 DOI 10.1109/TDSC.2013.2295798

# Fragmentation in presence of data dependencies

Sabrina De Capitani di Vimercati, *Senior Member, IEEE,* Sara Foresti, *Member, IEEE,* Sushil Jajodia, *Fellow, IEEE,* Giovanni Livraga, *Member, IEEE,* Stefano Paraboschi, *Member, IEEE,* Pierangela Samarati, *Fellow, IEEE*

**Abstract**—Fragmentation has been recently proposed as a promising approach to protect the confidentiality of sensitive associations whenever data need to undergo external release or storage. By splitting attributes among different fragments, fragmentation guarantees confidentiality of the associations among these attributes under the assumption that such associations cannot be reconstructed by re-combining the fragments. We note that the requirement that fragments do not have attributes in common, imposed by previous proposals, is only a necessary, but not sufficient, condition to ensure that information in different fragments cannot be recombined as dependencies may exist among data enabling some form of linkability. In this paper, we identify the problem of improper information leakage due to data dependencies, provide a formulation of the problem based on a natural graphical modeling, and present an approach to tackle it in an efficient and scalable way.

**Index Terms**—Data dependencies, data fragmentation, confidentiality, visibility requirements, CSP

## 1 Introduction

Recent years have seen a tremendous explosion of the amount of information shared, collected, processed, and externally stored or published. Outsourcing, cloud, big data have become common terms with reference to recent and emerging scenarios. A common aspect among them is the presence of data (differently collected, stored, or accessed) which is exposed to external parties, for storage, processing or even release. Hence, the inevitable concerns by end users, and the attention from the academic and industrial communities on possible improper exposure of sensitive information (e.g., [1], [2]).

Several techniques and approaches have been proposed for providing protection of sensitive information, tackling different aspects of the problem with varying assumptions and applicable to different scenarios. Among such proposals, data fragmentation is a promising approach for withholding sensitive attributes and breaking sensitive associations across different data fragments so to ensure their protection from unauthorized eyes (e.g., [3], [4], [5]). Fragmentation can find applicability in different scenarios, covering both cases where data are externally managed for storage and processing (as in the cloud) and cases where some views over data need to be published (as in the case of public or semi-public release as well as data sharing among different parties). In scenarios where data are externally stored or processed, fragmentation has the advantage of maintaining data in the clear (in contrast to encrypted), thus supporting a more efficient query execution and accommodating a large variety of queries. In scenarios where data are released to the public or to parties from which specific sensitive associations need to be withheld, it provides the needed data views while ensuring confidentiality of sensitive associations. The guarantee of confidentiality offered by fragmentation relies on the requirement that fragments do not have attributes in common and on the consequent assumption that this prevents their linkability. In other words, there is an implicit assumption that attributes are independent. Clearly, such an assumption does not typically hold, and often there are attributes whose values might depend on the values of other attributes to which they are related and on which therefore they can indirectly leak information. For instance, the treatment given to a patient typically depends on her disease, hence visibility of the treatment given to a patient also leaks information on her disease. The inference enabled by data dependencies can put confidentiality at risk improperly exposing sensitive attributes (as in the case where disease is sensitive), sensitive associations (as in the case where treatment appears with some attributes whose association with disease is sensitive), or actually enabling some form of linkability among fragments (as in the case where disease appears in other fragments). A successful approach to data fragmentation therefore has to take into consideration dependencies that can exist among data.

In this paper, we address this problem and extend fragmentation to the consideration of data depen-


- *S. De Capitani di Vimercati, S. Foresti, G. Livraga, and P. Samarati are with the Università degli Studi di Milano, Italy. E-mail:* firstname.lastname@*unimi.it*
- *S. Jajodia is with George Mason University, USA. E-mail: jajodia@gmu.edu*
- *S. Paraboschi is with the Università degli Studi di Bergamo, Italy. E-mail: parabosc@unibg.it*

dencies. We build our approach on existing proposals accommodating both confidentiality constraints (capturing sensitive information and associations) and visibility constraints (capturing requirements of data views) and enrich them with the consideration of data dependencies, thus providing a comprehensive modeling of the problem. As we will note, consideration of data dependencies not only increases the expressive power of the approach (enabling the specification and consideration of inferences due to relationships among data and therefore the protection against them in the fragmentation), but also simplifies the specification of confidentiality constraints (as constraints that were due to some data dependencies do not need to be determined and specified anymore).

The contribution of this paper is fourfold. First, we characterize the limitation of existing proposals identifying the problem of improper information leakage due to data dependencies. Second, we provide a natural and intuitive formulation of the problem based on its representation in terms of graphs and their coloring. Third, we identify an approach to tackle the problem that translates the (otherwise naturally recursive) control into a condition that can be verified simply on fragments without entailing recursive evaluation. Fourth, we provide a representation of the problem as a Constraint Satisfaction Problem, thus enabling exploitation of existing off-the-shelf solvers for its solution in a natural, efficient, and scalable way.

The remainder of this paper is organized as follows. Section 2 summarizes the basic concepts of the data fragmentation approach on which we build. Section 3 introduces data dependencies and illustrates the problem of improper information exposure due to them. Section 4 provides a formulation of the fragmentation problem based on an intuitive graphical representation and on graph coloring. Section 5 formally defines improper exposures due to data dependencies and the requirements that a fragmentation should satisfy to ensure that data dependencies cannot be exploited for violating the confidentiality protection offered by fragmentation. Section 6 illustrates a translation of the problem into a Constraint Satisfaction Problem and presents our experimental results. Section 7 discusses related work. Finally, Section 8 concludes the paper.

## 2 Preliminary concepts

Consistently with other proposals (e.g., [3], [4], [5], [6]), we consider a scenario where data undergoing external release or storage are organized in a single relation $r$, defined over relation schema $R(a_1, \ldots, a_n)$. In the following, when clear from the context, we use $R$ to denote either the relation schema $R$ or the set $\{a_1, \ldots, a_n\}$ of attributes composing it. We assume data in $r$ to be subject to *confidentiality* and *visibility* constraints.

Confidentiality constraints model sensitive attributes or associations among attributes in $R$, and are formally defined as follows [3].

*Definition 2.1 (Confidentiality constraint):* Let $R(a_1, \ldots, a_n)$ be a relation schema. A *confidentiality constraint* $c$ over $R$ is a subset of $\{a_1, \ldots, a_n\}$.

This definition captures both *singleton* and *association constraints*. A singleton constraint $\{a\}$ states that the values of attribute $a$ are sensitive. A non-singleton constraint $\{a_i, \ldots, a_j\}$ states that the association of values of attributes $a_i, \ldots, a_j$ (i.e., their joint visibility) is sensitive. Figure 1(b) illustrates an example of confidentiality constraints for relation PATIENTS in Figure 1(a), which state that: Social Security Numbers are sensitive ($c_1$), the association between patients' names and diseases is sensitive ($c_2$), and the association between patients' ZIP codes and insurance premiums is sensitive ($c_3$).

Visibility constraints are complementary to confidentiality constraints and model data views that should be made visible (for publication or plaintext storage). Formally, visibility constraints are defined as follows [5].

*Definition 2.2 (Visibility constraint):* Let $R(a_1, \ldots, a_n)$ be a relation schema. A *visibility constraint* $v$ over $R$ is a monotonic Boolean formula over attributes in $R$.

Visibility constraints are monotonic Boolean formulas (i.e., they contain only AND and OR connectives) since negations model requirements of non-visibility, which are already captured by confidentiality constraints. For simplicity and without loss of generality, in the following we assume visibility constraints to be expressed in disjunctive normal form. Figure 1(c) illustrates an example of visibility constraints for relation PATIENTS in Figure 1(a). Intuitively, a visibility constraint requires the inclusion or the joint inclusion of attributes within a single fragment. More precisely, visibility constraint $v{=}a$ is satisfied iff attribute $a$ belongs to a fragment (e.g., $v_4$ requires attribute `Insurance` to belong to a fragment). Visibility constraint $v{=}a_i \wedge \ldots \wedge a_j$ is satisfied iff attributes $a_i, \ldots, a_j$ belong to the same fragment (e.g., $v_3$ requires attributes `Treatment` and `Premium` to appear in the same fragment). Visibility constraint $v{=}v_i \vee \ldots \vee v_j$ is satisfied iff at least one among $v_i, \ldots, v_j$ is satisfied by a fragment (e.g., $v_1$ requires the existence of a fragment that contains either attribute `Name` or both attributes `Birth` and `ZIP`).

Confidentiality and visibility constraints are enforced by vertically fragmenting relation $R$ (e.g., [3], [4], [5], [6]). A *fragment* $F$ of a relation $R$ is a subset of the attributes in $R$. Formally, a fragmentation is defined as follows.

*Definition 2.3 (Fragmentation):* Let $R(a_1, \ldots, a_n)$ be a relation schema. A *fragmentation* $\mathcal{F}$ of $R$ is a set

PATIENTS

| SSN | Name | Birth | ZIP | Job | Insurance | Premium | Disease | Treatment |
|---|---|---|---|---|---|---|---|---|
| 123-45-6789 | Andrew | 56/12/07 | 94101 | miner | industry | 100 | silicosis | bronchodilators |
| 234-56-7654 | Bob | 79/03/01 | 94123 | miner | industry | 100 | silicosis | bronchodilators |
| 345-67-8123 | Carol | 51/11/11 | 95173 | lawyer | law | 500 | CVD | collyrium |
| 456-78-9876 | David | 67/05/09 | 96234 | secretary | law | 100 | CVD | collyrium |
| 567-89-0534 | Erin | 80/11/12 | 94143 | doctor | medical | 500 | ARS | stem cell transplant |
| 678-90-1234 | Fred | 60/07/11 | 94123 | secretary | medical | 200 | stroke | nitroglycerin |
| 789-01-2345 | Greg | 50/02/25 | 94145 | carpenter | industry | 200 | broken leg | leg cast |
| 890-12-3456 | Hal | 45/12/31 | 94178 | nurse | medical | 200 | fever | paracetamol |
| 901-23-4567 | Irene | 80/10/07 | 96234 | trainee | law | 100 | dyspepsia | antacid |

(a)

$\mathcal{C}$

$c_1$ = {SSN}
$c_2$ = {Name, Disease}
$c_3$ = {ZIP, Premium}

(b)

$\mathcal{V}$

$v_1$ = Name ∨ (Birth ∧ ZIP)
$v_2$ = (Disease ∧ Birth) ∨ (Disease ∧ Treatment)
$v_3$ = Treatment ∧ Premium
$v_4$ = Insurance

(c)

Fig. 1. An example of relation (a), confidentiality (b) and visibility constraints (c)

$F_1$

| Birth | ZIP | Disease |
|---|---|---|
| 56/12/07 | 94101 | silicosis |
| 79/03/01 | 94123 | silicosis |
| 51/11/11 | 95173 | CVD |
| 67/05/09 | 96234 | CVD |
| 80/11/12 | 94143 | ARS |
| 60/07/11 | 94123 | stroke |
| 50/02/25 | 94145 | broken leg |
| 45/12/31 | 94178 | fever |
| 80/10/07 | 96234 | dyspepsia |

$F_2$

| Insurance | Premium | Treatment |
|---|---|---|
| industry | 100 | bronchodilators |
| industry | 100 | bronchodilators |
| law | 500 | collyrium |
| law | 100 | collyrium |
| medical | 500 | stem cell transplant |
| medical | 200 | nitroglycerin |
| industry | 200 | leg cast |
| medical | 200 | paracetamol |
| law | 100 | antacid |

Fig. 2. An example of correct fragmentation of relation PATIENTS in Figure 1(a) w.r.t. the confidentiality and visibility constraints in Figures 1(b)-(c)

$\{F_1, \ldots, F_l\}$ of fragments, where each fragment $F_i$, $i = 1, \ldots, l$, is a subset of $\{a_1, \ldots, a_n\}$.

