Loose associations to increase utility in data publishing [1]

Sabrina De Capitani di Vimercati [a,*], Sara Foresti [a], Sushil Jajodia [b], Giovanni Livraga [a], Stefano Paraboschi [c], and Pierangela Samarati [a]

[a] *Dipartimento di Informatica, Università degli Studi di Milano - 26013 Crema, Italy*
E-mail: firstname.lastname@*unimi.it*
[b] *Center for Secure Information Systems, George Mason University - 22030-4422 Fairfax, VA, USA*
E-mail: jajodia@gmu.edu
[c] *Dipartimento di Ingegneria, Università degli Studi di Bergamo - 24044 Dalmine, Italy*
E-mail: parabosc@unibg.it

Abstract. Data fragmentation has been proposed as a solution for protecting the confidentiality of sensitive associations when releasing data for publishing or external storage. To enrich the utility of data fragments, a recent approach has put forward the idea of complementing a pair of fragments with some (non precise, hence *loose*) information on the association between them. Starting from the observation that in presence of multiple fragments the publication of several independent associations between pairs of fragments can cause improper leakage of sensitive information, in this paper we extend loose associations to operate over an arbitrary number of fragments.

We first illustrate how the publication of multiple loose associations between different pairs of fragments can potentially expose sensitive associations, and describe an approach for defining loose associations among an arbitrary set of fragments. We investigate how tuples in fragments can be grouped for producing loose associations so to increase the utility of queries executed over fragments. We then provide a heuristics for performing such a grouping and producing loose associations satisfying a given level of protection for sensitive associations, while achieving utility for queries over different fragments. We also illustrate the result of an extensive experimental effort over both synthetic and real datasets, which shows the efficiency and the enhanced utility provided by our proposal.

Keywords: Loose associations, fragmentation, confidentiality constraints, privacy, data publishing

1. Introduction

The amount of information being published, shared, and/or externally stored or processed in today's society is growing at an incredible pace everyday. Together with the praise for the benefits and convenience of this scenario, comes however an ever increasing worry about the need to guarantee confidential-

[1] A preliminary version of this paper appeared under the title "Extending loose associations to multiple fragments," in *Proc. of the 27th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy* (DBSec 2013), Newark, NJ, USA, July 2013 [12].

[*] Corresponding author. E-mail: sabrina.decapitani@unimi.it.

ity of sensitive information. The problem of protecting confidentiality of sensitive information in these contexts has been receiving much attention by the research and development communities (as well as end users and individuals) and many approaches have been proposed (e.g., [19,25]). An important aspect to be taken into consideration in the development and application of protection techniques for ensuring confidentiality of sensitive information is the need to maintain utility in the data, avoiding their over-protection, where utility encompasses both the availability of certain information as well as the ability to perform queries over the data. While confidentiality can be provided by wrapping data with an encryption layer, query evaluation over encrypted data requires to adopt either indexing techniques (e.g., [5]), or specific encryption approaches such as homomorphic encryption (e.g., [18]), which however provide only limited capabilities for querying (as they support evaluation of only specific conditions). The need to maintain data in the clear to provide better support for queries has been, for example, one of the key observations in the design of solutions relying on *fragmentation* for protecting sensitive associations in data outsourcing (e.g. [1,7,13]). Fragmentation protects sensitive associations among data by splitting them in different fragments (vertical data views) that are not linkable one to the other. The advantage of fragmentation over data encryption is in fact the ability to query actual data (in contrast to indexes used when data are completely encrypted) and therefore to provide more convenience in terms of data accessibility and query performance. Fragmentation represents also a useful paradigm to enforce protection requirements and to produce different views over data that can be publicly released without the risk of disclosing sensitive information.

Along the same line of enhancing utility, *loose associations* have been recently proposed as a complement to fragmentation [12,14]. In fact, providing complete protection to sensitive associations can, in many cases, be considered an overdo and smaller protection guarantees (meaning uncertainty over the associations among data) are considered acceptable. Loose associations complement then data fragmentation by providing some information on the association among tuples in different fragments. Intuitively, being fragments unlinkable without loose associations, a tuple in a fragment could have, as its corresponding tuple, any tuple in another fragment. Loose associations provide information on the relationships between tuples of different fragments at the granularity of groups of tuples (in contrast to individual tuples) thus maintaining some degree of protection over the association. The original definition of loose associations operates assuming that a fragmentation includes two fragments only and a single loose association is defined between this pair of fragments. A fragmentation may however include an arbitrary number of fragments, and therefore multiple loose associations might need to be defined. The presence of multiple loose associations may unfortunately open the door to privacy breaches. In fact, while the associations released in loose form are protected, the publication of multiple loose associations could indirectly expose other sensitive associations (i.e., a recipient could be able to reconstruct them).

In this paper, we present a general approach to define loose associations that operate on an arbitrary number of fragments. With our approach, the data owner can specify multiple associations among different pairs of fragments with the assurance that their combination cannot introduce leakages. Also, it allows specifying loose associations involving more than two fragments. Our approach is based on the definition of a universal loose association, encompassing all the fragments and all the confidentiality constraints. Any projection of this universal loose association (including the complete association itself) provides a loose association involving a different subset of fragments and is guaranteed to maintain the aimed degree of protection to the sensitive associations. In [12] we presented an early version of our proposal that here is extended by introducing an approach that takes into account queries to be executed so to build loose associations that enhance utility for them. We then provide a heuristic algorithm for the computation of a loose association, and present the results of an extensive experimental analysis over

synthetic and real datasets aimed at evaluating the efficiency, efficacy, and scalability of our algorithm, as well as the utility of the computed loose association.

The remainder of this paper is organized as follows. Section 2 introduces the basic concepts on which our approach builds. Section 3 illustrates the privacy risks caused by the release of multiple loose associations. Section 4 presents our definition of loose association, taking into account an arbitrary number of fragments. This section also introduces the properties that need to be guaranteed to ensure that a loose association satisfies a given privacy degree, and provides some observations on loose associations. Section 5 discusses the utility of loose associations in terms of providing better response to queries. Section 6 illustrates a heuristic algorithm for the computation of a loose association. Section 7 presents our experimental analysis, on synthetic as well as on real datasets, showing the efficiency of our approach and the utility provided in query execution. Section 8 discusses related work. Finally, Section 9 concludes the paper.

2. Basic concepts

We consider a scenario where a data owner wishes to release her data for publication or external storage. Data, represented for convenience as a single relation s over relational schema $S(a_1, \ldots, a_m)$, are subject to *confidentiality constraints* stating that certain information (individual attributes or associations among them) is to be considered sensitive and should therefore not be disclosed. A confidentiality constraint is formally defined as follows [1,7].

Definition 2.1 (Confidentiality constraint) *Given a relation schema $S(a_1, \ldots, a_m)$, a* confidentiality constraint *c over S is a subset of the attributes $\{a_1, \ldots, a_m\}$ in S.*

Confidentiality constraints are enforced before release by avoiding disclosure of sensitive attributes (singleton constraints), and fragmenting the relation into vertical views so to break sensitive attribute associations (non-singleton constraints). Note that non-singleton constraints can also be enforced by non releasing a subset of the attributes in the constraint. At the schema level, fragmentation splits then S into a set $\mathcal{F} = \{F_1, \ldots, F_n\}$ of fragments. Each F_i corresponds, at the instance level, to the vertical view f_i obtained projecting s over F_i. Given a fragmentation $\mathcal{F} = \{F_1, \ldots, F_n\}$, sensitive information modeled by a set $\mathcal{C}$ of confidentiality constraints is protected by ensuring that: *i)* no individual fragment $F \in \mathcal{F}$ contains all the attributes involved in a confidentiality constraint (i.e., $\forall F \in \mathcal{F}$, $\forall c \in \mathcal{C}$: $c \not\subseteq F$); and *ii)* fragments are disjoint (i.e., $\forall F_i, F_j \in \mathcal{F}$, $i \neq j : F_i \cap F_j = \emptyset$). We assume data to be fragmented no more than necessary to satisfy the constraints, and therefore fragmentations to be *minimal*, that is, merging any two fragments would violate at least one constraint. Figure 1 illustrates an example of relation to be released, of confidentiality constraints over it, and of a fragmentation satisfying the constraints. Note that fragments could cover either all the attributes that are not sensitive by themselves (i.e., not appearing in singleton confidentiality constraints) or only a subset of them, as in Figure 1(c). In this paper, we do not impose any requirement on this, as our model is independent from such an assumption.

Fragmentation completely breaks the associations among attributes appearing in different fragments. In fact, since attributes are assumed to be independent,[1] any tuple appearing in a fragment could have,

[1] We maintain such an assumption of the original proposal to not complicate the treatment with aspects not related to loose associations. Dependencies among attributes can be taken into consideration simply by extending the requirement of unlinkability among fragments to include the consideration of such dependencies (for more details, see [13]).

PATIENTS

	Name	YoB	Edu	ZIP	Job	MarStatus	Disease	Race	InsCompany	Salary	InsAmount
t_1	Alice	1974	B.Sc	90015	Assistant	Married	Flu	Black	BestCompany	1000	150
t_2	Bob	1965	MBA	90038	Manager	Widow	Diabetes	White	BestCompany	5000	70
t_3	Carol	1976	Ph.D	90001	Manager	Married	Calculi	Black	MyCompany	2000	100
t_4	David	1972	M.Sc	90087	Doctor	Divorced	Asthma	Asian	HerCompany	4000	150
t_5	Greg	1975	M.Sc	90025	Doctor	Single	Flu	Indian	MyCompany	1000	70
t_6	Hal	1970	Th.D	90007	Clerk	Single	Calculi	Indian	BestCompany	2000	120
t_7	Eric	1960	Primary	90025	Chef	Divorced	Diabetes	White	YourCompany	4000	110
t_8	Fred	1974	Ed.D	90060	Teacher	Widow	Asthma	Asian	YourCompany	5000	60

(a)

$\mathcal{C}$

$c_1 = \{\texttt{YoB}, \texttt{Edu}\}$
$c_2 = \{\texttt{ZIP}, \texttt{Job}\}$
$c_3 = \{\texttt{Name}, \texttt{Disease}\}$
$c_4 = \{\texttt{YoB}, \texttt{ZIP}, \texttt{Disease}\}$
$c_5 = \{\texttt{YoB}, \texttt{ZIP}, \texttt{MarStatus}\}$

(b)

F_l

	Name	YoB
l_1	Alice	1974
l_2	Bob	1965
l_3	Carol	1976
l_4	David	1972
l_5	Greg	1975
l_6	Hal	1970
l_7	Eric	1960
l_8	Fred	1974

F_m

Edu	ZIP	
B.Sc	90015	m_1
MBA	90038	m_2
Ph.D	90001	m_3
M.Sc	90087	m_4
M.Sc	90025	m_5
Th.D	90007	m_6
Primary	90025	m_7
Ed.D	90060	m_8

(c)

Fig. 1. An example of relation (a), a set $\mathcal{C}$ of confidentiality constraints over it (b), and a fragmentation that satisfies the confidentiality constraints in $\mathcal{C}$ (c)

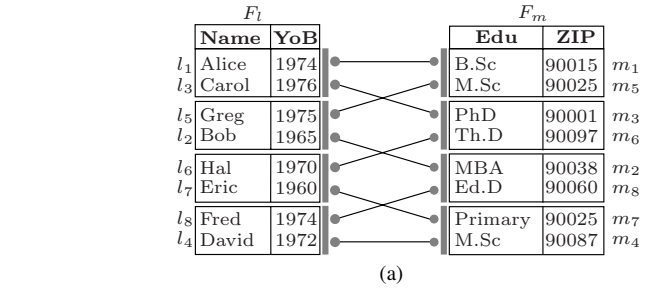
as its corresponding part, any other tuple appearing in another fragment. In some cases, such protection can be an overkill and a lower uncertainty on the association could instead be preferred, to mitigate information loss. A way to achieve this consists in publishing an association among tuples in fragments at the level of groups of tuples (in contrast to individual tuples), where the cardinality of the groups impacts the uncertainty over the association, which therefore remains *loose*. Hence, group associations are based on grouping of tuples in fragments, as follows.

Definition 2.2 (k-Grouping) *Given a fragment F_i, its instance f_i, and a set GID_i of group identifiers, a* k-grouping *function over f_i is a surjective function $\mathcal{G}_i$:$f_i \rightarrow \mathsf{GID}_i$ such that $\forall g_i \in \mathsf{GID}_i : |\mathcal{G}_i^{-1}(g_i)| \geq k$.*

A group association is the association among groups, induced by the grouping enforced in fragments. Looseness is defined with respect to a degree k of protection corresponding to the uncertainty of the association among tuples in groups within the fragments (or, more correctly, among values of attributes involved in confidentiality constraints whose attributes appear in the fragments). Group associations have been introduced in [14] and defined over a pair of fragments. Given two fragment instances, f_l and f_m, and a (k_l,k_m)-grouping over them (meaning a k_l-grouping over f_l and a k_m-grouping over f_m) group association A_{lm} contains a pair ($\mathcal{G}_l(t[F_l])$,$\mathcal{G}_m(t[F_m])$), for each tuple $t \in s$.

Figure 2(a) illustrates a (2,2)-grouping over fragments f_l and f_m in Figure 1(c), and the induced group association over them. The group association, graphically represented by the edges among the rectangles corresponding to groups of tuples in Figure 2(a), is released as a table containing the pairs of group identifiers in A_{lm} and by complementing fragments with a column reporting the identifier of the group to which each tuple belongs (Figure 2(b)). In the following, for simplicity, given a tuple t in the original relation, we denote with l (m, respectively) tuple $t[F_l]$ ($t[F_m]$, respectively) in fragment f_l (f_m, respectively).

