On the Inference-Proofness of Database Fragmentation Satisfying Confidentiality Constraints[*]

Joachim Biskup[1], Marcel Preuß[1], and Lena Wiese[2]

[1] Technische Universität Dortmund, Dortmund, Germany
{biskup,preuss}@ls6.cs.tu-dortmund.de
[2] National Institute of Informatics, Tokyo, Japan
wiese@nii.ac.jp

Abstract. Confidentiality of information should be preserved despite the emergence of data outsourcing. An existing approach is supposed to achieve confidentiality by vertical fragmentation and without relying on encryption. Although prohibiting unauthorised (direct) accesses to confidential information, this approach has so far ignored the fact that attackers might infer sensitive information logically by deduction. In this article vertical fragmentation is modelled within the framework of Controlled Query Evaluation (CQE) allowing for inference-proof answering of queries. Within this modelling the inference-proofness of fragmentation is proved formally, even if an attacker has some a priori knowledge in terms of a rather general class of semantic database constraints.

Keywords: Database Security, Information Dissemination Control, Inference-Proofness, First-Order Logic, Outsourcing, Fragmentation.

1 Introduction

In these days information has become one of the most important resources, which has to be protected. In order to protect information from undesired disclosures, confidentiality requirements are declared by setting up a confidentiality policy. According to such a confidentiality policy a system should enforce the declared confidentiality requirements autonomously as for example surveyed in [3].

Moreover, there is an increasing need for storing data cost-efficiently in our economy-driven society. One approach to achieve this goal is called "database as a service" paradigm and leads to third party service providers specialized on hosting database systems and offering the use of these database systems to their customers via Internet in return for payment of rent [12]. These customers may save money because they are freed from purchasing expensive hard- and software and dealing with difficult administrative and maintenance tasks such as upgrading hard- and software or eliminating technical malfunctions.

[*] This work has been partially supported by the DFG (SFB 876/A5), and a postdoctoral research grant of the German Academic Exchange Service (DAAD).

Patient	SSN	Name	DoB	ZIP	Illness	Doctor
	12345	Hellmann	03.01.1981	94142	Hypertension	White
	98765	Dooley	07.10.1953	94141	Obesity	Warren
	24689	McKinley	12.02.1952	94142	Hypertension	White
	13579	Ripley	03.01.1981	94139	Obesity	Warren

Fig. 1. Example of a relational instance containing sensitive associations

Obviously, there is a goal conflict between the discussed "database as a service" paradigm and confidentiality requirements because the service provider cannot be restrained from reading all cleartext information stored in its systems. One natural approach to cope with that conflict lies in encrypting all outsourced data on the user side [12,13]. But, unfortunately, such an approach often makes the efficient evaluation of queries on the server side impossible [2,9].

The benefit of encryption of data lies in making these data – and also the information contained in these data – illegible. But often, in relational database systems single pieces of information are not confidential per se. Due to the storage of data according to some (static) relational schema, semantic associations between different pieces of information are represented and often only these associations are confidential [9]. For example, in a hospital the list of illnesses cured and the list of patients are both not particularly sensitive per se. In contrast, an association between a patient's name and a specific illness is very sensitive and has to be protected. An example adapted from [9] of a relational instance containing this sensitive association among others is given in Fig. 1.

To achieve this protection, some authors suggest to break sensitive associations by splitting relational instances vertically, which is referred to as vertical fragmentation. There are several different approaches to achieving confidentiality based on vertical fragmentation surveyed in [13] and for each of these approaches the corresponding authors describe how fragments of an original relational instance can be outsourced so that unauthorised (direct) accesses to confidential information are prohibited. But it is *not* shown that confidential information cannot be inferred by employing inferences, which may offer the possibility to infer confidential information based on the knowledge of non-confidential information [11]. Moreover, it is *not* considered that an attacker often has some a priori knowledge, which might enable him to infer confidential information [6].

In contrast, there are several approaches to so-called Controlled Query Evaluation (CQE) surveyed in [4] and for each of these approaches it is proven that a declared confidentiality policy is enforced so that any harmful inferences are avoided. "Inference-proofness" is achieved by limiting a user's information gain so that this user cannot infer protected information reliably based on his a priori knowledge and the (possibly distorted) answers to his queries.

The main novel contribution of this article consists of a formal analysis of a specific approach to vertical fragmentation – splitting a relational instance into one externally stored part and one locally-held part – w.r.t. its inference-proofness. More specifically, based on the seminal ideas proposed in [7,8], a formalisation of this approach to vertical fragmentation is developed in Sect. 2. After introducing the framework of CQE briefly in Sect. 3, a logic-oriented

F_o	tid	SSN	Name	DoB
	1	12345	Hellmann	03.01.1981
	2	98765	Dooley	07.10.1953
	3	24689	McKinley	12.02.1952
	4	13579	Ripley	03.01.1981

F_s	tid	ZIP	Illness	Doctor
	1	94142	Hypertension	White
	2	94141	Obesity	Warren
	3	94142	Hypertension	White
	4	94139	Obesity	Warren

Fig. 2. Possible fragmentation of the instance given in Fig. 1

modelling of the approach to vertical fragmentation presented in Sect. 2 within the framework of CQE is introduced in Sect. 4 and subsequently analysed w.r.t. its inference-proofness in Sect. 5. Thereby an attacker's a priori knowledge in terms of a rather general class of semantic database constraints is respected.

2 Confidentiality by Fragmentation

Now, the approach to vertical fragmentation (in the following simply referred to as fragmentation) presented in [7,8] is extended. In this approach all data is stored in a single relational instance r over a relational schema $\langle R|A_R|SC_R\rangle$ with relational symbol R and the set $A_R = \{a_1, \ldots, a_n\}$ of attributes. Moreover, the set SC_R contains some semantic (database) constraints, which must be satisfied by each relational instance constructed over this schema. Note that semantic constraints are not considered in [7,8] (hence there $SC_R = \emptyset$).

The approach considered is built on the assumption of a client-server architecture, in which the server is managed by a third party service provider. This third party service provider is not considered to be trustworthy in terms of confidentiality and might actively monitor all queries processed and all data stored on its server. But it is assumed to be guaranteed that this service provider does not manipulate data maliciously so that all data received from the server is always correct in terms of integrity. The client used in this architecture is assumed to be completely trustworthy and also has the ability to store (a limited amount of) data locally. But as this local storage is assumed to be more expensive than the external storage, it is desirable to store as much data as possible on the server.

The idea for achieving confidentiality despite outsourcing (some) data lies in splitting the original instance r over schema $\langle R|A_R|SC_R\rangle$ into two fragment instances f_o and f_s stored instead of r. While f_s may be outsourced to an external server, f_o can only be stored locally on the client.[1] To build f_o and f_s, the attribute set A_R of schema $\langle R|A_R|SC_R\rangle$ is partitioned into two sets $\bar{A}_{F_o}$ and $\bar{A}_{F_s}$ (items (i), (iii) of Def. 1). Then fragment instance f_o (f_s, respectively) is in essence the projection of r on $\bar{A}_{F_o}$ ($\bar{A}_{F_s}$) (item (a) of Def. 1). Obviously, in terms of confidentiality no sensitive information or association is allowed to be contained in fragment instance f_s. Such a fragmentation of the instance given in Fig. 1 (e.g., breaking the name-illness association) is depicted in Fig. 2.