Given a fragment $F$ in $\mathcal{F}$, the corresponding fragment instance is the set of tuples in the original relation $r$ projected over the attributes in $F$, maintaining possible duplicate tuples. In the following, we use the term fragment to also refer to a fragment instance, when clear from the context.

A fragmentation is *correct* iff: *1)* no confidentiality constraint is violated by its fragments, *2)* every visibility constraint is satisfied by at least one of its fragments, and *3)* fragments are disjoint (to avoid fragmentation to be "un-done" by joining fragments and hence possibly violating confidentiality constraints). These conditions are formally captured by the definition below.

*Definition 2.4 (Correct fragmentation):* Let $R(a_1, \ldots, a_n)$ be a relation schema, $\mathcal{C}$ and $\mathcal{V}$ be a set of confidentiality constraints and visibility constraints, respectively, over $R$. A fragmentation $\mathcal{F}$ of $R$ is *correct* with respect to $\mathcal{C}$ and $\mathcal{V}$ iff:

1) $\forall c \in \mathcal{C}$, $\forall F \in \mathcal{F}$ : $c \not\subseteq F$ (*confidentiality*);
2) $\forall v \in \mathcal{V}$, $\exists F \in \mathcal{F}$ : $F$ satisfies $v$ (*visibility*);
3) $\forall F_i, F_j \in \mathcal{F}$, $i \neq j$: $F_i \cap F_j = \emptyset$ (*unlinkability*).

Figure 2 illustrates an example of a correct fragmentation of relation PATIENTS in Figure 1(a) with respect to the confidentiality and visibility constraints in Figures 1(b)-(c).

Consistently with previous proposals, we consider only fragmentations that are *minimal*. A fragmentation is minimal iff merging any two of its fragments would violate at least one confidentiality constraint. Minimality ensures that data are not unnecessarily fragmented (as excessive fragmentation can decrease data utility/performance). Also, we do not impose any condition on the presence (or absence) of attributes whose inclusion in the fragmentation is irrelevant for the satisfaction of the visibility constraints (e.g., attribute Job in Figure 1), leaving our modeling as general as the original proposals [5], [7]. We only note that the presence of such attributes can be forced by simply considering a default visibility constraint $v{=}a$ for each attribute $a{\in}R$ that does not appear in a singleton confidentiality constraint.

## 3 PROBLEM DEFINITION

Fragmentation enforces protection of sensitive attributes and/or associations by ensuring that the involved attributes do not appear together in a fragment and that fragments do not have attributes in common (so to ensure their unlinkability). However, such a protection guarantee is based on an implicit assumption that attributes are independent and therefore: *i*) attributes not appearing in any fragment are assumed to remain confidential, and *ii)* absence of common attributes (i.e., satisfaction of the unlinkability condition in Definition 2.4) prevents the correlation among tuples of different fragments.

Unfortunately, such assumptions do not always hold, as in some cases there might be relationships among attributes and the value of an attribute might depend on the values of other attributes. In this case, the values of the former might be inferrable (with precision and full confidence or some uncertainty) from the values of the latter. For instance, the treatment given to a patient can convey information on the disease being cured. In this case, we say that attribute Disease "depends on" attribute Treatment. When data dependencies are present, even if fragments do not explicitly violate confidentiality constraints, they might do so indirectly (as an attribute not belonging to a fragment can be inferred from other attributes in it, thus exposing the attribute or the associations in which it is involved). Also, while fragments might be apparently unlinkable, a correlation among their tuples can be reconstructed based on attributes that can be inferred.

Before illustrating the problem of information leakage they cause, we formally define data dependencies.

$\mathcal{D}$
$d_1 =$ {Birth, ZIP}⇝Name
$d_2 =$ {Treatment}⇝Disease
$d_3 =$ {Disease}⇝Job
$d_4 =$ {Insurance, Premium}⇝Job

Fig. 3. An example of data dependencies over relation PATIENTS in Figure 1(a)

In line with what is assumed for confidentiality and visibility constraints, while noting that in some cases a dependency might hold only for specific values, we consider dependencies at the schema level. Though losing a little in expressiveness, this schema-level treatment nicely fits with the modeling we build upon (where all constraints are specified at the schema level) and provides the great advantage of compact representation and clear modeling for the data administrator. Data dependencies are defined as follows.

*Definition 3.1 (Data dependency):* Let $R(a_1, \ldots, a_n)$ be a relation schema. A *data dependency* $d$ over $R$ is an expression of the form $X \rightsquigarrow Y$, with $X, Y \subset R$ and $X \cap Y = \emptyset$.

A data dependency $X \rightsquigarrow Y$ models the fact that the values of attributes in $Y$ depend on the values of attributes in $X$. In other words, for each tuple $t$ in $r$, $t[X]$ conveys information on $t[Y]$ (determining its exact value or reducing the uncertainty on it). Note that a data dependency $X \rightsquigarrow Y$, with $Y=\{a_i, \ldots, a_j\}$, can be written as a set $\{X \rightsquigarrow a_i, \ldots, X \rightsquigarrow a_j\}$ of data dependencies. In the following, we then assume that the right-hand side of a data dependency always contains a single attribute. Also, given a data dependency $d=X \rightsquigarrow a$, we use the terms *premise* and *consequence* of $d$ (denoted $d.premise$ and $d.consequence$) respectively, to refer to set $X$ of attributes and to attribute $a$, respectively. Figure 3 illustrates an example of data dependencies over relation PATIENTS in Figure 1(a). These dependencies state that: the date of birth and the ZIP code of a patient can disclose her name ($d_1$), capturing the fact that pair (Birth,ZIP) can work as a quasi-identifier [8]; the treatment with which a patient is cured can leak information on her disease ($d_2$); the disease suffered by a patient can leak information on her job ($d_3$); and the combined knowledge of the insurance and the corresponding premium of a patient can leak information on her job ($d_4$).

In general, data dependencies are not symmetric, meaning that it might be that there is a data dependency $a_i \rightsquigarrow a_j$, while the vice-versa ($a_j \rightsquigarrow a_i$) does not hold. For instance, while the name of a patient may determine her sex, the vice-versa is not true. When dependencies are symmetric, both directions (e.g., $a_i \rightsquigarrow a_j$ and $a_j \rightsquigarrow a_i$) should be explicitly specified.

Let $\mathcal{F}$ be a (correct) fragmentation of relation $R$ and $\mathcal{D}$ be a set of data dependencies among the attributes of the same relation. We can identify three basic kinds of confidentiality violations that can be caused by data dependencies, as illustrated in the following (examples refer to the relation and constraints in Figure 1, the data dependencies in Figure 3, and the fragmentation in Figure 2, which is also reported in Figure 4(a) for the reader's convenience).

- *A sensitive attribute or association is exposed by the attributes in a fragment.* Formally, $\exists c \in \mathcal{C}$, $d \in \mathcal{D}$, $F \in \mathcal{F}$: $d.premise \subseteq F$, $c \subseteq F \cup d.consequence$. For instance, as visible in Figure 4(b), sensitive association $c_2$={Name, Disease} is exposed by fragment $F_1$ because of dependency $d_1$={Birth, ZIP}⇝Name.
- *An attribute appearing in a fragment is also derivable from (as it depends on) some attributes in another fragment, thus enabling linkability among such fragments* (which, being the fragmentation minimal, brings to the violation of at least one confidentiality constraint [5]). Formally, $\exists d \in \mathcal{D}$, $F_i, F_j \in \mathcal{F}$, $i \neq j$: $d.premise \subseteq F_i$, $d.consequence \in F_j$. For instance, because of data dependency $d_2$=Treatment⇝Disease, fragment $F_2$ conveys information about attribute Disease, explicitly represented in $F_1$ (see Figure 4(c)). Hence, the tuples in the two fragments can be correlated violating confidentiality constraint $c_3$={ZIP, Premium}.
- *An attribute is derivable (independently) from attributes appearing in different fragments, thus enabling linkability among these fragments.* Formally, $\exists d_i, d_j \in \mathcal{D}$, $i \neq j$, $F_k, F_l \in \mathcal{F}$, $k \neq l$: $d_i.premise \subseteq F_k$, $d_j.premise \subseteq F_l$, $d_i.consequence = d_j.consequence$. For instance, dependencies $d_3$={Disease}⇝Job and $d_4$={Insurance, Premium}⇝Job cause the implicit representation of attribute Job in both fragments $F_1$ and $F_2$, thus enabling their linkability (see Figure 4(d)).

Such cases are only simple examples and more complex violations can be caused by the cascade effect of the exploitation of data dependencies in chains. For instance, dependency $d_2$={Treatment}⇝Disease causes the implicit representation of attribute Disease in fragment $F_2$. In turn, attribute Disease permits to exploit $d_3$={Disease}⇝Job, which causes the implicit representation also of attribute Job in the same fragment.

