Detecting Information Leakage in Database Access Control With Help from Data Exchange

Rada Chirkova
Department of Computer Science
NC State University, Raleigh, NC 27695, USA
chirkova@csc.ncsu.edu

Ting Yu
Department of Computer Science
NC State University, Raleigh, NC 27695, USA
yu@csc.ncsu.edu

ABSTRACT

We study the following problem: Given a set of answers, MV, to some fixed queries, $\mathcal{V}$, on an (unavailable) database instance of interest, I, and given another query, Q: Which of the answer tuples to Q on I are "deterministically assured" by the contents of MV? That is, which tuples $\bar{t}$ must necessarily be present in the answer to the query Q on the instance I, based on the information in $\mathcal{V}$, in MV, and (optionally) in the set Σ of integrity constraints that must hold on I? (We say that there is *information leakage* of Q via MV and $\mathcal{V}$ iff at least one such tuple $\bar{t}$ exists.) Note that the instance I is not available for making the determination.

We perform a theoretical investigation of the above *problem of information-leak disclosure.* We focus on the relational setting, and assume that Q and the queries in $\mathcal{V}$ are conjunctive queries, and the set Σ of integrity constraints (when present) is "weakly acyclic" [18]. We use data-exchange techniques to develop a sound and complete algorithm for solving the problem in this setting. The results of this paper are applicable to fundamental problems in information management, especially in database security and privacy. One immediate application is in contexts where database users are assigned data-access privileges using fine-grained access control, intuitively via view definitions. In such contexts, our algorithm would permit database owners to detect information leakage in cases where a database user, or a group of possibly colluding users, are given privileges that will enable the users to deterministically derive some sensitive information contained in the database.

1. INTRODUCTION

In this paper we consider the following basic question: Given a set of answers, MV, to some fixed queries on some (unavailable) database instance of interest, I, and given another query, Q: Which of the answer tuples to Q on I are "deterministically assured" by the contents of MV? Formally, consider an instance I of some database schema $\mathbf{P}$. For a set $\mathcal{V}$ of $n \geq 1$ queries V_1, V_2, ..., V_n defined on the schema $\mathbf{P}$, denote by MV the union of the answers $V_j(I)$ to the queries V_j on the instance I, for all $j \in [1, n]$. The problem is as follows: Given a query Q over the schema $\mathbf{P}$, return the set of all tuples, $\bar{t}$, of domain values, such that each $\bar{t}$ must be in the answer $Q(I)$ to the query Q on the instance of interest I. (We say that there is *information leakage* of Q via MV and $\mathcal{V}$ if and only if at least one such tuple $\bar{t}$ exists.) The information available for making the determination of whether a tuple $\bar{t}$ must be in $Q(I)$ includes the following: the definition of the schema $\mathbf{P}$; the definitions of the queries Q, V_1, V_2, ..., V_n; the instance MV; and (optionally) integrity constraints Σ that must hold on all instances of the schema $\mathbf{P}$. Note that the instance I is not available for making the determination. This problem builds on the problems considered in [1] by adding integrity constraints to the problem inputs.

It turns out that this abstract problem arises naturally in fine-grained database-access control. Indeed, in organizational databases that are designed for shared access, individual users typically access the stored data based on their privileges. In many cases, such access privileges are expressed using detailed levels of granularity of the database data [7]. Representative mechanisms for formally specifying fine-grained user-access privileges include Oracle Virtual Private Database (VPD) [27, 28] and label-based access control in DB2 [9].

Not surprisingly, management of fine-grained access control is rather challenging. In one particular challenge that we address in this paper, a user or a group of users may obtain sensitive data using more than one data-access policy at a time. (E.g., a user may work on multiple projects, with each project independently granting partial access to the same data.) As a result, sensitive information may be leaked inadvertently to unauthorized users when they combine privileges from multiple access-control rules, each of which is seemingly safe. Then it is natural for the owners of the data to want to determine whether any such information leakage is possible when certain data have been disclosed to the users. What complicates this problem further is that an adversary may not only be aware of specific access-control rules, but also be equipped with domain knowledge, as reflected by database integrity constraints. Formalizing "sensitive information" as our query Q of interest, and formalizing access-control rules as views $\mathcal{V}$, leads naturally to the formal problem outlined in the beginning of this section. Consider an illustration.

EXAMPLE 1.1. *Suppose a relation* `Emp` *stores information about employees of a company. Let the attributes of* `Emp` *be* `Name`*,* `Dept`*, and* `Salary`*, with self-explanatory attribute names:* `Emp(Name,Dept,Salary)`*.*

We assume for simplicity that no integrity constraints hold on the database schema $\mathbf{P}$ *containing the relation* `Emp`*. (In particular, the only primary key of* `Emp` *is all its attributes.) Thus, the set* Σ *of dependencies holding on the schema* $\mathbf{P}$ *is the empty set.*

Let a "secret query" `Q` *ask for the salaries of all the employees. We can formulate the query* `Q` *in SQL as*

```
(Q): SELECT DISTINCT Name, Salary FROM Emp;
```

Consider two views, V *and* W*, that are defined for some class(es) of users, in SQL on the schema* **P**. *The view* V *returns the department name for each employee, and the view* W *returns the salaries in each department:*

```
(V): DEFINE VIEW V(Name, Dept) AS
     SELECT DISTINCT Name, Dept FROM Emp;
(W): DEFINE VIEW W(Dept, Salary) AS
     SELECT DISTINCT Dept, Salary FROM Emp;
```

Suppose that some user(s) are authorized to see the answers to V *and* W*, and that at some point the user(s) can see the following set* MV *of answers to these views.*

MV = { V(JohnDoe, Sales), W(Sales, $50000) } .

Consider a tuple $\bar{t}$ = (JohnDoe, $50000) *of domain values in* MV. *Assume that these users are interested in finding out whether* $\bar{t}$ *is in the answer to the secret query* Q *on the "back-end" instance,* I. *(That is, applying the queries* V *and* W *to* I *has generated the instance* MV.*) We can show that the relation* Emp *in such an instance* I *is uniquely determined by* V, W, *and* MV*:*

Emp *in* I *is* { Emp(JohnDoe, Sales, $50000) } .

The only answer to the secret query Q *on this instance* I *is the above tuple* $\bar{t}$. *Thus, the presence of the tuple* $\bar{t}$ *in the answer to* Q *on the organizational database of interest is in this case deterministically assured by the information that these users are authorized to access.* □

As detecting information leakage is an important challenge in real-life database access, a variety of formalizations of the problem have been studied, please see [7, 12] for overviews. In influential papers [23, 24] by Miklau and Suciu, the problem considered in [23, 24] is the same at the informal level as our basic question above. (As a result, the solutions that we obtain in this current paper address, at the pragmatic level, the same real-life challenges as the solutions developed in [23, 24].) That is, the papers [23, 24] focus on the problem of determining, for a fixed finite set $\mathcal{V}$ of views to be published, whether the published answers will logically disclose information about a fixed confidential query Q.

At the same time, the formalization of this problem in [23, 24], inspired by Shannon's notion of perfect secrecy [29], is as follows: There is no information leakage of the query Q via the views $\mathcal{V}$ if and only if the *probability* of an adversary guessing the answer to Q is the same (or, in another scenario, is almost the same) whether the adversary knows the answers to $\mathcal{V}$ or not. The information-leakage problem is addressed in [23, 24] using this formalization, in the absence of any specific fixed answers to the views $\mathcal{V}$ and using the assumption that the database of interest is given as a probability distribution over a fixed finite domain of values. Data-exchange approaches [6] are not used in the technical development in [23, 24]; rather, the term "data exchange" is used in [23, 24] informally as a reference to today's universal sharing of data (as in, e.g., on the Web).

It is possible that the solutions given in [23, 24] could be extended to address part of our basic question, as follows. Suppose that, for a set of databases of interest (given via an appropriate domain and probability distribution), for a query Q, and for views $\mathcal{V}$, there is no information leakage of Q via $\mathcal{V}$ by the definition of [23, 24]. Then, conceivably, one could show formally that for each possible set, MV, of answers to the views $\mathcal{V}$, there is no *deterministic* information leakage of Q w.r.t. $\mathcal{V}$ and MV, in the sense of our basic question. While such a result might presumably be proved, the proof would still leave open the following possibility. Suppose that for a particular set MV^*, there is no deterministic information leakage of Q w.r.t. $\mathcal{V}$ and MV^*, even though by the results of [23, 24] there is probabilistic information leakage of Q via $\mathcal{V}$, intuitively due to some other set MV' of answers to the views $\mathcal{V}$. Clearly, any such set MV' must not be disclosed to the users, due to the associated threat of (deterministic) leakage of the sensitive information Q. At the same time, any set MV^* as above can be safely disclosed to the users, even in presence of the "general" information leakage of Q via $\mathcal{V}$ shown using the results of [23, 24].

In this paper, we formalize the basic question outlined in the beginning of this section; our formalization is a natural extension of that in [1]. We then perform a theoretical study of the formal problem, in the relational setting and under the following restrictions, which we will be referring to collectively as "the CQ weakly-acyclic setting": (1) The given queries Q, V_1, ..., V_n are all SQL select-project-join queries with equality comparisons and possibly with constants. (That is, we assume that Q, V_1, ..., V_n are expressed in the common language of *conjunctive (CQ) queries.*) (2) We assume that for each given set Σ of integrity constraints *(dependencies)* on the input database schema **P**, the set Σ is a finite set of "weakly-acyclic dependencies" [18], which is a common assumption in the literature.

Our contributions. The specific contributions of this work are as follows:

- We formalize the problem of "deterministic assurance" of the presence of some tuples in the answer to a fixed query, Q, on some database of interest, I, by the contents of the answers MV to other fixed queries $\mathcal{V}$ on the same database I. In our formalization, a problem instance, usually denoted $\mathcal{D}_s$, includes a schema **P**, a set Σ of dependencies on **P**, a set $\mathcal{V}$, a set MV, and a query Q. If any instance I of schema **P** exists such that I satisfies Σ and such that the set of answers on I to the views $\mathcal{V}$ is exactly MV, then we say that $\mathcal{D}_s$ is a *valid* problem instance. For each valid problem instance $\mathcal{D}_s$, the problem of information-leak disclosure is to determine the set of tuples that must be present in the answer to the query Q on the instance I due to the information given by $\mathcal{D}_s$.
- We perform a theoretical study of the problem of information-leak disclosure in the CQ weakly-acyclic setting. Specifically, we introduce and study two approaches that arise naturally in the context of the problem: the "rewriting approach" (Section 5) and the "data-exchange approach" (Section 6). While both approaches are sound, neither approach yields a sound and complete algorithm for all CQ weakly-acyclic inputs.
- Then, in Section 7 we introduce a modification of our "data-exchange approach" of Section 6. The resulting "view-verified data-exchange approach" yields a sound and complete algorithm for all problem inputs in the CQ weakly-acyclic setting.
- We prove that the problem of information-leak disclosure is Π_2^p complete for the class of CQ weakly acyclic instances. (We assume, same as in [33],

that the parts $\mathbf{P}$, Σ, and $\mathcal{V}$ of the problem input $\mathcal{D}_s$ are fixed, while Q and MV can vary.)

- Finally, we provide an algorithm for determining whether a given CQ weakly acyclic instance $\mathcal{D}_s$ of the problem of information-leak disclosure is valid.

The results of this paper are applicable to a number of fundamental problems in information management, especially in database security and privacy. Consider, for instance, a database where user privileges are defined through views. Following the principles of least privilege and separation of duty [16], it would be natural to ask whe- ther a user, or a group of possibly colluding users, are given privileges that will enable the users to deterministically derive some sensitive information. Questions of this type are particularly important and challenging for fine-grained access control, where privileges may be granted to users in a much more elaborate manner than table-level or column-level access control. As a result, sensitive information may be leaked in very subtle ways, which are hard to discover by manual inspection of access-control rules. Similarly, when a data owner wants to share information with a third party (e.g., through views), it is crucial to understand whether, e.g., private information of individuals may be leaked because of such sharing [31, 12].

The remainder of this paper is organized as follows. In Section 2 we review related work. We then provide the background definitions in Section 3, and formalize in Section 4 the problem of information-leak disclosure, which we focus on in this paper. In Sections 5–7 we present three approaches to addressing the problem in the CQ weakly acyclic setting. Further, in Section 7 we solve the problem of determining the validity of a CQ weakly acyclic problem input, and show that the problem of information-leak disclosure is Π_2^p complete in the CQ weakly acyclic setting, even when $\Sigma = \emptyset$.

2. RELATED WORK

As observed in Section 1, to the best of our knowledge, the formal problem that we address in this current paper has not been considered in the open literature. (Abiteboul and Duschka in [1] consider the special case where the set of dependencies is empty, apply in their analysis a different type of complexity metric than we do in this paper, and do not provide algorithms alongside their complexity results.) The work [23, 24] addresses a problem that is similar to ours at the informal level; see Section 1 for a detailed discussion of [23, 24]. Generally, the literature on privacy-preserving query answering and data publishing is represented by work on data anonymization and on differential privacy; [12] is a recent survey. Most of that work focuses on probabilistic inference of private information, while in this paper we focus on the possibilistic situation, where an adversary can deterministically derive sensitive information. Further, our model of sensitive information goes beyond associations between individuals and their private sensitive attributes.

Policy analysis has been studied for various types of systems, including operating systems, role-based access control, trust management, and firewalls [20, 4, 5, 22]. Typically, two types of properties are studied. The first type is static properties: Given the current security setting (e.g., non-management privileges of users), can certain actions or events (e.g., separation of duty) happen? Our analysis of information leakage in database policies falls into this category. What is different is that our policy model is much more elaborate, as we deal with policies defined by database query languages. The other type of properties in policy analysis is dynamic properties when a system evolves; that direction is not closely related to the topic of our paper.

The problem of *inference control,* with a focus on preventing unauthorized users from computing sensitive information, has been studied extensively in the database-security literature. The inputs to the problem can be considered the same as ours; the set $\mathcal{V}$ is interpreted as free-form queries, defined (within a given query language) by the user, who asks them sequentially on a database instance I. In addition, a fixed procedure for inference of the sensitive information is specified; the adversary user is assumed to use only that procedure in computing sensitive information. The "security monitor" in the system logs all the queries, and temporarily withholds the answer to the latest query posed by the user. It then applies the fixed inference procedure to the log and to the latest answer. In case sensitive information is derived in this process, the monitor chooses what to do (e.g., to refuse to give the user the latest answer) to prevent the leakage. Work in this direction has been done in statistical databases [17] and in controlled query evaluation (CQE) [8]. Some of the work (e.g., [10]) uses chase with dependencies to determine leakage.

In contrast, our goal is to determine whether there exists any procedure that would be guaranteed to derive all and only the sensitive information, for all problem instances in a given class. None of the procedures that we have seen in the literature on inference control yields sound and complete algorithms for our class of interest in this current paper, CQ weakly acyclic instances.

Zhang and Mendelzon in [33] addressed the problem of letting users access authorized data, via rewriting the users' queries in terms of their authorization views; this problem is different from ours. Toward that goal, [33] explored the notion of "conditional query containment," which in this current paper we extend to the case $\Sigma \neq \emptyset$. The results of [33], which we use in our work, include a powerful reduction of the problem of testing conditional containment of CQ queries to that of testing *unconditional* containment of modifications of the queries.

Our results of Sections 6–7 build on the influential framework for data exchange [18] by Fagin and colleagues. Our MV-induced dependencies of Section 7 resemble target-to-source dependencies Σ_{ts} introduced into (peer) data exchange in [19]. The difference is that Σ_{ts} are embedded dependencies defined at the schema level. In contrast, our disjunctive MV-induced dependencies embody the given set of view answers MV.

Finally, our problem can be linked at the abstract level to the work of Nash and colleagues [26] on whether a query Q is determined by views $\mathcal{V}$. The focus in [26] is on whether views $\mathcal{V}$ determine the entire answer to the query Q. Our interest in this paper is in determining the maximal set $Q_s(I)$ of the tuples in the answer to Q on an instance I, such that $Q_s(I)$ is deterministically assured by the set MV of answers to $\mathcal{V}$ on I.

3. PRELIMINARIES

3.1 Instances and Queries

Schemas and instances. A *schema* $\mathbf{P}$ is a finite sequence $< P_1, \ldots, P_m >$ of relation symbols, with each

P_i having a fixed arity $k_i \geq 0$. An *instance I of* $\mathbf{P}$ assigns to each $P_i \in \mathbf{P}$ a finite k_i-ary relation $I[P_i]$, which is a set of tuples. For tuple membership in relation $I[P_i]$, we use the notation $\bar{t} \in I[P_i]$. Each element of each tuple in an instance belongs to one of two disjoint infinite sets of values, CONST and VAR. We call elements of CONST *constants,* and denote them by lowercase letters a, b, c, ... ; the elements of VAR are called *(labeled) nulls,* denoted by symbols $\perp$, $\perp_1$, $\perp_2$, ...

Sometimes we use the notation $P_i(\bar{t}) \in I$ instead of $\bar{t} \in I[P_i]$, and call $P_i(\bar{t})$ a *fact* of I. When all the values in $P_i(\bar{t})$ are constants, we say that $P_i(\bar{t})$ is a *ground fact,* and $\bar{t}$ is a *ground tuple.* The *active domain* of instance I, denoted $adom(I)$, is the set of all elements of CONST $\cup$ VAR that occur in any facts in I. When each fact in I is a ground fact, we call I a *ground instance.* End users of relational data are interested only in ground instances; non-ground instances are typically used as formal technical tools, as in, e.g., data exchange [6].

Queries. In this paper we focus on the class of queries called "conjunctive queries" *(CQ queries).* In the definitions, we will be using the following notion of *relational atom.* Let QVAR be an infinite set of values disjoint from CONST $\cup$ VAR; we call QVAR *the set of (query) variables.* We denote variables by uppercase letters X, Y, ... Then $P(\bar{t})$, with P a k-ary relation symbol and $\bar{t}$ a k-vector, is a *relational atom* whenever each value in $\bar{t}$ is an element of CONST $\cup$ QVAR.

A *CQ-rule* over schema $\mathbf{P}$, with *output relation symbol* $Q \notin \mathbf{P}$, of arity $k \geq 0$ of Q, is an expression of the form

$$Q(\bar{X}) \leftarrow P_1(\bar{U}_1) \wedge \ldots \wedge P_n(\bar{U}_n).$$

Here, $n \geq 1$; the vector $\bar{X}$ has k elements; for each $i \in [1, n]$, P_i is a relation symbol in $\mathbf{P}$, of arity $k_i \geq 0$; and $Q(\bar{X})$, $P_1(\bar{U}_1)$, ..., $P_n(\bar{U}_n)$ are all relational atoms. We consider only *safe* rules: That is, each variable in $\bar{X}$ also occurs in at least one of $\bar{U}_1$, ..., $\bar{U}_n$. All the variables of the rule that do not appear in $\bar{X}$ (i.e., the *nonhead variables* of the rule) are assumed to be existentially quantified. We call the atom $Q(\bar{X})$ the *head* of the rule, call $\bar{X}$ the *head vector* of the rule, and call the conjunction of its remaining atoms the *body* of the rule. Each atom in the body of the rule is called a *subgoal* of the rule. The conjunction in the body of a rule is usually written using commas, as $P_1(\bar{U}_1), \ldots, P_n(\bar{U}_n)$.

A *conjunctive query (CQ query)* is a query defined by a single *CQ*-rule. Consider a CQ query defined by a CQ-rule with a k-ary ($k \geq 0$) output relation symbol Q and with head vector $\bar{X}$. Wherever clear from the context, we will be referring to the query as $Q(\bar{X})$, or even just as Q. We will be using $head_{(Q)}$ and $body_{(Q)}$ as names for the head and body of the (rule for) Q.

Semantics of CQ queries. We now define the semantics of a CQ query Q. In the definition, we will need the notions of *homomorphism* and of *valuation.* Consider two conjunctions, $\varphi(\bar{Y})$ and $\psi(\bar{Z})$, of relational atoms. Then a mapping h from the set of elements of $\bar{Y}$ to the set of elements of $\bar{Z}$ is called a *homomorphism* from $\varphi(\bar{Y})$ to $\psi(\bar{Z})$ whenever (i) $h(c) = c$ for each constant c in $\bar{Y}$, and (ii) for each conjunct of the form $p(\bar{U})$ in $\varphi(\bar{Y})$, the relational atom $p(h(\bar{U}))$ is a conjunct in $\psi(\bar{Z})$. (For a vector $\bar{S} = [s_1 s_2 \ldots s_l]$, for some $l \geq 1$, we define $h(\bar{S})$ as the vector $[h(s_1)h(s_2)\ldots h(s_l)]$. By convention, a homomorphism is an identity mapping when applied to empty vectors and to empty tuples.)

We define homomorphisms in the same way for the case where either one of $\varphi(\bar{Y})$ and $\psi(\bar{Z})$ (or both) is a conjunction of facts. We will denote homomorphisms by lowercase letters g, h, ... , possibly with subscripts.

We call each homomorphism from a conjunction $\varphi(\bar{Y})$ of relational atoms to a conjunction $\psi(\bar{Z})$ of facts, a *valuation* from $\varphi(\bar{Y})$ to $\psi(\bar{Z})$. We will use Greek letters μ, ν, ... , possibly with subscripts, for valuations.

Consider a k-ary CQ query $Q(\bar{X})$ and an instance I, which we interpret as a conjunction of all the facts in I. Then *the answer to Q on I*, denoted $Q(I)$, is

$$Q(I) = \{\ \nu(\bar{X}) \mid \nu \text{ is a valuation from } body_{(Q)} \text{ to } I\ \}\ .$$

(When $\bar{X}$ is the empty vector, i.e., Q is a *Boolean query,* which is the case $k = 0$, then $\nu(\bar{X})$ is the empty tuple.)

Query containment. A query Q_1 *is contained in query* Q_2, denoted $Q_1 \sqsubseteq Q_2$, if $Q_1(I) \subseteq Q_2(I)$ for every instance I. A classic result of [11] by Chandra and Merlin states that a necessary and sufficient condition for the containment $Q_1 \sqsubseteq Q_2$, for CQ queries Q_1 and Q_2 of the same arity, is the existence of a containment mapping from Q_2 to Q_1. A *containment mapping* [11] from query $Q_2(\bar{X}_2)$ to query $Q_1(\bar{X}_1)$ is a homomorphism h from $body_{(Q_2)}$ to $body_{(Q_1)}$ such that $h(\bar{X}_2) = \bar{X}_1$.

Canonical database. Every CQ query Q can be regarded as a symbolic instance $I^{(Q)}$. $I^{(Q)}$ is defined as the result of turning each subgoal $P_i(\ldots)$ of Q into a tuple in the relation $I^{(Q)}[P_i]$. The procedure is to keep each constant in the body of Q, and to replace consistently each variable in the body of Q by a distinct constant different from all the constants in Q. The tuples that correspond to the resulting ground facts are the only tuples in the *canonical database* $I^{(Q)}$ for Q, which is unique up to isomorphism.

