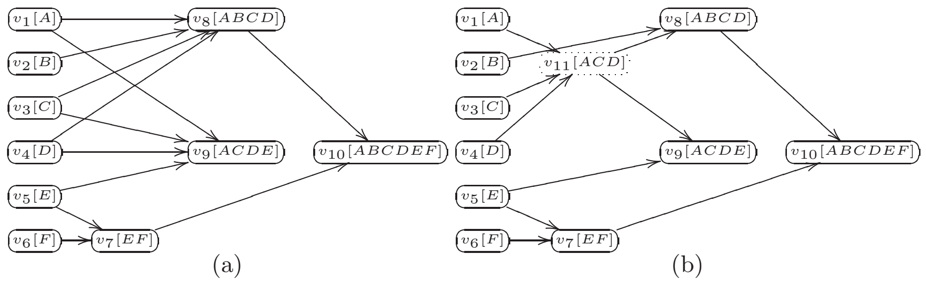

Fig. 10. Hierarchy with only the needed vertices (a) and minimized hierarchy (b)

number of tokens. The correct enforcement of the authorization policy is then guaranteed by properly connecting the vertices in the hierarchy, that is, for each vertex v_i representing $\{u\}$, there exists a path connecting v_i with all vertices v_j representing a set of users including u.

Example 7. Figure 10(a) illustrates a key derivation hierarchy that correctly enforces the authorization policy in Figure 8 and that contains only the vertices representing singleton sets of users and the acls of the tuples. Note however that vertices v_8 and v_9 have three common ancestors representing users A, C, and D. If we then insert vertex v_{11} representing $\{A, C, D\}$, the total number of tokens is reduced by one. The corresponding key derivation hierarchy is illustrated in Figure 10(b).

Since each resource r is encrypted with the key k associated with the vertex representing $acl(r)$, if the data owner updates the authorization policy, the resources involved possibly need to be re-encrypted to guarantee the correct enforcement of the new authorization policy. In fact, if $acl(r)$ is changed, r must be decrypted and re-encrypted with a key that all and only users in $acl(r)$ can derive. A re-encryption operation is however expensive from the data owner point of view, since she needs to download the encrypted version of r from the server, decrypt it, re-encrypt the resource with a key that only the set $acl(r)$ of users can derive, and send the new encrypted version of r to the server. Moreover, the key derivation hierarchy may need to be updated accordingly.

To limit the burden for the data owner for managing updates to the access control policy, in [31, 36] the authors propose a solution based on two layers of encryption. A *Base Encryption Layer* (BEL) is applied by the data owner before transmitting the resources to the server and consists in encrypting the resources according to the authorization policy existing at initialization time. A *Surface Encryption Layer* (SEL) is performed by the server over the resources already encrypted by the data owner. It enforces the dynamic changes over the policy. A user can then only access a resource if she knows or can derive the key used for encrypting the resources at both levels. The combination of the policies applied at BEL and SEL allows the outsourcing of the management of the authorization policy defined by the data owner.

3.3 Open Issues

We now outline some of the key aspects and challenges that have to be further investigated in the data outsourcing scenario.

- *Protecting the authorization policy.* The current approaches that use selective encryption for enforcing access control on the outsourced data are based on the definition of a key derivation hierarchy and consequently on a set of tokens that are stored at the server side. The public availability of tokens, and therefore of the corresponding key derivation hierarchy, makes visible the relationship between users and resources they are authorized to access, thus disclosing the authorization policy. There are however situations where the owners do not wish to publicly declare to whom they give (or not give) access to their resources. In these cases, the authorization policy should be kept confidential. In [37] the authors have presented a first proposal that addresses this issue. Alternative solutions however may be envisioned.
- *Write accesses.* Current proposals in the data outsourcing scenario are based on the assumption that write operations can be executed only by the data owner. Although this assumption is realistic in the data outsourcing scenario, there may exist other contexts where the consideration of read privileges only is limiting. This problem has been partially addressed in some proposals (e.g., [38–40]) that however focus on the problem on verifying the integrity of the outsourced resources. It would instead be interesting to extend current approaches for enforcing selective access to the consideration of write operations.
- *Depart from encryption.* The typical assumption underlying approaches in data outsourcing scenarios is that all the data are equally sensitive and therefore they have to be all encrypted. This assumption however may be an overdue. As a matter of fact, there are situations where data are not sensitive per se; what is sensitive is their association with other data (e.g., in a hospital the list of illnesses cured or the list of patients could be made publicly available, while the association of specific illnesses to individual patients must be protected). An interesting evolution would be therefore the development of new solutions where encryption should be applied only when explicitly demanded by the privacy requirements. Recently some proposals have put forward the idea of combining fragmentation and encryption [19–21] to reduce the use of encryption.
- *Multiple owners.* The current proposals enforcing access control in the data outsourcing scenario assume that the outsourced data belongs to a single data owner. However, a server may be responsible for managing and sharing data of different data owners that may also need to collaborate. An interesting open problem is then how to extend the current approaches to consider a multi-owners scenario.