Our goal is to compute a fragmentation so that data dependencies cannot be exploited for violating the confidentiality constraints. This requires the design of a new fragmentation approach that takes data dependencies into account. One might think that the problem could be easily tackled simply by extending confidentiality constraints to consider possible inferences caused by data dependencies and rewriting constraints such that, for every dependency $X \rightsquigarrow a$ and every constraint $c$ such that $a \in c$, a new constraint $(c \setminus \{a\} \cup X)$ is added. For instance, constraint {Name, Treatment} should be added to the set of confidentiality constraints in Figure 1(b) to take

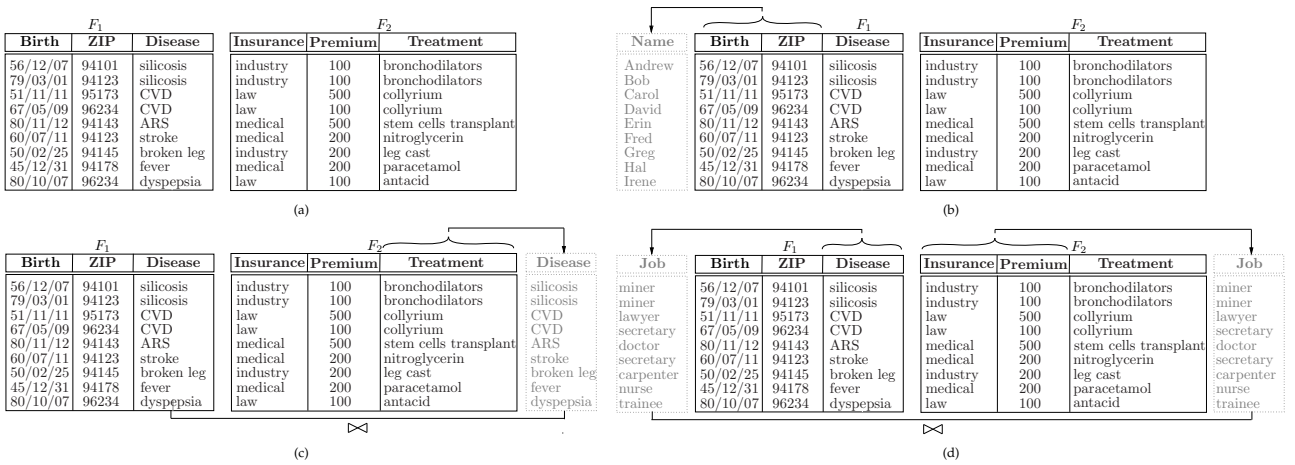



Fig. 4. Information exposure by data dependencies

into account risks of violation of constraint {Name, Disease}, based on inferences that exploit dependency {Treatment}⇝Disease. Unfortunately, the problem is not so simple. As a matter of fact, such a simple extension of the constraints would permit only to capture violations due to exposure of confidential attributes or associations but not of possible linking due to attributes inferrable from one or more fragments (Figures 4(c)-(d)).

Connected with this observation, we also note that our consideration of data dependencies not only simplifies the definition of the confidentiality constraints, but also extends their expressive power, providing stronger protection guarantees than the consideration of constraints only. The simplification of the confidentiality constraints comes from the fact that possible derived confidentiality constraints do not need to be explicitly stated but can be implicitly taken into account simply by defining dependencies. For instance, in presence of possible quasi-identifiers and of association constraints between an individual's identity and some sensitive attributes, previous approaches forced the specification of additional confidentiality constraints involving the quasi-identifier and the sensitive attributes. In our approach, such a burden is removed, since it is sufficient to specify a dependency between the quasi-identifier and the identity to automatically have the implicit additional constraints taken into account. The stronger protection guarantees come from the fact that data dependencies capture possible leakages that could never be expressed with confidentiality constraints alone (as it is the case of improper information exposure in Figures 4(c)-(d)).

As a final remark, note that while data dependencies may resemble functional dependencies, they model a different concept. In fact, a functional dependency of the form $X \rightarrow a$ states that the values of the attributes in $X$ uniquely determine the values of attribute $a$. However, the knowledge of the values of attributes in $X$ does not necessarily imply that we can infer the value of the corresponding attribute $a$. For instance, a typical functional dependency is the one existing between the primary key of a relation and all the other attributes. The primary key can be, for example, a tuple identifier from which no information can be inferred. A data dependency $X \rightsquigarrow a$ models a more generic relationship between the values of attributes in $X$ and the values of attribute $a$ stating that the knowledge of the values of $X$ can convey information on the values of $a$. Of course, such a relationship may hold with different precision and confidence for different values.

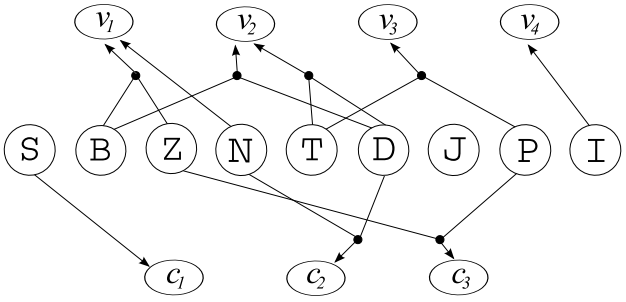
## 4 Problem Formulation and Modeling

Data dependencies can cause exposure of information not explicitly released but that can be inferred exploiting dependencies directly (from the attributes in one fragment) or indirectly (correlating apparently unlinkable fragments). To reason about, and represent, fragments and their satisfaction of the constraints, even in presence of data dependencies, we adopt a graphical representation of the problem. We first model the problem without data dependencies and then inject data dependencies in it, capturing information leakage.

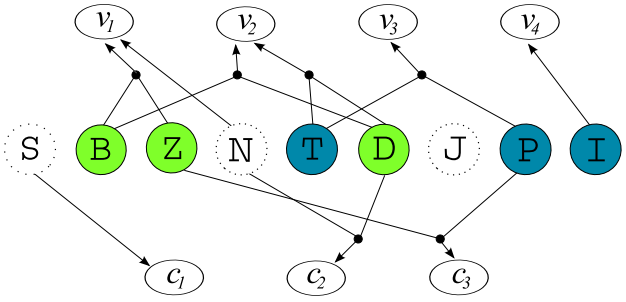
### 4.1 Constraint and fragmentation graph

We define a *constraint and fragmentation graph* as a colored directed hypergraph where:

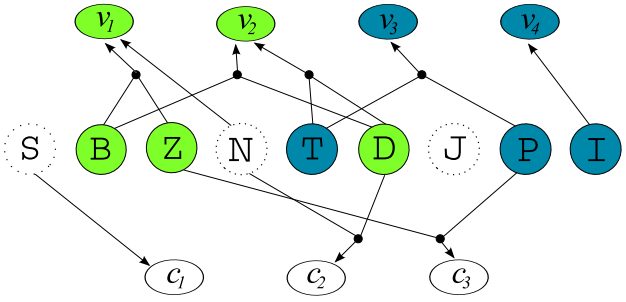
- every attribute in the original relation and every (confidentiality or visibility) constraint corresponds to a node; to graphically distinguish attribute nodes from constraint nodes, we denote attribute nodes with circles and constraint nodes with ovals;
- every confidentiality constraint $c = \{a_1, \ldots, a_n\}$ is translated into a hyperarc connecting nodes $a_1, \ldots, a_n$ to node $c$;




(a) Constraint and fragmentation graph




(b) Colored fragments in the graph




(c) Propagation of the colors to the constraints

Fig. 5. A constraint and fragmentation graph

- every visibility constraint of the form $v = a_1 \wedge \ldots \wedge a_n$ is translated into a hyperarc connecting nodes $a_1, \ldots, a_n$ to node $v$;
- every visibility constraint of the form $v = v_1 \vee \ldots \vee v_n$ (where each $v_j$ is a conjunction of attributes as above) is translated into different hyperarcs, one for each $v_j$, $j = 1, \ldots, n$, connecting nodes that represent attributes in $v_j$ to node $v$.

Figure 5(a) illustrates such a graph for the problem in Figure 1, where, for simplicity, attributes are denoted with their initial.

A fragmentation can be easily represented on our graph by coloring its nodes. Each fragment is associated with a color, which is then associated with all the attributes belonging to the fragment (we use a dotted line for attributes that remain "neutral", that is, with no color associated since they do not belong to any fragment). Figure 5(b) represents the fragmentation in Figure 2, where $F_1$ is green (lighter color in b/w printouts) and $F_2$ is blue (darker color in b/w printouts).

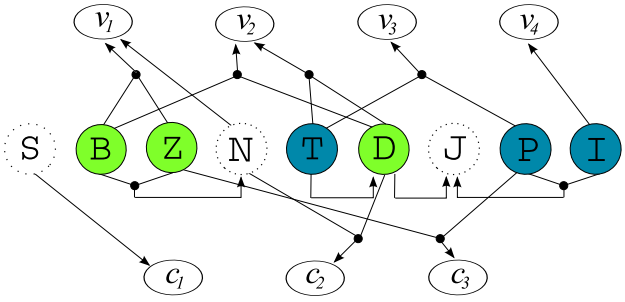
With the graph and the coloring above, satisfaction of the constraints can be easily checked by propagating colors through hyperarcs, where a color propagates along a hyperarc if all its sources have it. Figure 5(c) illustrates such a propagation: $v_1$ is green from `B` and `Z`, $v_2$ is green from `B` and `D`, $v_3$ is blue from `T` and `P`, and $v_4$ is blue from `I`. No color propagates to confidentiality constraints.

Since fragments should satisfy constraints by ensuring the visibility of attributes as demanded by visibility constraints and the protection of sensitive attributes/associations as demanded by confidentiality constraints, a fragmentation (meaning a coloring of the attributes in the graph) is correct iff: *1)* no node representing a confidentiality constraint is colored, *2)* all the nodes representing visibility constraints have at least one color, and *3)* nodes representing attributes have only one color. Note how they correspond to the three conditions in Definition 2.4. It is easy to see (Figure 5(c)) that the fragmentation in Figure 2 is correct.

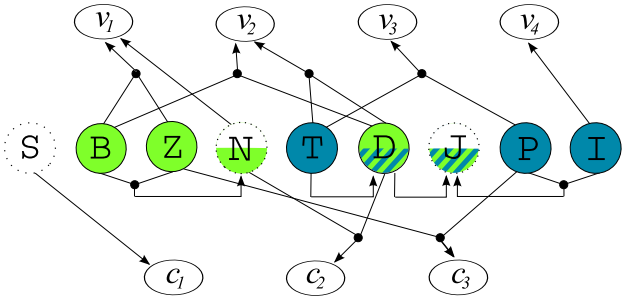
### 4.2 Considering data dependencies

Data dependencies enable inference of some attributes based on other attributes made visible by the fragments (or themselves indirectly exposed via inference from the fragments). Data dependencies can be easily captured in our graphical representation as hyperarcs connecting the attributes in the premise with the attribute in the consequence: each data dependency $d$=$\{a_i, \ldots, a_j\} \rightsquigarrow a$ is translated into a dependency hyperarc connecting $a_i, \ldots, a_j$ to $a$. Figure 6(a) extends the graph in Figure 5(b) with hyperarcs representing the data dependencies in Figure 3.