The degree of looseness guaranteed by a group association depends on the uncertainty given not only by the cardinality of the groups, but also by the association among attribute values for those attributes



(a)

F_l

	Name	YoB	G_l
l_1	Alice	1974	ny1
l_2	Bob	1965	ny2
l_3	Carol	1976	ny1
l_4	David	1972	ny4
l_5	Greg	1975	ny2
l_6	Hal	1970	ny3
l_7	Eric	1960	ny3
l_8	Fred	1974	ny4

A_{lm}

G_l	G_m
ny1	ez1
ny1	ez2
ny2	ez1
ny2	ez3
ny3	ez2
ny3	ez4
ny4	ez3
ny4	ez4

F_m

G_m	Edu	ZIP	
ez1	B.Sc	90015	m_1
ez3	MBA	90038	m_2
ez2	Ph.D	90001	m_3
ez4	M.Sc	90087	m_4
ez1	M.Sc	90025	m_5
ez2	Th.D	90007	m_6
ez4	Primary	90025	m_7
ez3	Ed.D	90060	m_8

(b)

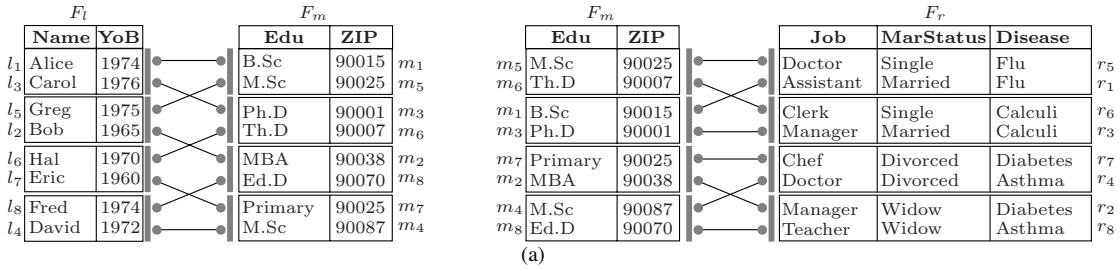
Fig. 2. Graphical representation (a) and corresponding relations (b) of a 4-loose association between fragments F_l and F_m in Figure 1(c)

appearing together in a confidentiality constraint c that is covered by the fragments (i.e., $c \subseteq F_l \cup F_m$). For instance, a looseness of $k = 4$ for the association in Figure 2 ensures that for each value of $t[F_l \cap c]$ there are at least $k = 4$ different values for $t[F_m \cap c]$, for each confidentiality constraint c covered by F_l and F_m. If the (k_l,k_m)-grouping satisfies the heterogeneity properties given in [14], the association among the fragments is ensured to be k-loose with $k = k_l \cdot k_m$. These heterogeneity properties demand diversity in the definition of the group association (i.e., no two groups can be associated more than once), as well as in the values of attributes that appear in a confidentiality constraint for those tuples that belong to the same group (in either F_l or F_m), or to groups in F_l (F_m, respectively) that are associated with the same group in F_m (F_l, respectively).

Note that the release of a group association between F_l and F_m may only put at risk constraints whose attributes are all contained in the fragments (i.e., all confidentiality constraints c such that $c \subseteq F_l \cup F_m$). For instance, the association in Figure 2 may put at risk only the sensitive association modeled by confidentiality constraint c_1={YoB, Edu}, and satisfies k-looseness for $k = 4$ (and therefore also for any lower k). In fact, any value of YoB is associated with at least (exactly, in this case) four different values of Edu and viceversa.

3. Problem statement

The proposal in [14] supports group associations between pairs of fragments. Given a generic fragmentation $\mathcal{F}$ composed of an arbitrary number of fragments, different group associations can be published on different fragments pairs. A simple example shows how such a publication, while guaranteeing protection of the specific associations released in loose form, can however expose other sensitive associations.



(a)

F_l

	Name	YoB	G_l
l_1	Alice	1974	ny1
l_2	Bob	1965	ny2
l_3	Carol	1976	ny1
l_4	David	1972	ny4
l_5	Greg	1975	ny2
l_6	Hal	1970	ny3
l_7	Eric	1960	ny3
l_8	Fred	1974	ny4

F_m

	G_{m1}	Edu	ZIP	G_{m2}
m_1	ez11	B.Sc	90015	ez22
m_2	ez13	MBA	90038	ez23
m_3	ez12	Ph.D	90001	ez22
m_4	ez14	M.Sc	90087	ez24
m_5	ez11	M.Sc	90025	ez21
m_6	ez12	Th.D	90007	ez21
m_7	ez14	Primary	90025	ez23
m_8	ez13	Ed.D	90060	ez24

F_r

	G_r	Job	MarStatus	Disease
r_1	jmd1	Assistant	Married	Flu
r_2	jmd4	Manager	Widow	Diabetes
r_3	jmd2	Manager	Married	Calculi
r_4	jmd3	Doctor	Divorced	Asthma
r_5	jmd1	Doctor	Single	Flu
r_6	jmd2	Clerk	Single	Calculi
r_7	jmd3	Chef	Divorced	Diabetes
r_8	jmd4	Teacher	Widow	Asthma

A_{lm}

G_l	G_{m1}
ny1	ez11
ny1	ez12
ny2	ez11
ny2	ez13
ny3	ez12
ny3	ez14
ny4	ez13
ny4	ez14

A_{mr}

G_{m2}	G_r
ez21	jmd1
ez21	jmd2
ez22	jmd1
ez22	jmd2
ez23	jmd3
ez23	jmd4
ez24	jmd3
ez24	jmd4

(b)

Fig. 3. Graphical representation (a) and corresponding relations (b) of a 4-loose association A_{lm} between F_l and F_m, and a 4-loose association A_{mr} between F_m and F_r, with F_l, F_m, and F_r three fragments of relation PATIENTS in Figure 1(a)

Consider the example in Figure 3, where relation PATIENTS in Figure 1(a) has been split into three fragments $F_l = \{\texttt{Name}, \texttt{YoB}\}$, $F_m = \{\texttt{Edu}, \texttt{ZIP}\}$, and $F_r = \{\texttt{Job}, \texttt{MarStatus}, \texttt{Disease}\}$. The (2,2)-grouping over F_l and F_m, and the (2,2)-grouping over F_m and F_r induce two 4-loose group associations: A_{lm} between F_l and F_m, and A_{mr} between F_m and F_r, respectively. The looseness of A_{lm} guarantees protection with respect to constraint $c_1 = \{\texttt{YoB}, \texttt{Edu}\}$, covered by F_l and F_m, ensuring that for each value of YoB in F_l the group association provides at least four possible values of Edu in F_m (and viceversa). The looseness of A_{mr} guarantees protection with respect to constraint $c_2 = \{\texttt{ZIP}, \texttt{Job}\}$, covered by F_m and F_r, ensuring that for each value of ZIP in F_m the group association provides at least four possible values of Job in F_r (and viceversa). This independent definition of the two associations does not take into consideration constraints expressing sensitive associations among attributes that are not covered by the pairs of fragments on which a group association is specified (c_3, c_4, and c_5 in our example), which can then be exposed. Consider, for example, constraint $c_3 = \{\texttt{Name}, \texttt{Disease}\}$. Alice (tuple l_1 in f_l) is mapped to group ny1 which is associated by A_{lm} with groups ez11 and ez12, that is, tuples m_1, m_3, m_5, and m_6 in f_m. These tuples are also grouped as ez22 and ez21, associated by A_{mr} with groups jmd1 and jmd2, that is, tuples r_1, r_3, r_5, and r_6 in f_r. Hence, by combining the information of the two associations, we know that l_1 in f_l is associated with one of these four tuples in f_r. While an

uncertainty of four is guaranteed with respect to the association among tuples, such an uncertainty is not guaranteed at the level of values, which could then expose sensitive associations. In particular, since the disease in both r_1 and r_5 is Flu and the disease in both r_3 and r_6 is Calculi, there are only two possible diseases associated with Alice, each of which has 50% probability of being the real one.

This simple example shows how group associations between pairs of fragments, while guaranteeing protection of the associations between the attributes in each pair of fragments, could indirectly expose other associations, which are not being released in loose form. To counteract this problem, group associations should be specified in a concerted form. In the next section, we extend and redefine group associations and the related properties to guarantee a given looseness degree to be enforced over an arbitrary number of associations and fragments.

4. Loose associations

Our approach to ensure that the publication of different associations does not cause improper leakage is based on the definition of a single loose association encompassing all the fragments on which the data owner wishes to specify associations so to take into account *all* the confidentiality constraints. Any projection over this "universal" group association will then produce different group associations, over any arbitrary number of fragments, which are not exposed to linking attacks such as the one illustrated in the previous section.

4.1. k-Looseness

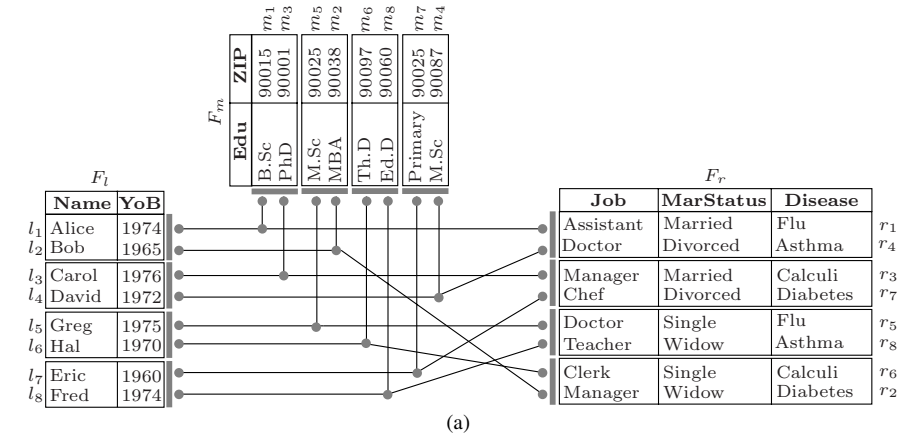
We start by identifying the constraints that are potentially exposed by the release of group associations involving a set $\mathcal{T}$ of fragments, as follows.

Definition 4.1 (Relevant constraints) *Given a set $\mathcal{T} = \{F_1, \ldots, F_n\}$ of fragments and a set $\mathcal{C}$ of confidentiality constraints, the set $\mathcal{C}_\mathcal{T}$ of* relevant constraints *for $\mathcal{T}$ is defined as $\mathcal{C}_\mathcal{T} = \{c \in \mathcal{C} : c \subseteq F_1 \cup \ldots \cup F_n\}$.*

Intuitively, the constraints relevant for a set of fragments are all those constraints covered by the fragments (i.e., all confidentiality constraints that are a subset of the union of the fragments). For instance, the only constraint among those reported in Figure 1(b) that is relevant for the set of fragments in Figure 1(c) is c_1; all constraints are instead relevant for the set of fragments in Figure 3.

The definition of a group association over different fragments is a natural extension of the case with two fragments, where the association is induced by groupings enforced within the different fragments. The consideration of the universal group association implies that only one grouping is applied within each fragment. Hence, given a fragmentation $\mathcal{F}=\{F_1, \ldots, F_n\}$, a $(k_1, \ldots, k_n)$-grouping is a set $\{\mathcal{G}_1, \ldots, \mathcal{G}_n\}$ of grouping functions defined over fragments $\{f_1, \ldots, f_n\}$ (i.e., a set of k_i-groupings over f_i, $i = 1, \ldots, n$). Figure 4 illustrates a (2,2,2)-grouping over fragments $F_l=\{\texttt{Name}, \texttt{YoB}\}$, $F_m=\{\texttt{Edu}, \texttt{ZIP}\}$ and $F_r=\{\texttt{Job}, \texttt{MarStatus}, \texttt{Disease}\}$ of relation PATIENTS in Figure 1(a), and the induced group association A_{lmr}.

Like for the case of two fragments, a group association permits to establish relationships among the tuples in the different fragments, while maintaining the uncertainty on which tuple in each fragment is actually associated with each tuple in another fragment. Such an uncertainty is given by the cardinality of the groups. The reconstruction made available by a group association, and obtained as the joins of the fragments in $\mathcal{F}$ and A, generates in fact all possible combinations among the tuples of associated



(a)

F_l

	Name	YoB	G_l
l_1	Alice	1974	ny1
l_2	Bob	1965	ny1
l_3	Carol	1976	ny2
l_4	David	1972	ny2
l_5	Greg	1975	ny3
l_6	Hal	1970	ny3
l_7	Eric	1960	ny4
l_8	Fred	1974	ny4

F_m

	Edu	ZIP	G_m
m_1	B.Sc	90015	ez1
m_2	MBA	90038	ez2
m_3	Ph.D	90001	ez1
m_4	M.Sc	90087	ez4
m_5	M.Sc	90025	ez2
m_6	Th.D	90007	ez3
m_7	Primary	90025	ez4
m_8	Ed.D	90060	ez3

F_r

	Job	MarStatus	Disease	G_r
r_1	Assistant	Married	Flu	jmd1
r_2	Manager	Widow	Diabetes	jmd4
r_3	Manager	Married	Calculi	jmd2
r_4	Doctor	Divorced	Asthma	jmd1
r_5	Doctor	Single	Flu	jmd3
r_6	Clerk	Single	Calculi	jmd4
r_7	Chef	Divorced	Diabetes	jmd2
r_8	Teacher	Widow	Asthma	jmd3

A_{lmr}

G_l	G_m	G_r
ny1	ez1	jmd1
ny1	ez2	jmd4
ny2	ez1	jmd2
ny2	ez4	jmd1
ny3	ez2	jmd3
ny3	ez3	jmd4
ny4	ez4	jmd2
ny4	ez3	jmd3

(b)

Fig. 4. Graphical representation (a) and corresponding relations (b) of a 4-loose association among three fragments F_l, F_m, and F_r of relation PATIENTS in Figure 1(a)

groups. Let us denote with $\mathcal{F}\bowtie A$ such a join. Guaranteeing k-looseness for the sensitive associations represented by relevant constraints requires ensuring that the reconstruction of tuples, made possible by the association among groups, is such that: for each constraint c relevant for $\mathcal{F}$ and for each fragment F, there are at least k tuples $t_1^a, \ldots, t_k^a$ in $\mathcal{F}\bowtie A$ such that, if $t_1^a[F \cap c] = \ldots = t_k^a[F \cap c]$, then $t_1^a[c \setminus F] \neq \ldots \neq t_k^a[c \setminus F]$. The k-looseness requirement must then take into consideration not only the number of tuples in other fragments with which a tuple can be associated but also the diversity of their values for the attributes involved in confidentiality constraints. In fact, different tuples that have the same values for these attributes do not provide the diversity needed to ensure k-looseness. We then start by identifying these tuples as follows.