3.2 Dependencies and Chase

Embedded dependencies. We consider dependencies σ of the form

$$\sigma : \phi(\bar{U}, \bar{V}) \rightarrow \exists \bar{W}\ \psi(\bar{U}, \bar{W})$$

with ϕ and ψ conjunctions of relational atoms, possibly with equations added. Such dependencies, called *embedded dependencies,* are expressive enough to specify all usual integrity constraints, such as keys, foreign keys, and inclusion dependencies [2]. If ψ comprises a single equation, then σ is an *equality-generating dependency (egd).* If ψ consists only of relational atoms, then σ is a *tuple-generating dependency (tgd).* We follow [18] in allowing constants in egds and tgds. Each set Σ of embedded dependencies is equivalent to a set of tgds and egds [2]. We write $I \models \Sigma$ if instance I satisfies all elements of Σ. All sets Σ we refer to are finite.

Query containment under dependencies. We say that *query Q is contained in query P under set of dependencies* Σ, denoted $Q \sqsubseteq_\Sigma P$, if for every instance $I \models \Sigma$ we have $Q(I) \subseteq P(I)$. Queries Q and P *are equivalent under* Σ, denoted $Q \equiv_\Sigma P$, if both $Q \sqsubseteq_\Sigma P$ and $P \sqsubseteq_\Sigma Q$ hold. Q and P *are equivalent (in the absence of dependencies),* denoted $Q \equiv P$, if $Q \equiv_\emptyset P$.

Chase for CQ query. Given a CQ query $Q(\bar{X}) \leftarrow \xi(\bar{X}, \bar{Y})$ and a tgd σ of the form $\phi(\bar{U}, \bar{V}) \rightarrow \exists \bar{W}\ \psi(\bar{U}, \bar{W})$; assume w.l.o.g. that Q has none of the variables in $\bar{W}$.

The *chase of Q with σ is applicable* if there is a homomorphism h from ϕ to ξ, such that h cannot be extended to a homomorphism h' from $\phi \wedge \psi$ to ξ. Then, a *chase step* on Q with σ and h is a rewrite of Q into $Q'(\bar{X}) \leftarrow \xi(\bar{X}, \bar{Y}) \wedge \psi(h(\bar{U}), \bar{W})$. It can be shown that $Q' \equiv_{\{\sigma\}} Q$ and that $Q' \sqsubseteq Q$.

We now define a chase step with an egd. Assume a CQ query Q as before and an egd σ of the form $\phi(\bar{U}) \rightarrow U_1 = U_2$. The *chase of Q with σ is applicable* if there is a homomorphism h from ϕ to ξ such that $h(U_1) \neq h(U_2)$ and at least one of $h(U_1)$ and $h(U_2)$ is a variable; let w.l.o.g. $h(U_1)$ be a variable. Then a *chase step* on Q with σ and h is a rewrite of Q into a query, Q', that results from replacing all occurrences of $h(U_1)$ in Q by $h(U_2)$. Again, we have $Q' \equiv_{\{\sigma\}} Q$ and $Q' \sqsubseteq Q$. If, for a homomorphism h as above, $h(U_1)$ and $h(U_2)$ are distinct constants, then we say that *chase with σ fails on Q*. In this case, $Q(I) = \emptyset$ on all $I \models \{\sigma\}$.

A *Σ-chase sequence C* (or just *chase sequence,* if Σ is clear from the context) *for CQ query Q_0* is a sequence of CQ queries $Q_0, Q_1, \ldots$ such that each query Q_{i+1} ($i \geq 0$) in C is obtained from Q_i by a chase step $Q_i \Rightarrow^{\sigma} Q_{i+1}$ using a dependency $\sigma \in \Sigma$. A chase sequence $Q = Q_0, Q_1, \ldots, Q_n$ is *terminating* if $I^{(Q_n)} \models \Sigma$, where $I^{(Q_n)}$ is the canonical database for Q_n. In this case we denote Q_n by $(Q)^{\Sigma}$ and say that $(Q)^{\Sigma}$ is the (terminal) *result* of the chase. All chase results for a given CQ query are equivalent in the absence of dependencies [14].

Weakly acyclic dependencies [18]. Let Σ be a set of tgds over schema $\mathbf{T}$. We construct the *dependency graph* of Σ, as follows. The nodes (positions) of the graph are all pairs (T, A), for $T \in \mathbf{T}$ and A an attribute of T. We now add edges: For each tgd $\varphi(\bar{X}) \rightarrow \exists\, \bar{Y}\ \psi(\bar{X}, \bar{Y})$ in Σ, and for each $X \in \bar{X}$ that occurs in φ in position (T, A) and that occurs in ψ, do the following.

- For each occurrence of X in ψ in position (S, B), add a *regular edge* from (T, A) to (S, B); and
- For each existentially quantified variable $Y \in \bar{Y}$ and for each occurrence of Y in ψ in position (R, C), add a *special edge* from (T, A) to (R, C).

For a set Σ of tgds and egds, with Σ^t the set of all tgds in Σ, we say that Σ is *weakly acyclic* if the dependency graph of Σ^t does not have a cycle going through a special edge. Chase of CQ queries terminates in finite time under sets of weakly acyclic dependencies [18].

The following result is immediate from [2, 13, 14].

THEOREM 3.1. *Given CQ queries Q_1, Q_2 and set Σ of embedded dependencies. Then $Q_1 \sqsubseteq_{\Sigma} Q_2$ iff $(Q_1)^{\Sigma} \sqsubseteq (Q_2)^{\Sigma}$ in the absence of dependencies.* □

Chase of instance. Given an instance I of schema $\mathbf{P}$ and given a set Σ of egds and tgds; we interpret I as a conjunction of its facts. We follow [14] in defining chase of I with Σ in the same way as chase of a CQ query with Σ. That is, in the chase steps we treat each null in I as a distinct variable (in the chase for CQ queries). Further, each chase step with a tgd that has existential variables introduces, in the result of the chase step, a distinct new null for each existential variable of the tgd. Chase sequences and chase termination are also defined in the same way as for CQ queries; the result I' of the chase of I with Σ always satisfies Σ, that is $I' \models \Sigma$.

4. THE PROBLEM STATEMENT

In this section we formalize the problem of information-leak disclosure. Suppose that we are given a schema $\mathbf{P}$ and a set of dependencies Σ on $\mathbf{P}$. Let $\mathcal{V}$ be a finite set of relation symbols not in $\mathbf{P}$, with each symbol *(view name)* $V \in \mathcal{V}$ of some arity $k_V \geq 0$. Each $V \in \mathcal{V}$ is associated with a k_V-ary query on the schema $\mathbf{P}$. We call $\mathcal{V}$ a *set of views on* $\mathbf{P}$, and call the query for each $V \in \mathcal{V}$ the *definition of the view V*, or *the query for V*. We assume that the query for each $V \in \mathcal{V}$ is associated with (V in) the set $\mathcal{V}$. We call a ground instance MV of schema $\mathcal{V}$ a *set of view answers for $\mathcal{V}$*.

Let I be a ground instance of schema $\mathbf{P}$. We say that *I is a Σ-valid base instance for $\mathcal{V}$ and MV*, denoted by $\mathcal{V} \Rightarrow_{I,\Sigma} MV$, whenever (a) $I \models \Sigma$, and (b) the answer $V(I)$ to the query for V on the instance I is identical to the relation $MV[V]$ for V in the instance MV. Further, we say that *MV is a Σ-valid set of view answers for $\mathcal{V}$*, denoted by $\mathcal{V} \Rightarrow_{*,\Sigma} MV$, whenever there exists a Σ-valid base instance for $\mathcal{V}$ and MV.

DEFINITION 4.1. *(**Specific instance of information-leak problem**) Given schema $\mathbf{P}$, set of dependencies Σ on $\mathbf{P}$, set $\mathcal{V}$ of views on $\mathbf{P}$, query Q over $\mathbf{P}$, and a (Σ-valid) set MV of view answers for $\mathcal{V}$. Then we say that the tuple $\mathcal{D}_s$ = ($\mathbf{P}$, Σ, $\mathcal{V}$, Q, MV) is a* (valid) specific instance of the information-leak problem. □

In Definition 4.1, the intuition for $\mathbf{P}$ and Σ is that of the schema and associated dependencies on a database of interest, which (database) belongs to some organization. The set $\mathcal{V}$ is, intuitively, the access policies given to users of the database, with each user accessing the database via a subset of $\mathcal{V}$. The query Q is a "secret query" on the database of interest. That is, the organization does not want any answer to Q disclosed to the users that access the database via the views $\mathcal{V}$. Finally, MV is a set of answers to the users' access policies $\mathcal{V}$ over some (single) instance of the database of interest. Example 1.1 illustrates this intuition.

CQ weakly acyclic information-leak instances. In this paper, we assume that a single malicious user *(attacker),* or several attackers colluding together, consider some *valid* specific instance $\mathcal{D}_s$ = ($\mathbf{P}$, Σ, $\mathcal{V}$, Q, MV) of the information-leak problem.[1] In the remainder of this paper we consider only those instances $\mathcal{D}_s$ where (i) $MV \neq \emptyset$. (This technical requirement is clearly nonrestrictive in the context of the problem.) We will focus on those instances where (ii) the query Q is a CQ query, and (iii) the queries for all the views in $\mathcal{V}$ are also CQ queries. (We call each set $\mathcal{V}$ that satisfies (iii) *a set of CQ views.*) In addition, for all the queries that we consider in this paper as defined in terms of the views in $\mathcal{V}$, we consider throughout this paper only (iv) *CQ* such queries, unless specified explicitly. We will refer to instances $\mathcal{D}_s$ that satisfy (i)–(iv) as *CQ instances of information leak.* Finally, we will focus on (v) *weakly acyclic* sets Σ of dependencies. Whenever the conditions (i)–(v) hold together on an instance $\mathcal{D}_s$ of the information-leak problem, we will refer to $\mathcal{D}_s$ as *a CQ weakly acyclic information-leak instance (or setting).*

Assumptions about the attackers' knowledge. We assume that the presumed attackers have access to all and only the given instance $\mathcal{D}_s$. That is, they know

[1] In Section 7.3 we address the problem of determining the validity of a given instance of the information-leak problem.

the schema $\mathbf{P}$ in $\mathcal{D}_s$ and the set Σ of dependencies on $\mathbf{P}$. They also have access to the definition of Q, to all the queries for $\mathcal{V}$, and to (the contents of) all the relations in the instance MV. We assume specifically that the attackers have not been given any instance I such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$, and that they do not have access to the answer to the query Q on any such instance I.

DEFINITION 4.2. *(***Information leak and its disclosure***) Given a valid specific instance $\mathcal{D}_s = (\mathbf{P}, \Sigma, \mathcal{V}, Q, MV)$ of the information-leak problem, with $k \geq 0$ the arity of the query Q. Consider a tuple $\bar{t}$ of k (not necessarily distinct) constants. (In case where $k = 0$, we set $\bar{t}$ to the empty tuple.) Then we say that:*

(1) $\bar{t}$ is a potential information leak (for Q) *in $\mathcal{D}_s$ iff for all I such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$ we have $\bar{t} \in Q(I)$;*

(2) $\bar{t}$ is a disclosed information leak in $\mathcal{D}_s$ *iff one can prove deterministically using the attackers' knowledge that $\bar{t}$ is a potential information leak in $\mathcal{D}_s$; and*

(3) There is a disclosure of an information leak in $\mathcal{D}_s$ *iff some tuple $\bar{t}$ is a disclosed information leak in $\mathcal{D}_s$. Any such $\bar{t}$ is called a* witness of the information-leak disclosure in $\mathcal{D}_s$. □

The intuition for Definition 4.2 is as follows. Suppose that attackers are examining a set of view answers MV that is available to them, trying to find answers to the secret query Q on the ground instance, call it I, of schema $\mathbf{P}$ such that I has actually given rise to the set MV. (That is, I is an actual instance of the organizational data, such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$ and such that the set MV was obtained by applying to I the queries for $\mathcal{V}$.) However, by our assumptions about the attackers' knowledge, the attackers have no way of determining which ground instance J of $\mathbf{P}$, with the property $\mathcal{V} \Rightarrow_{J,\Sigma} MV$, is that desired actual instance I behind the set MV. Thus, their best bet is to find all the tuples $\bar{t}$ that are in the answer to Q on *all* instances J as above. This reasoning justifies item (1) of Definition 4.2. Items (2) and (3) of Definition 4.2 concern approaches that the attackers might select for proving that a fixed tuple $\bar{t}$ satisfies item (1) as above. In this paper we consider three such approaches, please see Sections 5–7.

DEFINITION 4.3. *(***Problem of information-leak disclosure***) Given a valid specific instance $\mathcal{D}_s$ of the information-leak problem, the* problem of information-leak disclosure for $\mathcal{D}_s$ *is to determine whether there is a disclosure of an information leak in $\mathcal{D}_s$.* □

In this paper, we address the problem of information-leak disclosure by developing approaches that output the set of potential information leaks for each input $\mathcal{D}_s$. Clearly, only those approaches that are sound and complete algorithms (for certain classes of inputs) can be used to solve correctly the problem as in Definition 4.3.

The intent of the use of the word "specific" for an instance $\mathcal{D}_s$ is to point out the presence in $\mathcal{D}_s$ of a fixed set of view answers MV in the context of the problem of information-leak disclosure. In contrast, a "general" instance of the problem, one that does not fix any set MV, would be considered in the context of a more general question: Does there exist *any* (Σ-valid) instance MV that would permit attackers to deterministically derive any answer to the secret query Q on the database of interest? As we are not considering the more general question here, in the remainder of this paper we will not be using the word "specific" explicitly when referring to instances of the information-leak problem.

5. THE REWRITING APPROACH

In this section we introduce an approach, called "the rewriting approach," to solving the problem of information-leak disclosure for CQ inputs. We argue that this approach arises naturally in the context of the problem; in a sense, it is the first thing to come to one's mind when thinking about the problem. This approach results in a sound and complete algorithm for the special case where the input set of dependencies Σ is the empty set. Due to the space limit, all the technical details on this approach can be found in Appendix B.

5.1 The Big Picture

To the best of our knowledge, the problem of information-leak disclosure as in Definition 4.3 has not been considered in the open literature. (Abiteboul and Duschka in [1] consider the special case where the set of dependencies is empty, apply in their analysis a different type of complexity metric than we do in this paper, and do not provide algorithms alongside their complexity results.) Thus, we have undergone a systematic study of potential approaches to solving this problem. As discussed in Section 4, our focus has been on developing approaches that output the set of potential information leaks for each input $\mathcal{D}_s$.

In Sections 5–7 we introduce three classes of approaches to solving the problem of information-leak disclosure. As the approach of Section 7 results in a sound and complete algorithm for the class of CQ weakly acyclic inputs, we concentrate our discussion on that approach in the limited page space. At the same time, our results are also conclusive for the approaches of Sections 5 and 6; for this reason, we provide a brief outline of each approach in the main body of the paper. All the technical details on the approaches of Sections 5 and 6 can be found in Appendices B and F, respectively.

5.2 Rewritings to Find Secret-Query Answers

Our rewriting approach formalizes a likely thought process of the presumed attackers. (See Appendix A for an illustration). Recall that in any instance $\mathcal{D}_s$ of the problem of information-leak disclosure, attackers deal directly with a ground instance MV. They know that MV is a set of answers to the access-policy views $\mathcal{V}$ on the underlying database of interest. Thus, intuitively, a question that is natural for the attackers to ask is which values in $adom(MV)$ can be put together to form an answer to the secret query Q, on all possible underlying databases of interest. (Our approach can also incorporate constants from definitions of the views in $\mathcal{V}$, and is also applicable to the case where Q is a Boolean query.)

A natural way to formalize this idea is to put together a query, call it R, in terms of the relations in the instance MV, and to then prove that R is "contained," in some precise sense (in particular, w.r.t. the views in $\mathcal{V}$), in the secret query Q. We refer to all queries R over the schema of MV as "rewritings" (in terms of $\mathcal{V}$), as indeed they would be defined in terms of the relation symbols in $\mathcal{V}$, that is in terms of views. Hence our name for this approach to information-leak disclosure.

Observe that, in the set $R(MV)$ of answers to a rewriting R on an instance MV, not all the tuples in $R(MV)$ would necessarily be in the answer to the query Q. That

is, the formal containment that we are looking for would not hold for all rewritings R. Thus, we restrict the scope of rewritings R to those CQ rewritings whose head includes only constants. That is, for a ground k-ary tuple $\bar{t}$ built using only values from $adom(MV)$, we consider only those k-ary rewritings R whose head is exactly $\bar{t}$.

We have shown that to solve the problem of information-leak disclosure for CQ inputs, for each tuple $\bar{t}$ as above it is sufficient to consider one CQ rewriting, denoted $R^*_{\bar{t}}$, whose body is the conjunction of all the tuples in the instance MV. That is, for each tuple $\bar{t}$ the rewriting $R^*_{\bar{t}}$ is a representative, in a precise sense, of all the CQ rewritings of the form $R(\bar{t})$. Denote by $(R^*_{\bar{t}})^{exp}$ the equivalent expansion of $R^*_{\bar{t}}$ in terms of the schema $\mathbf{P}$.

Now our *rewriting approach* is as follows. Given a valid CQ input instance $\mathcal{D}_s$ of information leak, with k the arity of its query Q. For each k-tuple $\bar{t}$ that can be generated from the set $adom(MV)$, the approach outputs $\bar{t}$ if and only if it can be shown formally that $(R^*_{\bar{t}})^{exp}$ is "appropriately contained" in Q.

We have studied several alternative formal definitions of the term "appropriately contained." In particular, we have introduced a definition that generalizes a key notion of [33] due to Zhang and Mendelzon. Specifically, the definition of *conditional query containment* of [33] is a restriction of our Definition 5.1 to the case $\Sigma = \emptyset$.

DEFINITION 5.1. *(**Σ-conditional query containment**) Given schema $\mathbf{P}$, set Σ of dependencies on $\mathbf{P}$, and set MV of view answers for a set of views $\mathcal{V}$. For queries Q_1 and Q_2 over $\mathbf{P}$, we say that Q_1* is Σ-conditionally contained in Q_2 w.r.t. *MV, denoted $Q_1 \sqsubseteq_{\Sigma,MV} Q_2$, iff for each instance, I, of $\mathbf{P}$ such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$, we have $Q_1(I) \subseteq Q_2(I)$.* □

We show that when Definition 5.1 is the meaning of "appropriate containment" in our rewriting approach, the approach captures exactly (in a sound and complete way) the potential information leaks for a given CQ instance $\mathcal{D}_s$. That is, for each class $\mathcal{E}$ of sets of dependencies, we have that whenever the set Σ in $\mathcal{D}_s$ belongs to $\mathcal{E}$, then a k-tuple $\bar{t}$ is a potential information leak in the instance $\mathcal{D}_s$ if and only if $\bar{t}$ is output by our rewriting approach with $(R^*_{\bar{t}})^{exp} \sqsubseteq_{\Sigma,MV} Q$ the meaning of "appropriate containment" in the approach.

Clearly, the missing part is a decidable test for the containment $(R^*_{\bar{t}})^{exp} \sqsubseteq_{\Sigma,MV} Q$. The main result of [33] provides a decidable (and Π^p_2 complete) test for the CQ case with $\Sigma = \emptyset$. (Please see Appendix D for the details.) Thus, for the class of CQ instances $\mathcal{D}_s$ with $\Sigma = \emptyset$, our rewriting approach is a sound and complete algorithm for disclosing information leaks. By this algorithm, the tuple $\bar{t}$ of Example 1.1 is a potential information leak for the setting in the example.

To extend our rewriting approach to the case of inputs $\mathcal{D}_s$ with $\Sigma \neq \emptyset$, one could try to develop a decidable test for $Q_1 \sqsubseteq_{\Sigma,MV} Q_2$ for the case where $\Sigma \neq \emptyset$. As the Π^p_2 complete test of [33] for $\Sigma = \emptyset$ is quite elaborate, it is not immediately clear how such an extension can be done. However, we observe that to solve the problem of information-leak disclosure (in the CQ case), we do not need to solve the problem of Σ-conditional containment between two queries w.r.t. a given set MV of view answers. Instead, for our purpose it is sufficient to solve a restriction of this problem to the case where the query Q_1 "represents exactly" the set MV. (Recall our definition of the rewriting $R^*_{\bar{t}}$ in terms of tuple $\bar{t}$ and of set MV.) This realization permits us to solve in Section 7 the problem of information-leak disclosure in the CQ weakly acyclic setting, even though we have not been able to develop a decidable test for $Q_1 \sqsubseteq_{\Sigma,MV} Q_2$ for the case where Q_1 and Q_2 are CQ queries and $\Sigma \neq \emptyset$.

Finally, we have shown that our rewriting approach is *sound* for CQ inputs with $\Sigma \neq \emptyset$ whenever one of $\sqsubseteq$, $\sqsubseteq_\Sigma$, and $\sqsubseteq_{\emptyset,MV}$ is used as the notion of "appropriate containment" in the approach. We show (in Appendix E) that none of these containment notions makes the approach *complete* for CQ weakly acyclic inputs.

6. THE DATA-EXCHANGE APPROACH

In Section 5 we considered an approach in which attackers would put together ground tuples using constants in the set of view answers MV that is available to them. The hope of the attackers is that these tuples could be shown to also be answers to the secret query Q, on all possible organizational databases that could have generated MV. For those attackers that have taken a course on data exchange, Definition 4.2 might suggest immediately a more sophisticated approach to disclosing information leaks. Indeed, item (1) of Definition 4.2 spells out the notion of "certain answers" to the secret query Q, in a data-exchange setting where we are given a "source instance" MV and a "target schema" $\mathbf{P}$. In this current section we pursue this intuition.

The main, perhaps surprising, result of this section is that the high(er)-tech data-exchange approach is not as powerful as the rewriting approach of Section 5, even for those CQ instances $\mathcal{D}_s$ where the set Σ of dependencies is the empty set. (For instance, the sound rewriting approach returns a nonempty set of disclosed information leaks in the setting of Example 1.1, while the data-exchange approach returns the empty set for the same input.) As it turns out, the data-exchange approach can be extended to yield a sound and complete algorithm for the problem of information-leak disclosure for all CQ weakly-acyclic inputs, including those for which Section 5 provides no solutions. We introduce the sound and complete extension in Section 7.

In this section, we first review in Section 6.1 the basics of data exchange. Then, in Section 6.2, we introduce and discuss the data-exchange approach to disclosure of information leaks. All the technical details of the discussion can be found in Appendix F.