4 Conclusions

Information is today one of the most valuable resources that organizations collect, share, and manage. Often these huge collections of information contain sensitive data whose protection is of paramount importance. Users and organizations are in fact not willing to release and share their information if they do not have the guarantee that their privacy will be preserved. In this chapter, we discussed privacy and data protection problems arising in data publication and data outsourcing scenarios, presented some emerging solutions, and identified some research challenges to be looked at.

Acknowledgements This work was supported in part by the EU within the 7FP project "PrimeLife" under grant agreement 216483.

References

1. Golle, P.: Revisiting the uniqueness of simple demographics in the us population. In: Proc. of the 5th Workshop on Privacy in the Electronic Society (WPES 2006), Alexandria, VA, USA (October 2006)
2. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Transactions on Knowledge and Data Engineering **13**(6) (November 2001) 1010–1027
3. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Samarati, P.: Microdata protection. In Yu, T., Jajodia, S., eds.: Security in Decentralized Data Management. Springer, Berlin Heidelberg (2007)
4. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Samarati, P.: k-anonymity. In Yu, T., Jajodia, S., eds.: Security in Decentralized Data Management. Springer, Berlin Heidelberg (2007)
5. Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., Zhu, A.: Anonymizing tables. In: Proc. of the 10th International Conference on Database Theory (ICDT 2005), Edinburgh, Scotland (January 2005)
6. Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., Zhu, A.: Approximation algorithms for k-anonymity. Journal of Privacy Technology (November 2005)
7. Meyerson, A., Williams, R.: On the complexity of optimal k-anonymity. In: Proc. of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2004), Paris, France (June 2004)
8. Bayardo, R., Agrawal, R.: Data privacy through optimal k-anonymization. In: Proc. of the 21st IEEE International Conference on Data Engineering (ICDE 2005), Tokyo, Japan (April 2005)
9. LeFevre, K., DeWitt, D., Ramakrishnan, R.: Incognito: Efficient full-domain k-anonymity. In: Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2005), Baltimore, MD, USA (June 2005)
10. Iyengar, V.: Transforming data to satisfy privacy constraints. In: Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD 2002), Alberta, Canada (July 2002)
11. Machanavajjhala, A., Gehrke, J., Kifer, D.: ℓ-density: Privacy beyond k-anonymity. In: Proc. of the 22nd IEEE International Conference on Data Engineering (ICDE 2006), Atlanta, GA, USA (April 2006)

12. Li, N., T, L., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and ℓ-diversity. In: Proc. of the 23rd IEEE International Conference on Data Engineering (ICDE 2007), Istanbul, Turkey (April 2007)
13. Truta, T., Vinay, B.: Privacy protection: p-sensitive k-anonymity property. In: Proc. of the 22nd International Conference on Data Engineering Workshop (ICDEW 2006), Atlanta, Georgia, USA (April 2006)
14. Wong, R., Li, J., Fu, A., Wang, K.: (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 (SIGKDD 2006), Philadelphia, PA, USA (August 2006)
15. Chen, B., Ramakrishnan, R., LeFevre, K.: Privacy skyline: Privacy with multidimensional adversarial knowledge. In: Proc. of the 33rd International Conference on Very Large Data Bases (VLDB 2007), Vienna, Austria (September 2007)
16. Martin, D., Kifer, D., Machanavajjhala, A., Gehrke, J., Halpern, J.: Worst-case background knowledge for privacy-preserving data publishing. In: Proc. of the 23rd IEEE International Conference on Data Engineering (ICDE 2007), Istanbul, Turkey (April 2007)
17. Chi-Wing, R., Fu, A., Wang, K., Pei, J.: Minimality attack in privacy preserving data publishing. In: Proc. of the 33rd International Conference on Very Large Data Bases (VLDB 2007), Vienna, Austria (September 2007)
18. Xiao, X., Tao, Y.: Personalized privacy preservation. In: Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2006), Chicago, Illinois, USA (June 2006)
19. Aggarwal, G., Bawa, M., Ganesan, P., Garcia-Molina, H., Kenthapadi, K., Motwani, R., Srivastava, U., Thomas, D., Xu, Y.: Two can keep a secret: a distributed architecture for secure database services. In: Proc. of the Second Biennial Conference on Innovative Data Systems Research (CIDR 2005), Asilomar, CA, USA (January 2005)
20. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Fragmentation and encryption to enforce privacy in data storage. In: Proc. of the 12th European Symposium On Research In Computer Security (ESORICS 2007), Dresden, Germany (September 2007)
21. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Keep a few: Outsourcing data while maintaining confidentiality. In: Proc. of the 14th European Symposium On Research In Computer Security (ESORICS 2009), Saint Malo, France (September 2009)
22. Samarati, P., De Capitani di Vimercati, S.: Data protection in outsourcing scenarios: Issues and directions. In: Proc. of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS 2010), Beijing, China (April 2010)
23. Hacigümüş, H., Iyer, B., Mehrotra, S.: Providing database as a service. In: Proc. of the 18th IEEE International Conference on Data Engineering (ICDE 2002), San Jose, CA (February 2002)
24. Hacigümüş, H., Iyer, B., Mehrotra, S., Li, C.: Executing SQL over encrypted data in the database-service-provider model. In: Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2002), Madison, WI, USA (June 2002)
25. Damiani, E., De Capitani di Vimercati, S., Jajodia, S., Paraboschi, S., Samarati, P.: Balancing confidentiality and efficiency in untrusted relational DBMSs. In: Proc. of the 10th ACM Conference on Computer and Communications Security (CCS 2003), Washington, VA, USA (October 2003)