Definition 4.2 (Alike) *Given a fragmentation $\mathcal{F}$=$\{F_1, \ldots, F_n\}$ with its instance $\{f_1, \ldots, f_n\}$, and the set $\mathcal{C}_\mathcal{F}$ of confidentiality constraints relevant for $\mathcal{F}$, tuples t_i,$t_j \in f_z$, $z = 1, \ldots, n$, are said to be* alike *with respect to a constraint $c \in \mathcal{C}_\mathcal{F}$, denoted $t_i \simeq_c t_j$, iff $c \cap F_z \neq \emptyset$ and $t_i[c \cap F_z] = t_j[c \cap F_z]$. Two tuples are said to be* alike *with respect to a set $\mathcal{C}_\mathcal{F}$ of relevant constraints, denoted $t_i \simeq_{\mathcal{C}_\mathcal{F}} t_j$, if they are alike with respect to at least one constraint $c \in \mathcal{C}_\mathcal{F}$.*

According to this definition, given a fragmentation $\mathcal{F}$, two tuples in a fragment instance f_i of fragment $F_i \in \mathcal{F}$ are alike if they have the same values for the attributes in at least one constraint relevant for $\mathcal{F}$. For instance, with reference to the fragments in Figure 4, $r_4 \simeq_{c_3} r_8$ as

r_4[Disease]=r_8[Disease]=Asthma. Since we are interested in evaluating the alike relationship with respect to the set $\mathcal{C}_\mathcal{F}$ of relevant constraints, in the following we omit the subscript of the alike relationship whenever clear from the context (i.e., we write $t_i \simeq t_j$ instead of $t_i \simeq_{\mathcal{C}_\mathcal{F}} t_j$).

Based on the definition of relevant constraints and alike relationship, we can now define k-looseness of a group association. Intuitively, a group association A is k-loose if, for each relevant constraint $c \in \mathcal{C}_\mathcal{F}$, each tuple t^a in A represents at least k associations among sub-tuples in fragments including attributes in c, and these associations correspond to at least k different combinations of values of the attributes in c. In other words, t^a represents k tuples that may belong to the original relation and that are not alike with respect to c. Formally, k-looseness is defined as follows.

Definition 4.3 (k-Looseness) *Given a fragmentation $\mathcal{F} = \{F_1, \ldots, F_n\}$ with its instance $\{f_1, \ldots, f_n\}$, the set $\mathcal{C}_\mathcal{F}$ of confidentiality constraints relevant for $\mathcal{F}$, and a group association A over $\{f_1, \ldots, f_n\}$, A is said to be* k-loose *with respect to $\mathcal{C}_\mathcal{F}$ iff $\forall c \in \mathcal{C}_\mathcal{F}$, let $\mathcal{F}_c = \{F \in \mathcal{F} : F \cap c \neq \emptyset\}$, $\forall F_i \in \mathcal{F}_c$ and $\forall g_i \in \mathsf{GID}_i$ let $T = \bigcup_{t^a \in A} \{\mathcal{G}_j^{-1}(t^a[\mathtt{G}_j]) \times \ldots \times \mathcal{G}_l^{-1}(t^a[\mathtt{G}_l]) : t^a[\mathtt{G}_i] = g_i\}$ with $\{F_j, \ldots, F_l\} = \mathcal{F}_c \setminus \{F_i\} \Longrightarrow |T| \geq k$ and $\forall t_x, t_y \in T$, $x \neq y$, $t_x \not\simeq_c t_y$.*

As an example, consider the group association in Figure 4 and confidentiality constraint c_5=$\{$YoB, ZIP, MarStatus$\}$ in Figure 1. The first tuple in A_{lmr} represents 8 possible associations among sub-tuples in F_l, F_r, and F_m (i.e., any possible combination of tuples from sets $\{l_1, l_2\}$, $\{m_1, m_3\}$, and $\{r_1, r_4\}$). These associations correspond to the following 8 possible combinations of values of attributes YoB, ZIP, and MarStatus that may belong to the original relation: ⟨1974, 90015, Married⟩, ⟨1974, 90015, Divorced⟩, ⟨1974, 90001, Married⟩, ⟨1974, 90001, Divorced⟩, ⟨1965, 90015, Married⟩, ⟨1965, 90015, Divorced⟩, ⟨1965, 90001, Married⟩, ⟨1965, 90001, Divorced⟩.

k-Looseness guarantees that none of the sensitive associations represented by relevant constraints can be reconstructed with confidence higher than $1/k$. Figure 4 illustrates a fragmentation of the relation in Figure 1(a) and a group association A_{lmr} that guarantees 4-looseness for the sensitive associations expressed by the confidentiality constraints in Figure 1(b).

Clearly, there is a correspondence between the size of the groupings and the k-looseness of the association induced by them. Trivially, a $(k_1, \ldots, k_n)$-grouping cannot provide k-looseness for a $k > \prod_{i=1}^{n} k_i$. Consider a constraint c, which includes attributes in F_i and F_j only. The $(k_1, \ldots, k_n)$-grouping can provide uncertainty over the associations existing among the attributes in c for a $k \leq k_i \cdot k_j$. Indeed, any tuple in f_i is associated with at least $k_i \cdot k_j$ tuples in f_j, which have different values for the attributes in c if groups are properly defined. With reference to the group association in Figure 4, the sensitive association represented by constraint c_1=$\{$YoB, Edu$\}$ enjoys a k-looseness of four: each value of YoB can be indistinguishably associated with at least four possible values of Edu, and viceversa. A constraint c involving more than two fragments may enjoy higher protection from the same $(k_1, \ldots, k_n)$-grouping. Considering the example in Figure 4, the sensitive association expressed by constraint c_5=$\{$YoB, ZIP, MarStatus$\}$ enjoys a k-looseness of eight: for each value of YoB there are at least eight possible different pairs of (ZIP, MarStatus); for each value of ZIP there are at least eight possible different pairs of (YoB, MarStatus); and for each value of MarStatus there are at least eight possible different pairs of (YoB, ZIP). For instance, value 1965 for attribute YoB can be associated with ⟨90015, Married⟩, ⟨90015, Divorced⟩, ⟨90001, Divorced⟩, ⟨90001, Married⟩, ⟨90038, Widow⟩, ⟨90038, Single⟩, ⟨90025, Widow⟩, ⟨90025, Single⟩. Since we consider minimal fragmentations, for each pair of fragments F_i, F_j in $\mathcal{F}$ there exists at least a confidentiality constraint c relevant for F_i and F_j only (i.e., $\forall \{F_i, F_j\} \in \mathcal{F}$, $i \neq j$, $\exists c \in \mathcal{C}$ s.t. $c \subseteq F_i \cup F_j$, Theorem A.2 in [14]), which enjoys a k-looseness of $k_i \cdot k_j$. Hence, a $(k_1, \ldots, k_n)$-grouping can ensure k-looseness with $k \leq min\{k_i \cdot k_j : i, j = 1, \ldots, n, i \neq j\}$ for the constraints in $\mathcal{C}_\mathcal{F}$. Whether

the $(k_1, \ldots, k_n)$-grouping provides k-looseness for lower values of k depends on how the groups are defined. In the following, we introduce three heterogeneity properties of grouping (revising and extending those provided in [14]) whose satisfaction ensures k-looseness for k=*min*$\{k_i \cdot k_j : i, j = 1, \ldots, n, i \neq j\}$.

4.2. Heterogeneity properties

The heterogeneity properties ensure diversity of the induced associations, which are defined as sensitive by confidentiality constraints. They operate at three different levels: groupings, group associations, and value associations.

The first property we introduce is *group heterogeneity*, which ensures diversity within each group by imposing that groups in a fragment do not include tuples with the same values for the attributes in relevant constraints. In this way, the minimum size k_i of the groups in fragment F_i, $i = 1, \ldots, n$, reflects the minimum number of different values in the group for each subset of attributes that appear together in a relevant constraint.

Property 4.4 (Group heterogeneity) *Given a fragmentation $\mathcal{F}$=$\{F_1, \ldots, F_n\}$ with its instance $\{f_1, \ldots, f_n\}$, and the set $\mathcal{C}_\mathcal{F}$ of confidentiality constraints relevant for $\mathcal{F}$, grouping functions $\mathcal{G}_i$ over f_i, $i = 1, \ldots, n$, satisfy* group heterogeneity *iff* $\forall t_z, t_w \in f_i$ *with* $t_z \simeq t_w \Longrightarrow \mathcal{G}_i(t_z) \neq \mathcal{G}_i(t_w)$.

This property is a straightforward extension of the one operating on two fragments, as its enforcement is local to each individual fragment to take into account all constraints relevant for $\mathcal{F}$ (not only those relevant for a pair). For instance, in Figure 4 the grouping functions of the three fragments satisfy group heterogeneity for $\mathcal{C}_\mathcal{F}$=$\{c_1, \ldots, c_5\}$ while the grouping function of fragment F_r in Figure 5(a) violates it with respect to confidentiality constraint c_4=$\{$`YoB`,`ZIP`,`Disease`$\}$. In fact, fragment instance f_r includes a group with tuples assuming the same value for attribute $c_4 \cap F_r$ =`Disease` (i.e., Diabetes).

The second property we introduce is *association heterogeneity*, which imposes diversity in the group association. For a group association A between two fragments, this property requires that A does not include duplicate tuples, that is, at most one association can exist between each pair of groups of the two fragments. By considering the more general case of a group association among an arbitrary number of fragments, this property requires that, for each constraint c in $\mathcal{C}_\mathcal{F}$, each group in a fragment f_i such that $F_i \cap c \neq \emptyset$ appears in at least k tuples in A that differ *at least* in the group of one of the fragments f_j storing attributes in c (i.e., $c \cap F_j \neq \emptyset$). In other words, the association heterogeneity property implies that A cannot have two tuples with the same group identifier for all attributes $\mathtt{G}_{i_j}$, $j = 1, \ldots, l$ corresponding to fragments storing attributes that appear in a constraint. Since we consider minimal fragmentations, there exists at least one relevant constraint for each pair of fragments in $\mathcal{F}$. Therefore, a group association A satisfies the association heterogeneity property if it does not have two tuples with the same group identifier for any pair of group attributes $\mathtt{G}_i$,$\mathtt{G}_j$, $i, j = 1, \ldots, n$, and $i \neq j$.

Property 4.5 (Association heterogeneity) *A group association A satisfies* association heterogeneity *iff* $\forall (g_{i_1}, \ldots, g_{i_n}), (g_{j_1}, \ldots, g_{j_n}) \in A$ *such that* $g_{i_z} = g_{j_z} \Longrightarrow g_{i_w} \neq g_{j_w}$, $w = 1, \ldots, n$ *and* $w \neq z$.

Figure 4 illustrates a group association that satisfies the association heterogeneity property, while the group association in Figure 5(b) violates it since a group of fragment f_l is associated twice with a group in fragment f_r.

The third property we introduce is *deep heterogeneity*, which captures the need of guaranteeing diversity in the associations of values behind the groups. Considering a pair of fragments f_i and f_j, deep heterogeneity requires that a group in f_i be associated with groups in f_j that do not include duplicated

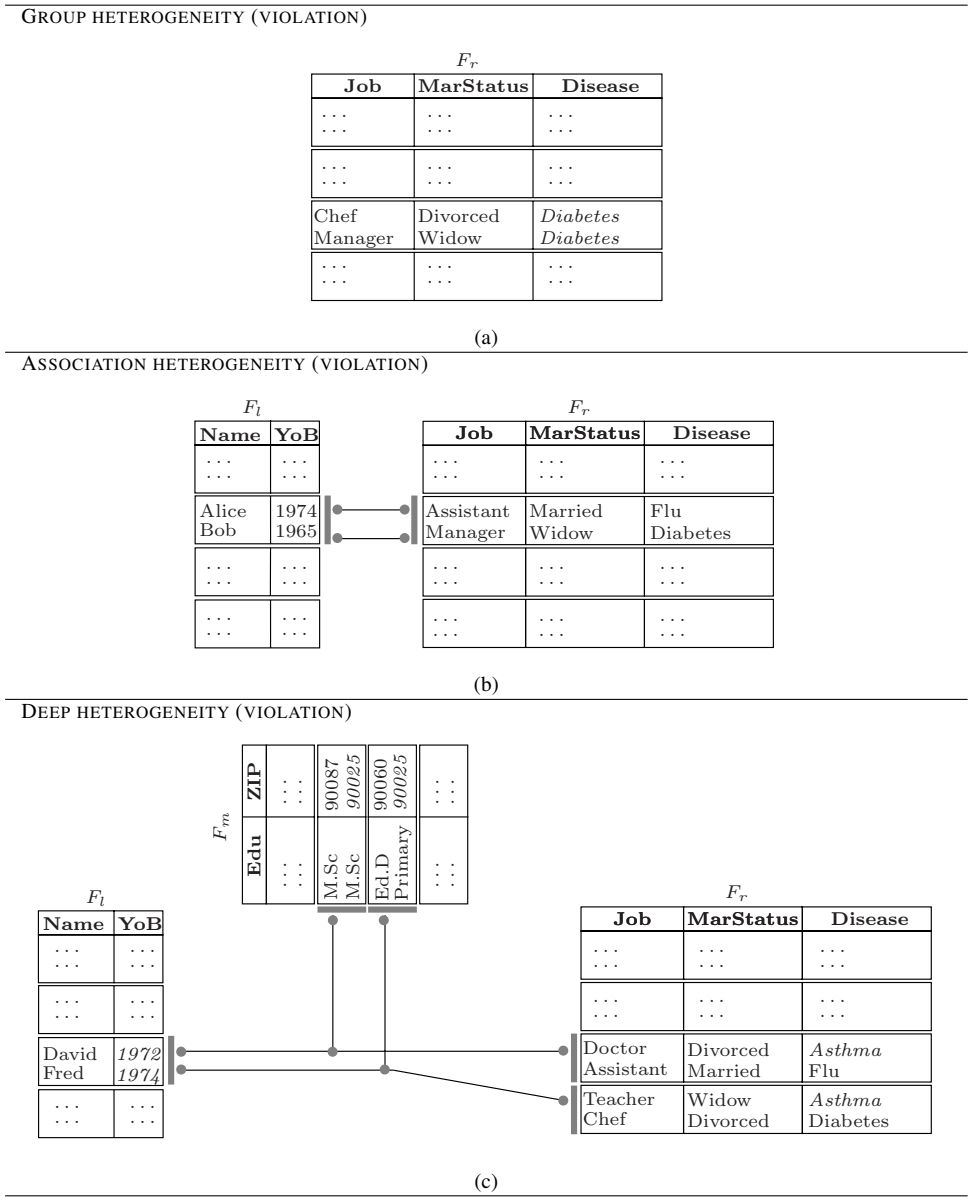


Fig. 5. Examples of violations of heterogeneity properties with respect to constraint c_4={YoB,ZIP,Disease}

values for the attributes in a constraint $c \subseteq F_i \cup F_j$ (i.e., tuples are not alike with respect to c). In fact, if the groups in f_j with which a group in f_i is associated contain alike tuples with respect to c, the $k_i \cdot k_j$ corresponding tuples do not contain $k_i \cdot k_j$ different values for the attributes in c, meaning that the group association offers less protection than expected. For instance, groups jmd1 and jmd3 in Figure 4 have the same values for attribute Disease (i.e., Flu and Asthma). Therefore, a group in f_l cannot be associated with both jmd1 and jmd3 because of constraint c_3={Name,Disease} (otherwise, the association between F_l and F_r would be 2-loose instead of 4-loose). Considering the more general case of a group association among an arbitrary number of fragments, and a constraint c composed of attributes

stored in fragments $\{F_1, \ldots, F_n\}$, deep heterogeneity requires a group f_i ($i = 1, \ldots, n$) be associated with groups in $\{f_1, \ldots, f_n\} \setminus \{f_i\}$ that do not permit to reconstruct (via the loose join $\mathcal{F}\bowtie A$) possible semi-tuples that have the same values for all the attributes in c. Note that deep heterogeneity does not require diversity over *all* the fragments storing the attributes composing a constraint, since this condition would be more restrictive than necessary to guarantee k-looseness. In fact, it is sufficient, for each tuple in a fragment f_i, to break the association with *one* of the fragments f_j ($j = 1, \ldots, n$, $i \neq j$) storing the attributes in c. For instance, with reference to the example in Figure 4, it is sufficient that each group in f_l be associated with groups of non alike tuples in either f_m or f_r to guarantee a 4-looseness for the sensitive association modeled by c_4. The deep heterogeneity property is formally defined as follows.