As an authorised user having access to the client as well as to the server should be able to query all information contained in the original instance r, the reconstructability of r based on the fragments f_o and f_s must be guaranteed. For

[1] Index s is for server and index o is for owner-side (local) storage.

this purpose each tuple in f_o and f_s is extended by a tuple ID (item (i) of Def. 1) so that in f_o and f_s exactly those two tuples (one tuple per fragment instance) which together correspond to a tuple of r share the same tuple ID (item (c) of Def. 1) being unique in f_o as well as in f_s (item (b) of Def. 1). Strategies for processing SQL queries referring to the original instance on the corresponding fragment instances efficiently are discussed in [7].

Based on the seminal ideas proposed in [7,8] a formalisation of this concept of fragmentation is developed in this article as follows:

Definition 1 (Fragmentation). *Given a relational schema $\langle R|A_R|SC_R\rangle$, a* vertical fragmentation *$\mathcal{F}$ of $\langle R|A_R|SC_R\rangle$ is a set*

$$\mathcal{F} = \{\langle F_o|A_{F_o}|SC_{F_o}\rangle, \langle F_s|A_{F_s}|SC_{F_s}\rangle\}$$

in which $\langle F_o|A_{F_o}|SC_{F_o}\rangle$ and $\langle F_s|A_{F_s}|SC_{F_s}\rangle$ are relational schemas called fragments *of $\mathcal{F}$. Moreover, for $i \in \{o, s\}$, it holds that*

(i) $A_{F_i} := \{a_{tid}\} \cup \bar{A}_{F_i}$ with $a_{tid} \notin A_R$ and $\bar{A}_{F_i} \subseteq A_R$,
(ii) $SC_{F_i} := \{a_{tid} \rightarrow \bar{A}_{F_i}\}$ with $a_{tid} \rightarrow \bar{A}_{F_i}$ being a functional dependency,
(iii) $\bar{A}_{F_o} \cup \bar{A}_{F_s} = A_R$ and $\bar{A}_{F_o} \cap \bar{A}_{F_s} = \emptyset$.

Given a relational instance r over $\langle R|A_R|SC_R\rangle$, the fragment instances *f_o and f_s over $\langle F_o|A_{F_o}|SC_{F_o}\rangle$ and $\langle F_s|A_{F_s}|SC_{F_s}\rangle$ are created by inserting both the tuple ν_o into f_o and the tuple ν_s into f_s for each tuple $\mu \in r$. Thereby, for $i \in \{o, s\}$:*

(a) $\nu_i[a] = \mu[a]$ for each attribute $a \in \bar{A}_{F_i}$,
(b) $\nu'[a_{tid}] \neq \nu''[a_{tid}]$ for tuples $\nu', \nu'' \in f_i$ with $\nu' \neq \nu''$,
(c) $\nu_o[a_{tid}] = \nu_s[a_{tid}]$ for attribute $a_{tid} \in A_{F_o}, A_{F_s}$.

Other tuples do not exist in f_o and f_s.

Note that even for two different tuples of r which are equal w.r.t. all attributes of $\bar{A}_{F_s}$ ($\bar{A}_{F_o}$, respectively) there are also two different tuples in f_s (f_o) which are equal w.r.t. all attributes of $\bar{A}_{F_s}$ ($\bar{A}_{F_o}$) because of the existence of unique tuple IDs. Hence, for each tuple μ of r there is *exactly* one tuple ν_s in f_s as well as one tuple ν_o in f_o. If there were no tuple IDs, all duplicates of tuples in f_s (f_o) would be removed. In terms of the example in Fig. 2 the first and the third tuple of f_s would be consolidated without the existence of tuple IDs.

As the goal is to achieve confidentiality by fragmentation, a formal declaration of confidentiality requirements is indispensable. In [7,8] this is obtained by defining a set of so-called confidentiality constraints on the schema level.

Definition 2 (Confidentiality Constraint). *Let $\langle R|A_R|SC_R\rangle$ be a relational schema. A* confidentiality constraint *c over $\langle R|A_R|SC_R\rangle$ is a subset $c \subseteq A_R$.*

Semantically a confidentiality constraint c claims that each combination of values allocated to the set $c \subseteq A_R$ of attributes in an instance r over schema $\langle R|A_R|SC_R\rangle$ should not be contained completely in f_s. In f_o such a combination of values may be contained completely since f_o is only stored locally.

$$c_1 = \{\texttt{SSN}\} \qquad c_3 = \{\texttt{Name}, \texttt{Illness}\}$$
$$c_2 = \{\texttt{Name}, \texttt{DoB}\} \qquad c_4 = \{\texttt{DoB}, \texttt{ZIP}, \texttt{Illness}\}$$

Fig. 3. Set $\mathcal{C}$ of confidentiality constraints over *Patient*

Definition 3 (Confidentiality of Fragmentation). *Let $\langle R|A_R|SC_R\rangle$ be a relational schema, $\mathcal{F}$ a fragmentation of $\langle R|A_R|SC_R\rangle$ according to Def. 1 and $\mathcal{C}$ a set of confidentiality constraints over $\langle R|A_R|SC_R\rangle$ according to Def. 2. Fragmentation $\mathcal{F}$ is* confidential *w.r.t. $\mathcal{C}$ iff $c \not\subseteq A_{F_s}$ for each $c \in \mathcal{C}$.*

Note that in case of a singleton constraint $c = \{a_i\}$ a fragmentation can only be confidential if the column of values allocated to a_i in an instance r over $\langle R|A_R|SC_R\rangle$ is not contained in f_s. So, a singleton constraint states that the values allocated to an attribute are sensitive per se. In case of a non-singleton constraint c at least one attribute $a_i \in c$ must not be contained in A_{F_s} and as a consequence of that the corresponding column of an instance r over $\langle R|A_R|SC_R\rangle$ is not in f_s. But in terms of Def. 3 it is irrelevant which of the attributes in c is chosen for not being in A_{F_s} and hence only associations between values allocated to the attributes of c in an instance r over $\langle R|A_R|SC_R\rangle$ are protected.

An example of a set of confidentiality constraints in terms of the running example introduced in Fig. 1 is given in Fig. 3. The fragmentation depicted in Fig. 2 is confidential w.r.t. this set of confidentiality constraints.

In terms of Def. 3 one trivial but feasible solution always is to store all data locally on the client. But since as much data as possible should be stored externally, an optimization problem proven to be NP-hard in [8] has to be solved. To achieve that, an approximation algorithm and several metrics to compare the qualities of computed solutions are presented in [8].

3 Controlled Query Evaluation

In the remainder of this article the inference-proofness of fragmentation is discussed based on a logic-oriented modelling of fragmentation within the framework of Controlled Query Evaluation (CQE). This framework comprises several inference-proof approaches to CQE, which are all based on the same classes of components. These classes and the general procedures of CQE will be introduced briefly on a fairly abstract level based on [4] now.