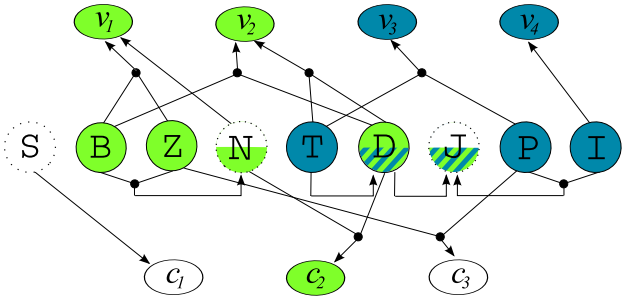
Information leakage caused by data dependencies can then be captured by the color propagation enabled by dependency hyperarcs similarly to what done above for checking constraints. In this color propagation, we need to take into account the fact that visibility constraints require explicit presence of the attributes in fragments, hence only colors originally assigned to the attributes (not those propagated via dependencies) should flow through hyperarcs corresponding to visibility constraints. We then maintain the original color assigned to a node (i.e., its fragment) distinguishable from the colors propagated to it via dependencies and represent propagated colors only in the bottom semi-half of attribute nodes. The different behavior of hyperarcs corresponding to visibility constraints is nicely visible by having such hyperarcs leaving from the top of the nodes (which maintain only the original color). A further important aspect to take into consideration is that, unlike propagation of colors through hyperarcs to constraints (which do not have any outgoing arc), propagation of colors to attributes has a cascade effect, and the propagation of a color to an attribute can fire further propagation.




(a) Colored fragments in the graph with dependencies




(b) Color-coded inferences due to dependencies




(c) Propagation of the colors to the constraints

Fig. 6. A constraint and fragmentation graph with dependencies

This recursive propagation reflects the fact that inferred attributes can in turn enable further inferences.

Figure 6(b) illustrates the propagation of colors to attributes enabled by dependency hyperarcs: N has a derived green (from B and Z), D has a derived blu from T, J has both a derived green from D and a derived blue from P and I as well as from D (derived from T through an indirect propagation). As said, derived colors are represented in the bottom half of attribute nodes. When more than one color (original or derived) is associated with a node, we represent the different colors with stripes.

Like before, propagation of colors from attributes to constraints via the hyperarcs allows us to easily check the satisfaction of the constraints. Clearly, injection of dependencies cannot cause problems to visibility constraints (since data visibility can only be augmented - indirectly - by dependencies). Dependencies can instead cause information exposure compromising confidentiality of sensitive attributes or associations. Improper exposure caused by dependencies translates into an indirect violation of the confidentiality or of the unlinkability conditions in Definition 2.4. Of course not all inferences due to dependencies create problems. For instance, a dependency disclosing an attribute that was not included in any fragment (as not needed to satisfy visibility constraints) but that is not involved in any confidentiality constraint does not create any violation. Our colored graph allows us to easily detect when dependencies compromise sensitive information: a confidentiality constraint becoming of a given color signals that the fragment with that color indirectly violates the constraint (i.e., exposes the attribute or association defined as sensitive by the constraint); an attribute becoming multi-colored signals the fact that the attribute can be inferred from the fragments of those colors hence enabling a correlation between the tuples of the fragments, thus indirectly violating the unlinkability condition. As an example, Figure 6(c) shows that the fragmentation in Figure 2 indirectly: *i)* violates constraint $c_2$, which becomes green from N and D; and *ii)* violates unlinkability enabling the correlation between the tuples of the fragments via D, which is present in $F_1$ (green) and inferable from $F_2$ (blue), and also via J, which is inferable from both fragments.

## 5 Capturing Exposure from Data Dependencies

The propagation of colors described in the previous section models the fact that attributes can be inferred, that is, that they are indirectly exposed by fragments. Intuitively, every time the premise of a dependency is present in a fragment, the consequence is also indirectly present in the fragment. The inference effect on attributes from dependencies is captured by the following definition.

*Definition 5.1 (Fragment and dependency composition):* Let $R(a_1, \ldots, a_n)$ be a relation schema, $F$ be a fragment of $R$, and $d$ be a data dependency among attributes in $R$. The *composition* of $F$ with $d$ is a set of attributes

$$F \otimes d = \begin{cases} F \cup d.consequence, & d.premise \subseteq F \\ F, & \text{otherwise} \end{cases}$$

For instance, with reference to our example:
{B,Z,D} $\otimes$ {B,Z}$\rightsquigarrow$N = {B,Z,D,N},
{B,Z,D} $\otimes$ {D}$\rightsquigarrow$J = {B,Z,D,J},
{T,P,I} $\otimes$ {T}$\rightsquigarrow$D = {T,P,I,D},
{T,P,I} $\otimes$ {P,I}$\rightsquigarrow$J = {T,P,I,J}.
As already observed, indirect exposure of attributes can clearly have a cascade effect as it can enable new inferences (new color propagation in terms of our graph). For instance, in our example, indirect inclusion of D in the blue fragment ($F_2$) causes also exposure of J in the same fragment (i.e., propagation of the blue color to it). (Note that J is exposed via two inference paths, as it is already exposed in the

blue fragment by the combination of P and I). Consideration of such a cascade effect implies the need to enable a further color derivation due to dependencies every time a dependency has some effect (i.e, some color is propagated), until a fixpoint is reached. In other words, the consideration of all possible effects of dependencies requires closing fragments with respect to the fragment and dependency composition in Definition 5.1, as captured by the following definition.

*Definition 5.2 (Closure):* Let $R(a_1, \ldots, a_n)$ be a relation schema, $\mathcal{F}=\{F_1, \ldots, F_m\}$ be a fragmentation of $R$, and $\mathcal{D}$ be a set of data dependencies over $R$. The *closure* of a fragment $F \in \mathcal{F}$ w.r.t. $\mathcal{D}$, denoted $F^*$, is a set of attributes such that $F \subseteq F^* \subseteq R$ and $\forall d \in \mathcal{D}$, $F^* \otimes d = F^*$. The *closure* of fragmentation $\mathcal{F}$ w.r.t. $\mathcal{D}$, denoted $\mathcal{F}^*$, is the set $\{F_1^*, \ldots, F_m^*\}$ of the closure of its fragments.

In terms of our graphical representation, the closure of a fragment includes all the nodes that - after performing all possible direct and indirect color propagations - include the color of the fragment. For instance, with reference to our running example, $F_1^*$ = {B, Z, D, N, J} and $F_2^*$ = {T, P, I, D, J}, as visible from Figure 6(b).

The closure of a fragmentation, by including all information derivable from its fragments via dependencies, permits to identify fragmentations that, though correct according to the original definition (Definition 2.4) cause improper information exposure due to *indirect* violation of the confidentiality and/or unlinkability conditions. In fact, such fragmentations are trivially all those whose closure violate the confidentiality and/or unlinkability conditions in Definition 2.4. We then extend the definition of correct fragmentation as follows.

*Definition 5.3 (Correct fragmentation – Extended):* Let $R(a_1, \ldots, a_n)$ be a relation schema, $\mathcal{C}$, $\mathcal{V}$, and $\mathcal{D}$ be a set of confidentiality constraints, visibility constraints, and data dependencies, respectively, over $R$. A fragmentation $\mathcal{F}$ of $R$ is *correct* with respect to $\mathcal{C}$, $\mathcal{V}$, and $\mathcal{D}$ iff:

1) $\forall c \in \mathcal{C}, \forall F \in \mathcal{F} : c \not\subseteq F^*$ (*confidentiality*);
2) $\forall v \in \mathcal{V}, \exists F \in \mathcal{F} : F$ satisfies $v$ (*visibility*);
3) $\forall F_i, F_j \in \mathcal{F}, i \neq j$: $F_i^* \cap F_j^* = \emptyset$ (*unlinkability*);

where $F^*$ is the closure of $F$ w.r.t. $\mathcal{D}$, for each $F$ in $\mathcal{F}$ (Definition 5.2)

Figure 7 illustrates an example of fragmentation that satisfies Definition 5.3 and Figure 8 illustrates the corresponding constraint and fragmentation graph after the color propagation due to data dependencies.

While natural and intuitive, the extended definition of correctness implies the need of controlling the satisfaction of conditions on the fragments' closure (instead of the fragments themselves) hence requiring the recursive computation of such a closure.

Our approach to avoid paying such a recursive computation is based on the consideration of fragments that are already closed with respect to the composition with dependencies, and on a result that allows us to check whether a fragmentation is closed simply looking at its fragments and the dependencies, without actually computing such a closure.

$F_1$

| Name | Insurance |
|---|---|
| Andrew | industry |
| Bob | industry |
| Carol | law |
| David | law |
| Erin | medical |
| Fred | medical |
| Greg | industry |
| Hal | medical |
| Irene | law |

$F_2$

| Premium | Disease | Treatment |
|---|---|---|
| 100 | silicosis | bronchodilators |
| 100 | silicosis | bronchodilators |
| 500 | CVD | collyrium |
| 100 | CVD | collyrium |
| 500 | ARS | stem cell transplant |
| 200 | stroke | nitroglycerin |
| 200 | broken leg | leg cast |
| 200 | fever | paracetamol |
| 100 | dyspepsia | antacid |

Fig. 7. An example of (extended) correct fragmentation of relation PATIENTS in Figure 1(a) w.r.t. the confidentiality constraints in Figure 1(b), the visibility constraints in Figure 1(c), and the data dependencies in Figure 3

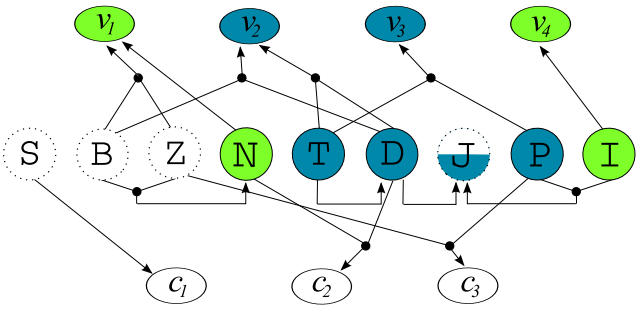



Fig. 8. Constraint and fragmentation graph with dependencies of the fragmentation in Figure 7

We start then by introducing closed fragmentations and proving that such a property can be verified simply on the fragmentation and the data dependencies.