Property 4.6 (Deep heterogeneity) *Given a fragmentation $\mathcal{F}$=$\{F_1, \ldots, F_n\}$ with its instance $\{f_1, \ldots, f_n\}$, and the set $\mathcal{C}_\mathcal{F}$ of constraints relevant for $\mathcal{F}$, a group association A over $\mathcal{F}$ satisfies* deep heterogeneity *iff* $\forall c{\in}\mathcal{C}_\mathcal{F}$; $\forall F_z \in \mathcal{F}$, $F_z \cap c \neq \emptyset$; $\forall\, (g_{i_1}, g_{i_2} \ldots g_{i_n}), (g_{j_1}, g_{j_2} \ldots g_{j_n}) \in A$ *the following condition is satisfied:* $g_{i_z} = g_{j_z} \Longrightarrow \bigvee_{l=1,\ldots,n,\ l\neq z} \nexists t_x, t_y\colon t_x{\in}\mathcal{G}_l^{-1}(g_{i_l}),\ t_y{\in}\mathcal{G}_l^{-1}(g_{j_l}),\ t_x \simeq_c t_y$.

Given a constraint c whose attributes appear in fragments $\{F_{i_1}, \ldots, F_{i_j}\}$, deep heterogeneity is satisfied with respect to c if no two tuples t, t' in A that have the same group g_y in f_{i_y} are associated with groups that include alike tuples with respect to c for all the fragments f_{i_x}, $x = 1, \ldots, j$ and $x \neq y$. This property must be true for all the groups in each fragment. This guarantees that, for each constraint, no sensitive association can be reconstructed with confidence higher than $1/k$. An example of group association that satisfies deep heterogeneity is illustrated in Figure 4. Note that deep heterogeneity is satisfied even though the two tuples in group ny2 for f_l are associated with groups jmd1 and jmd2 in f_r, which include tuples $r_1 \simeq_{c_5} r_3$ and $r_4 \simeq_{c_5} r_7$. In fact, constraint c_5 is not covered by F_l and F_r but by the three fragments all together, and heterogeneity of the associations in which r_1 and r_3 (r_4 and r_7, respectively) are involved is provided by the tuples in f_m. Figure 5(c) illustrates an example of violation of the deep heterogeneity property with respect to confidentiality constraint c_4=$\{$`YoB`,`ZIP`,`Disease`$\}$. In fact, the groups in the instances f_m and f_r with which a group in f_l is associated include tuples that are alike with respect to confidentiality constraint c_4 (i.e., two tuples in f_m have the same value for attribute `ZIP` and two tuples in f_r have the same value for attribute `Disease`), clearly reducing the protection offered by the association. In fact, the tuples that can be reconstructed by joining these two groups in f_m and f_r include occurrences of the same values for the attributes in c_4 (i.e., `ZIP`=90025 and `Disease`=Asthma). Hence, the association `YoB`=1972, `ZIP`=90025, and `Disease`=Asthma holds with probability higher than $1/k$.

If the three properties above are satisfied by a $(k_1, \ldots, k_n)$-grouping and its induced group association, then the group association is k-loose for any $k{\leq}min\{k_i \cdot k_j : i, j = 1, \ldots, n,\ i \neq j\}$, as stated by the following theorem.

Theorem 4.7 *Given a fragmentation $\mathcal{F}$=$\{F_1, \ldots, F_n\}$ with its instance $\{f_1, \ldots, f_n\}$, the set $\mathcal{C}_\mathcal{F}$ of constraints relevant for $\mathcal{F}$, and a $(k_1, \ldots, k_n)$-grouping that satisfies Properties 4.4, 4.5, and 4.6, the group association A induced by the $(k_1, \ldots, k_n)$-grouping is k-loose with respect to $\mathcal{C}_\mathcal{F}$ (Definition 4.3) for each $k{\leq}\min\{k_i \cdot k_j : i, j = 1, \ldots, n,\ i \neq j\}$.*

PROOF: (SKETCH). To assess the protection offered by the release of a $(k_1, \ldots, k_n)$-grouping that satisfies Properties 4.4, 4.5, and 4.6, we first analyze the protection provided to the sensitive association represented by an arbitrary confidentiality constraint c in $\mathcal{C}_\mathcal{F}$.

By Definition 2.2, each group $g_a \in \mathsf{GID}_i$ contains at least k_i tuples, $\forall F_i \in \mathcal{F}$. Each group $g_a \in \mathsf{GID}_i$ appears in at least k_i tuples in A, each associating g_a to a different group g_b in GID_j, for each fragment F_j in $\mathcal{F}$ by Property 4.5. Hence, each $g_a \in \mathsf{GID}_i$ is associated with at least k_i different groups in GID_j, $\forall F_j \in \mathcal{F} : F_j \cap c \neq \emptyset$, $i \neq j$. Each tuple t^a in A having $t^a[\mathtt{G}_i] = g_a$ has at least $\prod_j k_j$ (with $F_j \in \mathcal{F}$: $F_j \cap c \neq \emptyset$, $i \neq j$) occurrences in the join $\mathcal{F} \bowtie A$. Let us denote with *groups_*a_i the tuples in $\mathcal{F} \bowtie A$ of the occurrences of a tuple t_i^a in A. Tuples in *groups_*a_i are not alike with respect to c. In fact, by Properties 4.4, each group in GID_j is composed of at least k_j tuples that are not alike with respect to c, $\forall F_j \in \mathcal{F}$ such that $F_j \cap c \neq \emptyset$. By Property 4.6, for each pair of tuples t_x^a, t_y^a in A with $t_x^a[\mathtt{G}_i] = t_y^a[\mathtt{G}_i] = g_a$, the tuples in *groups_*$a_x \cup$ *groups_*a_y are not alike with respect to c. Hence, $\mathcal{F} \bowtie A$ has at least $k_i \cdot \prod k_j$ tuples, all with $t^a[\mathtt{G}_i] = g_a$, that are note alike with respect to c.

Then, a $(k_1, \ldots, k_n)$-grouping satisfying Properties 4.4, 4.5, and 4.6 induces a group association that is k-loose with respect to c for each $k \leq \prod k_i$, $\forall F_j \in \mathcal{F}$ such that $F_j \cap c \neq \emptyset$.

Since we consider minimal fragmentations only, for each pair of fragments F_i,F_j in $\mathcal{F}$ there exists at least a confidentiality constraint c that is relevant for F_i and F_j only. Hence, the $(k_1, \ldots, k_n)$-grouping satisfying Properties 4.4, 4.5, and 4.6 induces a group association that is k-loose with $k = k_i \cdot k_j$ between F_i and F_j (i.e., it guarantees a protection degree $k_i \cdot k_j$ to the constraints relevant for F_i and F_j).

We can then conclude that the $(k_1, \ldots, k_n)$-grouping satisfying Properties 4.4, 4.5, and 4.6 induces a group association that is k-loose with respect to $\mathcal{C}_\mathcal{F}$ for each $k \leq$ *min*$\{k_i \cdot k_j : i, j = 1, \ldots, n, i \neq j\}$. □

We note that the protection degree offered by a $(k_1, \ldots, k_n)$-grouping that satisfies Properties 4.4, 4.5, and 4.6 may be different (but not less than k) for each confidentiality constraint c in $\mathcal{C}_\mathcal{F}$. Indeed, the protection degree for a constraint c is *min*$\{k_i \cdot k_j : F_i, F_j \in \mathcal{F}$ and $F_i \cap c \neq \emptyset$ and $F_j \cap c \neq \emptyset\}$.

4.3. Some observations on k-looseness

The consideration of all the constraints relevant for the fragments involved in a group association guarantees that no sensitive association can be reconstructed with a probability greater than $1/k$. For instance, confidentiality constraint $c_3 = \{\mathtt{Name}, \mathtt{Disease}\}$ is properly protected, for a looseness of 4, by the group association in Figure 4 while, as already illustrated in Section 3, the two group associations in Figure 3 grant only a 2-looseness protection to it.

Given a k-loose association A among a set $\mathcal{F}$ of fragments, the release of this loose association is equivalent to the release of $2^n - n$, with $n = |\mathcal{F}|$, k-loose associations (one for each subset of fragments in $\mathcal{F}$). Indeed, the projection over a subset of attributes in A represents a k-loose association for the fragments corresponding to the projected group of attributes. This is formally captured by the following observation.

Observation 1 *Given a fragmentation $\mathcal{F} = \{F_1, \ldots, F_n\}$, a subset $\{F_i, \ldots, F_j\}$ of $\mathcal{F}$, and a k-loose association $A(\mathtt{G}_1, \ldots, \mathtt{G}_n)$ over $\mathcal{F}$, group association $A'(\mathtt{G}_i, \ldots, \mathtt{G}_j)$ is a k-loose association over $\{F_i, \ldots, F_j\}$.*

For instance, with respect to the 4-loose association in Figure 4, the projection of A over attributes $\mathtt{G}_l$ and $\mathtt{G}_m$ is a 4-loose association between F_l and F_m.

Since a k-loose association defined over a set $\mathcal{F}$ of fragments guarantees that sensitive associations represented by constraints in $\mathcal{C}_\mathcal{F}$ be properly protected, the release of multiple loose associations among arbitrary (and possibly overlapping) subsets of fragments in $\mathcal{F}$ provides the data owner with the same protection guarantee. The data owner can therefore decide to release either one loose association A

encompassing all the associations among the fragments in $\mathcal{F}$, or a subset of loose associations defined among arbitrary subsets of fragments in $\mathcal{F}$ by projecting the corresponding attributes from A. This is formally captured by the following observation.

Observation 2 *Given a fragmentation $\mathcal{F}=\{F_1,\ldots,F_n\}$ and a k-loose association $A(\mathtt{G}_1,\ldots,\mathtt{G}_n)$ over it, the release of an arbitrary set of k-loose associations $\{A_1(\mathtt{G}_h,\ldots,\mathtt{G}_i),\ldots,A_m(\mathtt{G}_j,\ldots,\mathtt{G}_k)\}$, with $\{\mathtt{G}_h,\ldots,\mathtt{G}_k\}\subseteq\{\mathtt{G}_1,\ldots,\mathtt{G}_n\}$, provides at least the same protection guarantees as the release of A.*

For instance, the data owner can decide to release the group associations obtained projecting $\langle \mathtt{G}_l,\mathtt{G}_m\rangle$ and $\langle \mathtt{G}_m,\mathtt{G}_r\rangle$ from the group association in Figure 4. This solution does not suffer from the privacy breach illustrated in Section 3, while providing associations between groups of the same size (i.e., the same utility for data recipients).

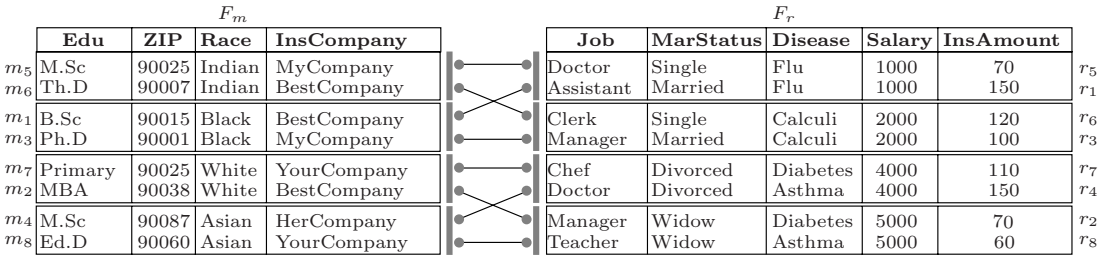
According to the two observations above, the data owner can release more than one group association among arbitrary subsets of fragments in $\mathcal{F}$ without causing privacy breaches. Note however that if the group associations of interest operate on disjoint subsets of fragments (i.e., no fragment is involved in more than one group association), they can be defined independently from each other without risks of unintended disclosure of sensitive associations. This is formally captured by the following observation.

Observation 3 *Given a fragmentation $\mathcal{F}$, and a set $\{\mathtt{F}_1,\ldots,\mathtt{F}_n\}$ of subsets of fragments in $\mathcal{F}$, the release of n loose associations A_i, $i=1,\ldots,n$ does not expose any sensitive association if $\mathtt{F}_i\cap\mathtt{F}_j=\emptyset$, $i,j=1,\ldots,n$, $i\neq j$.*

5. Queries and data utility with loose associations

The reason for publishing group associations among fragments, representing vertical views over the original data, is to provide some (not precise) information on the associations among the tuples in the fragments while ensuring not to expose the sensitive associations defined among their attributes (for which the degree of uncertainty k should be maintained). Group associations then increase the utility of the data released for queries involving different fragments. However, given a set of fragments, different group associations might be defined satisfying a given degree k of looseness to be provided. There are two different issues that have to be properly addressed in the construction of group associations: one is how to select the size k_i of the grouping of each fragment f_i such that the product of any two k_i is equal to or greater than k; and one is how to group tuples within the fragments so to maximize utility.