CQE is a framework with a server hosting a database instance. Although answering queries sent by users, one of the goals of the CQE system is to limit a user's information gain – even by considering information that a user could possibly obtain by employing logical inferences – according to some confidentiality policy. This is achieved by determining each piece of information a (rational) user can possibly infer based on his knowledge before interacting with this user.

To be able to do so, the monitoring of raw data a user receives as answers to his queries is not sufficient. The information contained in these data has to be extracted and represented suitably so that it can be processed. For that purpose the information system considered is assumed to be logic-based in the sense that its database instance is represented by a set of (closed) formulas of a well-defined language of logic (e.g., first-order logic) and the semantics of query evaluation is founded on the (well-defined) notions of validity and implication defined in the context of the semantics of this language.

As inferences can be often drawn by combining several pieces of knowledge, the computation of all inferences a user can employ (based on logical implications) presupposes that the CQE system needs to be aware of the complete knowledge this user has. In case of dynamic CQE – which aims at controlling a user's information gain at runtime – this obviously means that all answers a user receives in response to his queries have to be recorded as a set of formulas in order to be able to decide whether this knowledge combined with the (correct) answers to subsequent queries provides a basis for drawing harmful inferences.

Regardless of using dynamic or static CQE, a user's a priori knowledge expressed as a set of formulas always has to be considered. This a priori knowledge comprises knowledge a user has independently of answers given by the information system considered (e.g., semantic constraints declared for the schema of a relational database or knowledge about the world in general). Although not being harmful per se, such a priori knowledge combined with (uncontrolled) answers to his queries might enable a user to draw some harmful inferences.

To express the knowledge to be kept secret from a specific user, a confidentiality policy in terms of a set of potential secrets is set up for each user. A potential secret Ψ is a formula expressing that the pertinent user must not be able to infer that the information embodied in Ψ is true in the database considered. So, regardless of whether Ψ is actually true in this database or not, from the point of view of this user (established by his a priori knowledge and answers to his queries) it must always be possible that Ψ is *not* true. The conservative approach that a user is aware of the policy set up for him is usually followed.

As already hinted above, there are two general modes of inference control. The dynamic mode controls each answer to a user's query at runtime and therefore the CQE system has to check whether the user's (assumed) knowledge combined with the (correct) answer to his query could enable him to infer some knowledge declared as confidential (i.e., knowledge embodied in a potential secret is implied logically). If this is the case, the system has to distort the answer suitably by lying (i.e., giving a wrong answer) or by refusing an answer at all. The combined usage of both techniques is possible, too.

In static mode the CQE system precomputes an alternative database instance for each user, which is inference-proof according to the confidentiality policy set up for the pertinent user. Although being as close as possible to the original database instance, the alternative instance is distorted by lies or refusals (i.e., missing values) so that the user can query it freely without receiving knowledge enabling him to draw harmful inferences. So, corresponding to the idea of fragmentation that the server knows the externally stored fragment completely, the user's knowledge may comprise the complete alternative instance.

4 A Logic-Oriented View on Fragmentation

The goal of this article is to discuss the inference-proofness of the approach to fragmentation presented in Sect. 2. As CQE is known to be inference-proof, the main idea is to model fragmentation within the framework of CQE. As CQE relies on a logic-oriented view on databases, the approach to fragmentation discussed in Sect. 2 has to be modelled logic-orientedly, too.

For that purpose a language $\mathscr{L}$ of first-order logic with equality, which is suitable for modelling fragmentation logic-orientedly, is presented now. As the externally stored fragment instance f_s over $\langle F_s|A_{F_s}|SC_{F_s}\rangle$, which is assumed to be known to an attacker, must be modelled in $\mathscr{L}$, the set $\mathcal{P}$ of predicate symbols of $\mathscr{L}$ contains the predicate symbol $F_s \in \mathcal{P}$ with arity $k+1 = |A_{F_s}|$ (including the additional tuple ID attribute plus k original attributes (cf. Fig. 4)). As security should not rely on obscurity, it is assumed that an attacker is aware of the process of fragmentation and knows both the computed fragmentation $\mathcal{F}$ and the schema $\langle R|A_R|SC_R\rangle$ over which the original instance r – being the target of his attacks – is built. To be able to model an attacker's knowledge about r based on these assumptions, language $\mathscr{L}$ also contains a predicate symbol $R \in \mathcal{P}$ with arity $n = |A_R|$. Moreover, there is a distinguished binary predicate symbol $= \notin \mathcal{P}$ for expressing equality. A predicate symbol F_o is *not* needed since an attacker does not have access to the client by assumption.

The language $\mathscr{L}$ also comprises the set *Dom* of constant symbols, which will be employed for the universe of interpretations for $\mathscr{L}$ as well. In compliance with other approaches to CQE (e.g., [5]) this set is assumed to be fixed and infinite. Further, $\mathscr{L}$ includes an infinite set $\mathit{Var} = \{X_1, X_2, \ldots\}$ of variables.

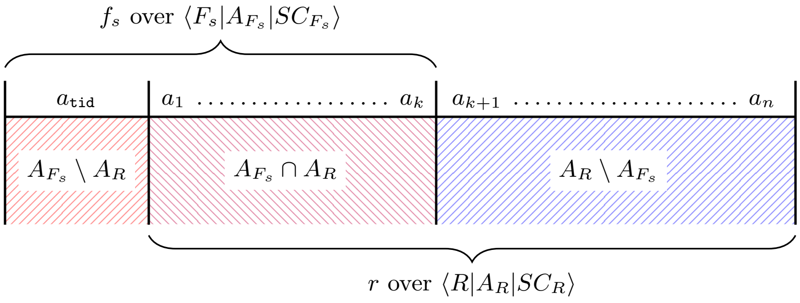
All formulas contained in $\mathscr{L}$ are constructed inductively in the natural fashion using the quantifiers $\forall$ and $\exists$ and the connectives $\neg$, $\wedge$, $\vee$ and $\Rightarrow$. Thereby each term is either a constant or a variable (functions are not allowed) and each variable is quantified (only closed formulas are in $\mathscr{L}$).

This syntactic specification has to be complemented with an appropriate semantics in which the characteristics of databases are reflected. Such a semantics is established by a so-called DB-Interpretation according to [5]:

Definition 4 (DB-Interpretation). *Given the language $\mathscr{L}$ of first-order logic with a fixed infinite set of constant symbols Dom and a finite set $\mathcal{P}$ of predicate symbols, an interpretation $\mathcal{I}$ over a universe $\mathcal{U}$ is a DB-Interpretation for $\mathscr{L}$ iff*

(i) $\mathcal{U} = \mathit{Dom}$,
(ii) $\mathcal{I}(v) = v \in \mathcal{U}$ *holds for every constant symbol* $v \in \mathit{Dom}$,
(iii) *every* $P \in \mathcal{P}$ *with arity* m *is interpreted by a finite relation* $\mathcal{I}(P) \subset \mathcal{U}^m$,
(iv) *the predicate symbol* $= \notin \mathcal{P}$ *is interpreted by* $\mathcal{I}(=) = \{(v, v) \mid v \in \mathcal{U}\}$.