6.1 Reviewing Data Exchange

Given schemas $\mathbf{S} = < S_1, \ldots, S_m >$ and $\mathbf{T} = < T_1, \ldots, T_n >$, with no relation symbols in common, denote by $<\mathbf{S}, \mathbf{T}>$ the schema $< S_1, \ldots, S_m, T_1, \ldots, T_n >$. If I is an instance of $\mathbf{S}$ and J an instance of $\mathbf{T}$, then (I, J) denotes an instance K of $< \mathbf{S}, \mathbf{T} >$ such that $K[S_i] = I[S_i]$ and $K[T_j] = J[T_j]$, for $i \in [1, m]$ and $j \in [1, n]$.

DEFINITION 6.1. *(**Data-exchange setting**) A* data-exchange setting *$\mathcal{M}$ is a triple $(\mathbf{S}, \mathbf{T}, \Sigma)$, where $\mathbf{S}$ and $\mathbf{T}$ are disjoint schemas and Σ is a finite set of dependencies over $<\mathbf{S}, \mathbf{T}>$. $\mathbf{S}$ in $\mathcal{M}$ is called the* source schema, *and $\mathbf{T}$ is called the* target schema. □

Instances of $\mathbf{S}$ are called *source* instances and are always ground instances. Instances of $\mathbf{T}$ are *target* instances. Given a source instance I, we say that a target instance J is a *solution for I (under $\mathcal{M}$)* if $(I, J) \models \Sigma$.

It is customary in the data-exchange literature[2] to restrict the study to the class of settings whose set Σ can be split into two sets Σ_{st} and Σ_t, as follows:

1. Σ_{st} is a set of *source-to-target* dependencies *(stds),* that is, tgds of the form $\varphi_{\mathbf{S}}(\bar{X}) \rightarrow \exists\, \bar{y}\ \psi_{\mathbf{T}}(\bar{X}, \bar{Y})$, where $\varphi_{\mathbf{S}}(\bar{X})$ and $\psi_{\mathbf{T}}(\bar{X}, \bar{Y})$ are conjunctions of relational atoms in $\mathbf{S}$ and $\mathbf{T}$, respectively; and
2. Σ_t, the set of *target* dependencies, is the union of a set of tgds and egds defined over the schema $\mathbf{T}$.

In this paper, we assume all data-exchange settings to be of the form $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where $\Sigma = \Sigma_{st} \cup \Sigma_t$, for Σ_{st} a set of stds and Σ_t a set of target dependencies. Intuitively, the stds can be viewed as a tool for specifying how the source data get translated into target data. In addition, the target dependencies are the usual database constraints, to be satisfied by the translated data. The data-exchange settings of this form are not restrictive from the database point of view.

Solutions for a given source instance are not necessarily unique, and there are source instances that have no solutions. *Universal solutions* are, intuitively, "the most general" solutions among all possible solutions. Formally, given a solution J for source instance I, we say that J is a *universal* solution for I if for every solution J' for I, there exists a homomorphism from J to J'. Constructing a universal solution for a given source instance I can be done by chasing I with $\Sigma_{st} \cup \Sigma_t$. The chase may never terminate or may fail; in the latter case, no solution exists [18]. If the chase does not fail and terminates, then the resulting target instance is guaranteed to be a universal solution for I.

The problem of checking for the existence of solutions is known to be undecidable, please see [6]. At the same time, the following positive result is due to [18].

THEOREM 6.1. *[18] Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st} \cup \Sigma_t)$ be a fixed data-exchange setting, such that Σ_t is weakly acyclic. Then there is a polynomial-time algorithm such that for every source instance I, the algorithm decides whether a solution for I exists. Then, whenever a solution for I exists, the algorithm computes a universal solution for I in polynomial time.* □

The universal solution of Theorem 6.1, called the *canonical* universal solution [18], is the result of the chase.

Query answering: Assume that a user poses a query Q over the target schema $\mathbf{T}$, and I is a given source instance. Then the usual semantics for the query answering is that of *certain answers* [6]. Let $\mathcal{M}$ be a data-exchange setting, let Q be a query over the target schema $\mathbf{T}$ of $\mathcal{M}$, and let I be a source instance. We define $certain_{\mathcal{M}}(Q, I)$, the set of *certain answers of Q with respect to I under $\mathcal{M}$*, as

$$certain_{\mathcal{M}}(Q, I) = \bigcap \{\ Q(J) \mid J \text{ is a solution for } I\ \}.$$

Computing certain answers for arbitrary FO queries is an undecidable problem. For unions of CQ queries *(UCQ queries),* which is a query language including all CQ queries, we have the following positive result:

THEOREM 6.2. *[18] Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st} \cup \Sigma_t)$ be a data-exchange setting with Σ_t a weakly acyclic set, and let Q be a UCQ query. Then the problem of computing certain answers for Q under $\mathcal{M}$ can be solved in polynomial time.* □

[2]We refer the reader to [6] for an excellent detailed survey of the literature on data exchange.

To compute the certain answers to a UCQ query Q w.r.t. a source instance I, we first check whether a solution for I exists. If there is no solution, the setting is inconsistent w.r.t. I. Otherwise, compute an arbitrary universal solution J for I, and then compute the set $Q_{\downarrow}(J)$ of all those tuples in $Q(J)$ that do not contain nulls. It can be shown that $Q_{\downarrow}(J) = certain_{\mathcal{M}}(Q, I)$.

6.2 Data Exchange for Information Leaks

Suppose that potential attackers are given a valid CQ instance $\mathcal{D}_s = (\mathbf{P}, \Sigma, \mathcal{V}, Q, MV)$ of the information-leak problem. By Definition 4.2, the attackers are interested in finding tuples $\bar{t}$ of elements of $adom(MV)$, such that for all instances I with $\mathcal{V} \Rightarrow_{I,\Sigma} MV$, we have $\bar{t} \in Q(I)$.

In this subsection we show how a straightforward reformulation of $\mathcal{D}_s$ turns the above problem into an instance of the problem of computing certain answers in data exchange. We first construct a set Σ_{st} of tgds, as follows. For a view V in the set of views $\mathcal{V}$ in $\mathcal{D}_s$, consider the query $V(\bar{X}) \leftarrow body_{(V)}(\bar{X}, \bar{Y})$ for V. (As $\mathcal{D}_s$ is a CQ instance, the query for each $V \in \mathcal{V}$ is a CQ query.) We associate with this $V \in \mathcal{V}$ the tgd $\sigma_V : V(\bar{X}) \rightarrow \exists \bar{Y}\ body_{(V)}(\bar{X}, \bar{Y})$. We then define the set Σ_{st} to be the set of tgds σ_V for all $V \in \mathcal{V}$. Then $\mathcal{D}_s$ can be reformulated into the following data-exchange setting:

$$\mathcal{M}^{(de)}(\mathcal{D}_s) = (\mathcal{V}, \mathbf{P}, \Sigma_{st} \cup \Sigma),$$

with a source instance MV and a query Q on the target schema $\mathbf{P}$. We call the triple $(\mathcal{M}^{(de)}(\mathcal{D}_s), MV, Q)$ *the associated data-exchange instance for $\mathcal{D}_s$.*

For valid CQ weakly acyclic instances $\mathcal{D}_s$, we introduce the following algorithm, which we call the *data-exchange approach to disclosing information leaks.* First, we compute the canonical universal solution, $J_{de}^{\mathcal{D}_s}$, for the source instance MV in the data-exchange setting $\mathcal{M}^{(de)}(\mathcal{D}_s)$. If $J_{de}^{\mathcal{D}_s}$ does not exist, then we output the empty set of answers. Otherwise we output, as a set of potential information leaks in $\mathcal{D}_s$, the set of all those tuples in $Q(J_{de}^{\mathcal{D}_s})$ that do not contain nulls. When we assume, similarly to [33], that everything in $\mathcal{D}_s$ is fixed except for MV and Q, then from Theorem 6.2 due to [18] we obtain immediately that this algorithm always terminates and runs in polynomial time. We have shown that this data-exchange approach is sound.

It turns out that our data-exchange approach is not complete for CQ weakly acyclic instances $\mathcal{D}_s$ with $\Sigma = \emptyset$, nor for those with $\Sigma \neq \emptyset$. (Due to the space limit, we have placed all the details into Appendix F.2.) We now discuss a feature of the data-exchange approach that prevents us from using it as a complete algorithm for the problem of disclosing information leaks. In Section 7 we will eliminate this feature of the data-exchange approach, in a modification that will yield a sound and complete algorithm for disclosing information leaks for all CQ weakly acyclic instances $\mathcal{D}_s$.

Why is the data-exchange approach not complete when applied to CQ weakly acyclic inputs? Intuitively, the problem is that its canonical universal solution $J_{de}^{\mathcal{D}_s}$ "covers too many target instances." Let us rewrite the set MV of Example 1.1 using, to save space, constants c, d, and f, as $MV = \{V(c, d), W(d, f)\}$. Now let us evaluate the queries for the views V and W of Example 1.1 over the canonical solution $J_{de}^{\mathcal{D}_s} = \{E(c, d, \perp_1), E(\perp_2, d, f)\}$ for that example. We obtain that the an-

swer to the view V on $J_{de}^{\mathcal{D}_s}$ is $\{V(c,d), V(\perp_2,d)\}$. Similarly, the answer to W on $J_{de}^{\mathcal{D}_s}$ is $\{W(d,\perp_1), W(d,f)\}$. Thus, if we replace $\perp_1$ in $J_{de}^{\mathcal{D}_s}$ by any constant except f, or replace $\perp_2$ by any constant except c, then any ground instance obtained from $J_{de}^{\mathcal{D}_s}$ using these replacements would "generate too many tuples" (as compared with MV) in the answer to either V or W.

We now generalize over this observation. Fix a valid CQ weakly acyclic instance $\mathcal{D}_s$, and consider the canonical universal solution (if one exists) $J_{de}^{\mathcal{D}_s}$ generated by the data-exchange approach with $\mathcal{D}_s$ as input. (In the remainder of this paper, we will refer to $J_{de}^{\mathcal{D}_s}$ as *the canonical data-exchange solution for* $\mathcal{D}_s$.) By definition of $J_{de}^{\mathcal{D}_s}$, for each $V \in \mathcal{V}$, the answer to the query for V on $J_{de}^{\mathcal{D}_s}$ is a superset of the relation $MV[V]$. Suppose that the answer on $J_{de}^{\mathcal{D}_s}$ to at least one view $V \in \mathcal{V}$ is not a subset of $MV[V]$, as it is the case in the example that we have just discussed. Then $J_{de}^{\mathcal{D}_s}$, as a template for instances of schema $\mathbf{P}$, describes not only instances that "generate" exactly the set MV in $\mathcal{D}_s$, but also those instances that generate proper supersets of MV. The latter instances are not of interest to the potential attackers. (Recall that attackers are interested only in the instances I of schema $\mathbf{P}$ such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$.) As a result, when the data-exchange approach uses $J_{de}^{\mathcal{D}_s}$ to obtain certain answers to the secret query Q, it can easily miss those certain answers that characterize only those instances that are of interest to the attackers.

7. VIEW-VERIFIED DATA EXCHANGE

The problem with the data-exchange approach of Section 6 is that its canonical universal solution, when turned into a ground instance, may produce a proper superset of the given set of view answers MV. (See Section 6.2.) That is, the canonical data-exchange solution does not necessarily describe ground solutions for $\mathcal{D}_s$ "tightly enough." The approach that we introduce in this section builds on data exchange, by "tightening" its universal solutions using $adom(MV)$. This approach, which we call *view-verified data exchange,* solves correctly the problem of disclosing information leaks in the CQ weakly-acyclic setting. We also use the approach of this section to solve the problem of deciding whether a given instance $\mathcal{D}_s$ of information leak is valid.

7.1 Chase with MV-Induced Dependencies

In Section 7.2 we will define view-verified data exchange for CQ weakly acyclic instances $\mathcal{D}_s$ of information leak. Given a $\mathcal{D}_s$ with set of views $\mathcal{V}$ and set of view answers MV, the idea of the approach is to force the canonical data-exchange solution $J_{de}^{\mathcal{D}_s}$ for $\mathcal{D}_s$ to generate only the relations in MV as answers to the queries for $\mathcal{V}$. (By definition of $J_{de}^{\mathcal{D}_s}$, the answer on $J_{de}^{\mathcal{D}_s}$ to the query for each $V \in \mathcal{V}$ is always a superset of the relation $MV[V]$.) We achieve this goal by chasing $J_{de}^{\mathcal{D}_s}$ using "MV-induced" dependencies. Intuitively, applying MV-induced dependencies to the instance $J_{de}^{\mathcal{D}_s}$ forces some nulls in $J_{de}^{\mathcal{D}_s}$ to become constants in $adom(MV)$. As a result, for at least one view $V \in \mathcal{V}$, some formerly non-ground tuples in the answer to V on the instance become ground tuples in $MV[V]$.

We now formally define MV-induced dependencies. Let $V(\bar{X}) \leftarrow \phi(\bar{X},\bar{Y})$ be a CQ query of arity $k_V \geq 0$, and MV be a ground instance of a schema that includes the k_V-ary relation symbol V. First, in case where $MV[V] = \emptyset$, we define the *MV-induced implication constraint (MV-induced ic) ι_V for V* as

$$\iota_V : \ \phi(\bar{X},\bar{Y}) \rightarrow \ false. \tag{1}$$

(Each MV-induced ic is an implication constraint, i.e., a Horn rule with the empty head. See [32] for the discussion and references on implication constraints.)

Second, in case where $k_V \geq 1$, suppose $MV[V] = \{\bar{t}_1, \bar{t}_2, \ldots, \bar{t}_{m_V}\}$, with $m_V \geq 1$. Then the *MV-induced generalized egd (MV-induced ged) τ_V for V* is

$$\tau_V : \phi(\bar{X},\bar{Y}) \rightarrow \vee_{i=1}^{m_V}(\bar{X} = \bar{t}_i). \tag{2}$$

Here, $\bar{X} = [S_1, \ldots, S_{k_V}]$ is the head vector of the query for V, with $S_j \in \textsc{Const} \cup \textsc{Qvar}$ for $j \in [1, k_V]$. For each $i \in [1, m_V]$ and for the ground tuple $\bar{t}_i = (c_{i1}, \ldots, c_{ik_V}) \in MV[V]$, we abbreviate by $\bar{X} = \bar{t}_i$ the conjunction $\wedge_{j=1}^{k_V}(S_j = c_{ij})$. MV-induced geds are a straightforward generalization of disjunctive egds of [15, 18].

We now define chase of instances with MV-induced dependencies. Consider first MV-induced implication constraints. Given an instance K of schema $\mathbf{P}$ and an MV-induced ic ι_V as in Eq. (1), suppose there exists a homomorphism h from the antecedent $\phi(\bar{X},\bar{Y})$ of ι_V to K. The intuition here is that we want to make sure that K does not "generate" any tuples in the relation $MV[V]$; however, by the existence of h, the instance K does generate at least one such tuple. We then say that *chase with ι_V (and h) fails on the instance K and produces the set* $\{\epsilon\}$, with ϵ denoting the empty instance.

Now let τ as in Eq. (2) be an MV-induced generalized egd for a $V \in \mathcal{V}$. The intuition here is that K must "generate" *only* the tuples in the relation $MV[V]$; we make this happen by assigning nulls in K to constants in $MV[V]$. (If such assignments are not possible, chase with τ fails on K.) Example 7.1 is the running example.

Our definition of the chase step with τ as in Eq. (2) is a straightforward extension of the definition of [18] for their disjunctive egds, as follows. Consider the consequent of τ, of the form $\vee_{i=1}^{m_V}(\bar{X} = \bar{t}_i)$. Recall that for each $i \in [1, m_V]$, the expression $\bar{X} = \bar{t}_i$ is of the form $\wedge_{j=1}^{k_V}(S_j = c_{ij})$. Denote by $\tau^{(1)}, \ldots, \tau^{(m_V)}$ the following m_V dependencies obtained from τ: $(\phi(\bar{X},\bar{Y}) \rightarrow \bar{X} = \bar{t}_1), \ldots, (\phi(\bar{X},\bar{Y}) \rightarrow \bar{X} = \bar{t}_{m_V})$, and call them *the dependencies associated with τ*. For each $i \in [1, m_V]$, $\tau^{(i)}$ is an embedded dependency that can be equivalently represented by k_V egds $\tau^{(i,1)}, \ldots, \tau^{(i,k_V)}$. Here, for each $j \in [1, k_V]$, the egd $\tau^{(i,j)}$ is $\phi(\bar{X},\bar{Y}) \rightarrow S_j = c_{ij}$.

Given a τ as in Eq. (2) and an instance K of schema $\mathbf{P}$, suppose that there exists a homomorphism h from $\phi(\bar{X},\bar{Y})$ to K such that $\wedge_{j=1}^{k_V}(h(S_j) = h(c_{ij}))$ is not a tautology for any $i \in [1, m_V]$. Then we say that τ *is applicable to K with the homomorphism h*. It is easy to see that it is also the case that each of $\tau^{(1)}, \ldots, \tau^{(m_V)}$ can be applied to K with h. That is, for each $i \in [1, m_V]$, the chase of K is applicable with at least one egd $\tau^{(i,j)}$ in the equivalent representation of $\tau^{(i)}$ as a set of egds. For each $i \in [1, m_V]$, let K_i be the result of applying all the egds $\tau^{(i,1)}, \ldots, \tau^{(i,k_V)}$ to K with h. Note that chase with $\tau^{(i,j)}$ and h can fail on K for some

i and j. For each such i, we say that *chase with* $\tau^{(i)}$ *fails on* K *and produces the empty instance* ϵ.

Similarly to [18], we distinguish two cases:

- If the set $\{K_1, \ldots, K_{m_V}\}$ contains only empty instances, we say that *chase with* τ *(and* h*) fails on* K *and produces the set* $\{\epsilon\}$.
- Otherwise, let $\mathcal{K}^{(\tau)} = \{K_{i_1}, \ldots, K_{i_p}\}$ be the set of all nonempty elements of $\{K_1, \ldots, K_{m_V}\}$. We say that $\mathcal{K}^{(\tau)}$ *is the result of applying* τ *to* K *with* h.

Similarly to the approach of [18], in addition to chase steps with MV-induced dependencies we will also use chase steps with egds and tgds as in Section 3.2. For the chase step of each type, we will use the set notation for uniformity: $K \Rightarrow^{\sigma,h} \mathcal{K}'$ denotes that a chase step with dependency σ and homomorphism h applied to instance K yields a set of instances $\mathcal{K}'$. Whenever chase with an egd fails on K, the set $\mathcal{K}'$ is the set $\{\epsilon\}$ by convention; in all other cases where σ is an egd or a tgd, the set $\mathcal{K}'$ is a singleton set. For σ as in Eq. (1)–(2), the set $\mathcal{K}'$ is in some cases $\{\epsilon\}$ as defined above.

DEFINITION 7.1. *(MV-**enhanced chase**) Let Σ be a set of egds and tgds, let $\Sigma^{(MV)}$ be a set of MV-induced dependencies, and let K be an instance.*

- *A* chase tree of K with $\Sigma \cup \Sigma^{(MV)}$ *is a tree (finite or infinite) such that:*
  - *The root is K, and*
  - *For every node K_j in the tree, let $\mathcal{K}_j$ be the set of its children. Then there must exist some dependency σ in $\Sigma \cup \Sigma^{(MV)}$ and some homomorphism h such that $K_j \Rightarrow^{\sigma,h} \mathcal{K}_j$.*
- *A* finite MV-enhanced chase of K with $\Sigma \cup \Sigma^{(MV)}$ *is a finite chase tree $\mathcal{T}$, such that for each leaf K_p of $\mathcal{T}$, (a) K_p is ϵ, or (b) there is no dependency σ in $\Sigma \cup \Sigma^{(MV)}$ and no homomorphism h such that σ can be applied to K_p with h.* □

EXAMPLE 7.1. *Consider $\mathcal{D}_s = (\{E\}, \emptyset, \{V, W\}, Q, MV'')$, with all the elements except MV'' as in Example 1.1;*[3] *$MV'' = \{ V(c,d), V(g,d), W(d,f) \}$.*

By definition, $\mathcal{D}_s$ is a CQ instance with $\Sigma = \emptyset$. $\mathcal{D}_s$ is also valid, as witnessed by instance $\{E(c,d,f), E(g,d,f)\}$. The data-exchange approach of Section 6 yields the following canonical data-exchange solution $J_{de}^{\mathcal{D}_s}$ for $\mathcal{D}_s$:

$$J_{de}^{\mathcal{D}_s} = \{ E(c,d,\perp_1),\ E(g,d,\perp_2),\ E(\perp_3,d,f) \} .$$

The set of answers without nulls to the query Q on $J_{de}^{\mathcal{D}_s}$ is empty. Thus, the data-exchange approach applied to $\mathcal{D}_s$ discovers no potential information leaks.

In applying the view-verified data-exchange approach to the input $\mathcal{D}_s$, we first construct the MV''-induced generalized egds, τ_V and τ_W, one for each of the two views in $\mathcal{D}_s$. (As MV'' has no empty relations, we do not need to construct MV''-induced ics for $\mathcal{D}_s$.)

$\tau_V : E(X,Y,Z) \rightarrow (X = c \wedge Y = d) \vee (X = g \wedge Y = d)$.
$\tau_W : E(X,Y,Z) \rightarrow (Y = d \wedge Z = f)$.

[3]Please see Example F.1 for the details.

The two dependencies associated with τ_V are $\tau_V^{(1)} : E(X,Y,Z) \rightarrow (X = c \wedge Y = d)$ and $\tau_V^{(2)} : E(X,Y,Z) \rightarrow (X = g \wedge Y = d)$. Each of $\tau_V^{(1)}$ and $\tau_V^{(2)}$ can be equivalently represented by two egds. For instance, the egd representation for $\tau_V^{(1)}$ is via $\tau_V^{(1,1)} : E(X,Y,Z) \rightarrow X = c$ and $\tau_V^{(1,2)} : E(X,Y,Z) \rightarrow Y = d$. Similarly, there is one dependency $\tau_W^{(1)}$ $(= \tau_W)$ associated with τ_W; an equivalent representation of $\tau_W^{(1)}$ is via two egds.

Consider homomorphism $h_V^{(1)} : \{X \rightarrow c,\ Y \rightarrow d,\ Z \rightarrow \perp_1\}$ from the antecedent $E(X,Y,Z)$ of τ_V to the instance $J_{de}^{\mathcal{D}_s}$. As applying $h_V^{(1)}$ to the consequent of $\tau_V^{(1)}$ gives us the tautology $(c = c \wedge d = d)$, we conclude that τ_V is not applicable to $J_{de}^{\mathcal{D}_s}$ with $h_V^{(1)}$.