With respect to the first issue of sizing the groups, there are different possible values of the different k_i that can satisfy the degree k of protection. For instance, for a group association between two fragments, we can use (k,1), ($\lceil\sqrt{k}\rceil$, $\lfloor\sqrt{k}\rfloor$), and (1,k). In the case of multiple fragments, the best utility can be achieved by distributing as much evenly as possible the sizing of the groups, hence imposing on each group a size close to $\sqrt{k}$. An uneven distribution would in fact result in an over-protection of the group associations over some of the fragments (a value of looseness much higher than the required k for constraints covered by a subset of the fragments in $\mathcal{F}$). Experiments show that this would lead to a significant reduction in the precision of the queries (see Section 7). For instance, a looseness of 12 over three fragments could be achieved with a (3,4,4)-grouping; a solution creating a (1,12,12)-grouping would indeed provide the required protection overall but would probably provide little utility for the association between the second and third fragments (whose association would in fact be 144-loose for the constraints that are relevant for the second and third fragment only).



	F_m					F_r					
	Edu	**ZIP**	**Race**	**InsCompany**		**Job**	**MarStatus**	**Disease**	**Salary**	**InsAmount**	
m_5	M.Sc	90025	Indian	MyCompany		Doctor	Single	Flu	1000	70	r_5
m_6	Th.D	90007	Indian	BestCompany		Assistant	Married	Flu	1000	150	r_1
m_1	B.Sc	90015	Black	BestCompany		Clerk	Single	Calculi	2000	120	r_6
m_3	Ph.D	90001	Black	MyCompany		Manager	Married	Calculi	2000	100	r_3
m_7	Primary	90025	White	YourCompany		Chef	Divorced	Diabetes	4000	110	r_7
m_2	MBA	90038	White	BestCompany		Doctor	Divorced	Asthma	4000	150	r_4
m_4	M.Sc	90087	Asian	HerCompany		Manager	Widow	Diabetes	5000	70	r_2
m_8	Ed.D	90060	Asian	YourCompany		Teacher	Widow	Asthma	5000	60	r_8

Fig. 6. An example of group association between two fragments F_m and F_r of relation PATIENTS in Figure 1(a) where tuples are grouped taking into account the similarity of the values of attributes `Race` and `Salary`

With respect to the issue of grouping within a fragment, we first note that queries that involve a single fragment (i.e., all the attributes in the query belong to the same fragment) are not affected by fragmentation as they can be answered exactly by querying the fragment. For instance, with respect to the fragments in Figure 6, query q ="SELECT AVG(`Salary`) FROM PATIENTS GROUP BY `Job`" involves attributes that belong to fragment F_r only. Hence, the execution of the query over fragment F_r returns exactly the same result as its execution over the original relation PATIENTS in Figure 1(a). Om the following, we focus our discussion on queries that involve two or more fragments, on which a group association is to be defined, with the goal of determining how to group the tuples in fragments so that the induced group associations maximize query utility. In particular, we consider aggregate queries of the form "SELECT *Att*, $\text{AGG}_i(a_i), \ldots, \text{AGG}_j(a_j)$ FROM S GROUP BY *Att*", where $\text{AGG}_i, \ldots, \text{AGG}_j$ are aggregation functions (e.g., COUNT, AVG, MIN, MAX functions), and $a_i, \ldots, a_j$ as well as set *Att* (this latter optional in the SELECT clause) are attributes that appear in fragments. For instance, with reference to Figure 6, query q="SELECT AVG(`Salary`) FROM PATIENTS GROUP BY `Race`" aims at computing the average salary grouped by the race of patients. We measure utility of a group association as the average improvement over the accuracy of the results (i.e., with less error) with respect to the results obtained in absence of the group association. Intuitively, utility is obtained as 1 minus the ratio of the average difference with respect to the real values in presence of group associations, and the average difference with respect to the real values in absence of group associations.

The execution of queries over group associations brings in, together with the real tuples on which the query should be executed, all the tuples together with them in their groups and the uncertainty – by definition – of which sub-tuples in a fragment are associated with which sub-tuples in other fragments. Our observation is therefore that groups within fragments should be formed so to contain as much as possible tuples that are similar for the attributes involved in the queries (have close values for continuous attributes). The intuition behind this is that, although the query is evaluated on a possibly larger number of tuples included in the returned groups, such tuples – assuming similar values – maintain the query result within a reasonable error, thus providing utility of the response. The more the attributes involved in the query on which such an observation has been taken into account in the grouping, the better the utility provided by the group association for the query. In fact, similarity of values within groups (even when ensuring diversity of the values) might provide limited uncertainty of values within a group. We therefore expect that not all the attributes involved in confidentiality constraints should be taken into account in this process.

Let us see now an example of queries over our fragmentation involving attributes such that none, some, or all of them have been subject to the observation above in the grouping (i.e., groups include

similar values for none of, some of, or all the attributes in the query). Consider the fragments and group association in Figure 6, computed over relation PATIENTS in Figure 1(a), where the group association has been produced grouping tuples with similar `Race` values for fragment f_m and similar `Salary` values for fragment f_r. We can then see the following three different cases.

- *A query q_{ns} involves none of the attributes whose similarity has been considered in the grouping.* An example of such a query on the group association in Figure 6 is q_{ns} = "SELECT `Edu`,AVG(`InsAmount`) FROM PATIENTS GROUP BY `Edu`", requiring the average insurance amount for the different education levels recorded. In this case, the utility of the association typically remains limited. We note however that the utility for this kind of queries has always been positive in our experimental analysis, reaching values close to 40% for some queries.
- *A query q_{as} involves also (but not only) attributes whose similarity has been considered in the grouping.* An example of such a query on the group association in Figure 6 is q_{as} = "SELECT `InsCompany`,AVG(`Salary`) FROM PATIENTS GROUP BY `InsCompany`," requiring, for each insurance company, the average salary of insurance holders. Enjoying the fact that the additional tuples involved in the computation will typically have salary close to the values of real tuples, this query provides quite appreciable utility with respect to the real result. As we will discuss in Section 7, utility for queries of this type has typically shown values close to 80% in our experimental evaluation.
- *A query q_{os} involves only attributes whose similarity has been considered in the grouping.* An example of such a query on the group association in Figure 6 is q_{os} = "SELECT `Race`, AVG(`Salary`) FROM PATIENTS GROUP BY `Race`" requiring the average salary of patients grouped by race. This query can benefit from the fact that similarity has been considered for both the attributes involved, which ensures that the additional tuples, brought-in in the evaluation because of the looseness of the association, have values for the involved attributes close to the ones on which the computation would have been executed if the query was performed on the original relation. As we will discuss in Section 7, the utility for this kind of queries is very high, and has typically shown values near 100% in our experimental evaluation.

Figure 7 shows the results of the queries above when executed over the the group association in Figure 6 or over the original relation in Figure 1(a).

6. Computing a k-loose association

Figure 8 illustrates our heuristic algorithm that computes a k-loose association aiming at providing greater utility in query evaluation. The algorithm takes as input a relation s defined over relation schema $S(a_1, \ldots, a_m)$, a fragmentation $\mathcal{F}$=$\{F_1, \ldots, F_n\}$ and its instance $\{f_1, \ldots, f_n\}$, a set $\mathcal{C}_\mathcal{F}$ of confidentiality constraints relevant for $\mathcal{F}$, privacy parameters $k_1, \ldots, k_n$, and a set $\mathcal{A}$ of attributes often involved in the expected queries. The algorithm returns a k-loose association (with $k = min\{k_i \cdot k_j : i, j = 1, \ldots, n, i \neq j\}$), and the corresponding grouping functions $\mathcal{G}_1, \ldots, \mathcal{G}_n$. Intuitively, the algorithm first identifies an optimal group for each tuple – without considering heterogeneity properties – in such a way that each group in a fragment contains tuples with similar values for attributes in $\mathcal{A}$. Our solution to identify such an optimal grouping is based on the observation that similarity can be conveniently translated into an ordering of values within the attribute domains. Maximum similarity can in fact be guaranteed by keeping in the same groups elements that are contiguous in the ordered sequence of attribute values. The

q_{ns}

Edu	AVG(**InsAmount**)
B.Sc	110
Ed.D	97.5
MBA	97.5
M.Sc	104
Ph.D	110
Primary	97.5
Th.D	110

q_{as}

InsCompany	AVG(**Salary**)
BestCompany	2500
HerCompany	4500
MyCompany	1500
YourCompany	4500

q_{os}

Race	AVG(**Salary**)
Asian	4500
Black	1500
Indian	1500
White	4500

(a) Execution over the group association in Figure 6

q_{ns}

Edu	AVG(**InsAmount**)
B.Sc	150
Ed.D	60
MBA	70
M.Sc	110
Ph.D	100
Primary	110
Th.D	120

q_{as}

InsCompany	AVG(**Salary**)
BestCompany	2666
HerCompany	4000
MyCompany	1500
YourCompany	4500

q_{os}

Race	AVG(**Salary**)
Asian	4500
Black	1500
Indian	1500
White	4500

(b) Execution over the original relation in Figure 1(a)

q_{ns}

Edu	AVG(**InsAmount**)
B.Sc	104
Ed.D	104
MBA	104
M.Sc	104
Ph.D	104
Primary	104
Th.D	104

q_{as}

InsCompany	AVG(**Salary**)
BestCompany	3000
HerCompany	3000
MyCompany	3000
YourCompany	3000

q_{os}

Race	AVG(**Salary**)
Asian	3000
Black	3000
Indian	3000
White	3000

(c) Execution over the fragments in Figure 6 without group association

Fig. 7. Results of sample queries on a group association (a), on the original relation (b), and on fragments without group association (c) for queries involving none (q_{ns}), some (q_{as}), or all (q_{os}) of the attributes that have been considered for similarity in the grouping

algorithm first orders tuples in the fragment instances based on their values for attributes in $\mathcal{A}$, and then partitions the tuples in groups of size k. In this way, each optimal group will contain k tuples that, thanks to the ordering, have similar values for attributes in $\mathcal{A}$. Clearly, different ordering criteria can be applied to different attributes, to properly model the similarity requirement. As an example, for numerical values, we adopt the traditional $\geq$ order relationship. The groupings obtained by ordering the tuples according to $\mathcal{A}$ are optimal with respect to similarity (and hence utility of query responses), but they do not provide any guarantee with respect to the confidentiality of sensitive associations. Optimal groupings are then used by the algorithm to drive the real allocation of tuples to groups: for each fragment, the algorithm tries to assign the tuples to the group closest to the optimal group so that also the heterogeneity properties are satisfied. In the assignment of tuples to groups, the algorithm follows two main criteria: *i)* it favors groups close to the optimal group of each tuple; and *ii)* it prefers groups of size k_i over larger groups. Our heuristic algorithm implementing this approach is presented in details in the remainder of this section.

INPUT
s: relation defined over relation schema $S(a_1, \ldots, a_m)$
$\mathcal{F}$: fragmentation composed of fragments $\{F_1, \ldots, F_n\}$
$\{f_1, \ldots, f_n\}$: instances of fragments $\{F_1, \ldots, F_n\}$
$\mathcal{C}_\mathcal{F}$: set of confidentiality constraints relevant for $\mathcal{F}$
$k_1, \ldots, k_n$: privacy parameters for $\{F_1, \ldots, F_n\}$
$\mathcal{A}$: set of attributes to be considered for similarity

OUTPUT
A: k-loose association with k = *min*$\{k_i \cdot k_j : i, j = 1, \ldots, n, i \neq j\}$
$\mathcal{G}_1$,...,$\mathcal{G}_n$: grouping function for $\{F_1, \ldots, F_n\}$

MAIN
1: *To_Place* := s /* tuples that need to be allocated to groups */
2: $A := \emptyset$
3: /* **Step 1**: pre-calculate groups to which sub-tuples in fragments should be allocated */
4: **for** $i = |\mathcal{F}| \ldots 1$ **do**
5: order s by $\mathcal{A} \cap F_i$ /* order tuples according to the attributes in $\mathcal{A}$ in fragment F_i */
6: *current* := 0 /* current group identifier */
7: j := 0 /* number of tuples in $g^i_{current}$ */
8: **for each** $t \in s$ **do** /* pre-allocate each semi-tuple to a group */
9: *OptimalGrouping*[t][F_i] := $g^i_{current}$ /* optimal assignment for t in F_i */
10: $j := j + 1$
11: **if** $j = k_i$ **then** /* create a new group */
12: *current* := *current* + 1
13: $j := 0$
14: /* **Step 2**: allocate tuples to groups without generating over-quota groups */
15: **for each** $t \in$ *To_Place* **do** /* allocate t to a group in each fragment and define the corresponding tuple *ta* in A */
16: *ta* := **Find_Assignment**(t, NULL, 1, FALSE, *OptimalGrouping*)
17: **if** *ta* $\neq$ NULL **then** /* t has been assigned to a group in each fragment */
18: $A := A \cup \{ta\}$
19: *To_Place* := *To_Place* $\setminus \{t\}$
20: /* **Step 3**: allocate non-assigned tuples, possibly generating over-quota groups */
21: **for each** $t \in$ *To_Place* **do**
22: *ta* := **Find_Assignment**(t, NULL, 1, TRUE, *OptimalGrouping*)
23: **if** *ta* $\neq$ NULL **then** /* t has been assigned to a group in each fragment */
24: $A := A \cup \{ta\}$
25: *To_Place* := *To_Place* $\setminus \{t\}$
26: /* **Step 4**: re-allocate tuples in under-quota groups and delete unassigned tuples */
27: *To_Empty* := $\{g \in \mathsf{GID}_i, i = 1, \ldots, m : 0 < |\{ta \in A : ta[i]=g\}| < k_i\}$ /* set of non-empty under-quota groups */
28: *To_Place* := *To_Place* $\cup$ **Re_Assign**(*To_Empty*)
29: **for each** $F_i \in \mathcal{F}$ **do** /* remove non-assigned tuples */
30: **for each** $t \in$ *To_Place* **do** $f_i := f_i \setminus \{t[F_i]\}$

Fig. 8. Heuristic algorithm that computes a k-loose association

The algorithm starts by initializing variable *To_Place*, representing the set of tuples that still need to be allocated to groups, to the tuples in s, and by creating an empty group association A (lines 1–2). The algorithm then operates in four steps as follows.