The semantics of satisfaction of formulas in $\mathscr{L}$ by a DB-Interpretation is the same as in usual first-order logic. A set $\mathcal{S} \subset \mathscr{L}$ of formulas implies a formula $\Phi \in \mathscr{L}$ (written as $\mathcal{S} \models_{DB} \Phi$) iff each DB-Interpretation $\mathcal{I}$ satisfying $\mathcal{S}$ (written as $\mathcal{I} \models_M \mathcal{S}$) also satisfies Φ (written as $\mathcal{I} \models_M \Phi$).



f_s over $\langle F_s | A_{F_s} | SC_{F_s} \rangle$
$a_{\texttt{tid}}$
a_1 .... a_k
a_{k+1} .... a_n
$A_{F_s} \setminus A_R$
$A_{F_s} \cap A_R$
$A_R \setminus A_{F_s}$
r over $\langle R | A_R | SC_R \rangle$

Fig. 4. Rearrangement of columns of r

$$
\begin{array}{rllllll}
db^+_{f_s} = \{ & F_s\,(& 1, & 94142, & \text{Hypertension}, & \text{White} &), \\
 & F_s\,(& 2, & 94141, & \text{Obesity}, & \text{Warren} &), \\
 & F_s\,(& 3, & 94142, & \text{Hypertension}, & \text{White} &), \\
 & F_s\,(& 4, & 94139, & \text{Obesity}, & \text{Warren} &)\ \}
\end{array}
$$

Fig. 5. Positive knowledge of f_s in a logic-oriented model

From now on suppose w.l.o.g. that the columns of a relational instance r under investigation are rearranged so that the first k columns of r correspond to the projection of f_s on $A_{F_s} \cap A_R$. This convention is visualised in Fig. 4.

As an attacker is supposed to know the outsourced fragment instance f_s, the knowledge contained in f_s obviously has to be modelled within the logic-oriented view representing an attacker's knowledge. The positive knowledge in terms of the tuples explicitly recorded in f_s can be simply modelled logic-orientedly by adding an atomic formula $F_s(\nu[a_{\texttt{tid}}], \nu[a_1], \dots, \nu[a_k])$ for each tuple $\nu \in f_s$. Regarding the fragmentation of Fig. 2 such a set of formulas is given in Fig. 5

But as the original instance r – and so its fragment instance f_s – is assumed to be complete[2], each piece of information *not* contained in r (f_s, respectively) is considered to be *not* valid by Closed World Assumption (CWA). Hence, an attacker knows that each combination of values $(v_{\texttt{tid}}, v_1, \dots, v_k) \in \mathit{Dom}^{k+1}$ not contained in any tuple of f_s is not true. This implicitly induces information expressed as $\neg F_s(v_{\texttt{tid}}, v_1, \dots, v_k)$. But as Dom is infinite, there is also an infinite number of such combinations not contained in the finite instance f_s.

As this negative knowledge is not explicitly enumerable, it is expressed implicitly by a so-called completeness sentence (cf. [5]) having a universally quantified variable X_j for each attribute $a_j \in A_{F_s}$. This completeness sentence is constructed so that it is satisfied by a DB-Interpretation $\mathcal{I}$ iff $\mathcal{I}$ satisfies each formula $\neg F_s(v_{\texttt{tid}}, v_1, \dots, v_k)$ with $(v_{\texttt{tid}}, v_1, \dots, v_k) \in \mathit{Dom}^{k+1}$ being a constant combination (substituting the universally quantified variables $X_{\texttt{tid}}, X_1, \dots X_k$ of the completeness sentence) which is *not* contained in any tuple of f_s.

[2] As there are no statements about the completeness of r or f_s in [7,8] this article relies on the assumption of complete instances.

In terms of the running example the knowledge implicitly known to be *not* valid by CWA can be expressed as the following completeness sentence:

$$\begin{array}{lllllllll}
(\forall X_t)(\forall X_Z)(\forall X_I)(\forall X_D)\ [& & & & & & & & \\
(X_t = 1 & \wedge & X_Z = 94142 & \wedge & X_I = \text{Hypert.} & \wedge & X_D = \text{White}) & \vee \\
(X_t = 2 & \wedge & X_Z = 94141 & \wedge & X_I = \text{Obesity} & \wedge & X_D = \text{Warren}) & \vee \\
(X_t = 3 & \wedge & X_Z = 94142 & \wedge & X_I = \text{Hypert.} & \wedge & X_D = \text{White}) & \vee \\
(X_t = 4 & \wedge & X_Z = 94139 & \wedge & X_I = \text{Obesity} & \wedge & X_D = \text{Warren}) & \vee \\
\neg F_s(X_t, X_Z, X_I, X_D) & & & & & & &]
\end{array}$$

Based on this insight an attacker's knowledge about the fragment instance f_s can be formalised logic-orientedly as follows:

Definition 5 (Logic-Oriented View on f_s). *Given a fragment instance f_s over $\langle F_s|A_{F_s}|SC_{F_s}\rangle$ according to Def. 1 with $A_{F_s} = \{a_{tid}, a_1, \ldots, a_k\}$, the positive knowledge contained in f_s is modelled in $\mathscr{L}$ by the set of formulas*

$$db^+_{f_s} := \{F_s(\nu[a_{tid}], \nu[a_1], \ldots, \nu[a_k]) \mid \nu \in f_s\}\ . \tag{1}$$

The implicit negative knowledge contained in f_s is modelled in $\mathscr{L}$ by the singleton set $db^-_{f_s}$ containing the completeness sentence

$$(\forall X_{tid})\ldots(\forall X_k)\left[\bigvee_{\nu\in f_s}\left(\bigwedge_{a_j\in A_{F_s}}(X_j = \nu[a_j])\right) \vee \neg F_s(X_{tid}, X_1, \ldots, X_k)\right]. \tag{2}$$

Moreover the functional dependency $a_{tid} \rightarrow \{a_1, \ldots, a_k\} \in SC_{F_s}$ is modelled in $\mathscr{L}$ by the singleton set fd_{F_s} containing the formula

$$\begin{aligned}(\forall X_{tid})\,(\forall X_1)\ldots(\forall X_k)\,(\forall X'_1)\ldots(\forall X'_k)\,[\,F_s(X_{tid}, X_1, \ldots, X_k) \wedge \\ F_s(X_{tid}, X'_1, \ldots, X'_k) \Rightarrow (X_1 = X'_1) \wedge \ldots \wedge (X_k = X'_k)\,]\end{aligned} \tag{3}$$

Overall the logic-oriented view on f_s in $\mathscr{L}$ is $db_{f_s} := db^+_{f_s} \cup db^-_{f_s} \cup fd_{F_s}$.

As already stated above, an attacker is assumed to know the process of fragmentation. This allows him to know that for each tuple $\nu \in f_s$ there is also a tuple $\mu \in r$ which is equal to ν w.r.t. the values allocated to the attributes of $A_{F_s} \cap A_R$. Moreover, being aware of both $\langle F_s|A_{F_s}|SC_{F_s}\rangle$ and $\langle R|A_R|SC_R\rangle$, an attacker knows that the values allocated to the attributes of $A_R \setminus A_{F_s}$ are kept hidden from him in each tuple of r. Regarding the logic-oriented modelling of an attacker's knowledge the ignorance of these values can be stated by using an existentially quantified variable for each term representing such a value.