Consider now the homomorphism $h_V^{(2)} : \{X \rightarrow \perp_3,\ Y \rightarrow d,\ Z \rightarrow f\}$ from the antecedent of τ_V to $J_{de}^{\mathcal{D}_s}$. Applying $h_V^{(2)}$ to the consequent of τ_V gives us the expression $(\perp_3 = c \wedge d = d) \vee (\perp_3 = g \wedge d = d)$, which has no tautologies among its disjuncts. Thus, τ_V is applicable to $J_{de}^{\mathcal{D}_s}$ with $h_V^{(2)}$. The chase step with τ_V and $h_V^{(2)}$ transforms $J_{de}^{\mathcal{D}_s}$ into instances J_1 and J_2, as follows.

$$J_1 = \{ E(c,d,\perp_1),\ E(g,d,\perp_2),\ E(c,d,f) \} .$$
$$J_2 = \{ E(c,d,\perp_1),\ E(g,d,\perp_2),\ E(g,d,f) \} .$$

(J_1 results from assigning $\perp_3 := c$, and J_2 from $\perp_3 := g$.)

We then use the same procedure to apply τ_W to each of J_1 and J_2. In each case, the chase steps assign the value f to each of $\perp_1$ and $\perp_2$. As a result, the following instance $J_{vv}^{\mathcal{D}_s}$ is obtained from each of J_1 and J_2:

$$J_{vv}^{\mathcal{D}_s} = \{ E(c,d,f),\ E(g,d,f) \} .$$ □

7.2 Solving CQ Weakly Acyclic Instances

We now define the view-verified data-exchange approach to the problem of disclosing information leaks.

Let $\mathcal{D}_s = (\mathbf{P}, \Sigma, \mathcal{V}, Q, MV)$ be a CQ instance of information leak. Then *the set $\Sigma_{\mathcal{D}_s}^{(MV)}$ of MV-induced dependencies for $\mathcal{D}_s$* is a set of up to $|\mathcal{V}|$ elements, as follows. For each $V \in \mathcal{V}$ such that $k_V \neq 0$ or $MV[V] \neq \{()\}$, $\Sigma_{\mathcal{D}_s}^{(MV)}$ has one MV-induced implication constraint or one MV-induced generalized egd, by the rules as in Eq. (1)–(2) in Section 7.1.[4]

For CQ weakly acyclic instances $\mathcal{D}_s$ we introduce the following *view-verified data-exchange approach to information-leak disclosure.* First, we compute (as in Section 6) the canonical universal solution $J_{de}^{\mathcal{D}_s}$ for the source instance MV in the data-exchange setting $\mathcal{M}^{(de)}(\mathcal{D}_s)$. If $J_{de}^{\mathcal{D}_s}$ does not exist, we stop and output the answer that $\mathcal{D}_s$ is not valid. Otherwise we obtain a chase tree of $J_{de}^{\mathcal{D}_s}$ with $\Sigma \cup \Sigma_{\mathcal{D}_s}^{(MV)}$, where $\Sigma_{\mathcal{D}_s}^{(MV)}$ is the set of MV-induced dependencies for $\mathcal{D}_s$. If the chase tree is finite, denote by $\mathcal{J}_{vv}^{\mathcal{D}_s}$ the set of all the nonempty

[4]We omit from $\Sigma_{\mathcal{D}_s}^{(MV)}$ the dependencies, of the form $\phi(\bar{X}, \bar{Y}) \rightarrow \mathit{true}$, for the case where $k_V = 0$ *and* $MV[V] \neq \emptyset$. By the results in this section, adding these dependencies to $\Sigma_{\mathcal{D}_s}^{(MV)}$ would not change any chase results.

leaves of the tree. We call each $J \in \mathcal{J}_{vv}^{\mathcal{D}_s}$ a *view-verified universal solution for* $\mathcal{D}_s$. If $\mathcal{J}_{vv}^{\mathcal{D}_s} = \emptyset$, then we stop and output the answer that $\mathcal{D}_s$ is not valid. Otherwise, for each $J \in \mathcal{J}_{vv}^{\mathcal{D}_s}$ we compute the set $Q_{\downarrow}(J)$ of all the tuples in $Q(J)$ that do not contain nulls. Finally, the output of the approach for the input $\mathcal{D}_s$ is the set

$$\bigcap_{J \in \mathcal{J}_{vv}^{\mathcal{D}_s}} Q_{\downarrow}(J). \tag{3}$$

The view-verified data-exchange approach to disclosing information leaks addresses the shortcoming of the data-exchange approach, see Section 6. Recall that the canonical universal solution $J_{de}^{\mathcal{D}_s}$ of the latter approach might not cover "tightly enough" all the instances of interest to the attackers. In the view-verified approach, we address this problem, by using our extension of the chase to generate from $J_{de}^{\mathcal{D}_s}$ a set $\mathcal{J}_{vv}^{\mathcal{D}_s}$ of instances that are each "tighter" than $J_{de}^{\mathcal{D}_s}$ in this sense.

In Section 7.3 we will show that the view-verified data-exchange approach is a sound and complete algorithm for the problem of disclosing information leaks for CQ weakly acyclic inputs. In particular, we will see that the set $\mathcal{J}_{vv}^{\mathcal{D}_s}$ is well defined, in that the chase tree in the view-verified data-exchange approach is always finite. We will also see that the set $\mathcal{J}_{vv}^{\mathcal{D}_s}$ is "just tight enough," in the following sense: Denote by $certain(Q, \mathcal{D}_s)$ the set of all the potential information leaks for $\mathcal{D}_s$, see Definition 4.2 (1). Then the expression in Eq. (3), which is the intersection of all the "certain-answer expressions" for Q and for the individual elements of the set $\mathcal{J}_{vv}^{\mathcal{D}_s}$, is exactly the set $certain(Q, \mathcal{D}_s)$.

EXAMPLE 7.2. *Recall the instance $\mathcal{D}_s$ of information leak in Example 7.1, and the instance $J_{vv}^{\mathcal{D}_s}$ obtained in that example. $J_{vv}^{\mathcal{D}_s}$ is the (only) view-verified universal solution for $\mathcal{D}_s$. The set of answers without nulls to the query Q on $J_{vv}^{\mathcal{D}_s}$ is $\{(c, f), (g, f)\}$. Thus, (c, f), (g, f) are potential information leaks discovered for the instance $\mathcal{D}_s$ by the view-verified data-exchange approach. Both (c, f) and (g, f) (and nothing else) are also discovered by the rewriting algorithm of Section 5, which is sound and complete for $\mathcal{D}_s$.* □

7.3 Correctness, Validity, and Complexity

In this subsection, we show that the view-verified data-exchange approach is sound and complete for all CQ weakly acyclic inputs. We also show how the approach can be used to decide whether a CQ weakly acyclic instance $\mathcal{D}_s$ is valid. Finally, we prove that the problem of information-leak disclosure is Π_2^p complete for CQ weakly acyclic inputs, under the assumption, same as in [33], that all elements of $\mathcal{D}_s$ except MV and Q are fixed. That is, the *size* of a given instance $\mathcal{D}_s$ is the size of its instance MV and query Q, with the remaining elements of $\mathcal{D}_s$ being fixed.

View-verified data exchange is an algorithm. We begin by obtaining a basic observation that builds on the results of [18] for chase with tgds and disjunctive egds (as defined in [18]). It is immediate from Proposition 7.1 that view-verified data exchange always terminates in finite time for CQ weakly acyclic inputs.

PROPOSITION 7.1. *Given a CQ weakly acyclic instance $\mathcal{D}_s$ of information leak, such that its canonical data-exchange solution $J_{de}^{\mathcal{D}_s}$ exists. Then we have that:*

(1) MV-enhanced chase of $J_{de}^{\mathcal{D}_s}$ with $\Sigma \cup \Sigma_{\mathcal{D}_s}^{(MV)}$ is a finite tree, $\mathcal{T}$, such that:
  (a) $\mathcal{T}$ is of polynomial depth in the size of $\mathcal{D}_s$, and
  (b) The number of leaves in $\mathcal{T}$ is up to exponential in the size of $\mathcal{D}_s$; and

(2) For each nonempty leaf J of $\mathcal{T}$, we have that:
  (a) J is of polynomial size in the size of $\mathcal{D}_s$, and
  (b) Each grounded version of J is a Σ-valid base instance for $\mathcal{V}$ and MV. □

(A *grounded version* of instance K results from replacing consistently all its nulls with distinct new constants.)

The proof of Proposition 7.1 relies heavily on the results of [18], particularly on its Theorem 3.9. Recall the "decomposition," in Section 7.1, of MV-induced generalized egds into egds as defined in Section 3.2. Intuitively, given a CQ weakly acyclic instance $\mathcal{D}_s$ and for each node K on each path from the root $J_{de}^{\mathcal{D}_s}$ of the tree $\mathcal{T}$ for $\mathcal{D}_s$, we can obtain K by chasing the root of $\mathcal{T}$ using only egds and weakly acyclic tgds.[5] The key observation here is that even though the set $\Sigma_{\mathcal{D}_s}^{(MV)}$ of dependencies is not fixed (in fact, its size is linear in the size of the instance MV in $\mathcal{D}_s$), all the constants that contribute to the size of $\Sigma_{\mathcal{D}_s}^{(MV)}$ are already used in the root $J_{de}^{\mathcal{D}_s}$ of the tree $\mathcal{T}$, by definition of $J_{de}^{\mathcal{D}_s}$. In addition, the antecedent of each MV-induced generalized egd in $\Sigma_{\mathcal{D}_s}^{(MV)}$ is of constant size, by definition of the size of $\mathcal{D}_s$. As a result, we can build on Theorem 3.9 and Proposition 5.6 of [18] to obtain items (1)(a) and (2)(a) of our Proposition 7.1.

Item (2)(b) of Proposition 7.1 is by definition of MV-enhanced chase, and (1)(b) is by construction of the tree $\mathcal{T}$. Appendix G provides a lower bound, via an example where for a CQ instance $\mathcal{D}_s$ with $\Sigma = \emptyset$, the number of leaves in a chase tree is exponential in the size of $\mathcal{D}_s$.

Soundness and completeness. By Proposition 7.1 (2)(b), the view-verified data-exchange approach is a *complete* algorithm when applied to CQ weakly acyclic instances $\mathcal{D}_s$. (That is, for each potential information-leak tuple $\bar{t}$ for a $\mathcal{D}_s$ in this class, view-verified data exchange outputs $\bar{t}$.) We now make a key observation toward a proof that this algorithm is also *sound* for such instances. (Soundness means that for each tuple $\bar{t}$ that this approach outputs for a $\mathcal{D}_s$ in this class, $\bar{t}$ is a potential information leak for $\mathcal{D}_s$.)

PROPOSITION 7.2. *Given a CQ weakly acyclic instance $\mathcal{D}_s = (\mathbf{P}, \Sigma, \mathcal{V}, Q, MV)$ of information leak. Then, for each instance I such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$, there exists a homomorphism from some view-verified universal solution for $\mathcal{D}_s$ to I.* □

The intuition for the proof of Proposition 7.2 is as follows. For a given $\mathcal{D}_s$, whenever an instance I exists such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$, a canonical data-exchange solution $J_{de}^{\mathcal{D}_s}$ for $\mathcal{D}_s$ must also exist. By definition of $J_{de}^{\mathcal{D}_s}$, there must be a homomorphism from $J_{de}^{\mathcal{D}_s}$ to the instance I. We then start applying MV-enhanced chase to $J_{de}^{\mathcal{D}_s}$, to

[5] Besides the egds and tgds of Section 3.2, chase on each path in $\mathcal{T}$ may use MV-induced implication constraints. However, the only role of the latter constraints is to obtain the instance ϵ and thus to terminate the respective path in $\mathcal{T}$.

simulate some rooted path, $P_{(\mathcal{T})}$, in the chase tree $\mathcal{T}$ for $\mathcal{D}_s$. (The tree is finite by Proposition 7.1.) In following the path $P_{(\mathcal{T})}$ via the chase, we make sure that there is a homomorphism from each node in the path to I, by always choosing an "appropriate" associated dependency $\tau^{(i)}$ for each MV-induced generalized egd τ that we are applying in the chase. By $\mathcal{V} \Rightarrow_{I,\Sigma} MV$, such a choice always exists, and the path $P_{(\mathcal{T})}$ terminates in finite time in a nonempty instance, J. By definition, J is a view-verified universal solution for $\mathcal{D}_s$. By our simulation of the path $P_{(\mathcal{T})}$ "on the way to" I, there exists a homomorphism from J to I.

Validity of instance $\mathcal{D}_s$. By the results of [18], when for a given $\mathcal{D}_s$ no canonical data-exchange solution exists, then $\mathcal{D}_s$ is not a valid instance. We refine this observation into a sufficient and necessary condition for validity of CQ weakly acyclic instances $\mathcal{D}_s$. (The only-if part of Proposition 7.3 follows from Proposition 7.2, and its if part is by Proposition 7.1 (2)(b).)

PROPOSITION 7.3. *Given a CQ weakly acyclic instance $\mathcal{D}_s$ of information leak, $\mathcal{D}_s$ is valid iff the set $\mathcal{J}_{vv}^{\mathcal{D}_s}$ of view-verified universal solutions for $\mathcal{D}_s$ is not empty.* □

Correctness of view-verified data exchange. By Proposition 7.2, view-verified data exchange is sound. By Proposition 7.3, it outputs a set of potential information leaks iff its input is valid. We now conclude:

THEOREM 7.1. *View-verified data exchange is a sound and complete algorithm for discovering potential information leaks for CQ weakly acyclic input instances.* □

Complexity of information-leak disclosure. We now show that finding potential information leaks for CQ weakly acyclic inputs is Π_2^p complete. (We assume, as in [33], that all elements of $\mathcal{D}_s$ except MV and Q are fixed.) By Theorem 7.1, this result also determines the runtime complexity of view-verified data exchange.

We first observe that the problem is in Π_2^p.

PROPOSITION 7.4. *The problem of information-leak disclosure is in Π_2^p for CQ weakly acyclic inputs.* □

Indeed, to compute the expression of Eq. (3) for a given CQ weakly acyclic instance $\mathcal{D}_s$, by Proposition 7.1 we can compute in polynomial time each view-verified universal solution J for $\mathcal{D}_s$ as a nonempty leaf in the chase tree for $\mathcal{D}_s$. We then compute in polynomial time the set $Q_{\downarrow}(J)$, and intersect $Q_{\downarrow}(J)$ with the partial expression for Eq. (3). We maintain the latter expression between creation of successive instances J. The result of Proposition 7.4 follows.

It turns out that the problem is also Π_2^p hard.

THEOREM 7.2. *The problem of information-leak disclosure is Π_2^p hard for CQ inputs with $\Sigma = \emptyset$.* □

The Π_2^p-completeness result is an immediate corollary:

THEOREM 7.3. *The problem of information-leak disclosure is Π_2^p complete for CQ weakly acyclic inputs.* □

The result of Theorem 7.2 is by reduction from the $\forall\exists$-CNF problem. Starting off from the reduction of [25], we use a modification that is similar in spirit to that suggested in [33]. The goal of our modification is to comply with our assumptions about the input size, specifically with the assumption that the input view definitions are fixed. (In [25] it is assumed that both the queries and the view definitions can vary.) The proof of Theorem 7.2 can be found in Appendix H.

8. REFERENCES

[1] S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. In *PODS*, pages 254–263, 1998.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[3] A. V. Aho, Y. Sagiv, and J. D. Ullman. Equivalences among relational expressions. *SIAM J. Comput.*, 8:218–246, 1979.

[4] E. Al-Shaer, H. Hamed, R. Boutaba, and M. Hasan. Conflict classification and analysis of distributed firewall policies. *IEEE JSAC*, 23(10):2069–2084, October 2005.

[5] P. Ammann and R. S. Sandhu. Safety analysis for the extended schematic protection model. In *Proc. IEEE Symposium on Security and Privacy*, pages 87–97, 1991.

[6] P. Barceló. Logical foundations of relational data exchange. *SIGMOD Record*, 38(1):49–58, 2009.

[7] E. Bertino, G. Ghinita, and A. Kamra. Access control for databases: Concepts and systems. *Foundations and Trends in Databases*, 3(1-2):1–148, 2011.

[8] J. Biskup and P. Bonatti. Controlled Query Evaluation for Known Policies by Combining Lying and Refusal. In *Int'l Sympos. Foundations of Inf. and Knowl. Sys.*, Feb. 2002.

[9] R. Bond, K. Y.-K. See, C. K. M. Wong, and Y.-K. H. Chan. *Understanding DB2 9 Security*. IBM Press, 2006.

[10] A. Brodsky, C. Farkas, and S. Jajodia. Secure databases: Constraints, inference channels, and monitoring disclosures. *IEEE TKDE*, 12(6):900–919, 2000.

[11] A. Chandra and P. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *STOC*, 1977.

[12] B.-C. Chen, D. Kifer, K. LeFevre, and A. Machanavajjhala. Privacy-preserving data publishing. *Foundations and Trends in Databases*, 2(1-2):1–167, 2009.

[13] A. Deutsch. *XML Query Reformulation over Mixed and Redundant Storage*. PhD thesis, Univ. Pennsylvania, 2002.

[14] A. Deutsch, A. Nash, and J. Remmel. The chase revisited. In *PODS*, pages 149–158, 2008.

[15] A. Deutsch and V. Tannen. Optimization properties for classes of conjunctive regular path queries. In *DBPL*, 2001.

[16] S. D. C. di Vimercati, S. Paraboschi, and P. Samarati. Access control: principles and solutions. *Softw., Pract. Exper.*, 33(5):397–421, 2003.

[17] J. Domingo-Ferrer, editor. *Inference Control in Statistical Databases, From Theory to Practice*. Springer-Verlag, 2002.

[18] R. Fagin, P. Kolaitis, R. Miller, and L. Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336:89–124, 2005.

[19] A. Fuxman, P. G. Kolaitis, R. J. Miller, and W.-C. Tan. Peer data exchange. *ACM TODS*, 31(4):1454–1498, 2006.

[20] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Comm. ACM*, 19:461–471, 1976.

[21] A. Levy, A. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *PODS*, 1995.

[22] N. Li, W. H. Winsborough, and J. C. Mitchell. Beyond proof-of-compliance: Safety and availability analysis in trust management. In *Proc. IEEE Symposium on Security and Privacy*, pages 123–139, 2003.

[23] G. Miklau and D. Suciu. A formal analysis of information disclosure in data exchange. In *ACM SIGMOD*, 2004.

[24] G. Miklau and D. Suciu. A formal analysis of information disclosure in data exchange. *JCSS*, 73(3):507–534, 2007.

[25] T. D. Millstein, A. Y. Halevy, and M. Friedman. Query containment for data integration systems. *JCSS*, 66, 2003.

[26] A. Nash, L. Segoufin, and V. Vianu. Views and queries: Determinacy and rewriting. *ACM TODS*, 35(3), 2010.

[27] Oracle. *The Virtual Private Database in Oracle9iR2: An Oracle Technical White Paper*, Jan. 2002.

[28] Oracle Cooperation. *Oracle Database: Security Guide*, December 2003. Available at www.oracle.com.

[29] C. E. Shannon. Communication theory of secrecy systems. *Bell Syst. Techn. J.*, 28:656–715, 1949.

[30] L. J. Stockmeyer. The polynomial-time hierarchy.

Theoretical Computer Science, 3(1):1–22, 1976.

[31] C. Yao, L. Wang, X. S. Wang, C. Bettini, and S. Jajodia. Evaluating privacy threats in database views by symmetric indistinguishability. *J. Comp. Sec.*, 17:5–42, 2009.

[32] X. Zhang and M. Özsoyoglu. Implication and referential constraints: A new formal reasoning. *IEEE TKDE*, 9, 1997.

[33] Z. Zhang and A. O. Mendelzon. Authorization views and conditional query containment. In *ICDT*, 2005.

APPENDIX

A. ELABORATION ON EXAMPLE 1.1

In this appendix we elaborate on Example 1.1, to illustrate the intuition behind the "rewriting approach" of Section 5.

EXAMPLE A.1. *Consider a relation* `Emp`*, which is used for storing information about employees of a company. Let the attributes of* `Emp` *be* `Name`*,* `Dept`*, and* `Salary`*, with self-explanatory attribute names:* `Emp(Name, Dept, Salary)`*.*

We assume that no integrity constraints hold on the database schema $\mathbf{P}$ *containing the relation* `Emp`*. (In particular, the only primary key of* `Emp` *is all its attributes.) Thus, the set* Σ *of dependencies holding on schema* $\mathbf{P}$ *is the empty set.*

Let a secret query `Q` *ask for the salaries of all the employees. We can formulate the query* `Q` *in SQL as*

```
(Q): SELECT DISTINCT Name, Salary FROM Emp;
```

Consider two views, `V` *and* `W`*, that are defined for some class(es) of users, in SQL on the schema* $\mathbf{P}$*. The view* `V` *returns the department for each employee, and the view* `W` *returns the salaries in each department:*

```
(V): DEFINE VIEW V(Name, Dept) AS
     SELECT DISTINCT Name, Dept FROM Emp;

(W): DEFINE VIEW W(Dept, Salary) AS
     SELECT DISTINCT Dept, Salary FROM Emp;
```

Suppose that some user(s) are authorized to see the answers to `V` *and* `W`*, and that at some point in time the user(s) can see the following set* MV *of answers to these views.*

$MV = \{$ `V(JohnDoe, Sales)`, `W(Sales, $50000)` $\}$.

Then one "conjunctive fact-expression" C_{MV} *that the user(s) can put together based on this database* MV *is*

$C_{MV =}$ `V(JohnDoe, Sales) AND W(Sales, $50000)` .

Let tuple $\bar{t} =$ (`JohnDoe`, `$50000`) *be the tuple that the user hypothesizes is in the answer to the secret query* `Q` *on all the instances of the relation* `Emp` *that satisfy the (empty set of) dependencies* Σ *and that generate the above database* MV*. Observe that the tuple* $\bar{t}$ *is made up from values* `JohnDoe` *and* `$50000`*, which "are generated by" the expression* C_{MV}*. Thus, knowing the associations between the values in the tuple* $\bar{t}$ *and the respective attribute names in* MV*, we can "put together" this expression* C_{MV} *and this tuple* $\bar{t}$ *as a SQL query,* `Rvw`*, in terms of the views* `V` *and* `W` *and in presence of the constants from the database* MV*, as follows:*

```
(Rvw): SELECT DISTINCT Name, Salary FROM V, W
       WHERE Name = 'JohnDoe' AND V.Dept = W.Dept
       AND V.Dept = 'Sales' AND Salary = '$50000';
```

That is, by defining the query `Rvw` *we formalize the rather natural process of the user "putting together" tuples in the available database* MV *and of his then using some of the values from the selected tuples to put forth a tuple of constants that is hypothetically in the answer to the secret query.*

By definition of the SQL query `Rvw`*, the above tuple* $\bar{t}$ = (`JohnDoe`, `$50000`) *is the only possible answer to* `Rvw` *on all possible instances of the relations* `V` *and* `W`*. It is easy to see that this answer to the query* `Rvw` *is compatible with (i.e., can be obtained by asking the query* `Rvw` *on) the above database* MV*. (Intuitively, this is true because we have constructed* `Rvw` *from the tuples in the above database* MV*.)* □

B. THE REWRITING APPROACH

In this appendix we provide the technical details on our rewriting approach (see Section 5) to the disclosure of information leaks.

