Available online at www.sciencedirect.com

ScienceDirect

Journal of Computer and System Sciences 73 (2007) 507–534

JOURNAL OF COMPUTER AND SYSTEM SCIENCES

www.elsevier.com/locate/jcss

A formal analysis of information disclosure in data exchange [☆]

Gerome Miklau [a,*], Dan Suciu [b]

[a] Department of Computer Science, University of Massachusetts at Amherst, 140 Governors Drive, Amherst, MA 01003, USA
[b] Department of Computer Science and Engineering, University of Washington, Box 352350, Seattle, WA 98195, USA

Received 15 January 2005; received in revised form 3 March 2006

Available online 13 November 2006

Abstract

We perform a theoretical study of the following *query-view security problem*: given a view V to be published, does V logically disclose information about a confidential query S? The problem is motivated by the need to manage the risk of unintended information disclosure in today's world of universal data exchange. We present a novel information-theoretic standard for query-view security. This criterion can be used to provide a precise analysis of information disclosure for a host of data exchange scenarios, including multi-party collusion and the use of outside knowledge by an adversary trying to learn privileged facts about the database. We prove a number of theoretical results for deciding security according to this standard. We also generalize our security criterion to account for prior knowledge a user or adversary may possess, and introduce techniques for measuring the magnitude of partial disclosures. We believe these results can be a foundation for practical efforts to secure data exchange frameworks, and also illuminate a nice interaction between logic and probability theory.

Keywords: Database security; Information disclosure; Inference control; Privacy

1. Introduction

Traditional security mechanisms protect data at the *physical* level. For example, firewalls and other perimeter mechanisms prevent the release of raw data, as do conventional access controls for file systems and databases. In data exchange, however, such mechanisms are limited since they can only protect the data up to the first authorized recipient. When data is exchanged with multiple partners, information may be unintentionally disclosed, even when all physical protection mechanisms work as intended. As an extreme example, Sweeney demonstrated this [26] when she retrieved the privileged medical data of William Weld, former governor of the state of Massachusetts, by linking information from two publicly available databases, each of which was considered secure in isolation.[1]

[☆] An extended abstract of this paper originally appeared as [Gerome Miklau, Dan Suciu, A formal analysis of information disclosure in data exchange, in: Conference on Management of Data (SIGMOD), ACM Press, 2004, pp. 575–586].

[*] Corresponding author.
E-mail addresses: miklau@cs.umass.edu (G. Miklau), suciu@cs.washington.edu (D. Suciu).

[1] A voter registration list for governor Weld's home city included the voters' names, zip, birth-date, and sex. The state's Group Insurance Commission published a database containing "anonymized" medical data, including only the patients' zip, birth-date, and sex, but omitting the name. Only one person matched governor Weld's zip, birth-date, and sex, allowing Sweeney to retrieve his medical records.

We address the case when data originates at a single source, but may be exchanged with multiple partners and further disseminated. This does not address Sweeney's case, but that of a single source protecting its data. To prevent unintended information disclosure, we need a formalism to protect the data at the *logical* level. Specifically, we study the following fundamental problem, called the *query-view security* problem: given views $V_1, V_2, \ldots$ that we want to publish, do they logically disclose any information about a query S that we want to keep secret? The views are expressed in a query language, and may remove data items through projections or selections, or break associations between data items. Adopting the nomenclature of the cryptographic community we say that Alice wants to give Bob the views $V_1, V_2, \ldots$ over a public channel, but would like to prevent an adversary Mallory (or even Bob) from learning the answer to secret query S. The query expressions $S, V_1, V_2, \ldots$ are known by the adversary, the answers to $V_1, V_2, \ldots$ are published, while the underlying database remains secret.

In many cases a disclosure may not reveal a fact with complete certainty or an exact answer to a query. Consider a database consisting of a single relation:

$$\text{Employee}(\textit{name}, \textit{department}, \textit{phone}).$$

Imagine Alice publishes the view projecting on (*name*, *department*) to Bob, and publishes the view projecting (*department*, *phone*) to Carol:

$$V_{Bob} = \Pi_{name,department}(\text{Employee}),$$
$$V_{Carol} = \Pi_{department,phone}(\text{Employee}).$$

Alice wants to protect employees' phone numbers, so the secret query would be:

$$S = \Pi_{name,phone}(\text{Employee}).$$

If Bob and Carol collude, they cannot compute the answer to S since the association between *name* and *phone* is not present in V_{Bob} and V_{Carol}. However a *partial disclosure* has occurred, and it is a potentially serious threat to the security of S. For example, if only four people work in each department then an adversary can guess any person's phone number with a 25% chance of success by trying any of the four phone numbers in her department.