*Definition 5.4 (Closed fragmentation):* Let $R(a_1, \ldots, a_n)$ be a relation schema and $\mathcal{D}$ be a set of data dependencies over $R$. A fragmentation $\mathcal{F}$ of $R$ is *closed* with respect to $\mathcal{D}$ iff $\mathcal{F}=\mathcal{F}^*$.

A straightforward approach for computing a closed fragmentation consists in starting from an arbitrary fragmentation and computing its closure, considering then the latter as a solution. Clearly, this would imply the execution of the recursive dependency composition (i.e., color propagation) illustrated above, which we actually aim at avoiding. Our approach to avoid such a computation is based on the observation that the composition (i.e., color propagation) of a dependency $d$ with a fragment $F$ has some effects on the fragment iff the following two conditions hold: *1)* the fragment contains the premise of the dependency (i.e., all nodes in $d.premise$ have color $F$); and *2)* the fragment does not contain the consequence of the dependency (i.e., $d.consequence$ does not have color $F$, since otherwise it is ineffective). This observation allows us to translate the control on the fact that a fragmentation is closed into a condition that does not require recursive evaluation, as stated by the follow-

$F_1$

| Name | Insurance |
|---|---|
| Andrew | industry |
| Bob | industry |
| Carol | law |
| David | law |
| Erin | medical |
| Fred | medical |
| Greg | industry |
| Hal | medical |
| Irene | law |

$F_2$

| Job | Premium | Disease | Treatment |
|---|---|---|---|
| miner | 100 | silicosis | bronchodilators |
| miner | 100 | silicosis | bronchodilators |
| lawyer | 500 | CVD | collyrium |
| secretary | 100 | CVD | collyrium |
| doctor | 500 | ARS | stem cell transplant |
| secretary | 200 | stroke | nitroglycerin |
| carpenter | 200 | broken leg | leg cast |
| nurse | 200 | fever | paracetamol |
| trainee | 100 | dyspepsia | antacid |

Fig. 9. An example of closed and correct fragmentation of relation PATIENTS in Figure 1(a) w.r.t. the confidentiality constraints in Figure 1(b), the visibility constraints in Figure 1(c), and the data dependencies in Figure 3

ing theorem. (The proof of the theorem is provided as supplemental material.)

*Theorem 5.5:* Given a relation schema $R(a_1, \ldots, a_n)$, a fragment $F$ of $R$, and a set $\mathcal{D}$ of data dependencies over $R$, the following implication holds:
$F{=}F^* \iff \forall d{\in}\mathcal{D}$: $d.premise{\not\subseteq}F$ or $d.premise \cup d.consequence \subseteq F$.

According to this theorem, the recursive process of computing the closure of a fragmentation $\mathcal{F}$ to check its correctness (which is necessary to take into consideration the cascade effect caused by data dependencies) can be translated into an equivalent static set of conditions that a closed and correct fragmentation must satisfy. It is straightforward to see that a closed fragmentation $\mathcal{F}$ that satisfies Definition 2.4 also satisfies Definition 5.3. In fact, when $\mathcal{F}{=}\mathcal{F}^*$ the two definitions are equivalent.

The only tricky question that remains open is whether our consideration of closed fragmentations might prevent us from finding a solution to the problem (i.e., a fragmentation that satisfies Definition 5.3) when such a solution would instead exist simply because there is no solution which is closed. Luckily, such a case cannot exist and while indeed some fragmentations satisfying Definition 5.3 might be not closed, their closure must be also a solution, as formalized by the following theorem. (The proof of the theorem is provided as supplemental material.)

*Theorem 5.6:* Given a relation schema $R(a_1, \ldots, a_n)$, and sets $\mathcal{C}$, $\mathcal{V}$, and $\mathcal{D}$ of confidentiality constraints, visibility constraints, and data dependencies, respectively, over $R$ the following implication holds:
$\exists\mathcal{F}$ satisfying Definition 5.3 $\iff$ $\exists\mathcal{F}'$ satisfying Definitions 2.4 and 5.4.

As a last remark, we note that our consideration of closed fragmentations may cause the inclusion in the solution of attributes whose presence or absence is irrelevant for the satisfaction of the visibility constraints. For instance, with reference to our running example, a closed fragmentation would include attribute `Job` (Figure 9), while the same fragmentation without such an attribute would also be a (non closed) solution to the problem (Figure 7). While noting that such a presence is indeed not a problem (the original proposal did not put specific requirements in this respect and treated solutions with or without such attributes as equally good), we also note that they could be removed from the solution with a post-processing scanning data dependencies and checking if their consequence can be removed from the fragment in which it appears (if any) without affecting the satisfaction of the visibility constraints.

The observations and results above, and the intuition of considering closed fragmentations, simplify our problem removing the need for a recursive evaluation. Our goal is then to compute a closed fragmentation that satisfies Definition 2.4 (equivalently, Definition 5.3). Also, to avoid excessive fragmentation, such solution should be minimal (as already commented in Section 2). We note that the minimality requirement (i.e., merging any pair of fragments violates at least one constraint) was imposed in previous approaches as an approximation of a minimum requirement (i.e., minimum number of fragments) to simplify the problem and to approach it heuristically. Tackling the problem with a tool that efficiently solves the well known constraint satisfaction problem (to which our problem easily reduces, as illustrated in the following section), we avoid imposing such a simplification to the problem and require our solution to be minimum (i.e., be composed of the smallest number of fragments). It is trivial to see that a minimum solution is also minimal, while the vice-versa is not true [4]. In fact, a minimum fragmentation $\mathcal{F}$ is also minimal, since any fragmentation $\mathcal{F}'$ obtained by merging two fragments in $\mathcal{F}$ has a smaller number of fragments than $\mathcal{F}$ and therefore violates at least one constraint, else $\mathcal{F}$ would not be minimum. By contrast, given a minimal fragmentation $\mathcal{F}$ composed of $n$ fragments, while we are guaranteed that no fragmentation $\mathcal{F}'$ obtained by $\mathcal{F}$ combining any two fragments satisfies the constraints, it could be that there is a fragmentation $\mathcal{F}''$ satisfying the constraints with a smaller number of fragments. Consider, as an example, relation PATIENTS in Figure 1(a), the confidentiality constraints in Figure 1(b) and an additional constraint $c_4$={`N`,`Z`}, the visibility constraints in Figure 1(c), and the data dependencies in Figure 3. Fragmentation $\mathcal{F}$={{`N`}, {`J`,`P`,`D`,`T`}, {`Z`,`I`}} is minimal, but it is not minimum. In fact, fragmentation $\mathcal{F}'$={{`N`,`I`}, {`J`,`P`,`D`,`T`}} in Figure 9 is correct and composed of less fragments (2) than $\mathcal{F}$.

Our problem of finding a Minimum Closed and Correct Fragmentation (MCCF) is then formally defined as follows.

*Problem 5.7 (MCCF):* Let $R(a_1, \ldots, a_n)$ be a relation schema, $\mathcal{C}$, $\mathcal{V}$, and $\mathcal{D}$ be a set of confidentiality constraints, visibility constraints, and data dependencies, respectively, over $R$. Determine a *closed* fragmentation (Definition 5.4) $\mathcal{F}$ over $R$ that satisfies Definition 2.4 such that $\nexists\mathcal{F}'$ satisfying the same definition and such

that $|\mathcal{F}'| < |\mathcal{F}|$.

For instance, with reference to our running example, Figure 9 represents a minimum closed and correct fragmentation of relation PATIENTS with respect to the constraints in Figure 1 and the data dependencies in Figure 3.

In the statement of the problem we do not impose requirements on the closure of $\mathcal{F}'$, since it is irrelevant w.r.t. the problem (a fragmentation and its closure always have the same number of fragments).

The MCCF problem is NP-hard since the MINFRAG problem introduced in [4] (i.e., the problem of computing a minimum fragmentation with respect to Definition 2.4) is NP-hard and it can be reduced to the MCCF problem in polynomial time by simply assuming $\mathcal{D}$=$\emptyset$. In the next section, we describe how to solve the MCCF problem.

## 6 COMPUTING A MINIMUM FRAGMENTATION

To solve the MCCF problem we represent it as a Constraint Satisfaction Problem (CSP), which can then be conveniently solved with off-the-shelf CSP solvers [9], providing efficiency and scalability.

The CSP is formulated as follows: given a triple $\langle X, D, C\rangle$, with $X$ a set of variables, $D$ the domain of variables in $X$, and $C$ a set of constraints over $X$, find an assignment $w : X \rightarrow D$ that satisfies all the constraints in $C$. A solution of the CSP is a function $w : X \rightarrow D$ that assigns a value in $D$ to each variable in $X$, in such a way that all the constraints in $C$ are satisfied. Our translation naturally interprets:

- the attributes $\{a_1, \ldots, a_n\}$ in the schema of the original relation $R$ as the set $X$ of variables;
- the set $\{0, \ldots, m\}$ of integers, with $m$ the number of fragments in a fragmentation $\mathcal{F}$ of $R$, as the domain $D$ of the variables in $X$;
- the conditions that a closed and correct fragmentation must satisfy (Definitions 2.4 and 5.4) as the set $C$ of constraints.

A solution to the problem so defined corresponds to an allocation of attributes to fragments, where $a \in F_i$ iff $w(a) = i$ and $i \neq 0$. Value 0 is assigned to attributes that do not belong to any fragment. In terms of our graphical modeling, $w$ corresponds to a coloring function over our constraint and fragmentation graph, where $D$ is the domain of colors (with 0 representing the neutral color) that can be associated with the nodes representing attributes in $R$. An assignment function is a minimum fragmentation iff it employs the minimum possible number of colors.

We now show how the confidentiality, visibility, and unlinkability conditions (Definition 2.4), as well as our additional requirement of considering as solutions only fragmentations that are closed (Definition 5.4) are translated into equivalent CSP constraints that the assignment function $w$ must satisfy. In the following, we refer our examples to relation PATIENTS in Figure 1(a) and, for clarity, the name of the variables representing attributes corresponds to the attributes' initials.