Step 1: Ordered grouping. For each fragment F_i, the algorithm identifies the optimal grouping, allocating tuples with similar values for the attributes in $\mathcal{A} \cap F_i$ to the same group(s). To this purpose, the algorithm orders the tuples in s according to the attributes in $\mathcal{A} \cap F_i$ (line 5). It then partitions the ordered tuples in sets of k_i contiguous tuples each, and assigns to each partition a different group identifier $g^i_{current}$ (lines 6–13). In this way, contiguous tuples in the ordered relation are ideally assigned to the same group (or to contiguous groups). The result of this step is a matrix *OptimalGrouping* with one row

for each tuple t in s, one column for each fragment F in $\mathcal{F}$, and where each cell *OptimalGrouping*$[t][F_i]$ contains the identifier of the optimal group for tuple t in fragment F_i.

Step 2: Under-quota grouping. The algorithm tries to assign each tuple to the group closest to the optimal one that satisfies all the heterogeneity properties, but without generating *over-quota* groups (i.e., groups in F_i with more than k_i tuples) to maximize utility. In fact, large groups limit the utility that can be obtained in query evaluation. For each tuple t in *To_Place*, the algorithm calls function **Find_Assignment** in Figure 9 (lines 15–16), which allocates tuples to groups according to the heterogeneity properties.

Function **Find_Assignment** receives as input a tuple t, the candidate tuple *assoc_tuple* that represents t in the group association A (which is NULL when t has not been assigned to any group), a fragment identifier i, a Boolean variable *over_quota* (which is TRUE only if over-quota groups are permitted), and the array *OptimalGrouping* computed in Step 1. This function then tries to assign t to a group in F_i close to the optimal one. Function **Find_Assignment** first checks whether t can be inserted into *OptimalGrouping*$[t][F_i]$, that is, it checks whether the heterogeneity properties are satisfied. If the heterogeneity properties are not satisfied, the function checks the groups of fragment F_i, denoted $g^i_{candidate}$, in increasing order of distance from *OptimalGrouping*$[t][F_i]$ (lines 3–13). In fact, similar values are ideally assigned by Step 1 to groups close to the optimal one. The satisfaction of the heterogeneity properties is verified by calling function **Try_Assignment** in Figure 9 (line 9 and line 13). Function **Try_Assignment** takes as input tuple t, the candidate tuple *assoc_tuple* that represents t in the group association A, the fragment identifier i, and a group identifier g, and returns TRUE if $t[F_i]$ can be assigned to g; FALSE, otherwise. When a correct assignment of t to a group in F_i is found, if F_i is the last fragment in $\mathcal{F}$, function **Find_Assignment** returns tuple *assoc_tuple* representing the computed assignment of t to groups (lines 14–15). Otherwise, function **Find_Assignment** recursively calls itself to assign t to a group in fragment F_{i+1} (line 16). If the recursive call succeeds (i.e., it returns a group association for t), function **Find_Assignment** returns *assoc_tuple*; it tries to allocate t to a group at higher distance from *OptimalGrouping*$[t][F_i]$, otherwise (line 18–19). If t cannot be assigned to any group in F_i, the function returns NULL (line 20).

If function **Find_Assignment** returns a tuple *ta*, the algorithm inserts *ta* into the group association A and removes t from *To_Place* (lines 17–19).

Step 3: Over-quota grouping. If Step 2 could not allocate all the tuples in s, the algorithm tries to allocate the remaining tuples in *To_Place* to the existing groups, thus possibly creating over-quota groups. To this purpose, for each tuple t in *To_Place*, the algorithm calls function **Find_Assignment** with variable *over_quota* set to TRUE. The algorithm updates A and *To_Place* according to the result returned by function **Find_Assignment** (lines 20–25).

Step 4: Re-assignment. Once tuples in s (or a subset thereof) have been allocated to groups, the algorithm determines the set *To_Empty* of groups generated by Steps 2-3 that are under-quota, that is, the groups that do not include the minimum number of tuples necessary to provide privacy guarantees (line 27). The algorithm then calls function **Re_Assign** in Figure 9 (line 28).

Function **Re_Assign** receives as input the set *To_Empty* of non-empty but under-quota groups, and tries to reallocate their tuples to other groups (lines 30–38). Tuples in under-quota groups that cannot be reallocated will be removed from the fragmentation and are inserted into *To_Remove* (lines 39–40). When a tuple t is inserted into *To_Remove*, the corresponding tuple *ta* in A is removed from the group association and, for each fragment F_l, $\mathcal{G}_l(t)$ is set to NULL (lines 44–45). Due to the removal of t, the group g^l to which t belong in F_l loses a tuple and it might become under-quota with the consequence

FIND_ASSIGNMENT(t, *assoc_tuple*, i, *over_quota*, *OptimalGrouping*)

if *over_quota* = FALSE **then** *max* := k_i **else** *max* := $2k_i$ /* maximum number of tuples in groups */
allocated := FALSE /* variable that is TRUE if t has been allocated to a group in F_i */
g_j^i := *OptimalGrouping*[t][F_i] /* optimal allocation for t in F_i */
distance := 0 /* distance from the optimal allocation */
num_groups := $\left\lceil \frac{|s|}{k_i} \right\rceil$ /* number of groups for F_i */
while *distance*$<\lfloor \frac{1}{2}$*num_groups* $\rfloor$ **do**
  candidate := (*j* + *distance*) $\mod$ *num_groups* /* candidate allocation */
  if $|\{ta \in A : ta[i]{=}g^i_{candidate}\}| <$ *max* **then** /* the candidate group can allocate other tuples */
    allocated := **Try_Assignment**(t, *assoc_tuple*, i, $g^i_{candidate}$)
  if *allocated*=FALSE $\wedge$ *distance*$\neq 0$ **then**
      candidate := (*j* − *distance*) $\mod$ *num_groups* /* alternative candidate allocation */
      if $|\{ta \in A : ta[i]{=}g^i_{candidate}\}| <$ *max* **then**
        allocated := **Try_Assignment**(t, *assoc_tuple*, i, $g^i_{candidate}$)
  if *allocated* = TRUE **then**
    if $i = |\mathcal{F}|$ **then return**(*assoc_tuple*) /* the tuple has been allocated to a group in each fragment */
    assoc_tuple := **Find_Assignment**(t, *assoc_tuple*, $i+1$, *over_quota*, *OptimalGrouping*) /* recursive call for F_{i+1} */
    if *assoc_tuple* $\neq$ NULL **then return**(*assoc_tuple*)
  $\mathcal{G}_i(t)$:= NULL /* try a different allocation at higher distance from the optimal one */
  distance := *distance* + 1
return(NULL)

TRY_ASSIGNMENT(t, *assoc_tuple*, i, g)

assoc_tuple[i] := g /* insert t into group g for fragment F_i*/
$\mathcal{G}_i(t) := g$
if g satisfies Definition 4.4 $\wedge$ /* group heterogeneity */
  $A \cup$ {*assoc_tuple*} satisfies Definition 4.5 $\wedge$ /* association heterogeneity */
  $A \cup$ {*assoc_tuple*} satisfies Definition 4.6 /* deep heterogeneity */ **then**
  return(TRUE) /* correct assignment */
return(FALSE)

RE_ASSIGN(*To_Empty*)

To_Remove := $\emptyset$ /* tuples that need to be removed from the fragmentation */
while *To_Empty* $\neq \emptyset$ **do**
    g^i := **Extract_Group**(*To_Empty*)
    let i be the fragment F_i of g^i
    for each *ta*$\in A$ s.t. *ta*[i]=g^i **do**
      let $t \in s$ be the tuple represented by *ta*
      CandidateGroups := $\{g \in \mathsf{GID}_i : |\{ta \in A : ta[i]{=}g\}| \geq k_i\}$ /* set of over-quota groups of F_i */
      allocated := FALSE
      while (*allocated*=FALSE) $\wedge$(*CandidateGroups*$\neq\emptyset$) **do** /* try to re-assign the semi-tuple */
        g^i_j := **Extract_Group**(*CandidateGroups*) /* candidate group */
        allocated := **Try_Assignment**(t, *ta*, i, g^i_j)
      if *allocated* = FALSE **then** /* if the reassignment failed, the tuple must be removed */
        To_Remove := *To_Remove* $\cup \{t\}$
        for $l = 1 \ldots |\mathcal{F}|$ **do**
          g^l := *ta*[l] /* check whether the removal of t generates new under-quota groups */
          if $|\{ta \in A : ta[l]{=}g^l\}| <$ k_l+1 **then** *To_Empty* := *To_Empty* $\cup \{g^l\}$
          ta[l] := NULL
          $\mathcal{G}_l(t)$:= NULL
return(*To_Remove*)

Fig. 9. Pseudocode of functions **Find_Assignment**, **Try_Assignment**, and **Re_Assign**

that it should be removed. If this is the case, g^l is inserted into *To_Empty* (line 43). Function **Re_Assign** returns the set *To_Remove* of tuples to be removed from the fragmentation (line 46).

The algorithm then deletes from each fragment both the tuples that have never been assigned to groups and the tuples returned by function **Re_Assign** (lines 29–30).

The utility of the k-loose association computed by this heuristic algorithm as well as its efficiency are evaluated in Section 7.

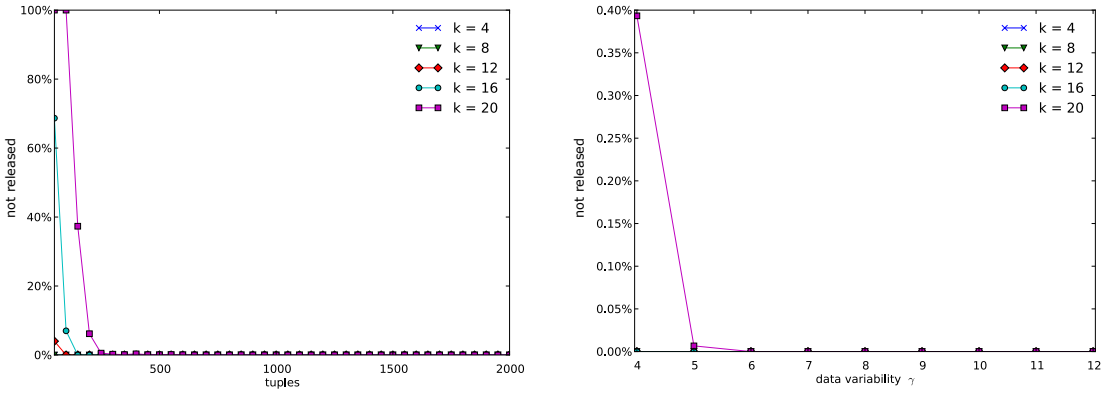
7. Coverage, performance, and utility

We implemented a prototype, written in Python, of the algorithm described in the previous section, and ran several sets of experiments to the aim of evaluating the ability of our approach to compute a k-loose association, while limiting the number of suppressed tuples (i.e., tuples that cannot be included in any group), and of assessing its performance (Section 7.2). We then analyzed the utility provided in query evaluation (Section 7.3). We now present the experimental setting (Section 7.1) and then discuss the experimental results.

7.1. Experimental setting

We considered both synthetic and real-world datasets. Synthetic data allow us to fully control all the parameters used for data generation, such as the variability in the distribution of attributes values, leading to a robust analysis of the behavior of our technique. Real data allow us to assess the applicability of our technique in a concrete setting. Synthetic datasets were generated starting from a relation schema composed of 7 attributes PATIENTS(`Name`, `YoB`, `Education`, `ZIP`, `MarStatus`, `Disease`, `Salary`), split over two fragments (F_l=$\{$`Name`, `MarStatus`, `Salary`$\}$ and F_m=$\{$`YoB`, `Education`, `ZIP`, `Disease`$\}$), to satisfy confidentiality constraints c_1=$\{$`Name`, `ZIP`, `MarStatus`, `Disease`$\}$ and c_2=$\{$`Name`, `YoB`, `Education`, `MarStatus`$\}$. The datasets were randomly generated adopting distinct characterizing parameters for each attribute. A statistical correlation was introduced between `Salary` and `Education`; all the other attributes were set using independent distributions. This allowed us to have knowledge of information in the protected data that we were interested in retrieving through the query computing the average `Salary` of patients with the same `Education` level.