For a more complex example, consider a manufacturing company that needs to exchange XML messages with several partners. Each message type is a dynamic view that computes on request some information about the company's manufacturing data: V_1 contains detailed information about parts for a specific product, to be exchanged with suppliers; V_2 contains detailed information about products' features, options, and selling prices, to be exchanged with retailers and customers; while V_3 provides labor cost information to be sent to a tax consulting firm. The company wants to keep secret the internal manufacturing cost for its products, which can be expressed in terms of a query S. Conventional protection mechanisms can ensure that each message is received and read only by authorized users, but are powerless beyond that point. At a minimum, the company would like to audit each of the three views and ensure that it does not disclose the secret information to its authorized recipient. But in addition the company would like to understand if any information is disclosed when views are combined (colluded), for example, when V_3 is accidentally or intentionally sent to a supplier, or when the tax consulting firm merges with one of the customers. None of these tasks are performed today, manually or automatically, because there is no clear understanding of when and how much information is disclosed at the logical level.

Our study of logical information disclosure applies directly to the following data exchange scenarios:

Multi-party collusion. Alice would like to publish n views $V_1, \ldots, V_n$ to n different users. Given a secret query S, which are the multiple party collusions among the users that will violate the confidentiality of S?

Prior knowledge. Suppose Mallory, the adversary, has some knowledge about the database represented in the form of a query K. Can Alice publish V without disclosing anything about the secret query S? The prior knowledge may be common knowledge, such as the fact that social security numbers are unique or that phone numbers with the same area code are in the same state, or may be more specific knowledge about the domain that Mallory has acquired somehow.

Relative security. Alice has already published a view U. This already leaked some information about a secret query S, but was considered an acceptable risk by Alice. Now she wishes to publish an additional view V. Does V disclose any *additional* information about S over what was already disclosed by U? This is not the

Table 1
Pairs of views and queries, over relation Emp(*name*, *department*, *phone*) and an informal description of their information disclosure

	View(s)	Query	Information disclosure	Query-view security
(1)	$V_1(n,d){:}-\mathrm{Emp}(n,d,p)$	$S_1(d){:}-\mathrm{Emp}(n,d,p)$	Total	No
(2)	$V_2(n,d){:}-\mathrm{Emp}(n,d,p)$	$S_2(n,p){:}-\mathrm{Emp}(n,d,p)$	Partial	No
	$V_2'(d,p){:}-\mathrm{Emp}(n,d,p)$			
(3)	$V_3(n){:}-\mathrm{Emp}(n,d,p)$	$S_3(p){:}-\mathrm{Emp}(n,d,p)$	Minute	No
(4)	$V_4(n){:}-\mathrm{Emp}(n,\mathsf{Mgmt},p)$	$S_4(n){:}-\mathrm{Emp}(n,\mathsf{HR},p)$	None	Yes

same as saying that the query S is secure with respect to the pair of views U, V, because S is not secure with respect to U.

Encrypted views. Given an adversary who has access to a version of the database in which each attribute has been encrypted, is confidential query S secure? These "views" are increasingly available to users, for instance in controlled publishing scenarios [21], private computation protocols [2] or in architectures designed to support untrusted database service providers [17]. Encrypted views can be handled with the techniques described here. We apply our results to this setting in Section 5.4.

Our first contribution is a formal definition of query-view security that captures the disclosure of partial information. Inspired by Shannon's notion of *perfect secrecy* [24], the definition compares the likelihood of an adversary guessing the answer to secret query S with and without knowledge of views $V_1, V_2, \ldots, V_n$. When the difference is zero, we say that the query is secure with respect to the views. To the best of our knowledge this is the first attempt to formalize logical information disclosure in databases. Our second contribution consists of a number of theoretical results about query-view security: we prove a necessary and sufficient condition for query-view security, and show that the security problem for conjunctive queries is Π_2^p-complete; we generalize query-view security to account for pre-existing knowledge; and when the query is not secure with respect to a view, we characterize the magnitude of disclosure. These theoretical results illuminate an interesting connection between logic and probability theory.

1.1. A spectrum of information disclosure

Table 1 contains a set of query-view pairs referring to an Employee relation, along with an informal description of the information the views disclose about the query. These examples represent a spectrum of information disclosure, beginning with total disclosure, and ending with a secure query and view. The first query and view is an obvious example of a total disclosure because S_1 is answerable using V_1.

Example (2) is precisely the example mentioned above in which Bob (given V_2) and Carol (given V_2') collude to cause a partial disclosure. It is worth noting that a contained rewriting of S_2 using V_2, V_2' exists here. For small departments, it may be easy for Mallory to guess the association between names and phone numbers.

As another example of partial disclosure, consider example (3), whose view is $V_3 = \Pi_{name}(\text{Employee})$. We ask whether query $S_3 = \Pi_{phone}(\text{Employee})$ is secure when this view is published. In this case the view omits phone entirely and would seem to reveal nothing about phone numbers in S_3. Surprisingly, the view does disclose some information about the secret query. In particular, it can reveal something about the size of the Employee relation, and therefore contains some small amount of information about the omitted column. We describe this further in Section 4.