- *Confidentiality* (Definition 2.4). A confidentiality constraint $c$=$\{a_i, \ldots, a_j\}$ is satisfied if it is not the case that all the attributes in it are assigned to a same fragment. In terms of the CSP assignment function, we express this condition by requiring that either there exists at least a pair of attributes with different values or (if they are all equal) that their value is 0. In this latter case, being all the attributes equal, it is sufficient to evaluate that only one attribute (the first one for simplicity) is equal to 0. Formally, each $c \in \mathcal{C}$ is translated into the following constraint:

$$\bigvee_{a_l, a_k \in c, l \neq k} (a_l \neq a_k) \vee (a_i = 0)$$

- *Visibility* (Definition 2.4). A visibility constraint $v$=$a_i \wedge \ldots \wedge a_j$ is satisfied if all the attributes in $v$ are released (i.e., $\forall a \in v$, $a \neq 0$) and stored in the same fragment (i.e., $\forall a_l, a_k \in v$, $a_l = a_k$). Formally, $v$ is translated into condition ($a_i$=$\ldots$=$a_j \wedge a_i \neq 0$). Note that it is sufficient to require that one (any) attribute in $v$ (the first one for simplicity) is different from 0 to guarantee the release of all the attributes in $v$ as they must all have the same value.

  A visibility constraint $v$=$v_1 \vee \ldots \vee v_n$ (where each $v_i$ in $v$ is a conjunction of attributes as above) is satisfied if at least one $v_i$ in $v$ is satisfied. Formally, each $v \in \mathcal{V}$ is translated into the following constraint:

$$\bigvee_{v_k \in v} (a_i = \ldots = a_j \wedge a_i \neq 0)$$

  with $v_k$=$a_i \wedge \ldots \wedge a_j$.
- *Unlinkability* (Definition 2.4). The unlinkability condition is automatically guaranteed by the fact that each variable can be assigned one value only in $D$, meaning that an attribute cannot be included in more than one fragment. Hence no further condition needs to be imposed.
- *Closure* (Definition 5.4). A data dependency $d$ cannot be exploited for inference when either *d.premise* is broken by fragmentation, or *d.premise* and *d.consequence* are stored in the same fragment. The premise of a dependency $d$ is broken if, similarly to confidentiality constraints, it is not the case that all the attributes in it are assigned to a same fragment. In terms of the CSP, the premise is broken if either there are at least two attributes with different values or, if they are all equal, their value is 0 (i.e., at least one of them is 0). Analogously, *d.premise* and *d.consequence* are stored in the same fragment if all the attributes in *d.premise* and *d.consequence* have the same value. Formally, each $d \in \mathcal{D}$, with

| | Input | CSP translation |
|---|---|---|
| CONFIDENTIALITY | $c_1$ = {SSN} | S=0 |
| | $c_2$ = {Name, Disease} | (N≠D) ∨ (N=0) |
| | $c_3$ = {ZIP, Premium} | (Z≠P) ∨ (Z=0) |
| VISIBILITY | $v_1$ = Name ∨ (Birth ∧ ZIP) | (N≠0) ∨ ((B=Z) ∧ (B≠0)) |
| | $v_2$ = (Disease ∧ Birth) ∨ (Disease ∧ Treatment) | ((D=B) ∧ (D≠0)) ∨ ((D=T) ∧ (D≠0)) |
| | $v_3$ = Treatment ∧ Premium | (T=P) ∧ (T≠0) |
| | $v_4$ = Insurance | I≠0 |
| DATA DEPENDENCIES | $d_1$ = (Birth, ZIP) ⇝ Name | (B≠Z) ∨ (B=0) ∨ ((B=Z) ∧ (Z=N)) |
| | $d_2$ = Treatment ⇝ Disease | (T=0) ∨ (T=D) |
| | $d_3$ = Disease ⇝ Job | (D=0) ∨ (D=J) |
| | $d_4$ = (Insurance, Premium) ⇝ Job | (I≠P) ∨ (I=0) ∨ ((I=P) ∧ (P=J)) |

Fig. 10. Constraints and data dependencies (Input) and their CSP formulation (CSP translation) that a closed and correct fragmentation of relation PATIENTS must satisfy

$d.premise$=$\{a_i, \ldots, a_j\}$ and $d.consequence$=$a_x$, is translated into the following constraint:

$$\bigvee_{a_l, a_k \in d.premise, l \neq k} ((a_l \neq a_k) \vee (a_i = 0)) \vee (a_i = \ldots = a_j = a_x)$$

Figure 10 illustrates the CSP constraints modeling our running example, reporting the CSP translation of the constraints in Figure 1 and the dependencies in Figure 3 over relation PATIENTS.

To compute a minimum solution to the problem, we formulate the CSP instance corresponding to our fragmentation problem illustrated above in terms of a Constraint Logic Program (CLP) [10], and use the well-known SWI Prolog interpreter enriched with the `CLP(fd)` Prolog library [11]. We first check if a solution to the problem does exist. Such a check is based simply on the following observation. Let $M$ be the minimum among: *i)* the number of attributes in $R$ that appear in a visibility constraint; *ii)* the number of confidentiality constraints in $\mathcal{C}$; and *iii)* the number of visibility constraints in $\mathcal{V}$. Any fragmentation composed of more than $M$ fragments cannot be a minimum solution [5]. Hence, if there is no correct fragmentation with at most $M$ fragments (Definition 5.3), there is no solution to the problem. We run the CLP solver with domain $D$=$\{0, 1, \ldots, M\}$. If no solution is found, the process is stopped since the constraints are in conflict. Otherwise, we proceed to compute a minimum fragmentation by iteratively evaluating the CLP solver over different instances of the problem, with domain $D$=$\{0, 1, \ldots, m\}$, starting with $m = 1$ (i.e., a fragmentation with one fragment) and, if no solution is found, increase $m$ by 1 at each iteration. We terminate the process when a solution - clearly one composed of the minimum number of fragments - is found (when $m = M$ in the worst case). For instance, with reference to our running example, $M$=$min\{7, 3, 4\}$=3, since 7 attributes appear in visibility constraints, $\mathcal{C}$ includes 3 confidentiality constraints, and $\mathcal{V}$ includes 4 visibility constraints. The CLP solver is first invoked with $m = M$ and returns an assignment, showing that a solution exists. Hence, the iterative process is started invoking the CLP solver with $m$=1, which returns *false*. Then, the solver is called with $m$=2 and returns assignment `S=B=Z=0`, `N=I=1`, and `D=T=J=P=2`, corresponding to the minimum fragmentation in Figure 9. Note that, since the translation into CSP constraints of the conditions that should be satisfied by a closed and correct fragmentation is independent from the number of fragments in a fragmentation, constraints in $C$ and variables in $X$ do not need to be updated in any way when $m$ changes.

We implemented a prototype written in Java and run several experiments on a server with two CPU Intel(R) Xeon(R) E5504 2.00GHz, 12GB RAM, one 240GB SSD disk, and Ubuntu 12.04 64bit operating system with Unity interface, Linux Kernel 3.2.0-56-generic. Our prototype operates in three steps. The first step randomly generates an instance of the Minimum Closed and Correct Fragmentation problem (Problem 5.7). Given a relational schema composed of $n$ attributes, it randomly generates a set of non-singleton (i.e., non-trivial) confidentiality constraints, a set of visibility constraints, and a set of data dependencies. The second step realizes the translation of the MCCF instance generated by the first step into an equivalent instance of the CLP, according to the syntax of SWI Prolog interpreter and the `CLP(fd)` Prolog library, as illustrated above. The third step invokes the services of the CLP solver to compute a solution to the instance of the CLP automatically built by the second step, if such a solution exists.

To evaluate the performance and scalability of our proposal, and the impact of confidentiality constraints, visibility constraints, and data dependencies in the evaluation, we have tested our prototype in different configurations. We considered a large variety of configurations, randomly generated varying the number $n$ of attributes in the schema of relation $R$ from 10 to 2000. For a relation schema with $n$ attributes, we considered: $n/4$, $n/8$, $n/16$ confidentiality constraints; $n/8$, $n/16$, $n/32$ visibility constraints; and $n/10$, $n/20$, $n/40$, 0 data dependencies. Each confidentiality constraint is composed of a number of attributes ranging between 2 and 10, while each visibility constraint involves between 1 and 5 at-

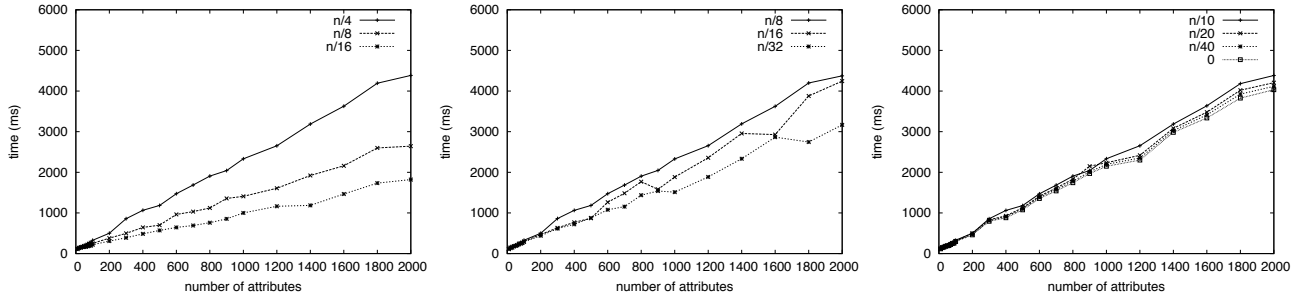



Fig. 11. Computational time varying the number of confidentiality constraints (a), visibility constraints (b), and data dependencies (c)

tributes in $R$. Data dependencies are characterized by premises including between $1$ and $8$ attributes, and consequences including one attribute (as noted in Section 3, consequences with more than one attribute can be simply expressed as different dependencies).