In our experiments we considered, as a base configuration, a dataset including $10,000$ tuples. We analyzed the behavior of the system varying several parameters. First, we considered the impact of variations of k, with values ranging between 4 and 20 (k was equal to 12 for experiments that did not change this parameter). Then, we considered changes on a parameter γ that drives the generation of the synthetic dataset, guiding the distribution of the attribute values. Low values of γ produce compact ranges of values for all the attributes and a high probability of similarity among tuples; high values of γ produce values for the attributes covering a wider range, with small similarity among tuples. The interval we considered in the experiments is between 4 and 12 (value 8 was used in experiments that did not consider variations of this parameter). Finally, we considered the impact of the variations of parameters k_l and k_m and, always choosing pairs of values such that $k_l \cdot k_m = k$, we considered several possible pairs (in experiments that did not consider variations of these parameters, we chose the pair k_l and k_m that had $k_l \geq k_m$ and minimum distance between k_l and k_m; e.g., when k=12, k_l=4 and k_m=3). As a real world dataset, we considered the IPUMS dataset [23], which has been widely used in the literature to test anonymization approaches. Among the attributes in the dataset, we considered the projection over attributes $\{$`Region`, `Statefip`, `Age`, `Sex`, `MarSt`, `Ind`, `IncWage`, `IncTot`, `Educ`, `Occ`, `HrsWork`, `Health`$\}$, repre-



(a) (b)

Fig. 10. Percentage of tuples in the original relation that are not released, varying the number of tuples in the relation (a) and the variability in the distribution of attribute values (b)

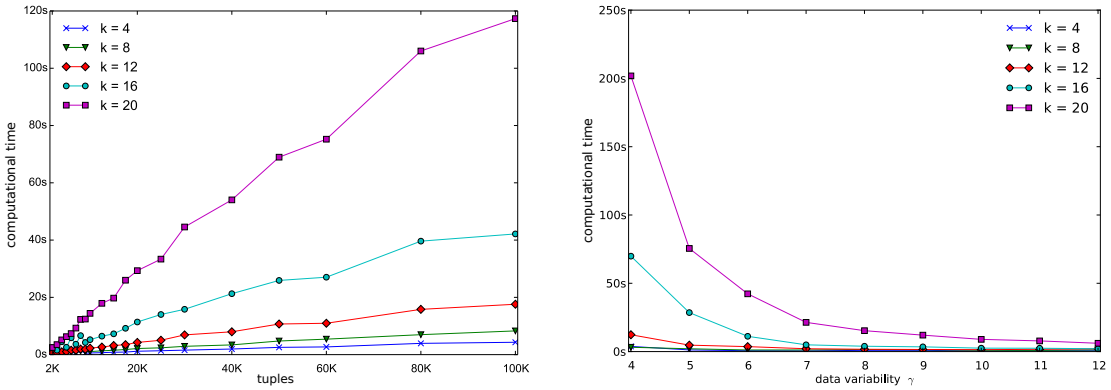
senting for each citizen: the region where she lives, the state where she lives, age, sex, marital status, the type of industry for which she works, salary, annual total income, education level, occupation, the number of hours she works per week, and health status rated on a five point scale. The relation includes $95,000$ tuples with a not null value for `IncWage` attribute. When considering the real dataset, we ran experiments over two fragmentations: the first one is composed of two fragments F_l={`Region`, `Statefip`, `Age`, `Sex`, `MarSt`, `Ind`, `IncWage`, `IncTot`} and F_m={`Educ`, `Occ`, `HrsWork`, `Health`} satisfying constraints c_1={`Statefip`, `Ind`, `Educ`, `Occ`, `Health`} and c_2={`Age`, `Sex`, `MarSt`, `Educ`, `Occ`, `Health`}; the second fragmentation has three fragments F_l={`Region`, `Statefip`, `Ind`, `IncWage`}, F_m={`Age`, `Sex`, `MarSt`, `IncTot`}, and F_r={`Educ`, `Occ`, `HrsWork`, `Health`} satisfying constraints c_1={`Statefip`, `Ind`, `Educ`, `Occ`, `Health`}, c_2={`Age`, `Sex`, `MarSt`, `Educ`, `Occ`, `Health`}, and c_3={`Statefip`, `Age`, `Sex`, `MarSt`, `Ind`}. Experiments have been run on a server with two Intel(R) Xeon(R) E5504 2.00GHz, 12GB RAM, one 240GB SSD disk, and Ubuntu 12.04 LTS 64bit operating system. The reported results have been computed as the average of a minimum of 5 (for the largest configurations) and a maximum of 120 (for more manageable configurations) runs of the same experiment.

7.2. Coverage and performance

We ran a first set of experiments on synthetic data aimed at assessing the coverage of the solution computed by our algorithm, that is, the number of tuples of the original relation that could not be published as they could not be allocated to any group without violating k-looseness. The experiments focused on evaluating how the number of tuples in the relation (Figure 10(a)) and the variability in the distribution of attribute values (Figure 10(b)) have an impact on the number of tuples that cannot be released, for different values of k.

Figure 10(a) shows that the algorithm is more likely to suppress tuples when operating over small datasets, as the number of candidate groups in each fragment is small. It is then harder to find an assignment for each tuple that satisfies all the heterogeneity properties. As it can be expected, the percentage of suppressed tuples grows with k, since it is harder to define larger groups of tuples (especially for small datasets).

Figure 10(b) illustrates the impact of the variability in the distribution of attribute values on the number of suppressed tuples. As visible from the figure, datasets characterized by low variability cause higher



(a) (b)

Fig. 11. Computational time necessary to determine a k-loose association, varying the number of tuples in the relation (a) and the variability in the distribution of attribute values (b)

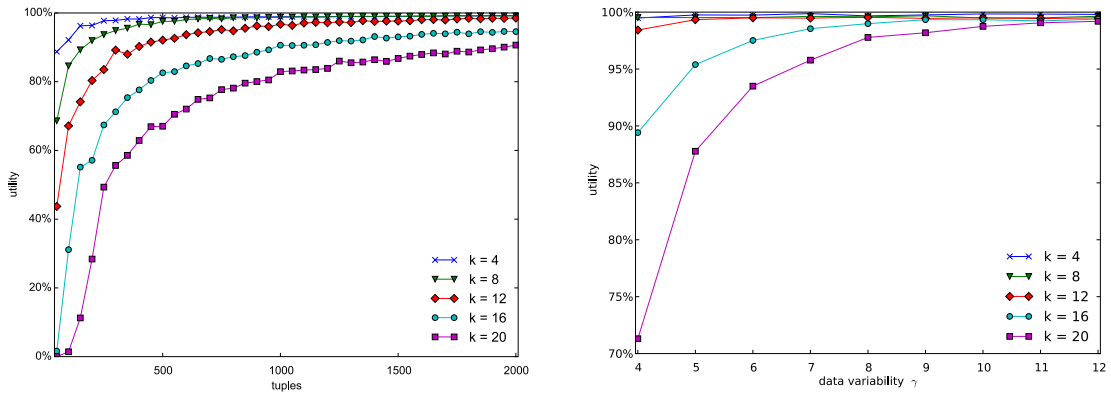
suppression. This is due to the fact that it is hard to assign tuples to groups when there is a higher probability that some of them have the same values for attributes in relevant confidentiality constraints. Indeed, two tuples with the same values for attributes in constraints cannot be assigned to the same group. In the experiments performed on the IPUMS dataset, no tuple has been suppressed.

A second set of experiments on synthetic data evaluated the impact of the size of the original relation and of the variability in the distribution of attribute values on the performance of our algorithm.

To prove the scalability of our approach, we ran our algorithm with large instances of the original relation, with a number of tuples varying between $2,000$ and $100,000$. Figure 11(a) illustrates the time necessary to compute a k-loose association and confirms the scalability of the proposed approach. In fact, our prototype is able to find a k-loose association for relations with $100,000$ tuples in less than one minute for k=16 (and we speculate that, according to publicly available Python/C performance ratios [16], an optimized C implementation would take less than one second).
Figure 11(b) illustrates the impact of the variability in the distribution of attribute values on the time necessary to compute a k-loose association, considering configurations with $10,000$ tuples. The figure confirms that, as already noted, the lower the variability, the harder the task to find a k-loose association. Both Figure 11(a) and Figure 11(b) also show that the computational time grows with the protection degree offered by the k-loose association: higher values of k require a higher computational cost.
In the experiments on the IPUMS dataset, our algorithm always computed a solution in less than 90 seconds.

7.3. Utility

We ran a set of experiments specifically focused on assessing the gain provided by loose associations, in terms of the utility of query results. We used as a reference the query that identifies the relationship between the `Education` level of patients and their `Salary` (i.e., SELECT AVG(`Salary`) FROM PATIENTS GROUP BY `Education`). We then defined a k-loose association that aims at keeping in the same group patients with the same `Education` level in fragment F_m and with similar `Salary` in fragment F_l.



(a) (b)

Fig. 12. Utility provided by a k-loose association, varying the number of tuples in the relation (a) and the variability in the distribution of attribute values (b)

Figure 12(a) compares the utility provided by the release of a k-loose association with different values for k, varying the number of tuples in the input dataset. The figure clearly shows that the release of a k-loose association permits to obtain high utility in query evaluation. In most of the considered configurations, utility is nearly 100%, meaning that the result computed over fragmented data complemented with loose associations is nearly the same obtained on the original relation. This figure also confirms that the quality of the loose association computed by our algorithm improves with the number of tuples in the dataset, as it becomes easier to have tuples with similar values for `Education` and `Salary` in the same group.

Figure 12(b) illustrates the impact of the variability in the distribution of attribute values on the obtained utility. Greater values of γ increase the variability and lead to a reduction in the probability for an attribute to present the same values in different tuples. Conflicts arise in a group when tuples present the same values on the attributes involved in a constraint. Then, the probability of conflicts decreases as γ increases. The utility provided by the release of a k-loose association is always high, and increases as the variability in the attribute values increases. Our experiments also clearly show that, in line with the observation that utility and privacy are two contrasting requirements, utility decreases as k increases. It is then expected that improvements in confidentiality guarantees of the solution correspond to worsening in the utility of the released data. It is however worth noticing that, also for the worst case in which k=20, if the size of the input dataset is not too limited (i.e., in the order of hundreds of tuples) the measured utility was higher than 80%, implying a high utility in query evaluation also when adopting privacy parameters higher than the values we expect to be used in real-world scenarios.

We ran a second set of experiments for evaluating the impact of keeping in the same group similar values for an attribute (or a set thereof) of interest for query evaluation (see Section 5). To this aim, we compared the utility of queries q_{os} (operating on both `Education` and `Salary`), q_{as} (operating only on `Education` or only on `Salary`), and q_{ns} (operating on neither `Education` nor `Salary`). Figure 13 compares the utility obtained with 7 different queries (query q_0 as representative for q_{os}, queries q_1, q_2, q_3 for q_{as}, and queries q_4, q_5, q_6 for q_{ns}) with k=12, varying k_l and k_m. Each query is represented by a group of bars, where each bar presents the utility obtained with one of the configurations for parameters k_l and k_m. The results clearly show that the query with highest utility (almost 100%) is q_0, which benefits from the ordering over both `Education` and `Salary` (i.e., query q_{os}). Queries q_2,q_3, and q_4 take advantage

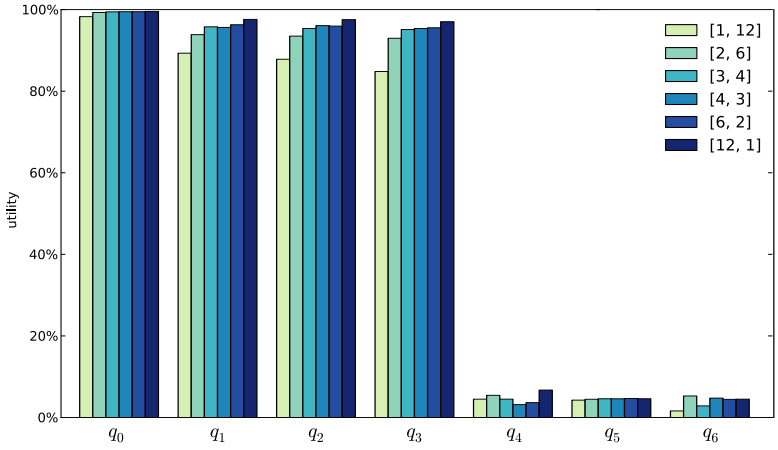
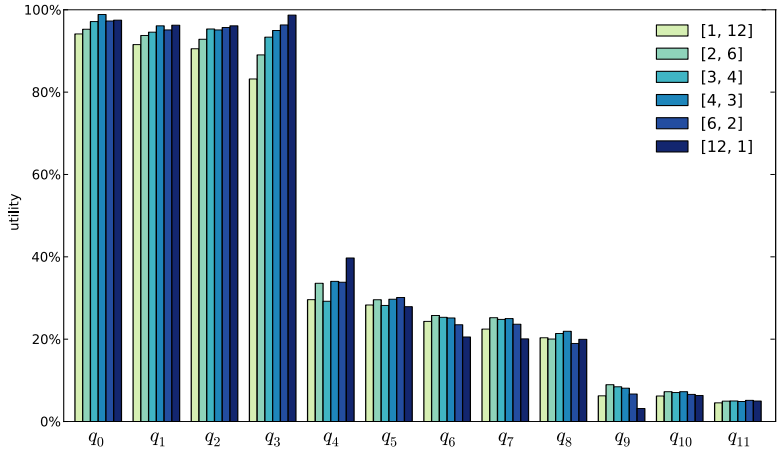


Fig. 13. Utility provided by a k-loose association with ordering on `Education` and `Salary`

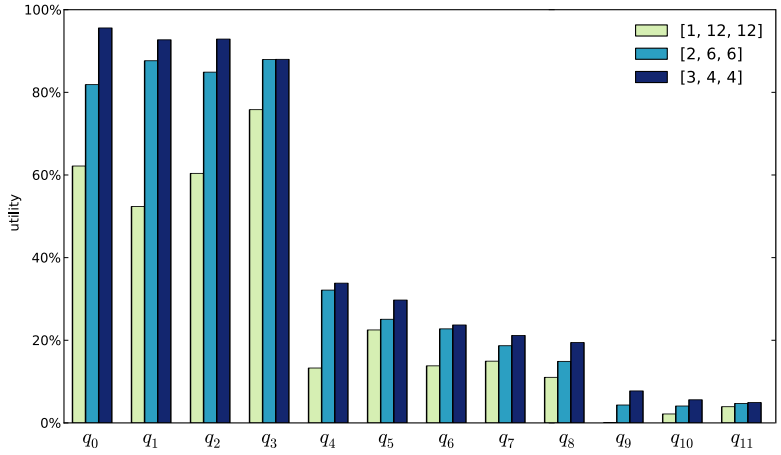
only of the ordering over one attribute, which however permits to obtain utility higher than 80% in all the considered configurations. Our experimental evaluation shows that the release of a loose association provides limited, but not null, utility also for queries q_{ns}. We can then conclude that keeping in the same group tuples with similar values permits to achieve better results in the utility of query evaluation.

In the experiments performed on the IPUMS dataset, we first defined a k-loose association between fragments F_l and F_m, and identified two representative sets of queries. The first set of queries operates on the attributes on which fragments of the loose association have been ordered (i.e., q_{os}, represented by query q_0, and q_{as}, represented by queries q_1, q_2, q_3). The second set of queries instead operates only on attributes different from those on which the ordering was performed (i.e., q_{ns}, represented by queries $q_4, \ldots, q_{11}$). Figure 14(a) illustrates the utility obtained in executing these two sets of queries over a k-loose association with k=12 and varying the values of k_l and k_m. Each query is represented by a group of bars in the figure. Queries involving at least one of the attributes on which the ordering has been performed (i.e., queries $q_0, \ldots, q_3$) showed excellent utility in the result, close to 100% for all queries. As expected, the utility in executing queries operating on unordered attributes only (i.e., queries $q_4, \ldots, q_{11}$) is lower. It is however worth noticing that the results are still appreciable, with most of the queries showing utility between 20% and 35%. Compared to the experiments on the synthetic dataset, the queries of type q_{ns} exhibit better utility. In particular, Figure 14(a) shows that queries q_{as} involving both ordered and unordered attributes exhibit utility values similar to query q_0 that involves ordered attributes only. Our explanation for the higher utility obtained on IPUMS dataset is that real data are more structured and present greater regularity and correlations among attribute values than the randomly generated data in our synthetic dataset, which is characterized by one correlation only (the one between attributes `Education` and `Salary`).