26. Agrawal, R., Kierman, J., Srikant, R., Xu, Y.: Order preserving encryption for numeric data. In: Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2004), Paris, France (June 2004)
27. Wang, H., Lakshmanan, L.V.S.: Efficient secure query evaluation over encrypted XML databases. In: Proc. of 32nd International Conference on Very Large Data Bases (VLDB 2006), Seoul, Korea (September 2006)
28. Wang, Z., Dai, J., Wang, W., Shi, B.: Fast query over encrypted character data in database. Communications in Information and Systems **4**(4) (December 2004) 289–300
29. Hacigümüş, H., Iyer, B., Mehrotra, S.: Efficient execution of aggregation queries over encrypted relational databases. In: Proc. of the 9th International Conference on Database Systems for Advanced Applications (DASFAA 2004), Jeju Island, Korea (March 2004)
30. Ceselli, A., Damiani, E., De Capitani di Vimercati, S., Jajodia, S., Paraboschi, S., Samarati, P.: Modeling and assessing inference exposure in encrypted databases. ACM Transactions on Information and System Security (TISSEC) **8**(1) (February 2005) 119–152
31. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Encryption policies for regulating access to outsourced data. ACM Transactions on Database Systems (2010) (to appear).
32. Akl, S., Taylor, P.: Cryptographic solution to a problem of access control in a hierarchy. ACM Transactions on Computer Systems **1**(3) (August 1983) 239–248
33. Atallah, M., Frikken, K., Blanton, M.: Dynamic and efficient key management for access hierarchies. In: Proc. of the 12th ACM Conference on Computer and Communications Security (CCS 2005), Alexandria, VA, USA (November 2005)
34. Crampton, J., Martin, K., Wild, P.: On key assignment for hierarchical access control. In: Proc. of IEEE Computer Security Foundation Workshop (CSFW 2006), Venice, Italy (July 2006)
35. Sandhu, R.: Cryptographic implementation of a tree hierarchy for access control. Information Processing Letters **27**(2) (February 1988) 95–98
36. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Over-encryption: Management of access control evolution on outsourced data. In: Proc. of the 33rd International Conference on Very Large Data Bases (VLDB 2007), Vienna, Austria (September 2007)
37. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Pelosi, G., Samarati, P.: Preserving confidentiality of security policies in data outsourcing. In: Proc. of the 7th Workshop on Privacy in the Electronic Society (WPES 2008), Alexandria, VA (October 2008)
38. Hacigümüş, H., Iyer, B., Mehrotra, S.: Ensuring integrity of encrypted databases in database as a service model. In: Proc. of the 17th IFIP WG11.3 Working Conference on Data and Application Security, Estes Park, CO, USA (August 2003)
39. Mykletun, E., Narasimha, M., Tsudik, G.: Authentication and integrity in outsourced databases. ACM Transactions on Storage **2**(2) (May 2006) 107–138
40. Narasimha, M., Tsudik, G.: DSAC: integrity for outsourced databases with signature aggregation and chaining. In: Proc. of the ACM 14th Conference on Information and Knowledge Management (CIKM 2005), Bremen, Germany (October-November 2005)