B.1 The Intuition

The rewriting approach formalizes the thought process of the presumed attackers, as outlined in Example 1.1. We argue that this approach is natural for attackers to use. Recall that in any instance of the problem of information-leak disclosure, attackers deal directly with a ground instance MV. They know that MV is a set of answers to the access-policy views $\mathcal{V}$ on the underlying database of interest. Thus, intuitively, a question that is natural for the attackers to ask is which values in $adom(MV)$ can be put together to form an answer to the secret query Q, on all possible underlying databases of interest. (In general, the attackers could consider in their pursuit not just values in $adom(MV)$, but also constants mentioned in the queries for $\mathcal{V}$ and in the secret query Q. It is straightforward to reflect this in our setting, by adding extra head arguments to the definitions of the respective views. Thus, we do not explicitly consider this extension in this paper.)

How can this question be answered deterministically, as required by Definition 4.2? A natural approach would be to put together a query, call it R, in terms of the relations in the instance MV, and to then prove that R is "contained," in some precise sense (in particular, w.r.t. the views in $\mathcal{V}$), in the secret query Q. We will be referring to all queries R over schema $\mathcal{V}$ as "rewritings" (in terms of $\mathcal{V}$), as indeed they would be defined in terms of the relation symbols in $\mathcal{V}$. (Another reason to refer to such queries R as "rewritings" is that we will need to define their expansions shortly.) Hence our name for this approach to information-leak disclosure.

A challenge arises immediately when attackers pursue this train of thought: In the set $R(MV)$ of answers to a rewriting R on an instance MV, not all the tuples in $R(MV)$ would necessarily be in the answer to the secret query Q. That is, the formal containment that we are looking for would not hold for all rewritings R. (As an illustration, suppose that Q returns names of employees with high salaries, and that R returns names of employees in the accounting department. Clearly, the answer to R is not necessarily a subset of the answer to Q, on any particular database of interest.)

At the same time, by Definition 4.2, for each individual tuple $\bar{t} \in R(MV)$, it makes sense to ask the question of whether *the query* $R(\bar{t})$ is contained in Q in the appropriate precise sense. The intuition is that $R(\bar{t})$ is the result of binding the head vector of R to a tuple, $\bar{t}$, in the relation $R(MV)$; as a result, $\bar{t}$ is the only answer to $R(\bar{t})$ on the instance MV. We focus on such rewritings $R(\bar{t})$ in this paper.

B.2 Defining the Rewriting Approach

We now formalize the rewriting approach. Our intent is to tie the definitions of rewritings that attackers can formulate on view answers, to the definition of the secret query Q. After defining rewritings of the form $R(\bar{t})$, we recall the standard notion of *expansion* of a view-based rewriting; an expansion of a rewriting is its equivalent reformulation over the schema $\mathbf{P}$ used to define the query Q. We then formalize the rewriting approach, using the notion of containment of queries over the same schema w.r.t. a set of view answers MV and a set of dependencies Σ.

Head-instantiated rewriting $R(\bar{t})$. Intuitively, an attackers' goal in this approach is to form candidate answers, $\bar{t}$, to the secret query by using constants that are in $adom(MV)$ and that thus presumably originate from the actual database I of interest, $\mathcal{V} \Rightarrow_{I,\Sigma} MV$. Observe that not all $\mathcal{V}$-based rewritings could be used toward this goal. Consider, for instance, a rewriting $R_f(f) \leftarrow V(X)$, defined using a constant f and a subgoal $V(X)$ for a view V and variable X. Clearly, regardless of the contents of the set MV of answers to the view V, the answer $R_f(f)(MV)$ to R_f on MV is always the set { (f) }. To rule out rewritings such as $R_f(f)$, we define a desirable type of rewritings as follows.

For an integer $k \geq 0$ and for a k-tuple $\bar{t}$ of constants, consider a safe k-ary CQ rewriting R over schema $\mathcal{V}$ and with head vector $\bar{t}$. We say that R is a *head-instantiated rewriting for* $\bar{t}$ iff there exists a safe k-ary CQ rewriting $R^{(g)}(\bar{X})$, a *grounding rewriting for* R, that satisfies two conditions. First, the head vector $\bar{X}$ of $R^{(g)}$ does not include constants. Second, there exists a mapping, h, that maps all the elements of $\bar{X}$ to constants and that maps the remaining terms in $R^{(g)}$ to themselves, such that the rewriting resulting from applying h to the definition of $R^{(g)}$ is exactly R.

EXAMPLE B.1. *Consider rewritings* R_{vw} *and* $\tilde{R}_{vw}$ *that use constants* c*,* d*, and* f*. (*$\tilde{R}_{vw}$ *also uses a variable* Z*.)*

$$R_{vw}(c, f) \leftarrow V(c, d), W(d, f).$$
$$\tilde{R}_{vw}(c, f) \leftarrow V(c, Z), W(Z, f).$$

Suppose that c*,* d*, and* f *stand for 'johnDoe', 'sales', and '*$50000*', respectively; then* R_{vw} *is an equivalent CQ reformulation of the rewriting* `Rvw` *of Example 1.1. By applying this "translation of constants" to the instance* MV *of Example 1.1, we obtain an instance* $MV' = \{V(c, d), W(d, f)\}$.

Each of R_{vw} *and* $\tilde{R}_{vw}$ *is a head-instantiated rewriting for* (c, f)*, as the respective grounding rewritings are*

$$R^{(g)}_{vw}(X, Y) \leftarrow V(X, d), W(d, Y).$$
$$\tilde{R}^{(g)}_{vw}(X, Y) \leftarrow V(X, Z), W(Z, Y).$$ □

By definition, for each head-instantiated rewriting R for a tuple $\bar{t}$, the answer to R on an instance I of schema $\mathcal{V}$ is nonempty (and is exactly { $\bar{t}$ }) iff there exists a valuation from the body of R onto a subset I' of I such that $adom(I')$ contains all constants in $\bar{t}$. Further, consider an arbitrary safe CQ rewriting R'' and any instance MV such that $R''(MV) \neq \emptyset$. Then for each tuple $\bar{t}$ in $R''(MV)$, the result $R''(\bar{t})$ of binding the head vector of R'' to $\bar{t}$ (while consistently renaming the terms in the

body of R'' as well) is a head-instantiated rewriting for $\bar{t}$, such that the answer to $R''(\bar{t})$ on MV is not empty.

Expansion of a rewriting. We now take a step back, from head-instantiated rewritings to general CQ rewritings, to recall the standard notion of expansion of a CQ rewriting. First, given a set of views $\mathcal{V}$ and a ground instance I of schema $\mathbf{P}$, consider an instance over schema $\mathcal{V} \cup \mathbf{P}$, which results from adding to I the relation $V(I)$ for each relation symbol $V \in \mathcal{V}$. We call the latter instance *the $\mathcal{V}$-enhancement of I,* and denote it by $I^{(+\mathcal{V})}$. Now given a rewriting R over the schema $\mathcal{V}$, consider a query, R', over the schema $\mathbf{P}$ such that for each instance I of $\mathbf{P}$ we have $R'(I) = R(I^{(+\mathcal{V})})$. We call such a query R' *an expansion of R (over $\mathbf{P}$)*, and denote it by R^{exp}. We will use the following straightforward but important property of R^{exp}:

PROPOSITION B.1. *For a set $\mathcal{V}$ of views over schema $\mathbf{P}$: Let R be a rewriting such that R^{exp} exists, and let MV be an instance of schema $\mathcal{V}$. Then for each instance I of schema $\mathbf{P}$ such that $\mathcal{V} \Rightarrow_{I,\emptyset} MV$, we have $R^{exp}(I) = R(MV)$.* □

In case where $\mathcal{V}$ is a set of CQ views and R is a CQ query over $\mathcal{V}$, the standard process in the literature of constructing R^{exp} is to replace each subgoal of R with the body of the query for the corresponding relation symbol in $\mathcal{V}$. In this process, care is taken to perform two operations on each query, of the form $V(\bar{X}) \leftarrow body(V)$, whose body in R^{exp} corresponds to a subgoal of R of the form $V(\bar{Z})$. First, we *bind the arguments of the query for V to the vector $\bar{Z}$*, in two steps, (A) and (B). The step (A) is to extend the homomorphism,[6] h, that maps each element of the head vector $\bar{X}$ of the query for V to the same-position element of $\bar{Z}$, to a homomorphism $h_{V(\bar{Z})}$, whose domain is the set of all arguments of $body(V)$, such that $h_{V(\bar{Z})}$ is the identity mapping for each value that is not in the domain of h. Then, (B) is to apply $h_{V(\bar{Z})}$ to $body(V)$, with conjunction of relational atoms $h_{V(\bar{Z})}(body(V))$ as the output. Second, before conjoining $h_{V(\bar{Z})}(body(V))$ with the current body, $bodyR^{exp}_{curr}$, of the query R^{exp}, we rename all the variables in $h_{V(\bar{Z})}(body(V))$ consistently into "fresh" variables not occuring in $bodyR^{exp}_{curr}$. The query R^{exp} that is obtained by this two-step process is unique up to variable renaming.

Conditional containment: We can now use containment to directly relate a rewriting R, via R^{exp}, to the secret query Q. The notion of containment we will use is relative to the fixed instance MV accessible to the attackers, and is also relative to the set Σ of dependencies known to hold on each instance (of schema $\mathbf{P}$) of interest to the attackers:

DEFINITION B.1. *(**Σ-conditional query containment**) Given schema $\mathbf{P}$, set Σ of dependencies on $\mathbf{P}$, and set MV of view answers for a set of views $\mathcal{V}$. For queries Q_1 and Q_2 over $\mathbf{P}$, we say that Q_1* is Σ-conditionally contained in Q_2 w.r.t. *MV, denoted $Q_1 \sqsubseteq_{\Sigma,MV} Q_2$, iff for each instance, I, of $\mathbf{P}$ such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$, we have $Q_1(I) \subseteq Q_2(I)$.* □

[6]It is easy to show that if such a h cannot be constructed, then R is unsatisfiable on all instances of the schema $\mathcal{V}$.

Definition 5.1 is more general than what is required in the context of the information-leak problem. In fact, in Definition 5.1 we generalize a key definition of [33] due to Zhang and Mendelzon. Specifically, the definition of *conditional query containment* of [33] is our Definition 5.1 in the special case where the set of dependencies Σ is the empty set.

For notational convenience in the results to follow, we now introduce Σ-conditional containment of a rewriting in a query modulo a set of views: For a rewriting R over $\mathcal{V}$ such that R^{exp} exists, and for a query Q over $\mathbf{P}$, we say that *R is Σ-conditionally contained in Q w.r.t. MV and modulo $\mathcal{V}$*, denoted $R \sqsubseteq_{\mathcal{V},\Sigma,MV} Q$, iff $R^{exp} \sqsubseteq_{\Sigma,MV} Q$ holds.

The rewriting approach: We are now ready to specify the rewriting approach to the problem of information-leak disclosure. For an instance MV of schema $\mathcal{V}$, we say that a head-instantiated rewriting $R(\bar{t})$ is *MV-validated* iff the set $R(\bar{t})(MV)$ is not the empty set. (The rewritings R_{vw} and $\tilde{R}_{vw}$ of Example B.1 are both MV'-validated.) Given a valid instance $\mathcal{D}_s$ of the information-leak problem, the *rewriting approach to the problem of information-leak disclosure for $\mathcal{D}_s$* is to find an MV-validated head-instantiated rewriting R for $\bar{t}$ such that $R \sqsubseteq_{\mathcal{V},\Sigma,MV} Q$. This approach is sound:

PROPOSITION B.2. *Given a valid instance $\mathcal{D}_s = (\mathbf{P}, \Sigma, \mathcal{V}, Q, MV)$ of the information-leak problem, with $k \geq 0$ the arity of the query Q. Let $\bar{t}$ be a k-tuple of values from $adom(MV)$. Suppose that there exists an MV-validated head-instantiated rewriting R for $\bar{t}$ such that $R \sqsubseteq_{\mathcal{V},\Sigma,MV} Q$. Then $\bar{t}$ is a potential information leak in $\mathcal{D}_s$.* □

The proof is very simple: Any rewriting R satisfying the conditions of Proposition B.2 must have $\bar{t}$ as its only answer on the given instance MV. Thus, by Proposition B.1, R^{exp} (which exists because the containment $R \sqsubseteq_{\mathcal{V},\Sigma,MV} Q$ is stated in Proposition B.2 to be well defined) has $\bar{t}$ as its only answer on all instances I such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$. From the containment $R^{exp} \sqsubseteq_{\Sigma,MV} Q$ we conclude that on all such instances I, the tuple $\bar{t}$ is an element of the set $Q(I)$. Hence, by Definition 4.2, $\bar{t}$ is a potential information leak in $\mathcal{D}_s$.

B.3 One Rewriting Is Enough

Suppose that we are given an instance $\mathcal{D}_s$ of the information-leak problem, with set of view answers MV. For $k \geq 0$ the arity of the secret query Q in $\mathcal{D}_s$, attackers can generate from $adom(MV)$ all k-tuples $\bar{t}$. Then, Proposition B.2 gives the attackers a tool for testing each such $\bar{t}$ as a potential information-leak tuple for $\mathcal{D}_s$, assuming that the attackers can come up with an "appropriate" rewriting R for each $\bar{t}$, and that there exists an algorithm for checking the containment $R \sqsubseteq_{\mathcal{V},\Sigma,MV} Q$ for each such R and $\bar{t}$. We will consider in the next subsection some such algorithms. However, in this current subsection we show that to solve this generate-and-test problem for a given instance $\mathcal{D}_s$, it is not necessary to also generate various bodies for rewritings R. Each valid $\mathcal{D}_s$ is associated with a single CQ rewriting for each $\bar{t}$, with all these rewritings (for $\mathcal{D}_s$) having the same body. The main result of this subsection is that for all CQ information-leak instances, these rewritings alone can be used to capture *exactly* the set of all potential information leaks.

Intuitively, we are to construct the desired rewritings from the facts in the instance MV. Indeed, by the requirement that MV in each $\mathcal{D}_s$ be a ground instance, each fact in MV can be viewed equivalently as a relational atom whose all arguments are constants. Given a fixed MV and a k-ary ($k \geq 0$) tuple $\bar{t}$ of values from $adom(MV)$, we say that a CQ rewriting R with head vector $\bar{t}$ is an *MV-induced rewriting for $\bar{t}$* iff each subgoal of R is a fact in MV. Further, an MV-induced rewriting R for $\bar{t}$ is a *maximal MV-induced rewriting for $\bar{t}$* iff each fact in MV is also a subgoal of R. In Example B.1, R_{vw} is a maximal MV'-induced rewriting for the tuple (c, f), and $\tilde{R}_{vw}$ is not an MV'-induced rewriting.

We now list useful properties of MV-induced rewritings.

PROPOSITION B.3. *Given a valid instance $\mathcal{D}_s$ of the information-leak problem. For a $k \geq 0$, let $\bar{t}$, $\bar{t}_1$, and $\bar{t}_2$ be k-tuples of values from $adom(MV)$, for the MV in $\mathcal{D}_s$. Then:*

(1) Each MV-induced rewriting R for $\bar{t}$ is an MV-validated head-instantiated rewriting for $\bar{t}$ whenever each element of $\bar{t}$ occurs in the body of R;

(2) For each $\bar{t}$, there is exactly one maximal MV-induced rewriting, which is an MV-validated head-instantiated rewriting for $\bar{t}$; and

(3) The maximal MV-induced rewritings for $\bar{t}_1$ and for $\bar{t}_2$ have the same body, for all choices of $\bar{t}_1$ and $\bar{t}_2$. □

The next result says that when we have the maximal MV-induced rewriting for some tuple $\bar{t}$ of values from $adom(MV)$, then we do not need to consider any other head-instantiated rewritings for $\bar{t}$ in our rewriting approach.

PROPOSITION B.4. *Given a valid CQ instance $\mathcal{D}_s$ of the information-leak problem. For $k \geq 0$ the arity of the query Q in $\mathcal{D}_s$, let $\bar{t}$ be a k-tuple of values from $adom(MV)$. Let R be an MV-validated head-instantiated rewriting for $\bar{t}$ such that $R \sqsubseteq_{\mathcal{V},\Sigma,MV} Q$. Then for the maximal MV-induced rewriting $R^*_{\bar{t}}$ for $\bar{t}$, we have $R^*_{\bar{t}} \sqsubseteq_{\mathcal{V},\Sigma,MV} Q$.* □

The following result says that maximal MV-induced rewritings alone can be used to capture exactly the potential information leaks in the CQ information-leak setting. This result is an immediate corollary of Propositions B.3 and B.4.

THEOREM B.1. *Given a valid CQ instance $\mathcal{D}_s$ of the information-leak problem, with $k \geq 0$ the arity of the query Q. For a k-tuple $\bar{t}$ of values from $adom(MV)$: The tuple $\bar{t}$ is a potential information leak in $\mathcal{D}_s$ iff for the maximal MV-induced rewriting $R^*_{\bar{t}}$ for $\bar{t}$, we have $R^*_{\bar{t}} \sqsubseteq_{\mathcal{V},\Sigma,MV} Q$.* □

PROOF. *If:* The proof of this direction parallels the proof of Proposition B.2.

Only-If: By Definition 4.2, for the given tuple $\bar{t}$ we have that $\bar{t}$ is in the set $Q(I)$ for all instances I of schema $\mathbf{P}$ such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$. By Proposition B.3, we have that $\bar{t}$ is the only answer on the instance MV to the maximal MV-induced rewriting $R^*_{\bar{t}}$ for $\bar{t}$. Thus, for the expansion of $R^*_{\bar{t}}$, denote this expansion by $(R^*_{\bar{t}})^{exp}$, we have by Proposition B.1 that for each instance I of schema $\mathbf{P}$ such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$, we have $(R^*_{\bar{t}})^{exp}(I) = \{(\bar{t})\}$. (In more detail, we have by Proposition B.1 that for each instance J of schema $\mathbf{P}$ such that $\mathcal{V} \Rightarrow_{J,\emptyset} MV$, we have $(R^*_{\bar{t}})^{exp}(J) = \{(\bar{t})\}$. The conclusion that $(R^*_{\bar{t}})^{exp}(I) = \{(\bar{t})\}$ for each instance I of schema $\mathbf{P}$ such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$ follows from the fact that the set of all such instances I is a subset of the set of all such instances J.) Thus, by the definitions of expansions of rewriting and of the containment $\sqsubseteq_{\mathcal{V},\Sigma,MV}$, we obtain immediately that $R^*_{\bar{t}} \sqsubseteq_{\mathcal{V},\Sigma,MV} Q$. □

It follows from Theorem B.1 that the converse of Proposition B.2 also holds. Hence we obtain the following result.

THEOREM B.2. *Given a valid CQ instance $\mathcal{D}_s$ of the information-leak problem, with $k \geq 0$ the arity of Q. For a k-tuple $\bar{t}$ of values from $adom(MV)$: There exists an MV-validated head-instantiated rewriting R for $\bar{t}$ such that $R \sqsubseteq_{\mathcal{V},\Sigma,MV} Q$ iff $\bar{t}$ is a potential information leak in $\mathcal{D}_s$.* □

B.4 Using the Rewriting Approach

It would be nice to be able to use Theorem B.2 to solve the problem of information-leak disclosure in the CQ setting. The missing part is an algorithm for checking the containment $R \sqsubseteq_{\mathcal{V},\Sigma,MV} Q$ for each R and $\bar{t}$. In this subsection, we locate an algorithm and provide a sound and complete solution for those instances $\mathcal{D}_s$ where $\Sigma = \emptyset$, and then suggest solutions for the case $\Sigma \neq \emptyset$. Our rewriting approach fails to provide complete solutions for the case where $\Sigma \neq \emptyset$. To address this issue, we develop in Section 7 an alternative approach for disclosing information leaks. The latter approach is sound and complete for all information-leak instances in the CQ weakly acyclic setting.

B.4.1 The Case Where $\Sigma = \emptyset$

We present here a sound and complete algorithm for solving the problem of information-leak disclosure for all instances $\mathcal{D}_s$ in the CQ setting with $\Sigma = \emptyset$. The algorithm builds on the results of [33]. Zhang and Mendelzon addressed in [33] the problem of letting users access authorized data, via rewriting the users' queries in terms of their authorization views. Toward that goal, [33] explored the notion of "conditional query containment." (We extended this notion to the case of dependencies in our Definition 5.1.) The results of [33] include a powerful reduction of the problem of testing conditional containment of CQ queries to that of testing *unconditional* containment of modifications of the queries. We present this result here in terms of the definitions introduced in Section 4 of this current paper. (Appendix D provides details on this result of [33]. The language $UCQ^{\neq}$ of unions of CQ queries with disequalities, which are mentioned in Theorem B.3, is defined in Appendix C.)

THEOREM B.3. *[33] Given a schema $\mathbf{P}$, a set of CQ views $\mathcal{V}$ on $\mathbf{P}$, an $\emptyset$-valid set MV of view answers for $\mathcal{V}$, and CQ queries Q_1 and Q_2 on the schema $\mathbf{P}$. Then $Q_1 \sqsubseteq_{\emptyset,MV} Q_2$ if and only if for the $UCQ^{\neq}$ query Q_1'' constructed from Q_1 by an algorithm given in [33], we have $Q_1'' \sqsubseteq Q_2$.* □

As the problem of unconditional containment of a $UCQ^{\neq}$ query in a CQ query is decidable [21] (see Appendix C), by Theorem B.3 so is the problem of determining whether $Q_1 \sqsubseteq_{\emptyset,MV} Q_2$ for CQ queries Q_1 and Q_2 w.r.t. a set $\mathcal{V}$ of CQ views and a set of view answers MV. It is shown in [33] that when $\mathbf{P}$ and $\mathcal{V}$ are fixed whereas MV, Q_1, and Q_2 are allowed to vary, the problem of deciding whether $Q_1 \sqsubseteq_{\emptyset,MV} Q_2$ holds is Π_2^p complete.

We can use Theorem B.3 to show that the tuple $\bar{t}$ of Example 1.1 is a potential information leak in the CQ information-disclosure setting of that example. Indeed, for the expansion R_{vw}^{exp} of the rewriting `Rvw` of Example 1.1, we have by Theorem B.3 that $R_{vw}^{exp}(\bar{t}) \sqsubseteq_{\emptyset,MV} Q$. The desired conclusion about the tuple $\bar{t}$ then follows from Proposition B.2.