Figure 13 and Figure 14(a) describe configurations that, keeping k constant, progressively increase the value of parameter k_l (and reduce k_m accordingly). The utility remains relatively stable across all these configurations, even if we can see that, for some queries, the utility decreases as k_l increases, while for a few queries the utility grows with k_l. Overall, we consider as preferable the intermediate solutions, with k_l and k_m near to $\sqrt{k}$, because they are not associated with the lowest levels of utility and because this criterion offers strong benefit when applied to fragmentations with more than two fragments, as observed in the following.



(a)

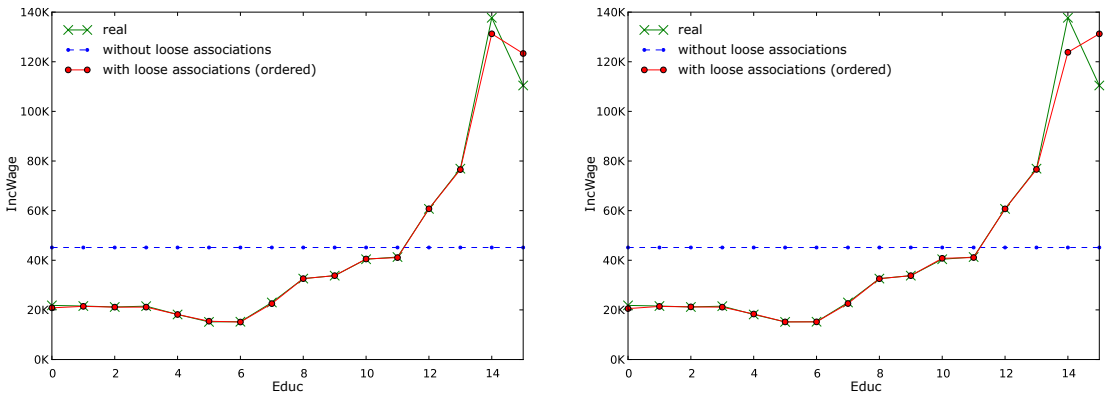


(b)

Fig. 14. Utility provided by a k-loose association over two (a) and three (b) fragments

To further evaluate our technique, we performed the same experimental analysis on the fragmentation of IPUMS dataset composed of three (rather than two) fragments. Figure 14(b) reports the utility of queries provided by the release of a k-loose association with k=12, comparing the results obtained with a (1,12,12)-, (2,6,6)- and (3,4,4)-grouping. The crucial difference among these configurations is that queries that combine the second and third fragment will operate on a k-loose association between the two fragments with k equal to 144, 36, and 16 (for the constraints that are relevant for the second and third fragment). As we already noticed, an increase in k leads to a reduction in utility. The graph in Figure 14(b) clearly shows the increase in utility that occurs going from (1,12,12)- to (3,4,4)-grouping for queries $q_0, \ldots, q_3$. This result proves that, for fragmentations with more than two fragments, there is a significant utility benefit in building loose associations with similar values for parameter k_i on all the fragments.

Figure 15 demonstrates in a different way the utility that can be obtained by the use of loose associations, analyzing query q_0 of the previous experiments. The graph has on the x-axis the different values



(a) (b)

Fig. 15. Query results computed ordering on `Educ` and `IncWage` on a k-loose association among two (a) and three (b) fragments

of attribute `Educ`, which represents the number of years of education reported in the census, and on the y-axis the average salary (attribute `IncWage`) for the cohort of people with that level of education. In essence, the graphs plot the result of query q_0="SELECT AVG(`IncWage`) FROM IPUMS_CENSUS GROUP BY `Educ`". The continuous green line (i.e., the line labeled "real") reports the results obtained on the real data. The dashed blue horizontal line (i.e., the line labeled "without loose associations") in the middle is the result that we can obtain without loose associations, because `IncWage` and `Educ` belong to different fragments and the global average salary will be returned for every education level. The continuous red line (i.e., the line labeled "with loose associations (ordered)") describes the result obtained with a k-loose association with k=12. Figure 15(a) shows the results obtained in the configuration with two fragments and assuming a (4,3)-grouping, while Figure 15(b) illustrates the results obtained in the configuration with three fragments and assuming a (3,4,4)-grouping. In both graphs, the green and red lines are overlapping over most of the range, clearly showing the utility that can be obtained by loose associations.

8. Related work

Several research efforts have addressed the problem of protecting privacy in data publishing, proposing approaches based on either sanitizing (e.g., [9,15,17,20,21,22,24,30]) or fragmenting data (e.g., [1,2,7, 8,10,11]) before their release. Our approach bears some similarity with k-anonymity [26] for the notion of grouping and more similarity with ℓ-diversity [22] for the consideration of the different values that are involved in the sensitive associations. Apart from this, our problem and solution are different from k-anomymity-like approaches, which are typically based on generalization of quasi-identifying attributes. By contrast, in our approach fragments contain the attribute values that appear in the original relation, while it is the association among fragments that is obfuscated. Most fragmentation works, although showing similarities with our proposal, address an orthogonal problem with respect to the one considered in this paper, as they aim at breaking sensitive associations while maximizing the ability of recipients of evaluating queries on fragments. Recent proposals have also considered the problem of improper information leakage due to data dependencies, which can be exploited to link different fragments [3,13].

Our loose associations can nicely complement these proposals, thus increasing the utility of fragmented data and ensuring proper protection of confidentiality constraints also in presence of data dependencies.

The works closest to our complement published (fragmented) data with information on their association, without disclosing sensitive information [11,14,31]. Our proposal is however more general than these solutions, since they operate on two fragments (or views/tables) only, while we consider an arbitrary number of fragments when defining loose associations. The work in [11] does not take into consideration the possible existence of duplicate values for non-key attributes. Sensitive associations on non-key attributes are therefore exposed to frequency-based attacks. Our heterogeneity properties overcome this issue by preventing the presence of duplicates in the same group of tuples. Anatomy [31] considers the specific problem of protecting the association between respondents' quasi-identifiers and a sensitive attribute while our solution permits to protect any association among attributes. Anatomy also partitions the original relation in groups of ℓ tuples before splitting attributes in two disjoint fragments. Our approach provides more flexibility in defining groups of tuples because for each fragment F_i we can consider a different privacy parameter k_i (and a different grouping function) instead of a single ℓ value. By comparing Anatomy with a loose association on two fragments, we can see that the two techniques provide the same protection guarantees and the same utility [14]. Anatomy can then be considered as a specific instance of our approach, where the original relation is partitioned in two fragments: F_l, storing the quasi-identifier, F_r, storing the sensitive attribute, and k_l=1 and k_r=ℓ (or viceversa).

Our work may also bring some resemblance with the proposals in [6,27,28]. The work in [28] adopts horizontal and vertical fragmentation to protect privacy of sparse multidimensional data (e.g., transactional data). The approach in [6] focuses instead on protecting recommendation data expressed by customers (i.e., Netflix movie ratings). Besides operating on different data models, both these proposals differ from our work since they are specifically targeted at protecting respondents' identities and their association with sensitive attributes. Also, they both adopt a dual approach with respect to loose associations, requiring homogeneity of values in fragments. The work in [27] addresses the problem of destroying the correlation between two disjoint subsets of attributes, preserving as much as possible the other correlations. Our approach does not aim at destroying correlations among attributes, as our goal is to preserve them as much as possible, while satisfying privacy constraints. Also, the solution in [27] adopts masking techniques, while our approach maintains data truthfulness.

Alternative approaches to protect privacy in data release are based on differential privacy (e.g., [15, 17]). Although addressing a similar problem, approaches based on differential privacy are not directly applied since they introduce noise in the dataset that depends on the expected users queries. Our approach instead aims at releasing truthful data.

A different line of work is represented by Controlled Query Evaluation (CQE) (e.g., [2,4]). These solutions provide the data owner storing a data collection with a technique for controlling whether users' queries should be permitted or denied depending on both confidentiality constraints and the history of past interactions. These approaches are not applicable to our scenario, since we publicly release a dataset.

9. Conclusions

We addressed the problem of extending loose associations to operate on multiple fragments. We first described how the publication of multiple loose associations between pairs of fragments can expose sensitive associations, and then presented an approach supporting the definition of a loose association among an arbitrary number of fragments. We also described a heuristic algorithm for the computation of

a loose association, and showed the results of an extensive experimental evaluation aimed at analyzing both the efficiency and the effectiveness of the proposed heuristics as well as the utility provided by loose associations in query execution. Our work leaves space for further investigations, including the consideration of external knowledge that data recipients can exploit to reconstruct sensitive associations among attributes in different fragments, and of dynamic datasets, where data in the original relation change over time.

Acknowledgements

The authors would like to thank Enrico Bacis for support in the implementation of the system and in the experimental evaluation. This work was supported in part by: the EC within the 7FP under grant agreement 312797 (ABC4EU) and within the H2020 program under grant agreement 644597 (ESCUDO-CLOUD); the Italian Ministry of Research within PRIN project "GenData 2020" (2010RTFWBH); and the NSF under grant IIP-1266147.

References

[1] G. Aggarwal et al. Two can keep a secret: A distributed architecture for secure database services. In *Proc. of CIDR 2005*, Asilomar, CA, USA, January 2005.

[2] J. Biskup. Dynamic policy adaptation for inference control of queries to a propositional information system. *JCS*, 20(5):509–546, 2012.

[3] J. Biskup and M. Preuß. Database fragmentation with encryption: Under which semantic constraints and a priori knowledge can two keep a secret? In *Proc. of DBSec 2013*, Newark, NJ, USA, July 2013.

[4] J. Biskup, M. Preuß, and L. Wiese. On the inference-proofness of database fragmentation satisfying confidentiality constraints. In *Proc. of ISC 2011*, Xi'an, China, October 2011.

[5] A. Ceselli, E. Damiani, S. De Capitani di Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Modeling and assessing inference exposure in encrypted databases. *ACM TISSEC*, 8(1):119–152, February 2005.

[6] C.-C. Chang, B. Thompson, H. (W.) Wang, and D. Yao. Towards publishing recommendation data with predictive anonymization. In *Proc. of ASIACCS 2010*, Beijing, China, April 2010.

[7] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Combining fragmentation and encryption to protect privacy in data storage. *ACM TISSEC*, 13(3):22:1–22:33, July 2010.

[8] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Selective data outsourcing for enforcing privacy. *JCS*, 19(3):531–566, 2011.

[9] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, and P. Samarati. k-Anonymous data mining: A survey. In C.C. Aggarwal and P.S. Yu, editors, *Privacy-Preserving Data Mining: Models and Algorithms*. Springer-Verlag, 2008.

[10] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Fragmentation and encryption to enforce privacy in data storage. In *Proc. of ESORICS 2007*, Dresden, Germany, September 2007.

[11] G. Cormode, D. Srivastava, T. Yu, and Q. Zhang. Anonymizing bipartite graph data using safe groupings. *The VLDB Journal*, 19(1):115–139, February 2010.

[12] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, G. Livraga, S. Paraboschi, and P. Samarati. Extending loose associations to multiple fragments. In *Proc. of the DBSec 2013*, Newark, NJ, USA, July 2013.

[13] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, G. Livraga, S. Paraboschi, and P. Samarati. Fragmentation in presence of data dependencies. *IEEE TDSC*, 2014. (pre-print).

[14] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Fragments and loose associations: Respecting privacy in data publishing. *PVLDB*, 3(1):1370–1381, September 2010.

[15] S. De Capitani di Vimercati, S. Foresti, G. Livraga, and P. Samarati. Protecting privacy in data release. In A. Aldini and R. Gorrieri, editors, *Foundations of Security Analysis and Design VI*. Springer, 2011.

[16] Ubuntu: Intel Q6600 one core – computer language benchmarks game. `http://benchmarksgame.alioth.debian.org/u32/performance.php?test=nbody`.

[17] C. Dwork. Differential privacy. In *Proc. of ICALP 2006*, Venice, Italy, July 2006.

[18] C. Gentry and S. Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In *Proc. of EUROCRYPT 2011*, Tallinn, Estonia, May 2011.

[19] R. Jhawar, V. Piuri, and P. Samarati. Supporting security requirements for resource management in cloud computing. In *Proc. of CSE 2012*, Paphos, Cyprus, December 2012.

[20] D. Kifer and J. Gehrke. Injecting utility into anonymized datasets. In *Proc. of SIGMOD 2006*, Chicago, IL, USA, June 2006.
[21] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and ℓ-diversity. In *Proc. of ICDE 2007*, Istanbul, Turkey, April 2007.
[22] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. ℓ-Diversity: Privacy beyond k-anonymity. *ACM TKDD*, 1(1):3:1–3:52, March 2007.
[23] Minnesota Population Center. IPMUS-USA (Integrated Public Use Microdata Series). http://www.ipums.org.
[24] T. Raeder, M. Blanton, N.V. Chawla, and K. Frikken. Privacy-preserving network aggregation. In *Proc. of PAKDD 2010*, Hyderabad, India, June 2010.
[25] P. Samarati. Data security and privacy in the cloud In *Proc. of ISPEC 2014*, Fuzhou, China, May 2014.
[26] P. Samarati. Protecting respondents' identities in microdata release. *IEEE TKDE*, 13(6):1010–1027, November 2001.
[27] Y. Tao, J. Pei, J. Li, X. Xiao, K. Yi, and Z. Xing. Correlation hiding by independence masking. In *Proc. of ICDE 2010*, Long Beach, CA, USA, March 2010.
[28] M. Terrovitis, N. Mamoulis, J. Liagouris, and S. Skiadopoulos. Privacy preservation by disassociation. *PVLDB*, 5(10):944–955, June 2012.
[29] J. Vaidya. A survey of privacy-preserving methods across vertically partitioned data. In C.C. Aggarwal and P.S. Yu, editors, *Privacy-Preserving Data Mining: Models and Algorithms*. Springer-Verlag, 2008.
[30] K. Wang and B. C. M. Fung. Anonymizing sequential releases. In *Proc. of KDD 2006*, Philadelphia, PA, USA, August 2006.
[31] X. Xiao and Y. Tao. Anatomy: Simple and effective privacy preservation. In *Proc. of VLDB 2006*, Seoul, Korea, September 2006.