In Figure 11, we report the results (obtained as the average over $10$ runs) of different configurations produced considering a base of $n/4$ confidentiality constraints, $n/8$ visibility constraints, and $n/10$ data dependencies, and then keeping two of the dimensions fixed at the base while varying the third one. In particular, we have: *i)* fixed visibility constraints at $n/8$ and data dependencies at $n/10$, varied confidentiality constraints at $n/4$, $n/8$, and $n/16$ (Figure 11(a)); *ii)* fixed confidentiality constraints at $n/4$ and data dependencies at $n/10$, varied visibility constraints at $n/8$, $n/16$, and $n/32$ (Figure 11(b)); and *iii)* fixed confidentiality constraints at $n/4$ and visibility constraints at $n/8$, varied data dependencies at $n/10$, $n/20$, $n/40$, and $0$ (Figure 11(c)). As visible from the figure, the computational time linearly grows with the number of attributes in the relation but it remains negligible for small configurations (less than one second for configurations with up to $400$ attributes independently from the number of constraints and dependencies). Even considering larger configurations, the time necessary to our prototype to compute a solution to our fragmentation problem (or detect that no solution exists) remains in the order of a few seconds: less than $5$ seconds for configurations with $2000$ attributes, $500$ confidentiality constraints, $250$ visibility constraints, and $200$ data dependencies, which are probably hard to find in real-world scenarios. Figure 11 also shows that the time necessary to compute a minimum fragmentation (if such a fragmentation exists) grows with the number of confidentiality constraints, visibility constraints, and data dependencies. Data dependencies, however, have a limited impact on the computational time with respect to confidentiality and visibility constraints (see Figure 11(c) reporting also the computational time of configurations with no data dependencies). In summary, the computational time obtained in our experiments, which consider configurations much more complex than those that can be expected to be found in practice, clearly proves the efficiency and scalability of our approach.

## 7 Related work

The growing interest of the research community on the emerging data outsourcing and cloud computing paradigms is testified by the huge amount of work addressing different security and privacy concerns including access control, data protection, and techniques for efficiently querying encrypted data (e.g., [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28]). With respect to the problem of protecting data confidentiality, which is the main goal of our work, the first proposals were based on the implicit assumption that all outsourced data are equally sensitive and therefore are protected through an encryption layer (e.g., [29], [30]). Since confidentiality demands that data decryption can be possible only at the user side, solutions have been also developed to enable external servers to execute queries on encrypted data (e.g., [16], [17]). Such solutions consist in defining indexes that the server storing the data can use to respond to specific queries. The main problem of these indexes is that they make query execution more expensive. To improve query execution efficiency, the research community has then proposed the use of fragmentation (possibly combined with encryption) as an alternative technique to protect sensitive data and associations among them (e.g., [3], [4], [6], [31]). In these approaches, the sensitive associations that need to be protected are modeled through a set of confidentiality constraints representing sets of attributes that cannot be jointly released. The sensitive associations are then protected by storing the data in different fragments that cannot be joined and by possibly encrypting some attributes. Following the fragmentation approach, some proposals put forward the idea of protecting sensitive associations while ensuring the visibility of specific data views (e.g., [5], [32]). These fragmentation-based proposals, although

interesting and effective, assume that no dependencies exist among data, thus being vulnerable to possible indirect information leakage. Recent solutions have considered the inference problem in a fragmentation scenario characterized by the presence of data dependencies (e.g., [33], [34]). However, these proposals consider fragmentations with two fragments only (stored at two non-communicating servers or one at the data owner side and the other one at an external server), and rely on a purely logical model for the representation of confidentiality constraints and inferences. Our proposal assumes an arbitrary number of fragments (which could even be stored at the same server), considers also visibility constraints, and provides a simple and natural characterization of exposures due to inferences as well as an efficient approach to compute a minimum fragmentation. A further line of work aims at studying the correlations that can be established among fragments through the exploitation of different kinds of information (e.g., the presence of indexes in the fragments [35] or the combination of indexes with the techniques supporting the selective access to the outsourced data [36]).

A similar (but not equal) problem of information leakage caused by data dependencies has been also investigated in the data publishing scenario (e.g., [37], [38]). The proposal in [37] is aimed at destroying the correlation between two disjoint and pre-defined subsets of attributes before their publication, while we are interested in computing a data fragmentation that does not suffer from confidentiality violations caused by data dependencies. The solution in [38] aims at guaranteeing $k$-anonymity [8] when publicly releasing a microdata table, assuming that the adversarial knowledge includes functional dependencies among attributes. Other works consider the inferences that can be drawn from the knowledge of the disclosure algorithm adopted (e.g., [39]).

Further related proposals are the classical studies on inferences in multilevel database systems, where most inference research addresses detection of inference channels within a database or at query processing time (e.g., [40], [41]). Although the problem addressed by these proposals presents some similarities with the problem considered in this paper, these solutions work in a different context and are not applicable to our fragmentation scenario.

## 8 Conclusions

Starting from the observation that a successful approach to data fragmentation for protecting privacy of sensitive associations must take into account possible indirect information exposure due to dependencies among data, in this paper we have extended the fragmentation approach to the consideration of data dependencies. Our approach aims then at providing a comprehensive and natural solution to protect sensitive information whenever data need to be shared, published, externally stored or processed. We believe that the availability of a simple and at the same time powerful and expressive model like the one presented in this paper can lead to an improvement in the management of large data collections, offering the opportunity to realize natural, convenient, and effective data protection solutions for emerging scenarios.

## Acknowledgments

The authors would like to thank Steven Capelli for support in the implementation of the system and in the experimental evaluation. This work was supported in part by: Italian MIUR PRIN project "GenData 2020", a Google Research Award, EC 7FP project PoSecCo (257129), NSF grants CT-20013A and IIP-1266147, ARO grant W911NF-13-1-0317, and ONR grant N00014-13-1-0703.

## References

[1] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Computing*, vol. 16, no. 1, pp. 69–73, Jan/Feb 2012.

[2] N. Virvilis, S. Dritsas, and D. Gritzalis, "A cloud provider-agnostic secure storage protocol," in *Proc. of CRITIS 2010*, Athens, Greece, Sep 2010.

[3] G. Aggarwal *et al.*, "Two can keep a secret: A distributed architecture for secure database services," in *Proc. of CIDR 2005*, Asilomar, CA, Jan 2005.

[4] 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*, vol. 13, no. 3, pp. 22:1–22:33, Jul 2010.

[5] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Fragments and loose associations: Respecting privacy in data publishing," *PVLDB*, vol. 3, no. 1, pp. 1370–1381, Sep 2010.

[6] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Selective data outsourcing for enforcing privacy," *JCS*, vol. 19, no. 3, pp. 531–566, 2011.

[7] 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 DBSec 2013*, Newark, NJ, Jul 2013.

[8] P. Samarati, "Protecting respondents' identities in microdata release," *IEEE TKDE*, vol. 13, no. 6, pp. 1010–1027, Nov/Dec 2001.

[9] E. Tsang, *Foundations of constraint satisfaction*. Academic press London, 1993, vol. 289.

[10] K. Marriott and P. Stuckey, *Programming with constraints: An introduction*. MIT press, 1998.

[11] M. Triska, "Library (`clpfd`): Constraint Logic Programming over Finite Domains," http://www.swi-prolog.org/man/clpfd.html.

[12] V. Atluri, B. Shafiq, S. Ae Chun, G. Nabi, and J. Vaidya, "UICDS-based information sharing among emergency response application systems," in *Proc. of DG.O 2011*, College Park, MD, Jun 2011.

[13] A. Basu, J. Vaidya, H. Kikuchi, and T. Dimitrakos, "Privacy-preserving collaborative filtering for the cloud," in *Proc. of CloudCom 2011*, Athens, Greece, Nov/Dec 2011.

[14] E. Damiani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Key management for multiuser encrypted databases," in *Proc. of StorageSS 2005*, Fairfax, VA, Nov 2005.

[15] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, and G. Livraga, "Enforcing subscription-based authorization policies in cloud scenarios," in *Proc. of DBSec 2012*, Paris, France, Jul 2012.

[16] H. Hacigümüs, B. Iyer, S. Mehrotra, and C. Li, "Executing SQL over encrypted data in the database-service-provider model," in *Proc. of SIGMOD 2002*, Madison, WI, Jun 2002.

[17] B. Hore, S. Mehrotra, M. Canim, and M. Kantarcioglu, "Secure multidimensional range queries over outsourced data," *VLDB J.*, vol. 21, no. 3, pp. 333–358, Jun 2012.

[18] R. Jhawar and V. Piuri, "Fault tolerance and resilience in cloud computing environments," in *Computer and Information Security Handbook, 2nd Edition*, J. Vacca, Ed. Morgan Kaufmann, 2013.

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

[20] R. Jhawar, V. Piuri, and M. Santambrogio, "Fault tolerance management in cloud computing: A system-level perspective," *IEEE ISJ*, vol. 7, no. 2, pp. 288–297, Jun 2013.

[21] M. Kandias, L. Mitrou, V. Stavrou, and D. Gritzalis, "Which side are you on? A new panopticon vs. privacy," in *Proc. of SECRYPT 2013*, Reykjavik, Iceland, Jul 2013.

[22] M. Kandias, V. Stavrou, N. Bozovic, L. Mitrou, and D. Gritzalis, "Can we trust this user? Predicting insider's attitude via YouTube usage profiling," in *Proc. of ATC 2013*, Italy, Dec 2013.

[23] M. Kandias, N. Virvilis, and D. Gritzalis, "The insider threat in cloud computing," in *Proc. of CRITIS 2011*, Lucerne, Switzerland, Sep 2011.

[24] Y. Li and H. Lu, "Privacy risk assessment with bounds deduced from bounds," *IJUFKS*, vol. 19, no. 4, pp. 685–715, Aug 2011.

[25] K. P. Smith, L. Seligman, and V. Swarup, "Everybody share: The challenge of data-sharing systems," *IEEE Computer*, vol. 41, no. 9, pp. 54–61, Sep 2008.

[26] J. Vaidya, "Privacy in the context of digital government," in *Proc. of DG.O 2012*, College Park, MD, Jun 2012.

[27] V. Varadharajan and U. Tupakula, "TREASURE: Trust Enhanced Security for Cloud Environments," in *Proc. of TRUSTCOM 2012*, Liverpool, UK, Jun 2012.

[28] L. Zhou, V. Varadharajan, and M. Hitchens, "A flexible cryptographic approach for secure data storage in the cloud using role-based access control," *IJCC*, vol. 1, no. 2/3, pp. 201–220, May 2012.

[29] E. Damiani, S. De Capitani di Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati, "Balancing confidentiality and efficiency in untrusted relational DBMSs," in *Proc. of ACM CCS 2003*, Washington, DC, Oct 2003.

[30] P. Samarati and S. De Capitani di Vimercati, "Data protection in outsourcing scenarios: Issues and directions," in *Proc. of ASIACCS 2010*, Beijing, China, Apr 2010.

[31] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Fragmentation design for efficient query execution over sensitive distributed databases," in *Proc. of ICDCS 2009*, Montreal, Canada, Jun 2009.

[32] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, G. Livraga, and P. Samarati, "An OBDD approach to enforce confidentiality and visibility constraints in data publishing," *JCS*, vol. 20, no. 5, pp. 463–508, 2012.