Theorems B.1 and B.3 suggest immediately a sound and complete algorithm for disclosing information leaks for CQ information-leak instances $\mathcal{D}_s$ with $\Sigma = \emptyset$, as follows.

The rewriting-based algorithm for disclosing information leaks for CQ instances $\mathcal{D}_s$ with $\Sigma = \emptyset$: First, for MV the set of view answers in $\mathcal{D}_s$ and for $k \geq 0$ the arity of the query Q in $\mathcal{D}_s$, generate all k-tuples of values from $adom(MV)$. Then, for each such tuple $\bar{t}$, check whether for the maximal MV-induced rewriting $R_{\bar{t}}^*$ for $\bar{t}$, we have $(R_{\bar{t}}^*)^{exp} \sqsubseteq_{\Sigma,MV} Q$. In case of each positive answer to the latter question, return the respective $\bar{t}$ as a potential information leak in $\mathcal{D}_s$. Assuming that $\mathbf{P}$ and $\mathcal{V}$ are fixed whereas MV and Q are allowed to vary (the same assumption as that made in [33]), we obtain that the total number of such tuples $\bar{t}$ is polynomial in the size of the problem input, and that the containment test for each such $\bar{t}$ is Π_2^p complete.

B.4.2 The Case Where $\Sigma \neq \emptyset$

Given an instance $\mathcal{D}_s$ and tuple $\bar{t}$ as in the setting of Theorem B.2, one could still approach the problem of disclosing information leaks in $\mathcal{D}_s$ using the rewriting-based algorithm for disclosing information leaks for instances $\mathcal{D}_s$ with $\Sigma = \emptyset$, see Section B.4.1. The caveat is that the algorithm is no longer complete in the case $\Sigma \neq \emptyset$.

Indeed, for a CQ instance $\mathcal{D}_s$ with $\Sigma \neq \emptyset$, for all k-tuples $\bar{t}$ of values from $adom(MV)$, and for each (fixed) MV-validated head-instantiated rewriting R for $\bar{t}$, observe that each of (1)–(3) below is a sufficient condition for (*):

(1) $R^{exp}(\bar{t}) \sqsubseteq Q$,

(2) $R^{exp}(\bar{t}) \sqsubseteq_{\Sigma} Q$, and

(3) $R^{exp}(\bar{t}) \sqsubseteq_{\emptyset,MV} Q$.

(*) $R^{exp}(\bar{t}) \sqsubseteq_{\Sigma,MV} Q$.

For each $i \in [1, 3]$, the logical implication $(i) \Rightarrow (*)$ is immediate from the definitions, once we observe that the containment in (*) can be obtained from the containment in (i) by dropping a subset of the instances in question. The containment test for each of (1) and (3) is decidable for CQ instances $\mathcal{D}_s$, and the containment test for (2) is decidable for weakly acyclic such instances, see Section 3.2.

Not surprisingly, one can show by counterexample that the implication $(*) \Rightarrow (i)$ does not hold for any $i \in [1, 3]$. Our example that works for all three values of i can be found in Appendix E.

The converse of the logical implication (3) $\Rightarrow$ (*) in particular does not necessarily hold. Thus, we can use the rewriting-based algorithm for CQ instances $\mathcal{D}_s$ with $\Sigma = \emptyset$ (Section B.4.1) only as a sound, but not complete, approach for solving the problem of disclosing information leaks in CQ instances $\mathcal{D}_s$ with $\Sigma \neq \emptyset$. Further, each of (1) and (2) can be substituted for (3) in the rewriting-based algorithm of Section B.4.1, to obtain a sound – but again not complete – algorithm for solving the problem of disclosing information leaks in CQ instances $\mathcal{D}_s$ with $\Sigma \neq \emptyset$. (In case where (2) is used in the above algorithm, the set Σ must be such (e.g., weakly acyclic) that the test $R^{exp}(\bar{t}) \sqsubseteq_{\Sigma} Q$ is decidable.)

To address the problem of incompleteness of the above three algorithms for solving the problem of disclosing information leaks in CQ instances $\mathcal{D}_s$ with $\Sigma \neq \emptyset$, one could in principle try to extend to the case $\Sigma \neq \emptyset$ the containment test of Theorem B.3 due to [33]. As the test of [33] is highly sophisticated, it is not immediately obvious how to approach this extension problem. Observe however that to solve the problem that we focus on in this current paper, we do not have to solve the problem of containment of queries, as the latter problem is more general than what we need. Recall that proving that a tuple $\bar{t}$ is a potential information leak for an instance $\mathcal{D}_s$ is a matter of showing that $\bar{t}$ is in the answer to the secret query Q on each instance, I, of schema $\mathbf{P}$, such that I "is compatible with" MV and Σ in $\mathcal{D}_s$. Thus, intuitively, we are considering the query Q w.r.t. the instance MV, rather than w.r.t. any query over MV. We develop this intuition fully in Section 7, where we introduce an alternative approach for disclosing information leaks. Our approach of Section 7 is sound and complete for all CQ weakly acyclic information-leak instances. We begin developing the ideas of Section 7 in Section 6, where we show that the idea of data exchange can be used toward solving our problem of disclosing information leaks.

C. $UCQ^{\neq}$ QUERIES

In this appendix we extend the definition of CQ queries given in Section 3.1, to the language of unions of CQ queries with nonequalities. We refer to the queries in the latter language as *$UCQ^{\neq}$ queries.*

In the definitions for $UCQ^{\neq}$ queries, we will be using the following notions of *relational atom* and of *(dis)equality atom.* Let QVAR be an infinite set of values disjoint from CONST ∪ VAR, we call QVAR *the set of (query) variables.* We will denote variables by uppercase letters $X, Y, \ldots$. Then $P(\bar{t})$, with P a k-ary relation symbol and $\bar{t}$ a k-vector of values, is a *relational atom* whenever each value in $\bar{t}$ is an element of CONST ∪ QVAR. Further, a *(dis)equality atom* is a built-in predicate of the form $S\ \theta\ T$, where θ is one of $=$ and $\neq$, and each of S and T is an element of CONST ∪ QVAR.

A *$CQ^{\neq}$-rule* over schema $\mathbf{P}$, with *output relation symbol* $Q \notin \mathbf{P}$, with arity $k \geq 0$ of Q, is an expression of the form

$$Q(\bar{X}) \ \leftarrow \ P_1(\bar{U}_1) \wedge \ldots \wedge P_n(\bar{U}_n) \wedge C.$$

Here, $n \geq 1$, the vector $\bar{X}$ has k elements, for each $i \in [1, n]$, P_i is a relation symbol in $\mathbf{P}$, of arity $k_i > 0$,

$Q(\bar{X})$, $P_1(\bar{U}_1), \ldots, P_n(\bar{U}_n)$ are relational atoms, and C is a (possibly empty) conjunction of (dis)equality atoms. We consider only *safe* rules: That is, each variable in $\bar{X}$, as well as each variable occurring in C, also occurs in at least one of $\bar{U}_1, \ldots, \bar{U}_n$. All the variables of the rule that do not appear in $\bar{X}$ (i.e., the *nonhead variables* of the rule) are assumed to be existentially quantified. We call the atom $Q(\bar{X})$ the *head* of the rule, call $\bar{X}$ the *head vector* of the rule, and call the conjunction of its remaining atoms the *body* of the rule. Each atom in the body of the rule is called a *subgoal* of the rule. The conjunction in the body of a rule is usually written using commas, as $P_1(\bar{U}_1), \ldots, P_n(\bar{U}_n), C$.

A *conjunctive query with disequalities (a $CQ^{\neq}$ query)* is a query defined by a single $CQ^{\neq}$-rule. Further, a *conjunctive query (a CQ query)* is a $CQ^{\neq}$ query with the empty set C of (dis)equalities in the body of its rule. Consider a $CQ^{\neq}$ query defined by a $CQ^{\neq}$-rule with a k-ary ($k \geq 0$) output relation symbol Q and with head vector $\bar{X}$. Wherever clear from the context, we will be referring to the query as simply $Q(\bar{X})$, or even just as Q. We will be using $head_{(Q)}$ and $body_{(Q)}$ as concise names for the head and for the body of the (rule for) Q.

Finally, for a k-ary relation symbol Q, with $k \geq 0$, let $\mathcal{S}(Q) = \{ Q^{(1)}, \ldots, Q^{(l)} \}$ be a finite nonempty set of $CQ^{\neq}$-rules over schema $\mathbf{P}$, such that Q is the output relation symbol in each rule. Then we say that the set $\mathcal{S}(Q)$ *defines a $UCQ^{\neq}$ query Q over* $\mathbf{P}$.

Semantics of $UCQ^{\neq}$ queries. We now define the semantics of a $UCQ^{\neq}$ query Q. In the definition, we will need the notions of *homomorphism* and of *valuation.* Consider two conjunctions, $\varphi(\bar{Y})$ and $\psi(\bar{Z})$, of relational atoms. Then a mapping h from the set of elements of $\bar{Y}$ to the set of elements of $\bar{Z}$ is called a *homomorphism* from $\varphi(\bar{Y})$ to $\psi(\bar{Z})$ whenever (i) $h(c) = c$ for each constant c in $\bar{Y}$, and (ii) for each conjunct of the form $p(\bar{U})$ in $\varphi(\bar{Y})$, the relational atom $p(h(\bar{U}))$ is a conjunct in $\psi(\bar{Z})$. (For a vector $\bar{S} = [s_1 s_2 \ldots s_l]$, for some $l \geq 0$, we define $h(\bar{S})$ as the vector $[h(s_1)h(s_2)\ldots h(s_l)]$.) We will denote homomorphisms by lowercase letters g, h, ... , possibly with subscripts.

We define homomorphisms in the same way for the case where either one of $\varphi(\bar{Y})$ and $\psi(\bar{Z})$ (or both) is a conjunction of *facts.* Further, for a conjunction C of (dis)equality atoms and two conjunctions $\varphi(\bar{Y})$ and $\psi(\bar{Z})$ of relational atoms or of facts, we say that each homomorphism, h, from $\varphi(\bar{Y})$ to $\psi(\bar{Z})$ is also a homomorphism from $\varphi(\bar{Y})$ to $\psi(\bar{Z}) \wedge C$.

Finally, suppose we are given a conjunction $\varphi(\bar{Y})$ of relational atoms, a conjunction $\psi(\bar{Z})$ of facts, and a conjunction C of (dis)equalities over variables in $\bar{Y}$ and over constants in CONST. Suppose that there is a homomorphism, h, from $\varphi(\bar{Y})$ to $\psi(\bar{Z})$, such that for each (dis)equality of the form $S\ \theta\ T$ in C, the values $h(S)$ and $h(T)$ are distinct elements of CONST $\cup$ VAR whenever θ is $\neq$, and are the same element of CONST $\cup$ VAR whenever θ is $=$. Then we say that h is a *valuation* from $\varphi(\bar{Y}) \wedge C$ to $\psi(\bar{Z})$. We will use Greek letters μ, ν, ... , possibly with subscripts, for valuations.

Now given a k-ary $CQ^{\neq}$ query $Q(\bar{X})$ and given an instance I, which we interpret as a conjunction of all the facts in I. Then *the answer to Q on I*, denoted $Q(I)$, is

$$Q(I) = \{\ \nu(\bar{X}) \mid \nu \text{ is a valuation from } body_{(Q)} \text{ to } I\ \}\ .$$

(When $\bar{X}$ is the empty vector, i.e., Q is a *Boolean query,* $\nu(\bar{X})$ is the empty tuple.) Further, for a $UCQ^{\neq}$ query Q defined by $l \geq 1$ $CQ^{\neq}$ rules $\{ Q^{(1)}, \ldots, Q^{(l)} \}$, and for an instance I, *the answer to Q on I* is the union $\cup_{i=1}^{l} Q^{(i)}(I)$.

Query containment. A query Q_1 *is contained in query Q_2,* denoted $Q_1 \sqsubseteq Q_2$, if $Q_1(I) \subseteq Q_2(I)$ for every instance I. A classic result [11] by Chandra and Merlin states that a necessary and sufficient condition for the containment $Q_1 \sqsubseteq Q_2$, for CQ queries Q_1 and Q_2 of the same arity, is the existence of a containment mapping from Q_2 to Q_1. Here, a *containment mapping* [11] from CQ query $Q_2(\bar{X}_2)$ to CQ query $Q_1(\bar{X}_1)$ is a homomorphism h from $body_{(Q_2)}$ to $body_{(Q_1)}$ such that $h(\bar{X}_2) = \bar{X}_1$. By the results in [21], this containment test of [11] remains true when Q_1 has built-in predicates. Thus, the same test holds in particular when Q_1 is a $CQ^{\neq}$ query. It follows that, for Q_1 a $UCQ^{\neq}$ query and for Q_2 a CQ query, determining whether $Q_1 \sqsubseteq Q_2$ is decidable. Indeed, this containment holds iff for each $CQ^{\neq}$ query $Q_1^{(i)}$ in the definition of Q_1, for all $i \in [1, l]$, we have $Q_1^{(i)} \sqsubseteq Q_2$.

D. CONDITIONAL CONTAINMENT FOR CQ QUERIES

Zhang and Mendelzon in [33] addressed the problem of letting users access authorized data, via rewriting the users' queries in terms of their authorization views. Toward that goal, [33] explored the notion of "conditional query containment." The results of [33] include a powerful reduction of the problem of testing conditional containment of CQ queries to that of testing *unconditional* containment of modifications of the queries. In this appendix we review these results of [33].

We begin by reviewing the definition of conditional containment of queries [33]. Some of the definitions here are restricted versions of the definitions given in Section 4. We provide the restricted definitions here for this appendix to be self contained.

Suppose that we are given a schema $\mathbf{P}$ and a set $\mathcal{V}$ of relation symbols not in $\mathbf{P}$, with each symbol *(view name)* $V \in \mathcal{V}$ of some arity $k_V \geq 0$. Each symbol $V \in \mathcal{V}$ is defined via a k_V-ary query on the schema $\mathbf{P}$. We call $\mathcal{V}$ a *set of views on* $\mathbf{P}$, and call the query for each $V \in \mathcal{V}$ the *definition of the view* V, or *the query for* V. We assume that the query for each $V \in \mathcal{V}$ is associated with (V in) the set $\mathcal{V}$. Consider a ground instance MV of schema $\mathcal{V}$, we call MV a *set of view answers for* $\mathcal{V}$. Then for a ground instance I of schema $\mathbf{P}$, we say that I *is a valid instance (of $\mathbf{P}$) for $\mathcal{V}$ and MV* [33] whenever for each $V \in \mathcal{V}$, the answer $V(I)$ to the query for V on the instance I is identical to the relation $MV[V]$ for V in the instance MV. For a given set MV of view answers for a set of views $\mathcal{V}$, we say that MV *is a valid set of view answers for* $\mathcal{V}$ whenever there exists at least one valid instance for $\mathcal{V}$ and MV.

Now given queries Q_1 and Q_2 on the schema $\mathbf{P}$, we say that Q_1 is *conditionally contained in Q_2 w.r.t. ($\mathcal{V}$*

and) MV [33], denoted[7] $Q_1 \sqsubseteq_{MV} Q_2$, if the relation $Q_1(I)$ is a subset of the relation $Q_2(I)$ for each valid instance I for $\mathcal{V}$ and MV.

It is easy to see that for all instances MV of all schemas $\mathcal{V}$, the containment $Q_1 \sqsubseteq Q_2$ is a sufficient condition for the containment $Q_1 \sqsubseteq_{MV} Q_2$. Not surprisingly, $Q_1 \sqsubseteq_{MV} Q_2$ does not imply $Q_1 \sqsubseteq Q_2$, something more sophisticated is clearly called for. The authors of [33] report the following powerful test for conditional containment of CQ queries. (We say that $\mathcal{V}$ is a set of CQ views if the query for each element of $\mathcal{V}$ is a CQ query. The language of $UCQ^{\neq}$ queries, which are mentioned in Theorem D.1, is defined in Appendix C.)

THEOREM D.1. *[33] Given a schema* $\mathbf{P}$*, a set of CQ views* $\mathcal{V}$ *on* $\mathbf{P}$*, a valid set* MV *of view answers for* $\mathcal{V}$*, and CQ queries* Q_1 *and* Q_2 *on the schema* $\mathbf{P}$*. Then* $Q_1 \sqsubseteq_{MV} Q_2$ *if and only if for the* $UCQ^{\neq}$ *query* Q_1'' *constructed for* Q_1 *by an algorithm given in [33], we have* $Q_1'' \sqsubseteq Q_2$*.* □

Theorem D.1 reduces the problem of testing conditional containment of CQ queries, $Q_1 \sqsubseteq_{MV} Q_2$, to the problem of testing (unconditional) containment in Q_2 of a $UCQ^{\neq}$ modification of Q_1. The latter containment can be decided by a test due to [21] (see Appendix C here). The required modification of Q_1 is done by an intricate algorithm given in [33]. We outline here briefly the intuition for the construction of Q_1'' from Q_1.

We say that an instance I of schema $\mathbf{P}$ *underproduces* MV if, for at least one $V \in \mathcal{V}$, the relation $V(I)$ is a proper subset of the relation $MV[V]$. By definition, each valid instance for $\mathcal{V}$ and MV does not underproduce MV.

The construction of Q_1'' from Q_1 proceeds in two steps. The first step guarantees that its output, a CQ query Q_1^*, has the empty answer on all instances of $\mathbf{P}$ that underproduce MV. The output of the second step, a $UCQ^{\neq}$ query Q_1'', has the same property as Q_1^* does. In addition, for each instance, I, of schema $\mathbf{P}$ such that I does *not* underproduce MV, and for each valuation, ν, from the query Q_1'' to I, all the facts in $\nu(body_{(Q_1'')})$ collectively constitute a valid instance for $\mathcal{V}$ and MV. The result of Theorem D.1 is shown in [33] to follow from these properties and from the fact that for all valid instances I for $\mathcal{V}$ and MV, $Q_1''(I) = Q_1(I)$.

E. NONCONTAINMENT EXAMPLE, $\Sigma \neq \emptyset$

In this appendix we show by example that when $R^{exp}(\bar{t}) \sqsubseteq_{\Sigma,MV} Q$ holds for some choice of R, $\bar{t}$, Σ, MV, and Q, then none of the following necessarily holds:

(1) $R^{exp}(\bar{t}) \sqsubseteq Q$,

(2) $R^{exp}(\bar{t}) \sqsubseteq_{\Sigma} Q$, and

(3) $R^{exp}(\bar{t}) \sqsubseteq_{\emptyset,MV} Q$.

EXAMPLE E.1. *Recall the relation* `Emp(Name, Dept, Salary)` *of Example 1.1. Suppose that the schema* $\mathbf{P}$ *that contains* `Emp` *also includes relations* `HQDept(Dept)` *and* `OfficeInHQ(Name, Address)`*. The intent of* `HQDept` *is to store the names of the departments that are located in the headquarters of the company,* `OfficeInHQ` *associates employees working in the headquarters with their office addresses.*

As before, we assume that the only primary key of the `Emp` *relation is its three attributes together. The key of the relation* `OfficeInHQ` *is* `Name`*, which we express using egd* τ*:*

$$\tau : OfficeInHQ(X,Y) \wedge OfficeInHQ(X,Z) \rightarrow Y = Z \ .$$

Suppose that for all the departments located in the company headquarters, all their employees have their offices in the headquarters. We express this constraint using a tgd σ*:*

$$\sigma : Emp(X,Y,Z) \wedge HQDept(Y) \rightarrow \exists S \ OfficeInHQ(X,S) \ .$$

We assume that σ *and* τ *constitute all the integrity constraints* Σ *on the schema* $\mathbf{P}$*, that is,* $\Sigma = \{ \sigma, \tau \}$*.*

Recall the views `V` *and* `W` *introduced in Example 1.1:*

```
(V): DEFINE VIEW V(Name, Dept) AS
     SELECT DISTINCT Name, Dept FROM Emp,

(W): DEFINE VIEW W(Dept, Salary) AS
     SELECT DISTINCT Dept, Salary FROM Emp,
```

Let `U` *be another view available to some class(es) of the database users.* `U` *returns the names of all the departments that are located in the company headquarters:*

```
(U): DEFINE VIEW U(Dept) AS SELECT * FROM HQDept,
```

Let the secret query `Q'` *return the names and salaries of all the employees who work in the company headquarters. (Note that* `Q'` *is different from the secret query* `Q` *of Example 1.1.)*

```
(Q'): SELECT DISTINCT E.Name, Salary FROM Emp E, OfficeInHQ
      WHERE E.Name = OfficeInHQ.Name,
```

Suppose that a user, or several users together, are authorized to see the answers to all three views `U`*,* `V`*, and* `W`*, and that at some point in time the user(s) can see the following set* MV' *of answers to these views.*

$MV' = \{$ `U(Sales)`, `V(JohnDoe, Sales)`, `W(Sales, $50000)` $\}$.

Recall the tuple $\bar{t}$ = (`JohnDoe`, `$50000`) *and the rewriting* R_{vw} *of Example 1.1:*

```
(Rvw): SELECT DISTINCT Name, Salary FROM V, W
       WHERE V.Name = 'JohnDoe' AND V.Dept = W.Dept
       AND V.Dept = 'Sales' AND Salary = '$50000',
```

We can show that the expansion R_{vw}^{exp} *of the rewriting* R_{vw} *is contained in the secret query* `Q'` *in presence of the set of dependencies* Σ *and of the database* MV'*. (Thus,* `Rvw` *is contained in* `Q'` *modulo the set of views* $\mathcal{V} = \{$ `U`, `V`, `W` $\}$ *and in presence of* Σ *and of* MV'*, i.e.,* $R_{vw} \sqsubseteq_{\mathcal{V},\Sigma,MV'} Q'$*.) At the same time, none of the following containments hold:* $R_{vw}^{exp} \sqsubseteq Q'$*,* $R_{vw}^{exp} \sqsubseteq_{\Sigma} Q'$*, and* $R_{vw}^{exp} \sqsubseteq_{MV'} Q'$*. We conclude that the information-leak tuple* $\bar{t}$*, fixed as above, is detectable by the containment* $R_{vw} \sqsubseteq_{\mathcal{V},\Sigma,MV'} Q'$*, but not by the containments* $R_{vw} \sqsubseteq_{\mathcal{V}} Q'$*,* $R_{vw} \sqsubseteq_{\mathcal{V},\Sigma} Q'$*, or* $R_{vw} \sqsubseteq_{\mathcal{V},MV'} Q'$*.*

We now prove all the containment and non-containment statements of the preceding paragraph, for the queries

[7] To avoid overcrowding the symbol $\sqsubseteq$, we assume that in the notation $\sqsubseteq_{MV}$, the name MV of an instance of schema $\mathcal{V}$ uniquely identifies the relevant set $\mathcal{V}$.