Moreover – because of the completeness of r and f_s – an attacker knows that for each combination of values $(v_{\texttt{tid}}, v_1, \ldots, v_k) \in Dom^{k+1}$ not contained in any tuple of f_s, there is no tuple $\mu \in r$ with $\mu[a_j] = v_j$ for each $j \in \{1, \ldots, k\}$. Otherwise there would be a tuple $\nu \in f_s$ containing this combination of values because of the process of fragmentation. So, equivalently, for each tuple $\mu \in r$ there exists a tuple $\nu \in f_s$ with $\nu[a_j] = \mu[a_j]$ for each $j \in \{1, \ldots, k\}$.

As already mentioned in Sect. 2, there is *exactly* one tuple in f_s for each tuple of r because of the existence of unique tuple IDs in f_s. So, if there are two different tuples $\nu_1, \nu_2 \in f_s$ being equal w.r.t. the values allocated to the attributes of $A_{F_s} \cap A_R$, an attacker can reason that there are also two tuples $\mu_1, \mu_2 \in r$ which are equal w.r.t. the values allocated to $A_{F_s} \cap A_R$, but differ in at least one of the values allocated to $A_R \setminus A_{F_s}$. Otherwise r would have two equal tuples $\mu_1 = \mu_2$ which is not possible in relational instances.

For now neglecting semantic constraints of the schema of r (i.e., $SC_R = \emptyset$), a logic-oriented view on the (hidden) original instance r based on the knowledge of the outsourced fragment instance f_s can be modelled as follows:

Definition 6 (Logic-Oriented View on r). *Let $\langle F_s | A_{F_s} | SC_{F_s} \rangle$ with $A_{F_s} = \{a_{tid}, a_1, \ldots, a_k\}$ be the outsourced fragment of a fragmentation $\mathcal{F}$ of a relational schema $\langle R | A_R | SC_R \rangle$ with $A_R = \{a_1, \ldots, a_k, \ldots, a_n\}$ and let f_s be a fragment instance over $\langle F_s | A_{F_s} | SC_{F_s} \rangle$ w.r.t. a relational instance r over $\langle R | A_R | SC_R \rangle$. The knowledge about r received from f_s is expressed by*

$$\begin{aligned}&(\forall X_1) \ldots (\forall X_k) \, [\, (\exists X_{tid}) \, F_s(X_{tid}, X_1, \ldots, X_k) \Leftrightarrow \\ &(\exists X_{k+1}) \ldots (\exists X_n) \, R(X_1, \ldots, X_k, X_{k+1}, \ldots, X_n) \,]\end{aligned} \tag{4}$$

and the knowledge received from preserving duplicates in f_s is expressed by

$$\begin{aligned}&(\forall X_1) \ldots (\forall X_k) \, [\, (\exists X_{tid}) \, (\exists X'_{tid}) \, [\, F_s(X_{tid}, X_1, \ldots, X_k) \wedge \\ &\qquad F_s(X'_{tid}, X_1, \ldots, X_k) \wedge (X_{tid} \neq X'_{tid}) \,] \Rightarrow \\ &(\exists X_{k+1}) \ldots (\exists X_n) \left(\exists X'_{k+1}\right) \ldots (\exists X'_n) \, [\, R(X_1, \ldots, X_k, X_{k+1}, \ldots, X_n) \wedge \\ &\qquad R(X_1, \ldots, X_k, X'_{k+1}, \ldots, X'_n) \wedge \bigvee_{j=k+1}^{n} (X_j \neq X'_j) \,] \,] \ .\end{aligned} \tag{5}$$

This view on r is referred to as the set of formulas db_r containing (4) and (5).

Before the inference-proofness of fragmentation can be analysed formally, the confidentiality policy according to which a fragmentation is computed has to be modelled logic-orientedly, too. A confidentiality constraint $c \subseteq A_R$ claims that each combination of values allocated to the attributes of c should not be revealed to an attacker completely. To specify this semantics more precisely, it is assumed[3] that c only protects those combinations of values which are explicitly allocated to the attributes of c in a tuple of r. In contrast, an attacker may get to know that a certain combination of values is *not* allocated to the attributes of c in any tuple of r. This semantics complies with the semantics of potential secrets known from the CQE framework (cf. Sect. 3).

The wish to protect a certain combination of values $(v_{i_1}, \ldots, v_{i_\ell}) \in Dom^{|c|}$ can be modelled as a potential secret $(\exists \boldsymbol{X}) \, R(t_1, \ldots, t_n)$ in which $t_j := v_j$ holds for each $j \in \{i_1, \ldots, i_\ell\}$. All other terms are existentially quantified variables.

[3] This assumption is needed because the semantics of confidentiality constraints on the instance level is not defined as exactly in [7,8].

But simply modelling each combination of values allocated to the pertinent attributes in r as a potential secret is not sufficient because an attacker is supposed to be aware of the confidentiality policy and could consequently read all sensitive information directly from the policy. This is prevented by protecting *each* combination of values possible according to *Dom*, regardless of whether it is contained in a tuple of r or not. But enumerating all of these combinations explicitly is not manageable since *Dom* is infinite.

Equivalently to the enumeration of all combinations of values, free variables $X_{i_1}, \ldots, X_{i_\ell}$ can be used in a potential secret instead of the constants $v_{i_1}, \ldots, v_{i_\ell}$. Then, one single potential secret per confidentiality constraint is sufficient. For that purpose the extended language $\mathscr{L}^f \supset \mathscr{L}$ of first-order logic expanding $\mathscr{L}$ by free variables is introduced.

Definition 7 (Confidentiality Policy). *Let $\mathcal{C}$ be a set of confidentiality constraints over schema $\langle R|A_R|SC_R\rangle$ according to Def. 2. Considering a confidentiality constraint $c_i \in \mathcal{C}$ with $c_i = \{a_{i_1}, \ldots, a_{i_\ell}\} \subseteq \{a_1, \ldots, a_n\} = A_R$ and the set $A_R \setminus c_i = \{a_{i_{\ell+1}}, \ldots, a_{i_n}\}$, constraint c_i can be modelled as a potential secret*

$$\Psi_i(\boldsymbol{X_i}) \;:=\; (\exists X_{i_{\ell+1}}) \ldots (\exists X_{i_n})\, R(X_1, \ldots, X_n)$$

in the extended language $\mathscr{L}^f$. Thereby $\boldsymbol{X_i} = (X_{i_1}, \ldots, X_{i_\ell})$ is the vector of free variables contained in $\Psi_i(\boldsymbol{X_i})$. The set containing exactly one potential secret $\Psi_i(\boldsymbol{X_i})$ constructed as above for every $c_i \in \mathcal{C}$ is called pot_sec($\mathcal{C}$).

To show the inference-proofness of fragmentation, it has to be proven that none of the potential secrets is implied by the set of formulas representing an attacker's knowledge. This proof cannot be produced (directly) for potential secrets containing free variables because DB-Interpretations are only defined for the language $\mathscr{L}$ not containing free variables. But as free variables of $\mathscr{L}^f$ represent constants of *Dom*, a so-called expansion of such formulas substituting free variables with constants can be constructed to enable the proof.