[33] 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, Jul 2013.

[34] 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, Oct 2011.

[35] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "On information leakage by indexes over data fragments," in *Proc. of PrivDB 2013*, Australia, Apr 2013.

[36] ——, "Private data indexes for selective access to outsourced data," in *Proc. of WPES 2011*, Chicago, IL, Oct 2011.

[37] 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, Mar 2010.

[38] H. Wang and R. Liu, "Privacy-preserving publishing data with full functional dependencies," in *Proc. of DASFAA 2010*, Tsukuba, Japan, Apr 2010.

[39] W. Liu, L. Wang, and L. Zhang, "k-jump strategy for preserving privacy in micro-data disclosure," in *Proc. of ICDT 2010*, Lausanne, Switzerland, Mar 2010.

[40] S. Dawson, S. De Capitani di Vimercati, P. Lincoln, and P. Samarati, "Minimal data upgrading to prevent inference and association attacks," in *Proc. of ACM PODS 1999*, Philadelphia, PA, May/Jun 1999.

[41] C. Farkas and S. Jajodia, "The inference problem: A survey," *SIGKDD Explor. Newsl.*, vol. 4, no. 2, pp. 6–11, Dec 2002.


**Sabrina De Capitani di Vimercati** is a professor at the Computer Science Department, Università degli Studi di Milano, Italy. Her research interests are in the area of security, privacy, and data protection. She has been a visiting researcher at SRI International, CA (USA), and George Mason University, VA (USA). She is chair of the IFIP WG 11.3 on Data and Application Security and Privacy.
http://www.di.unimi.it/decapita

**Sara Foresti** is an assistant professor at the Computer Science Department, Università degli Studi di Milano, Italy. Her PhD thesis received the ERCIM STM WG 2010 award. She has been a visiting researcher at George Mason University, VA (USA). She has been serving as PC chair and member of several conferences. Her research interests are in the area of security and privacy.
http://www.di.unimi.it/foresti

**Giovanni Livraga** is a PhD student in Computer Science at the Università degli Studi di Milano, Italy. He has been a visiting researcher at SAP Labs, France, and George Mason University, VA (USA). He has been serving as PC member for several conferences and as reviewer for several journals. His research interests are in the area of security and privacy in emerging scenarios.
http://www.di.unimi.it/livraga

**Sushil Jajodia** is a professor and the director of CSIS at George Mason University, VA (USA). His research interests include information security and privacy. He has authored or coauthored 6 books and more than 425 papers, and edited 41 books and conference proceedings. He holds 12 patents. He has received several awards. He has been named IEEE Fellow (2013).
http://csis.gmu.edu/jajodia

**Stefano Paraboschi** is a professor and deputy-chair at the Dipartimento di Ingegneria of the Università degli Studi di Bergamo, Italy. He has been a visiting researcher at Stanford University and IBM Almaden, CA (USA), and George Mason University, VA (USA). His research focuses on security and privacy, Web technology for data intensive applications, XML, and database technology.
http://cs.unibg.it/parabosc

**Pierangela Samarati** is a professor at the Computer Science Department, Università degli Studi di Milano, Italy. Her main research interests are in data protection, security and privacy. She has published more than 230 papers in journals, conference proceedings, and books. She has received several awards. She has been named ACM Distinguished Scientist (2009) and IEEE Fellow (2012).
http://www.di.unimi.it/samarati

# Supplemental material

The proof of the theorems in Section 5 have been omitted from the paper for space constraints. We provide them as supplemental material.

*Theorem 5.5:* Given a relation schema $R(a_1, \ldots, a_n)$, a fragment $F$ of $R$, and a set $\mathcal{D}$ of data dependencies over $R$, the following implication holds:
$F{=}F^* \iff \forall d{\in}\mathcal{D}$: $d.premise{\not\subseteq}F$ or $d.premise \cup d.consequence \subseteq F$.

*Proof:* By Definition 5.2, if $F{=}F^*$ then $\forall d{\in}\mathcal{D}$, $F^*{\otimes}d{=}F^*$ or equivalently $F{\otimes}d{=}F$. We then need to prove that, $\forall d{\in}\mathcal{D}$, $F{\otimes}d{=}F$ if condition $d.premise{\not\subseteq}F$ or condition $d.premise \cup d.consequence \subseteq F$ hold, and vice-versa.

Let us first assume that $d.premise{\not\subseteq}F$. By Definition 5.1, $F{\otimes}d{=}F$ (second case). Let us now assume that $d.premise{\cup}d.consequence{\subseteq}F$. By Definition 5.1, $F{\otimes}d{=}F{\cup}d.consequence$ since $d.premise{\subseteq}F$ (first case). However, $d.consequence{\subseteq}F$ by hypothesis and therefore $F{\otimes}d{=}F{\cup}d.consequence{=}F$.

Let us now assume that $F{\otimes}d{=}F$. This hypothesis holds in two cases: *i)* $d.premise{\subseteq}F$ and $F{\cup}d.consequence{=}F$ (first case in Definition 5.1), meaning that $d.premise{\cup}d.consequence{\subseteq}F$, or *ii)* $d$ does not apply to $F$, that is, $d.premise{\not\subseteq}F$ (second case in Definition 5.1). □

*Theorem 5.6:* Given a relation schema $R(a_1, \ldots, a_n)$, and sets $\mathcal{C}$, $\mathcal{V}$, and $\mathcal{D}$ of confidentiality constraints, visibility constraints, and data dependencies, respectively, over $R$ the following implication holds:
$\exists\mathcal{F}$ satisfying Definition 5.3 $\iff$ $\exists\mathcal{F}'$ satisfying Definitions 2.4 and 5.4.

*Proof:* We separately prove the two implications.

*$\exists\mathcal{F}$ satisfying Definition 5.3 $\implies$ $\exists\mathcal{F}'$ satisfying Definitions 2.4 and 5.4.*
We prove the implication by contradiction. Let us assume that $\exists\mathcal{F}$ satisfying Definition 5.3, but that $\nexists\mathcal{F}'$ satisfying Definitions 2.4 and 5.4. Let $\mathcal{F}^* = \{F^* : F{\in}\mathcal{F}\}$, that is, the closure of fragmentation $\mathcal{F}$ w.r.t. $\mathcal{D}$ (Definition 5.2). By assumption, $\mathcal{F}^*$ violates Definitions 2.4 and 5.4. We now consider each of the conditions in Definition 2.4, and the condition in Definition 5.4 separately.

1) *Confidentiality*: $\mathcal{F}^*$ violates confidentiality iff $\exists c{=}\{a_i, \ldots, a_j\}{\in}\mathcal{C}$ and $\exists F'{\in}\mathcal{F}^*$ such that $c{\subseteq}F'$. Since $F'$ is the closure $F^*$ of a fragment $F$ in $\mathcal{F}$, then $F^*$ violates $c$, contradicting the hypothesis that $\mathcal{F}$ satisfies Definition 5.3 (Condition 1 would be violated).
2) *Visibility*: $\mathcal{F}^*$ violates visibility iff $\exists v{\in}\mathcal{V}$ such that $\forall F'{\in}\mathcal{F}^*$, $F'$ does not satisfy $v$. Since visibility constraints are monotonic Boolean formulas and for each fragment $F$ in $\mathcal{F}$ there is a fragment $F'$ in $\mathcal{F}^*$ such that $F'$ is the closure $F^*$ of $F$, this contradicts the hypothesis that $\mathcal{F}$ satisfies Definition 5.3 (Condition 2 would be violated). In fact, $F{\subseteq}F^*$ by Definition 5.2 and then if $F$ satisfies $v$ also $F^*$ satisfies it.
3) *Unlinkability*: $\mathcal{F}^*$ violates unlinkability iff $\exists F_i{,}F_j{\in}\mathcal{F}^*$ such that $F_i{\cap}F_j{\neq}\emptyset$. Since $F_i{=}F_k^*$ and $F_j{=}F_l^*$ with $F_k{,}F_l{\in}\mathcal{F}$, this is equivalent to say that $F_k^*{\cap}F_l^*{\neq}\emptyset$, contradicting the hypothesis that $\mathcal{F}$ satisfies Definition 5.3 (Condition 3 would be violated).
4) *Closure*: $\mathcal{F}^*$ violates this condition iff $\exists d{\in}\mathcal{D}$ and $\exists F'{\in}\mathcal{F}^*$ such that $\nexists F{\in}\mathcal{F}$ such that $F'{=}F^*$. Since $F'$ is, by hypothesis, the closure of a fragment $F$ in $\mathcal{F}$, by Theorem 5.5, either $d.premise{\not\subseteq}F'$ or $d.consequence{\subseteq}F'$. This contradicts the hypothesis that $\mathcal{F}^*$ is the closure of $\mathcal{F}$.

*$\exists\mathcal{F}'$ satisfying Definitions 2.4 and 5.4 $\implies$ $\exists\mathcal{F}$ satisfying Definition 5.3.*
Let us assume, by contradiction, that $\exists\mathcal{F}'$ satisfying Definitions 2.4 and 5.4, but that $\nexists\mathcal{F}$ satisfying Definition 5.3. Before analyzing each condition separately, we recall that for each $F{\in}\mathcal{F}'$, $F{=}F^*$ by Definition 5.4.

1) *Confidentiality*: $\mathcal{F}'$ violates confidentiality iff $\exists c{=}\{a_i, \ldots, a_j\}{\in}\mathcal{C}$ and $\exists F{\in}\mathcal{F}'$ such that $c{\subseteq}F^*$. Since $F{=}F^*$ this implies that $c{\subseteq}F$ contradicting the hypothesis that $\mathcal{F}'$ satisfies Definition 2.4 (Condition 1 would be violated).
2) *Visibility*: $\mathcal{F}'$ violates confidentiality iff $\exists v{\in}\mathcal{V}$ such that $\forall F{\in}\mathcal{F}'$, $F$ does not satisfy $v$. This contradicts the hypothesis that $\mathcal{F}'$ satisfies Definition 2.4 (Condition 2 would be violated).
3) *Unlinkability*: $\mathcal{F}'$ violates unlinkability iff $\exists F_i{,}F_j{\in}\mathcal{F}'$, $i \neq j$, such that $F_i^*{\cap}F_j^*{\neq}\emptyset$. Since $F_i^*{=}F_i$ and $F_j^*{=}F_j$, then $F_i{\cap}F_j{\neq}\emptyset$, contradicting the hypothesis that $\mathcal{F}'$ satisfies Definition 2.4 (Condition 3 would be violated).

□