R^{exp}_{vw} *and* Q' *over the schema* **P**. *First, we render in Datalog the queries* Rvw, R^{exp}_{vw}, Q', *and the queries for the three views. (For conciseness, in the remainder of this example we will refer to the constants* JohnDoe, Sales, *and* $50000 *as c, d, and f, respectively, and will refer to each relation name in* **P** *by the first letter of the name.)*

$$\begin{array}{ll} R_{vw}(c,f) & \leftarrow V(c,d), W(d,f). \\ R^{exp}_{vw}(c,f) & \leftarrow E(c,d,Z), E(X,d,f). \\ Q'(X,Z) & \leftarrow E(X,Y,Z), O(X,S). \\ U(X) & \leftarrow H(X). \\ V(X,Y) & \leftarrow E(X,Y,Z). \\ W(Y,Z) & \leftarrow E(X,Y,Z). \end{array}$$

(1) The noncontainment of $R^{exp}_{vw}(c,f)$ *in* Q' *is immediate from the containment test of [11] and from the absence in the definition of* $R^{exp}_{vw}(c,f)$ *of a subgoal with predicate* OfficeInHQ. *(As a result, the subgoal of* Q' *with predicate* OfficeInHQ *cannot be mapped into the body of* $R^{exp}_{vw}(c,f)$.*) We conclude that* $R_{vw} \sqsubseteq_{\mathcal{V}} Q'$ *does not hold.*

(2) We observe that the result of chasing the query $R^{exp}_{vw}(c,f)$ *with the dependencies* Σ *is identical to* $R^{exp}_{vw}(c,f)$. *Recall that* $R^{exp}_{vw}(c,f) \sqsubseteq_{\Sigma} Q'$ *holds iff that chase result (which is identical to* $R^{exp}_{vw}(c,f)$*) is contained in* Q' *in the absence of dependencies. We then use the reasoning of item (1) to conclude that* $R^{exp}_{vw} \sqsubseteq_{\Sigma} Q'$ *does not hold. Thus,* $R_{vw} \sqsubseteq_{\mathcal{V},\Sigma} Q'$ *does not hold either.*

(3) Consider the following instance I of schema **P***:*

$$I = \{H(d), E(c,d,f)\}.$$

It is easy to verify that for the set $\mathcal{V} = \{$ U, V, W $\}$, *the result of applying the queries for* $\mathcal{V}$ *to I is exactly the database MV′ as given above. We check that* $Q'(I) = \emptyset$ *and that* $R^{exp}_{vw}(c,f)(I) = \{ (c,f) \}$. *As a result, the containment* $R^{exp}_{vw} \sqsubseteq_{MV'} Q'$ *does not hold. We conclude that* $R_{vw} \sqsubseteq_{\mathcal{V},MV'} Q'$ *does not hold either.*

(4) Finally, consider the result of chasing the instance I of (3) with the dependencies Σ*:*

$$I^{(\Sigma)} = \{H(d), E(c,d,f), O(c,j)\}.$$

(Here, j is some constant value.)

It is easy to argue that for each instance J of schema **P** *that satisfies* Σ *and that gives rise to MV′ (when the queries for* $\mathcal{V}$ *are applied to J), J must have the relations for* Emp *and for* HQDept *exactly as in I. Further, the relation for* OfficeInHQ *in any such instance J must have exactly one ground fact with c as the first argument, and any nonnegative number of ground facts with other values (than c) of the first argument. Finally, in any such instance J, each pair of distinct ground facts in the relation for* OfficeInHQ *must differ on the value of their first argument.*

It follows that on any instance J of schema **P** *that satisfies* Σ *and that gives rise to MV′, the relation* $Q'(J)$ *is* $Q'(J) = \{ (c,f) \}$, *and the relation* $R^{exp}_{vw}(c,f)(J)$ *is also* $\{ (c,f) \}$. *As a result, the containment* $R^{exp}_{vw} \sqsubseteq_{\Sigma,MV'} Q'$ *does hold. We conclude that* $R_{vw} \sqsubseteq_{\mathcal{V},\Sigma,MV'} Q'$ *holds as well.* □

F. THE DATA-EXCHANGE APPROACH

In this appendix we discuss all the technical details of the data-exchange approach of Section 6 to disclosure of information leaks.

F.1 A Sound Data-Exchange Approach

Suppose that potential attackers are given a valid CQ instance $\mathcal{D}_s = (\mathbf{P}, \Sigma, \mathcal{V}, Q, MV)$ of the information-leak problem. By Definition 4.2, the attackers are interested in finding tuples $\bar{t}$ of elements of $adom(MV)$, such that for all instances I satisfying $\mathcal{V} \Rightarrow_{I,\Sigma} MV$, we have $\bar{t} \in Q(I)$.

We now show how a straightforward reformulation of $\mathcal{D}_s$ turns the above problem into an instance of the problem of computing certain answers in data exchange. We first construct a set Σ_{st} of tgds, as follows. For a view V in the set of views $\mathcal{V}$ in $\mathcal{D}_s$, consider the query $V(\bar{X}) \leftarrow body_{(V)}(\bar{X}, \bar{Y})$ for V. (As $\mathcal{D}_s$ is a CQ instance, the query for each V in $\mathcal{V}$ is a CQ query.) We associate with this $V \in \mathcal{V}$ the tgd $\sigma_V : V(\bar{X}) \rightarrow \exists \bar{Y}\ body_{(V)}(\bar{X}, \bar{Y})$. We then define the set Σ_{st} to be the set of tgds σ_V for all $V \in \mathcal{V}$. Then $\mathcal{D}_s$ can be reformulated into a data-exchange setting

$$\mathcal{M}^{(de)}(\mathcal{D}_s) = (\ \mathcal{V},\ \mathbf{P},\ \Sigma_{st} \cup \Sigma\ ),$$

with a source instance MV and a query Q on the target schema **P**. We call the triple $(\ \mathcal{M}^{(de)}(\mathcal{D}_s),\ MV,\ Q\ )$ *the associated data-exchange instance for* $\mathcal{D}_s$.

The following observation is immediate from Definition 4.2 and from the definitions in Section 6.1.

PROPOSITION F.1. *Given a valid CQ instance* $\mathcal{D}_s$ *of the information-leak problem, with its associated data-exchange instance* $(\ \mathcal{M}^{(de)}(\mathcal{D}_s),\ MV,\ Q\ )$. *Then for each tuple* $\bar{t}$ *that is a certain answer of Q with respect to MV under the data-exchange setting* $\mathcal{M}^{(de)}(\mathcal{D}_s)$, *we have that* $\bar{t}$ *is a potential information leak in the instance* $\mathcal{D}_s$. □

For valid CQ weakly acyclic instances $\mathcal{D}_s$, we introduce the following algorithm, which we call the *data-exchange approach to disclosing information leaks*. First, we compute the canonical universal solution, $J^{\mathcal{D}_s}_{de}$, for the source instance MV in the data-exchange setting $\mathcal{M}^{(de)}(\mathcal{D}_s)$. If $J^{\mathcal{D}_s}_{de}$ does not exist, then we output the empty set of answers. Otherwise we output, as a set of potential information leaks in $\mathcal{D}_s$, the set of all those tuples in $Q(J^{(de)}(\mathcal{D}_s))$ that do not contain nulls. When we assume, same as in Section 5, that everything in $\mathcal{D}_s$ is fixed except for MV and Q, then from Theorem 6.2 due to [18] we obtain immediately that this algorithm always terminates and runs in polynomial time. By Proposition F.1, this data-exchange approach is sound.

F.2 The Data-Exchange Approach Is Not Complete

By Theorem B.2 and Proposition F.1, we have that for CQ weakly acyclic instances $\mathcal{D}_s$, the potential information leaks that can be disclosed using the data-exchange approach can in principle also be disclosed by our rewriting approach:

THEOREM F.1. *Given a valid CQ weakly acyclic instance* $\mathcal{D}_s$ *of the information-leak problem. Let* $\mathcal{T}$ *be the set of tuples output by the data-exchange approach applied to* $\mathcal{D}_s$. *Then for each* $\bar{t} \in \mathcal{T}$, *there exists an MV-validated head-instantiated rewriting R for* $\bar{t}$ *such that* $R \sqsubseteq_{\mathcal{V},\Sigma,MV} Q$. □

Even in the light of the result of Theorem F.1, we cannot just abandon the data-exchange approach in favor of the rewriting approach when working with valid CQ weakly acyclic instances $\mathcal{D}_s$ of information leak. It is true that we have a sound and complete rewriting approach for disclosing information leaks for CQ instances $\mathcal{D}_s$ with $\Sigma = \emptyset$, please see Section B.4.1. However, the lack of a decidable containment test for the containment $\sqsubseteq_{\Sigma,MV}$ prevents us from extending the algorithm of Section B.4.1 into a sound and complete rewriting approach for those CQ instances $\mathcal{D}_s$ where $\Sigma \neq \emptyset$, please see Section B.4.2. Thus, in principle, we could use the soundness (by Proposition F.1) of the data-exchange approach to try to disclose information-leak tuples for those CQ weakly acyclic instances $\mathcal{D}_s$ for which we do not have a sound and complete rewriting-based algorithm.

It turns out that, as we will see in this subsection, the data-exchange approach is not complete for CQ weakly acyclic instances $\mathcal{D}_s$ with $\Sigma = \emptyset$, as well as for those with $\Sigma \neq \emptyset$. In the remainder of this section, we discuss a feature of the data-exchange approach that prevents us from using it as a complete algorithm for the problem of disclosing information leaks. In Section 7 we will eliminate this feature of the data-exchange approach, in a modification that will give us a sound and complete algorithm for disclosing information leaks for all CQ weakly acyclic instances $\mathcal{D}_s$.

We now prove that the data-exchange approach is not complete for CQ instances $\mathcal{D}_s$ with $\Sigma = \emptyset$.

EXAMPLE F.1. *Recall Example 1.1; the Datalog versions of its two views and of its secret query Q are as follows:*

$V(X,Y) \leftarrow E(X,Y,Z).$
$W(Y,Z) \leftarrow E(X,Y,Z).$
$Q(X,Z) \leftarrow E(X,Y,Z).$

(We abbreviate the relation name `Emp` *to E.)*

Using the agreement as in Example B.1 for the constants used in Example 1.1, we represent the set of view answers of Example 1.1 as $MV' = \{ V(c,d), W(d,f) \}$. In the same notation, the tuple $\bar{t}$ of Example 1.1 is recast as (c,f).

Consider the information-leak instance $\mathcal{D}_s = (\{E\}, \emptyset, \{V,W\}, Q, MV')$, with all the elements as defined above. By definition, $\mathcal{D}_s$ is a CQ weakly acyclic instance. ($\mathcal{D}_s$ is also valid, by the existence of the instance $\{(c,d,f)\}$ of schema $\{E\}$.) The data-exchange approach applied to $\mathcal{D}_s$ yields the following canonical universal solution, $J_{de}^{\mathcal{D}_s}$, for the source instance MV' in the data-exchange setting $\mathcal{M}^{(de)}(\mathcal{D}_s)$:

$$J_{de}^{\mathcal{D}_s} = \{ E(c,d,\perp_1), E(\perp_2,d,f) \} .$$

(The first tuple in $J_{de}^{\mathcal{D}_s}$ is due to the tuple $V(c,d)$ in MV', and the second tuple is due to $W(d,f)$ in MV'.) It is easy to see that each of the two answers to the secret query Q on the instance $J_{de}^{\mathcal{D}_s}$ has nulls and thus cannot qualify as a potential information leak in $\mathcal{D}_s$. □

When given as input the instance $\mathcal{D}_s$ of Example F.1, the data-exchange approach outputs the empty set of potential information leaks. As that instance $\mathcal{D}_s$ is a CQ instance with $\Sigma = \emptyset$, the sound and complete rewriting-based algorithm of Section B.4.1 for disclosure of information leaks is applicable to $\mathcal{D}_s$, and outputs $\{ (c,f) \}$ when given $\mathcal{D}_s$ as input. We conclude that the data-exchange approach is incomplete when applied to CQ instances of information leak with $\Sigma = \emptyset$. Further, by applying similar reasoning to the example of Appendix E, we obtain that the data-exchange approach is also incomplete when applied to CQ weakly acyclic instances of information leak with $\Sigma \neq \emptyset$.

Why is the data-exchange approach not complete when applied to CQ weakly acyclic instances of information leak? Intuitively, the problem is that the canonical universal solution $J_{de}^{\mathcal{D}_s}$ in the data-exchange approach "covers too many target instances." Let us evaluate the queries for the views V and W of Example F.1 over the solution $J_{de}^{\mathcal{D}_s}$ of that example. We obtain that the answer to the view V on the instance $J_{de}^{\mathcal{D}_s}$ is $\{ V(c,d), V(\perp_2,d) \}$. Similarly, the answer to W on $J_{de}^{\mathcal{D}_s}$ is $\{ W(d,\perp_1), W(d,f) \}$. Thus, if we replace $\perp_1$ in $J_{de}^{\mathcal{D}_s}$ by any constant except f, or replace $\perp_2$ by any constant except c, then any ground instance obtained from $J_{de}^{\mathcal{D}_s}$ using these replacements would "generate too many tuples" (as compared with MV') either in the answer to V or in the answer to W.

We now generalize over this observation. Fix a valid CQ weakly acyclic instance $\mathcal{D}_s$, and consider the canonical universal solution $J_{de}^{\mathcal{D}_s}$ generated by the data-exchange approach when given $\mathcal{D}_s$ as input. (In the remainder of this paper, we will refer to $J_{de}^{\mathcal{D}_s}$ as *the canonical data-exchange solution for $\mathcal{D}_s$*.) By definition of $J_{de}^{\mathcal{D}_s}$, for each view $V \in \mathcal{V}$, the answer to the query for V on $J_{de}^{\mathcal{D}_s}$ is always a superset of the relation $MV[V]$. Suppose that the answer on $J_{de}^{\mathcal{D}_s}$ to at least one view $V \in \mathcal{V}$ is not a subset of $MV[V]$, as is the case in the example that we have just discussed. Then $J_{de}^{\mathcal{D}_s}$, as a template for instances of schema $\mathbf{P}$, describes not only instances that "generate" exactly the set MV in $\mathcal{D}_s$, but also those instances that generate proper supersets of MV. The latter instances are not of interest to the potential attackers. (Recall that attackers are interested only in the instances I of schema $\mathbf{P}$ such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$.) As a result, when the data-exchange approach uses $J_{de}^{\mathcal{D}_s}$ to obtain certain answers to the secret query Q, it can easily miss those certain answers that characterize only those instances that are of interest to the attackers.

G. THE NUMBER OF LEAVES IN MV-ENHANCED CHASE CAN BE EXPONENTIAL IN THE SIZE OF THE INPUT

In this appendix we show by example a family of CQ instances with $\Sigma = \emptyset$, such that the number of leaves in a chase tree for each instance in the family is exponential in the size of the instance. As usual and similarly to [33], we assume that the *size* of a given instance $\mathcal{D}_s$ is the size of its instance MV and query Q, with the remaining elements of $\mathcal{D}_s$ being fixed.

EXAMPLE G.1. *Consider schema $\mathbf{P}$ with two binary relations P and R, and with $\Sigma = \emptyset$. Let the set of views $\mathcal{V} = \{V,W\}$ be associated with two CQ queries,*

as follows:

$V(X) \leftarrow P(X,Y), R(Y,Z).$
$W(Z) \leftarrow R(Y,Z).$

For a $n \geq 1$, consider a set $MV_{(n)}$ of answers for $\mathcal{V}$, with $n+2$ tuples. The relation $MV_{(n)}[V]$ has n tuples $V(1)$, $V(2)$, ..., $V(n)$, and $MV_{(n)}[W]$ has tuples $W(0)$ and $W(1)$.

Let the secret CQ query Q be

$Q(X,Z) \leftarrow P(X,Y), R(Y,Z).$

Let the instance $\mathcal{D}_s^{(n)}$ be the tuple ($\mathbf{P}$, Σ, $\mathcal{V}$, Q, $MV_{(n)}$), with components as described above.

The canonical universal solution $J_{de}^{\mathcal{D}_s^{(n)}}$ for $MV_{(n)}$ has two tuples $P(i, \perp_{(i,1)})$ and $R(\perp_{(i,1)}, \perp_{(i,2)})$ for $V(i)$ in $MV_{(n)}$, for each $i \in [1, n]$. It also has the tuples $R(\perp_{(n+1,1)}, 0)$ and $R(\perp_{(n+2,1)}, 1)$ for $MV_{(n)}[W]$.

The process of creating view-verified universal solutions for $\mathcal{D}_s^{(n)}$ involves assigning either 0 or 1 independently to each of the nulls $\perp_{(i,2)}$, for all $i \in [1, n]$. It is easy to see that this process creates 2^n nonisomorphic databases, one for each assignment of zeroes and ones to each element of the vector $[\perp_{(1,2)}, \perp_{(2,2)}, \ldots, \perp_{(n,2)}]$. The expression 2^n is exponential in the size of the set $MV_{(n)}$ of view answers in $\mathcal{D}_s^{(n)}$. □

H. THE PROBLEM OF INFORMATION-LEAK DISCLOSURE IS Π_2^P HARD FOR CONJUNCTIVE INPUTS WITH $\Sigma = \emptyset$

In this appendix we prove Theorem 7.2, which states that the problem of information-leak disclosure with CQ inputs is Π_2^p hard, even when the set Σ of dependencies in the input instance $\mathcal{D}_s$ is the empty set.

PROOF. (Theorem 7.2) In this proof, we build on the constructions from the proof of Theorem 3.3 in [25]. The reason that we modify the reduction of [25] is that we need to comply with our assumptions about the size of the instances $\mathcal{D}_s$, specifically with the assumption that the input view definitions are fixed. (In [25] it is assumed that both the queries and the view definitions can vary.) Thus, our variation on the reduction of [25] is similar in spirit to the modification suggested in [33].

Similarly to the reduction in [25], we reduce the $\forall\exists$-CNF problem, known to be Π_2^p complete [30], to our problem. The $\forall\exists$-CNF problem is defined as follows: Given a 3-CNF propositional formula F with variables $\bar{X}$ and $\bar{Y}$, is it the case that for each truth assignment to $\bar{Y}$, there exists a truth assignment to $\bar{X}$ that satisfies F?

We are given a 3-CNF formula F, with variables

$\bar{Z} = \{X_1, \ldots, X_n\} \cup \{Y_1, \ldots, Y_m\}$

The formula F has clauses $\bar{C} = \{C_1, \ldots, C_l\}$. Clause C_i contains the three variables (either positive or negated) $Z_{i,1}$, $Z_{i,2}$, and $Z_{i,3}$.

For ease of exposition, in the main body of this proof we will assume that in the instance of the $\forall\exists$-CNF problem under consideration, we have $m \geq 1$. (In the last paragraph of the proof, we outline how to modify the main body of the proof so that it would also go through for each input instance with $m = 0$.)

For the input formula F, with $m \geq 1$, we begin building the corresponding instance $\mathcal{D}_s(F)$ of information leak. The schema $\mathbf{P} = \{P, R, S\}$ in $\mathcal{D}_s(F)$ uses three relation symbols: R of arity $k_R = 4$, and two binary relation symbols P and S. Intuitively, for each $i \in [1, l]$ and for the clause C_i in F, in each instance of schema $\mathbf{P}$ we will have in R a nonempty set of tuples whose fourth argument is the constant i. Further, for each $j \in [1, m]$ and for the variable Y_j in F, in each instance of schema $\mathbf{P}$ we will have in each of P and S a nonempty set of tuples whose second argument is the constant j.

The set Σ in the instance $\mathcal{D}_s(F)$ is the empty set: $\Sigma = \emptyset$.

We now define the set of views $\mathcal{V} = \{U, V, W\}$ in the instance $\mathcal{D}_s(F)$. First, for the clauses $\bar{C}$ in F we introduce the following view V:

$V(Z_1, Z_2, Z_3, i) \leftarrow R(Z_1, Z_2, Z_3, i).$

The answer to this view simply mirrors the relation R.

The instance $\mathcal{D}_s(F)$ includes a set MV of answers to the views in $\mathcal{V}$. The relation $MV[V]$ in MV records, for each $i \in [1, l]$ and for the clause C_i in F, the seven (out of the total eight possible) satisfying assignments for the clause. (We follow [25] in using 1 for *true* and 0 for *false*.) The fourth argument of each tuple in $MV[V]$ for these seven assignments (for C_i) is always i.

As a running example, we use the following example from the proof of Theorem 3.3 in [25]: Consider the formula

$F = (X_1 \vee X_2 \vee Y_1) \wedge (\neg X_1 \vee \neg X_2 \vee \neg Y_2).$

The seven satisfying assignments to X_1, X_2, and Y_1 in the first clause $C_1 = (X_1 \vee X_2 \vee Y_1)$ of F are $(1,1,1)$, $(1,1,0)$, $(1,0,1)$, $(1,0,0)$, $(0,1,1)$, $(0,1,0)$, and $(0,0,1)$. For the second clause $C_2 = (\neg X_1 \vee \neg X_2 \vee \neg Y_2)$ of F, the seven satisfying assignments to X_1, X_2, and Y_2 are $(0,0,0)$, $(0,0,1)$, $(0,1,0)$, $(0,1,1)$, $(1,0,0)$, $(1,0,1)$, and $(1,1,0)$.

By construction of the view V and by our intuition for the relation R, see above, we construct the relation $MV[V]$ from these fourteen assignments as follows. First, the seven assignments as above for C_1 are adorned, in the fourth argument of V, by the index 1 of C_1, as follows: $V(1,1,1,1)$, $V(1,1,0,1)$, $V(1,0,1,1)$, $V(1,0,0,1)$, $V(0,1,1,1)$, $V(0,1,0,1)$, and $V(0,0,1,1)$. Similarly, the seven assignments as above for C_2 get adorned, in the fourth argument of V, by the index 2 of C_2, as follows: $V(0,0,0,2)$, $V(0,0,1,2)$, $V(0,1,0,2)$, $V(0,1,1,2)$, $V(1,0,0,2)$, $V(1,0,1,2)$, and $V(1,1,0,2)$. These fourteen tuples together constitute the relation $MV[V]$ in the instance $\mathcal{D}_s(F)$ for the formula F in this running example.

We continue defining the views in the set $\mathcal{V} = \{U, V, W\}$, for the instance $\mathcal{D}_s(F)$. For the $m \geq 1$ variables $Y_1, \ldots, Y_m$ in F, we introduce a unary view W:

$W(j) \leftarrow P(Y_j, j), S(Y_j, j).$

In the set MV for $\mathcal{D}_s(F)$, the relation $MV[W]$ is $\{(1), (2), \ldots, (m)\}$. Intuitively, for each $j \in [1, m]$, the tuple (j) in $MV[W]$ witnesses, in each ground instance I of schema $\mathbf{P}$ such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$, the presence

of a ground P-atom $P(y_j, j)$ and of a ground S-atom $S(y_j, j)$ with the same arguments (y_j is some constant value).