Definition 8 (Expansion of Formulas). *Let $\Psi(\boldsymbol{X}) \in \mathscr{L}^f$ be a formula containing the vector $\boldsymbol{X} = (X_1, \ldots, X_\ell)$ of free variables. $\Psi(\boldsymbol{X}) \in \mathscr{L}^f$ is expanded to the set of formulas* $\mathrm{ex}(\Psi(\boldsymbol{X})) \subset \mathscr{L}$ *by substituting the free variables $\boldsymbol{X}$ with every constant combination $\boldsymbol{v} = (v_1, \ldots, v_\ell) \in Dom^\ell$, thereby creating a formula $\Psi(\boldsymbol{v}) \in \mathscr{L}$. The expansion of a set $\mathcal{S} \subset \mathscr{L}^f$ is* $\mathrm{ex}(\mathcal{S}) := \bigcup_{\Psi(\boldsymbol{X}) \in \mathcal{S}} \mathrm{ex}(\Psi(\boldsymbol{X}))$.

5 Inference-Proofness of Fragmentation

Until now the logic-oriented model only comprises knowledge an attacker has by knowing the outsourced fragment instance. The a priori knowledge an attacker might have has been completely neglected. But as shown in the following example, an attacker might generally employ this knowledge to draw harmful inferences. In terms of the running example, suppose an attacker knows that `Ripley` is the *only* patient who is treated by doctor `Warren` and lives in a small town with zip code `94139`. By knowing the set of formulas $db^{+}_{f_s}$ (see Fig. 5) and moreover knowing the relationship between f_s and its original instance described by formula (4) of Def. 6, an attacker might reason that `Ripley` suffers from `Obesity`, thereby violating confidentiality constraint c_3 of Fig. 3.

In this article an attacker is supposed to have a priori knowledge about the original instance r in terms of the set SC_R of semantic constraints declared for schema $\langle R|A_R|SC_R\rangle$. Here, SC_R is a set of arbitrary unirelational and typed semantic constraints as long as they belong to the rather general classes of so-called Equality Generating Dependencies (EGDs) or Tuple Generating Dependencies (TGDs), which together comprise nearly all semantic constraints (cf. [1]).

Intuitively expressed, an EGD claims that the presence of some tuples in r implies that certain components of these tuples are equal and a TGD claims that the presence of some tuples in r implies the existence of certain other tuples in r. Moreover, a constraint is unirelational if it refers to only one relational schema, and it is typed if there is an assignment of variables to column positions preventing the claim for equality of values being in different columns of r [1]. In the case of non-typed constraints, for example, an attacker might infer sensitive information based on a non-typed EGD stating that in some tuple of r a (non-hidden) value allocated to an attribute of $\bar{A}_{F_s}$ is equal to a (hidden) value allocated to an attribute of $\bar{A}_{F_o}$. A well known example for a unirelational and typed EGD is a functional dependency and an example for a unirelational and typed TGD is a join dependency.

According to [1,10] unirelational and typed EGDs and TGDs can be formalised as follows:

Definition 9 (Unirelational Typed TGDs/EGDs). *Let $\langle R|A_R|SC_R\rangle$ be a relational schema. Each* unirelational EGD/TGD *contained in SC_R can be expressed in $\mathscr{L}$ by a formula $(\forall \boldsymbol{X})\,[\alpha \Rightarrow \beta]$, in which*

(i) α is a conjunction of atomic formulas of the kind $R(X_1,\dots,X_n)$ with $X_1,\dots,X_n$ being variables of $\boldsymbol{X}$ and every variable of $\boldsymbol{X}$ appears in α,
(ii) in case of a/an
  - *EGD, β is an atomic formula of the kind $(X' = X'')$ with X' and X'' being distinct variables of $\boldsymbol{X}$*
  - *TGD, β is a formula $(\exists\, \boldsymbol{Y})\,\gamma$, in which γ is a conjunction of atomic formulas of the kind $R(X_1,\dots,X_n)$ with $X_1,\dots,X_n$ being variables of $\boldsymbol{X}$ and $\boldsymbol{Y}$.*

A unirelational EGD/TGD is typed *iff the set Var of Variables of $\mathscr{L}$ can be partitioned into n disjoint classes so that, for each atomic formula of the kind $R(X_{i_1},\dots,X_{i_n})$ of α or β, for $1 \le j \le n$, the variable X_{i_j} belongs to class j, and for each atomic formula of the kind $(X' = X'')$ both variables X' and X'' belong to the same class.*

For $n = 3$, for example, $(\forall \boldsymbol{X})\,[\,R(X_1,X_2,X_3) \Rightarrow (X_1 = X_3)\,]$ is a non-typed EGD and $(\forall \boldsymbol{X})\,[\,R(X_1,X_2,X_3) \Rightarrow R(X_1,X_3,X_2)\,]$ is a non-typed TGD. In contrast, $(\forall \boldsymbol{X})\,[\,R(X_1,X_2,X_3) \wedge R(X_1,X_2',X_3') \Rightarrow (X_3 = X_3')\,]$ is a typed EGD and replacing $(X_3 = X_3')$ with $R(X_1,X_2',X_3)$ results in a typed TGD.

Now, the inference-proofness of fragmentation can be proved formally based on the logic-oriented modelling of the view an attacker is supposed to have on the original instance r by knowing the externally stored fragment instance f_s and employing his a priori knowledge. Intuitively expressed, it is shown that a rational attacker always has to consider the existence of an alternative instance r' possible which is – from his point of view constituted by his knowledge – indistinguishable from r and does not violate a potential secret.

Theorem 1 (Inference-Proofness). *Let r be a relational instance over a relational schema $\langle R|A_R|SC_R\rangle$ with $A_R = \{a_1, \ldots, a_n\}$ and let $\mathcal{F}$ be a fragmentation of $\langle R|A_R|SC_R\rangle$ according to Def. 1, which is – according to Def. 3 – confidential w.r.t. a set $\mathcal{C}$ of confidentiality constraints constructed in terms of Def. 2. Moreover, let f_s be the fragment instance over fragment $\langle F_s|A_{F_s}|SC_{F_s}\rangle \in \mathcal{F}$ with $A_{F_s} = \{a_{\mathtt{tid}}, a_1, \ldots, a_k\}$ created w.r.t. instance r. It holds that*

$$\text{for all } \Psi(\boldsymbol{v}) \in \text{ex}(pot\_sec(\mathcal{C})) : \; db_{f_s} \cup db_r \cup prior_{SC_R} \not\models_{DB} \Psi(\boldsymbol{v}) \qquad (6)$$

with $\text{ex}(pot\_sec(\mathcal{C}))$ being the expansion (Def. 8) of $pot\_sec(\mathcal{C})$ constructed according to Def. 7 and db_{f_s} and db_r being constructed according to Def. 5 and Def. 6. Moreover, $prior_{SC_R}$ is a set of unirelational typed TGDs and EGDs contained in SC_R, which are constructed in terms of Def. 9 and satisfied by r.

Proof. To prove formula (6) of Theorem 1, it has to be shown that for an arbitrary $\tilde{\Psi}(\boldsymbol{v}) \in \text{ex}(pot\_sec(\mathcal{C}))$ with $\boldsymbol{v} = (v_{i_1}, \ldots, v_{i_\ell})$ there is a DB-Interpretation $\mathcal{I}^*$ which satisfies db_{f_s}, db_r and $prior_{SC_R}$ and does *not* satisfy $\tilde{\Psi}(\boldsymbol{v})$.

As $\tilde{\Psi}(\boldsymbol{v})$ with $\boldsymbol{v} = (v_{i_1}, \ldots, v_{i_\ell})$ is in $\text{ex}(pot\_sec(\mathcal{C}))$, there has to be the potential secret $\tilde{\Psi}(\boldsymbol{X}) \in pot\_sec(\mathcal{C})$ containing the vector $\boldsymbol{X} = (X_{i_1}, \ldots, X_{i_\ell})$ of free variables and therefore, by construction of $pot\_sec(\mathcal{C})$, there also exists a confidentiality constraint $c = \{a_{i_1}, \ldots, a_{i_\ell}\} \in \mathcal{C}$. Due to $\mathcal{F}$ being confidential by assumption, $c \not\subseteq A_{F_s}$ holds (see Def. 3) and as a consequence of that there is an attribute $a_m \in c$ with $a_m \notin A_{F_s}$. Hence, respecting the rearrangement of the columns of r (see Fig. 4), both $m \notin \{1, \ldots, k\}$ and $m \in \{k+1, \ldots, n\}$ hold.

As a first step towards the construction of $\mathcal{I}^*$, the interpretation of the predicate symbol F_s – that is $\mathcal{I}^*(F_s)$ – is defined as

$$\mathcal{I}^*(F_s) := \{ \, (\nu[a_{\mathtt{tid}}], \nu[a_1], \ldots, \nu[a_k]) \mid \nu \in f_s \, \} \qquad (7)$$

and obviously this interpretation satisfies all formulas of $db^+_{f_s}$ as well as the closed world assumption contained in $db^-_{f_s}$. Moreover, fd_{F_s} is satisfied because, by assumption, f_s satisfies the functional dependency contained in SC_{F_s} and hence also (3) is satisfied by $\mathcal{I}^*(F_s)$. So, $\mathcal{I}^* \models_M db_{f_s}$ already holds.

Continuing the construction of $\mathcal{I}^*$, the set $\mathcal{I}^*(R)$ is defined as

$$\mathcal{I}^*(R) := \{ \, (\mu[a_1], \ldots, \mu[a_{m-1}], \varphi_m(\mu[a_m]), \mu[a_{m+1}], \ldots, \mu[a_n]) \mid \mu \in r \, \} \qquad (8)$$

in which $\varphi_m : \mathcal{U}_m \to \mathcal{U} \setminus \{v_m\}$ is an *injective* function having the finite domain $\mathcal{U}_m := \{ \, \mu[a_m] \mid \mu \in r \, \}$ and the infinite range $\mathcal{U} \setminus \{v_m\}$ with $v_m \in \boldsymbol{v}$ and $\mathcal{U}$ being the universe of $\mathcal{I}^*$ (cf. Def. 4). Note that φ_m can always be constructed because of $|\mathcal{U} \setminus \{v_m\}| > |\mathcal{U}_m|$. Moreover, $\mathcal{I}^* \not\models_M \tilde{\Psi}(\boldsymbol{v})$ holds as $v_m \in (v_{i_1}, \ldots, v_{i_\ell})$ is excluded from the range of φ_m and $\mathcal{I}^*$ can only satisfy $\tilde{\Psi}(\boldsymbol{v})$ if there is a tuple $(u_1, \ldots, u_m, \ldots, u_n) \in \mathcal{I}^*(R)$ for which $u_j = v_j$ holds for each $j \in \{i_1, \ldots, i_\ell\}$.

Next, it is shown that $\mathcal{I}^* \models_M db_r$ holds by proving that $\mathcal{I}^*$ satisfies the formulas (4) and (5) of Def. 6. To prove the if-part of the equivalence, assume that $(\exists X_{\texttt{tid}})\, F_s(X_{\texttt{tid}}, X_1, \ldots, X_k)$ of (4) is satisfied by $\mathcal{I}^*$ under a constant substitution $(X_1/u_1), \ldots, (X_k/u_k)$ which is feasible according to *Dom*. Then, according to (7), there is a tuple $(w_{\texttt{tid}}, u_1, \ldots, u_k) \in \mathcal{I}^*(F_s)$ with $w_{\texttt{tid}} \in \mathcal{U}$ implying the existence of a tuple $\nu \in f_s$ with $\nu[a_j] = u_j$ for all $j \in \{1, \ldots, k\}$. As f_s is a fragment instance of r (see Def. 1) and the columns of r are rearranged as described above, there is a tuple $\mu \in r$ with $\mu[a_j] = \nu[a_j]$ for all $j \in \{1, \ldots, k\}$. According to (8) and because of $m \notin \{1, \ldots, k\}$ there is a tuple $(u_1, \ldots, u_k, w_{k+1}, \ldots, w_n) \in \mathcal{I}^*(R)$ satisfying the conclusion. To finally establish the equivalence, the only-if-part can be proved by applying the argumentation presented above backwards.

To prove formula (5) of Def. 6, assume that the premise of (5) is satisfied by $\mathcal{I}^*$ under a constant substitution $(X_1/u_1), \ldots, (X_k/u_k)$ which is feasible according to *Dom*. Then there are two tuples $(w_{\texttt{tid}}, u_1, \ldots, u_k)$ and $(w'_{\texttt{tid}}, u_1, \ldots, u_k)$ in $\mathcal{I}^*(F_s)$ and $w_{\texttt{tid}} \neq w'_{\texttt{tid}}$ holds for $w_{\texttt{tid}}, w'_{\texttt{tid}} \in \mathcal{U}$. Because of the construction of $\mathcal{I}^*(F_s)$ described in (7) there are two different tuples $\nu, \nu' \in f_s$ with $\nu[a_j] = \nu'[a_j] = u_j$ for all $j \in \{1, \ldots, k\}$. Due to the existence of *exactly* one tuple in f_s for each tuple in r (cf. Sect. 2), it can be reasoned that there are also two tuples $\mu, \mu' \in r$ with $\mu[a_j] = \mu'[a_j] = u_j$ for each $j \in \{1, \ldots, k\}$. As relational instances cannot contain two identical tuples, $\mu[a_p] \neq \mu'[a_p]$ must hold for some $p \in \{k{+}1, \ldots, n\}$ and according to (8) there are two tuples $(u_1, \ldots, u_k, w_{k+1}, \ldots, w_n)$ and $(u_1, \ldots, u_k, w'_{k+1}, \ldots, w'_n)$ in $\mathcal{I}^*(R)$. In the case of $p \neq m$, obviously $w_p \neq w'_p$ holds, and otherwise $w_m \neq w'_m$ holds because of φ_m being injective. Hence, the conclusion of (5) is satisfied by $\mathcal{I}^*$, too.