Finally, we define the view U in the set $\mathcal{V} = \{U, V, W\}$ for the instance $\mathcal{D}_s(F)$:

$$U(Y, j) \leftarrow S(Y, j).$$

The answer to this view simply mirrors the relation S. Now in the set MV that we are constructing, the relation for U provides the two possible assignments, 1 and 0, to each variable among $Y_1, \ldots, Y_m$ in the set of variables $\bar{Y}$ in the formula F. That is, for each $j \in [1, m]$, the relation $MV[U]$ has exactly two tuples, $U(1, j)$ and $U(0, j)$. For instance, the relation $MV[U]$ for our running example would have four tuples: $MV[U] = \{U(1,1), U(0,1), U(1,2), U(0,2)\}$. Here, the first two tuples correspond to the two possible assignments, 1 and 0, to the variable Y_1 in the formula F in the example. Similarly, the last two tuples in $MV[U]$ correspond to the two possible assignments, 1 and 0, to the variable Y_2 in the formula F in the example.

The above construction generates, for a given formula F, the elements $\mathbf{P}$, Σ, $\mathcal{V}$, and MV in the instance $\mathcal{D}_s(F)$ of information leak that we are producing for F. To complete the construction of $\mathcal{D}_s(F)$ for the given F, we now specify the secret query Q in $\mathcal{D}_s(F)$:

$$Q() \leftarrow \bigwedge_{j=1}^{m} P(Y_j, j) \ \bigwedge_{i=1}^{l} R(Z_{i,1}, Z_{i,2}, Z_{i,3}, i).$$

Intuitively, the Boolean query Q has a separate subgoal, $P(Y_j, j)$, for each $j \in [1, m]$, that is for each Y_j among the variables $Y_1, \ldots, Y_m$ of the input formula F. The query Q also has a separate subgoal, $R(Z_{i,1}, Z_{i,2}, Z_{i,3}, i)$, for each $i \in [1, l]$, that is for each $C_i(Z_{i,1}, Z_{i,2}, Z_{i,3})$ among the clauses $C_1, \ldots, C_l$ of the input formula F. By design, Q uses in its R-subgoals all the variables of the form $Z_{i,k}$ in the same way as they are used in the formula F. (Recall that for each variable of the form $Z_{i,k}$ in the clauses of F, this variable is either among the variables $\bar{X}$ of F or among the variables $\bar{Y}$ of F.) In addition, also by design of the query Q, for each variable Y_j among $Y_1, \ldots, Y_m$ in the formula F, the *same* variable name Y_j is used in the P-subgoal of Q with j the value of the second argument of the subgoal. It follows that for each variable that is used as the first argument of some P-subgoal of the query Q, this same variable must occur in the conjunction $\bigwedge_{i=1}^{l} R(Z_{i,1}, Z_{i,2}, Z_{i,3}, i)$ in the body of Q.

As an illustration, the query Q for the formula F in our running example is as follows:

$$Q() \leftarrow P(Y_1, 1), P(Y_2, 2), R(X_1, X_2, Y_1, 1), R(X_1, X_2, Y_2, 2).$$

We have completed the construction of the instance $\mathcal{D}_s(F)$ for an arbitrary input 3-CNF propositional formula F. By construction, the instance $\mathcal{D}_s(F) = (\mathbf{P}, \Sigma, \mathcal{V}, Q, MV)$ is a CQ instance of information leak. Moreover, $\Sigma = \emptyset$ in $\mathcal{D}_s(F)$. In this instance $\mathcal{D}_s(F)$, each of $\mathbf{P}$, Σ, and $\mathcal{V}$ does not depend on the input formula F; thus, both the formulation and the size of each of $\mathbf{P}$, Σ, and $\mathcal{V}$ are fixed across all the input formulae F. In contrast, the size of each of Q and MV in $\mathcal{D}_s(F)$ is linear in the size of the input formula F. As a result, the overall size of the instance $\mathcal{D}_s(F)$ is polynomial (linear) in the size of the input formula F.

It turns out that the instance $\mathcal{D}_s(F)$ is also valid, by the existence of a ground instance $I^{(1,1,\ldots,1)}(F)$ of schema $\mathbf{P}$ such that $\mathcal{V} \Rightarrow_{I^{(1,1,\ldots,1)}(F),\Sigma} MV$, as follows. (Intuitively, the m-tuple $(1, 1, \ldots, 1)$ in the name of this instance $I^{(1,1,\ldots,1)}(F)$ refers to the fact that this instance represents an assignment of the value 1 to each of the m variables $Y_1, \ldots, Y_m$ of the input formula F.) $I^{(1,1,\ldots,1)}(F)$ is the union of the instance $I^{(1,1,\ldots,1)}_{(P,S)}(F)$, as specified below, with the instance $I^{(1,1,\ldots,1)}_{(R)}(F)$ that has the same set of tuples for the relation R as the instance MV has in the relation for V. As an illustration, in our running example, the instance $I^{(1,1)}_{(R)}(F)$ has the same fourteen tuples as we saw above in the relation $MV[V]$, except that these same tuples are now in the relation R in $I^{(1,1)}(F)$:

$$\begin{aligned} I^{(1,1)}_{(R)}(F) = \{&R(1,1,1,1),\ R(1,1,0,1),\ R(1,0,1,1),\\ &R(1,0,0,1),\ R(0,1,1,1),\ R(0,1,0,1),\\ &R(0,0,1,1),\ R(0,0,0,2),\ R(0,0,1,2),\\ &R(0,1,0,2),\ R(0,1,1,2),\ R(1,0,0,2),\\ &R(1,0,1,2),\ R(1,1,0,2)\}. \end{aligned}$$

For the general case, the instance $I^{(1,1,\ldots,1)}_{(P,S)}(F)$ is of the following form:

$$\begin{aligned} I^{(1,1,\ldots,1)}_{(P,S)}(F) = \{&P(1,1), P(1,2), \ldots, P(1,m), S(1,1),\\ &S(0,1), S(1,2), S(0,2), \ldots, S(1,m), S(0,m)\}. \end{aligned}$$

As an illustration, for our running example, the instance $I^{(1,1)}_{(P,S)}(F)$ is as follows:

$$I^{(1,1)}_{(P,S)}(F) = \{P(1,1), P(1,2), S(1,1), S(0,1), S(1,2), S(0,2)\}.$$

The entire instance $I^{(1,1)}(F)$ for our running example is the union of the instances $I^{(1,1)}_{(P,S)}(F)$ and $I^{(1,1)}_{(R)}(F)$ as given above.

Intuitively, the tuples in the relation S in $I^{(1,1,\ldots,1)}_{(P,S)}(F)$ mirror the instance $MV[U]$, by definition of the view U and of $MV[U]$. The tuples $P(1,1)$, $P(1,2)$, $\ldots$, $P(1,m)$ in $I^{(1,1,\ldots,1)}_{(P,S)}(F)$ give us a key part of this proof, by representing a particular assignment of values 1 and 0, one value to each Y_j ($j \in [1, m]$) among the variables Y_1, $\ldots$, Y_m of the formula F. The particular assignment of values 1 and 0 to the variables $\bar{Y}$ of F that is represented in the instance $I^{(1,1,\ldots,1)}(F)$ is the assignment of the value 1 to each of the m variables. We represent this fact in the name of the instance $I^{(1,1,\ldots,1)}(F)$, by using this assignment as m-tuple $(1, 1, \ldots, 1)$ in the superscript in the name.

It is straightforward to verify that the ground instance $I^{(1,1,\ldots,1)}(F)$ satisfies $\mathcal{V} \Rightarrow_{I^{(1,1,\ldots,1)}(F),\Sigma} MV$. Further, it is straightforward to verify that there exist $2^m - 1$ more ground instances of schema $\mathbf{P}$, as follows. Each such instance, denote it for now by J, differs from the instance $I^{(1,1,\ldots,1)}(F)$ only in whether the first argument of one or more P-tuple(s) in J is the value 0, instead of 1 as it is in $I^{(1,1,\ldots,1)}(F)$. It is straightforward to verify that for each such instance J, we have that $\mathcal{V} \Rightarrow_{J,\Sigma} MV$

in the context of the instance $\mathcal{D}_s(F)$. Clearly, the total number of such instances J, including the instance $I^{(1,1,\ldots,1)}(F)$, is 2^m, as the value 1 of the first argument of $P(1, j)$ in $I^{(1,1,\ldots,1)}(F)$ can be flipped to 0 independently for each $j \in [1, m]$.

For each instance J constructed as above, we name the instance using the same notation as for the instance $I^{(1,1,\ldots,1)}(F)$, by adorning the name $I(F)$ with an m-tuple $(a_1, a_2, \ldots, a_m)$ in the superscript, where a_j for each $j \in [1, m]$ is either 1 or 0. For example, the instance $I^{(0,0,\ldots,0)}(F)$ is the instance that differs from the instance $I^{(1,1,\ldots,1)}(F)$ only in that the first argument of each P-tuple in J is the value 0, instead of 1 as it is in $I^{(1,1,\ldots,1)}(F)$.

To the 2^m instances of the form $I^{(a_1,a_2,\ldots,a_m)}(F)$ as defined above, we refer collectively as *the core instances (of schema* $\mathbf{P}$*) for the input formula* F. In addition to these core instances, there is an infinite number of other ground instances of schema $\mathbf{P}$, where for each instance, I, we have that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$ in the context of the instance $\mathcal{D}_s(F)$. By definition of $\mathcal{D}_s(F)$, each such instance I can be obtained by unioning one of the core instances for F, call the instance K, with a finite set of ground atoms of the form $P(c, d)$, where either (i) d is a constant that is not in the set $[1, m]$, or (ii) d is in the set $[1, m]$, while c is a constant distinct from the value g in the atom $P(g, d)$ in the "core part" K of the instance I.

For each ground instance I that satisfies $\mathcal{V} \Rightarrow_{I,\Sigma} MV$ and by definition of the query Q in $\mathcal{D}_s(F)$, we obtain the following useful observations.

LEMMA H.1. *Given a 3-CNF propositional formula* F *and the instance* $\mathcal{D}_s(F)$ *of information leak for* F*. Then, for all ground instances* I *of schema* $\mathbf{P}$ *such that* $\mathcal{V} \Rightarrow_{I,\Sigma} MV$ *in the context of the instance* $\mathcal{D}_s(F)$*, the relation* R *is the same in all the instances* I*, and the relation* S *is also the same in all the instances* I*.* □

This result is immediate from the definitions of the views U and V. The result of Lemma H.1 implies that in each ground instance I of schema $\mathbf{P}$ such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$, (i) the relation R is the same in I as the relation R in the fixed instance $I^{(1,1,\ldots,1)}(F)$ defined above, and (ii) the relation S is the same in I as the relation S in $I^{(1,1,\ldots,1)}(F)$.

LEMMA H.2. *Given a 3-CNF propositional formula* F *and the instance* $\mathcal{D}_s(F)$ *of information leak for* F*. Then, for each ground instance* I *of schema* $\mathbf{P}$ *such that* $\mathcal{V} \Rightarrow_{I,\Sigma} MV$ *in the context of the instance* $\mathcal{D}_s(F)$*, the relation* P *in* I *has for each* $j \in [1, m]$ *an atom of the form* $P(e_j, j)$*, with the constant* $e_j \in \{1, 0\}$*.* □

PROOF. This result is immediate from the definitions of the views U and W and from the specifications of the relations $MV[U]$ and $MV[W]$ in $\mathcal{D}_s(F)$. Indeed, recall that $MV[W] = \{(1), (2), \ldots, (m)\}$. By definition of the view W, for each $j \in [1, m]$ we have that the tuple (j) in $MV[W]$ witnesses, in each ground instance I of schema $\mathbf{P}$ such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$, the presence of a ground P-atom $P(e_j, j)$ and of a ground S-atom $S(e_j, j)$ with the same arguments. Now consider an arbitrary ground atom $S(d, f)$ in any ground instance I of schema $\mathbf{P}$ such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$. By definition of the view U and by the contents of the relation $MV[U]$, the value d in this atom $S(d, f)$ must be one of 1 and 0, and the value f in $S(d, f)$ must belong to the set $[1, m]$. Thus, via the definition of the view W and the contents of the relation $MV[W]$ as discussed above, we obtain that each ground atom of the form $P(e_j, j)$ as above must have its value e_j restricted to one of 1 and 0. The claim of the lemma follows. □

LEMMA H.3. *Given a 3-CNF propositional formula* F *and the instance* $\mathcal{D}_s(F)$ *of information leak for* F*. Let* I *be a ground instance of schema* $\mathbf{P}$ *such that* $\mathcal{V} \Rightarrow_{I,\Sigma} MV$ *in the context of the instance* $\mathcal{D}_s(F)$*. Then there exists a core instance* K *for* F *such that there is an identity homomorphism from* K *to* I*.* □

LEMMA H.4. *Given a 3-CNF propositional formula* F *and the instance* $\mathcal{D}_s(F)$ *of information leak for* F*. Let* I *be a ground instance of schema* $\mathbf{P}$ *such that* $\mathcal{V} \Rightarrow_{I,\Sigma} MV$ *in the context of the instance* $\mathcal{D}_s(F)$*. Then for each valuation,* μ*, from* Q *to* I*, the image of all the* P*-subgoals of* Q *under* μ *is the relation for* P *in one of the core instances for* F*.* □

(Toward the proof of Lemma H.4, recall that by design of the query Q, the first argument of each P-subgoal of the query Q must also occur as one of the first three arguments of at least one R-subgoal of Q.)

By the above lemmae and by the structure of all the ground instances I of schema $\mathbf{P}$ such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$, it must be that, in the context of the instance $\mathcal{D}_s(F)$:

(*) For each I such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$ and for each valuation, μ, from the query Q to I, the image $\mu(body_{(Q)})$ of the body of the query Q under μ is a subset of one of the core instances for F.

Specifically, by construction of the query Q, for any such core instance K for F, all the ground atoms in the relation P in K must be present in the set $\mu(body_{(Q)})$. (Recall that in each core instance, K, for the input formula F, the relation P intuitively represents exactly one specific assignment of values 1 an 0 to the $m \geq 1$ variables $\bar{Y}$ of the formula F. Further, each specific assignment of values 1 an 0 to the $m \geq 1$ variables $\bar{Y}$ of the formula F is represented by a separate core instance for F.)

As an illustration, consider the query Q of our running example and the instance $I^{(1,1)}(F)$ of schema $\mathbf{P}$ for that example. (Both the query Q and the instance $I^{(1,1)}(F)$ for this running example have already been given in this proof.) Consider a mapping $\mu = \{ X_1 \rightarrow 1, X_2 \rightarrow 0, Y_1 \rightarrow 1, Y_2 \rightarrow 1 \}$. We can show that μ is a valuation from the query Q to the instance $I^{(1,1)}(F)$. The image $\mu(body_{(Q)})$ of the body of the query Q under the valuation μ includes all the ground atoms in the relation P in the instance $I^{(1,1)}(F)$, that is both atoms $P(1, 1)$ and $P(1, 2)$ in $I^{(1,1)}(F)$.

We now proceed to show that for the input formula F and for the corresponding instance $\mathcal{D}_s(F)$ constructed as above, the following two statements are equivalent:

(I) For each assignment of values 1 and 0 to the variables $\bar{Y}$ in the formula F, there exists an assign-

ment of values 1 and 0 to the variables $\bar{X}$ in F such that F is true under these assignments; and

(II) The tuple () is an answer to the query Q on all the ground instances I of schema $\mathbf{P}$ such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$ in the context of the instance $\mathcal{D}_s(F)$.

Note that by Definition 4.2, the statement (II) says that the tuple () is a potential information-leak tuple in the instance $\mathcal{D}_s(F)$. Thus, once we show the equivalence of the statements (I) and (II), our proof of Π_2^p hardness of the problem of information-leak disclosure for CQ inputs with $\Sigma = \emptyset$ will be complete.

We begin the proof of the equivalence of the statements (I) and (II) by making the following observation. Denote by $\mathcal{R}_{(Q)}$ the conjunction of the R-subgoals of the query Q in the instance $\mathcal{D}_s(F)$. Further, denote by $\mathcal{P}_{(Q)}$ the conjunction of the P-subgoals of the query Q in the instance $\mathcal{D}_s(F)$. Consider any ground instance I of schema $\mathbf{P}$ such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$ in the context of the instance $\mathcal{D}_s(F)$. Let ν be any mapping of the set $\bar{Z}$ of the variables ($\bar{X}$ and $\bar{Y}$) of the formula F into the set $\{0, 1\}$. Similarly to the argument in [3] (as also used in the proof of Theorem 3.3 in [25]), we can show that any such mapping ν is a satisfying assignment for the formula F if and only if $\nu(\mathcal{R}_{(Q)})$ is a subset of the instance I. By Lemma H.1, we have that any such mapping ν is a satisfying assignment for the formula F if and only if $\nu(\mathcal{R}_{(Q)})$ is a subset of *each* ground instance J of schema $\mathbf{P}$ such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$ in the context of the instance $\mathcal{D}_s(F)$. From the above reasoning and from Lemmae H.2–H.3, we obtain the following result of Lemma H.5.

We first introduce some notation. In the remainder of the proof of Theorem 7.2, let $\mu_{(\bar{Y})}^{(a_1,\ldots,a_m)}$, with $a_j \in \{0, 1\}$ for each $j \in [1, m]$, denote the mapping from the set of variables $\bar{Y}$ of the formula F to the set $\{0, 1\}$, such that $\mu_{(\bar{Y})}^{(a_1,\ldots,a_m)}(Y_j) = a_j$ for each $j \in [1, m]$. Further, let $\mu_{(\bar{X})}$ denote a mapping from the set of variables $\bar{X}$ of the formula F to the set $\{0, 1\}$.

LEMMA H.5. *Given a 3-CNF formula F with $m \geq 1$ variables $\bar{Y}$ and with $n \geq 0$ variables $\bar{X}$, let $(a_1, \ldots, a_m)$ be an arbitrary m-tuple such that $a_j \in \{0, 1\}$ for each $j \in [1, m]$. Let $\mu_{(\bar{X})}$ be an arbitrary mapping from the set of variables $\bar{X}$ of the formula F to the set $\{0, 1\}$. Let $\mathcal{D}_s(F)$ be the instance of information leak that is associated with the formula F. Finally, let I be a ground instance of the schema $\mathbf{P}$, such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$ in the context of the instance $\mathcal{D}_s(F)$, and such that the set $\{P(a_1, 1), \ldots, P(a_m, m)\}$ is a subset of the instance I.*

Then the following two statements are equivalent:

- *The assignment $\mu_{(\bar{Y})}^{(a_1,\ldots,a_m)} \cup \mu_{(\bar{X})}$ of the variables of the formula F to elements of the set $\{0, 1\}$ is a satisfying assignment for the formula F; and*
- *The empty tuple () is in the answer to the query Q on the instance I due to the valuation $\mu_{(\bar{Y})}^{(a_1,\ldots,a_m)} \cup \mu_{(\bar{X})}$ from $body_{(Q)}$ to I.* □

We are now ready to show the equivalence of the statements (I) and (II) as formulated above.

(I) → (II): Suppose that for each m-tuple $(a_1, \ldots, a_m)$, such that $a_j \in \{0, 1\}$ for each $j \in [1, m]$, we have that there exists a mapping $\mu_{(\bar{X})}$ from the set of variables $\bar{X}$ of the formula F to the set $\{0, 1\}$, such that

$$\mu_{(\bar{Y})}^{(a_1,\ldots,a_m)} \cup \mu_{(\bar{X})}$$

is a satisfying assignment for the formula F. Fix an arbitrary ground instance I of the schema $\mathbf{P}$ such that $\mathcal{V} \Rightarrow_{I,\Sigma} MV$. Then, by Lemmae H.2 and H.5, the empty tuple () is in the relation $Q(I)$.

(II) → (I): Consider the set $\mathcal{K}$ of the 2^m core instances (of schema $\mathbf{P}$) for the formula F. By construction of the set $\mathcal{K}$, for each m-tuple $(a_1, \ldots, a_m)$ such that $a_j \in \{0, 1\}$ for each $j \in [1, m]$, there exists an instance $K \in \mathcal{K}$ such that the set $\{P(a_1, 1), \ldots, P(a_m, m)\}$ is a subset of the instance K, and the relation $K[P]$ has *no other tuples.*

Fix an arbitrary instance $K \in \mathcal{K}$; the relation $K[P] = \{P(a_1, 1), \ldots, P(a_m, m)\}$ specifies a particular m-tuple $(a_1, \ldots, a_m)$ such that $a_j \in \{0, 1\}$ for each $j \in [1, m]$. By our assumption (II), there exists a mapping $\mu_{(\bar{X})}$ from the set of variables $\bar{X}$ of the formula[8] F to the set $\{0, 1\}$, such that

$$\mu(K) = \mu_{(\bar{Y})}^{(a_1,\ldots,a_m)} \cup \mu_{(\bar{X})}$$

is a valuation from the query Q to the instance K that produces the empty tuple () in the relation $Q(K)$. Thus, by Lemma H.5, the mapping $\mu(K)$ of the variables of the formula F to the set $\{0, 1\}$ is a satisfying assignment for the formula F. The claim of (I) follows from the observation (made above) that for each m-tuple $(a_1, \ldots, a_m)$ such that $a_j \in \{0, 1\}$ for each $j \in [1, m]$, there exists an instance $K \in \mathcal{K}$ such that the set $\{P(a_1, 1), \ldots, P(a_m, m)\}$ is in the instance K. This completes the proof of Theorem 7.2.

We now outline how to modify the main body of the proof, given above, so that it would also go through for each input instance with $m = 0$. First, we replace $MV[W]$ by the relation $MV[W] = \{\ (0)\ \}$. Second, we replace the conjunction of the P-subgoals in the query Q by a single subgoal $P(Y_0, 0)$. (Note that for the first argument Y_0 of the subgoal $P(Y_0, 0)$ of the query Q, we have, unlike the case $m \geq 1$, that Y_0 is *not* in the conjunction of the R-subgoals of the query Q.) Third, we replace $MV[U]$ by $MV[U] = \{\ U(1, 0),\ U(0, 0)\ \}$. Fourth, we use $I^{(1)}$ and $I^{(0)}$ and as the only two "core" instances I. Here, $I^{(1)}$ has $P = \{P(1, 0)\}$, and $I^{(0)}$ has $P = \{P(0, 0)\}$. The other relations (R and S) in each of $I^{(1)}$ and $I^{(0)}$ mirror R and S in the core relations for $m \geq 1$. With these modifications, the proof above goes through for all input formulae with $m = 0$.

[8]Recall that the set $\bar{Z} = \bar{X} \cup \bar{Y}$ is the set of all variables of the formula F, and is also the set of all variables of the query Q in $\mathcal{D}_s(F)$.