As a last step $\mathcal{I}^* \models_M \mathit{prior}_{SC_R}$ has to be proved. This is prepared by constructing a temporary DB-Interpretation $\mathcal{I}_t$ for r as a set

$$\mathcal{I}_t(R) := \{\, (\mu[a_1], \ldots, \mu[a_m], \ldots, \mu[a_n]) \mid \mu \in r \,\} \tag{9}$$

and as (by assumption) r satisfies all constraints of SC_R, all formulas of prior_{SC_R} are satisfied by $\mathcal{I}_t$, too.

As there are no constants in formulas of prior_{SC_R} since all terms of these formulas are quantified variables (cf. Def. 9), an *arbitrary* DB-Interpretation $\mathcal{I}$ satisfying prior_{SC_R} does not need to contain tuples with specific combinations of values corresponding to combinations of constants in formulas of prior_{SC_R}. Hence, $\mathcal{I}$ still satisfies prior_{SC_R} if values in tuples of $\mathcal{I}$ are exchanged by other values of $\mathcal{U}$ so that all equalities (to satisfy equalities between variables in formulas of prior_{SC_R}) and diversities (to prevent the creation of further equalities resulting in more implications that need to be satisfied) between values of $\mathcal{I}$ are preserved. Moreover, as formulas of prior_{SC_R} are typed, values of $\mathcal{I}$ can be exchanged as long as all equalities and diversities between values of $\mathcal{I}$ are preserved within each column of $\mathcal{I}$.

Obviously, there is a tuple $(u_1, \ldots, u_m, \ldots, u_n) \in \mathcal{I}_t(R)$ iff there is a tuple $(u_1, \ldots, \varphi_m(u_m), \ldots, u_n) \in \mathcal{I}^*(R)$ and as φ_m is injective, two values u'_m and u''_m of $\mathcal{U}$ are equal iff $\varphi_m(u'_m) = \varphi_m(u''_m)$. Hence, $\mathcal{I}^* \models_M \mathit{prior}_{SC_R}$ holds. □

6 Conclusion and Future Work

Motivated by the wish to achieve confidentiality of information hosted by third party service providers without the usage of encryption, the approach to fragmentation presented in Sect. 2 is developed in [7,8]. In these articles the protection of information is discussed only in terms of direct accesses to data. It is *not* shown that confidential information cannot be inferred based on the knowledge of non-confidential information contained in the externally stored fragment instance f_s and a priori knowledge an attacker might possibly have.

This desirable result is presented in this article under the supposition that an attacker only has a priori knowledge in terms of Equality Generating Dependencies and Tuple Generating Dependencies which are all unirelational and typed. Regarding the possibilities to express knowledge about semantic constraints declared for the schema of an original instance r, this supposition is not very restrictive as most of the semantic constraints commonly used (e.g., functional dependencies, join dependencies) belong to these classes of constraints [1]. Moreover, reconsidering the proof of Theorem 1, it can be seen easily that the inference-proofness still holds if the restriction that semantic constraints have to be typed is replaced by the weaker restriction that the set of semantic constraints considered does not impose that any value of one of the columns $k+1, \ldots, n$ of r is equal to a value of one of the columns $1, \ldots, k$ of r.

Additionally to the knowledge about semantic constraints an attacker might also have some a priori knowledge about the world in general (e.g., a set of facts and inference rules) which cannot be expressed as a set of formulas complying with the restrictions stated above. But as shown in the introductory example of Sect. 5, the inference-proofness of fragmentation *cannot* be guaranteed under arbitrary a priori knowledge – even if no sensitive information can be inferred solely based on this a priori knowledge. So, further research on (weak) syntactic restrictions for modelling a priori knowledge without violating confidentiality requirements might lead to even more expressive languages for that purpose.

As there are other approaches to achieving confidentiality by vertical fragmentation than the one treated in this article (see e.g. [2,9]), another idea for future work might be to analyse the inference-proofness of these approaches. As these approaches free the client from storing data locally by resorting to encryption if necessary, the logic-oriented modelling of an attacker's knowledge has to be adapted suitably to reflect these circumstances. Moreover, approaches based on vertical fragmentation might be combined with approaches based on horizontal fragmentation, which partition the set of tuples of an original instance r with the help of selection criteria into several instances declared over the same set of attributes as r. An approach to achieving inference-proofness based on horizontal fragmentation is presented in [14]. This kind of "hybrid" fragmentation promises a higher amount of outsourced data, but it raises several confidentiality issues (like meta-inferences), which must be analysed with scrutiny.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)
2. Aggarwal, G., Bawa, M., Ganesan, P., Garcia-Molina, H., Kenthapadi, K., Motwani, R., Srivastava, U., Thomas, D., Xu, Y.: Two can keep a secret: A distributed architecture for secure database services. In: 2nd Biennial Conference on Innovative Data Systems Research, CIDR 2005, pp. 186–199 (2005)
3. Biskup, J.: Security in Computing Systems – Challenges, Approaches and Solutions. Springer, Heidelberg (2009)
4. Biskup, J.: Usability confinement of server reactions: Maintaining inference-proof client views by controlled interaction execution. In: Kikuchi, S., Sachdeva, S., Bhalla, S. (eds.) DNIS 2010. LNCS, vol. 5999, pp. 80–106. Springer, Heidelberg (2010)
5. Biskup, J., Bonatti, P.A.: Controlled query evaluation with open queries for a decidable relational submodel. Annals of Mathematics and Artificial Intelligence 50(1-2), 39–77 (2007)
6. Biskup, J., Embley, D.W., Lochner, J.: Reducing inference control to access control for normalized database schemas. Information Processing Letters 106(1), 8–12 (2008)
7. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Enforcing confidentiality constraints on sensitive databases with lightweight trusted clients. In: Gudes, E., Vaidya, J. (eds.) Data and Applications Security XXIII. LNCS, vol. 5645, pp. 225–239. Springer, Heidelberg (2009)
8. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Keep a few: Outsourcing data while maintaining confidentiality. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 440–455. Springer, Heidelberg (2009)
9. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Combining fragmentation and encryption to protect privacy in data storage. ACM Transactions on Information and System Security 13(3) (2010)
10. Fagin, R.: Horn clauses and database dependencies. Journal of the ACM 29(4), 952–985 (1982)
11. Farkas, C., Jajodia, S.: The inference problem: A survey. ACM SIGKDD Explorations Newsletter 4(2), 6–11 (2002)
12. Hacigümüs, H., Mehrotra, S., Iyer, B.R.: Providing database as a service. In: Proceedings of the 18th International Conference on Data Engineering, ICDE 2002, pp. 29–40. IEEE Computer Society, Los Alamitos (2002)
13. Samarati, P., De Capitani di Vimercati, S.: Data protection in outsourcing scenarios: Issues and directions. In: Feng, D., Basin, D.A., Liu, P. (eds.) ACM Symposium on Information, Computer and Communications Security, ASIACCS 2010, pp. 1–14. ACM, New York (2010)
14. Wiese, L.: Horizontal fragmentation for data outsourcing with formula-based confidentiality constraints. In: Echizen, I., Kunihiro, N., Sasaki, R. (eds.) IWSEC 2010. LNCS, vol. 6434, pp. 101–116. Springer, Heidelberg (2010)