The last example, Table 1(4), is a case where no information is disclosed. The names of employees in the Management department reveal nothing about the names of employees in the Human Resources department.

1.2. Paper organization

In the next section we provide an overview of related work, and explain why existing database techniques are insufficient for analyzing information disclosure. Section 3 presents notation and a probabilistic model of databases. Section 4 describes our definition of query-view security and its main results. Section 5 extends these results to include prior knowledge, and query-view security for encrypted views. Section 6 presents two approaches for relaxing the security standard. We conclude in Section 7. A complete proof of the hardness of deciding query-view security is provided in Appendix A.

2. Related work

Below we distinguish the query-view security problem from superficially-related database problems, describe connections to related theoretical results and alternative models, and provide pointers to contributions in this area that have appeared since the original publication [22] of our work.

2.1. Query answering

The basic problem addressed in query answering [18] is: given a view V (or several such views), answer a query S by using only the data in the view(s). A more refined version, called query rewriting, asks for the answer to be given as a query over the view(s) V. The connection to our problem is that, whenever S can be answered from V, then S is obviously not secure, as in Table 1(1). However, adopting non-answerability as a criterion for security would clearly be a mistake: it would classify example (2) as secure. As we claim above, even though the query S may be not answerable using V, substantial information about the query may be revealed, allowing an attacker to guess the secret information with a high probability of success.

A related strategy considers a view V and its answer $v = V(I)$ as a constraint on the set of possible database instances, making some impossible. The set of possible answers to a query S given V may therefore be reduced. (If this set happens to have size 1, then the answer to S is determined by $v = V(I)$.) We might say S is secure given V if *every* possible answer to S remains possible given V. This criterion would classify examples (2) and (3) correctly. That is, it would capture the partial disclosures in these cases, but not in others. However, this standard of security ignores the *likelihood* of the possible answers of S. For example, consider the boolean query and view below:

$$S():-\text{Employee}(\mathsf{Jane}, \mathsf{Shipping}, \mathsf{1234567}),$$

$$V():-\text{Employee}(\mathsf{Jane}, \mathsf{Shipping}, p), \text{Employee}(n, \mathsf{Shipping}, \mathsf{1234567}).$$

In the absence of the view, the query S (which asserts the presence of a particular tuple in the database) may be true or false. Given the answer to V on the database, S could still evaluate to true or to false. However, the probability that S is true is substantially higher given that V is true, and so a serious disclosure has occurred. In general, while V may not rule out any possible answers to S, some answers may become less likely, or in the extreme, virtually improbable without contradicting a security criterion based on possible answers. Our definition of query-view security handles this case.

2.2. Database security

Access control mechanisms in databases [5,8] are used to define the rights of authorized subjects to read or modify data elements, and therefore usually offer control at a physical level, rather than a logical level. For example, a simple access control policy might prohibit access to confidential columns in a relation. This is similar to publishing a view after projecting out those columns. We have shown that such a view can in fact contain a partial disclosure about the confidential column (see Example 4.2).

A statistical database publishes views consisting of aggregate functions over a subset of records. Information is thus hidden by aggregating data, and the security problem is to ensure that data in individual tuples remains secret. The security of statistical databases has a long history [1,8]. Disclosures are classified as exact disclosures of confidential statistics, negative disclosures (the answer to statistic S is not value x) and approximate disclosures (the answer to statistic S is in range $[l_1, l_2]$ with confidence $p\%$) [8]. A wide range of techniques have been developed to provide security including query restriction, data perturbation, and output perturbation [1]. The present work on query-view security is orthogonal: we do not hide data by aggregation, but rather through projections or selections, or breaking associations. Rather than statistics, we need to use probability theory to reason about query-view security.

The authors of [11] study formal definitions of privacy in the context of privacy preserving data mining. In this setting the goal is to permit accurate data mining models to be built over aggregates while preventing disclosure of individual items. Here the published view is the result of applying a randomization operator to data values or a distribution of data values. It is shown that a known information-theoretic definition of privacy may permit certain disclosures, and they propose an extended measure to account for this drawback.

A protection mechanism for relational databases was proposed in [3] based on a probabilistic definition of security similar to our own. An algorithm for deciding query-view security was not known, and relative security and quantified security were not addressed. In [4] the same authors focus on comparing probabilistic independence (closely related to our security criterion) with algebraic independence, but are not concerned with a decision procedure for probabilistic independence.

2.3. *Alternative models*

Query-view security could be defined by comparing the entropy [25] of S with the conditional entropy of S given V. If this is done for all possible answers s, $\bar{v}$ (computing the entropy of event $S = s$, and $S = s$ given $\bar{V} = \bar{v}$) then the condition is equivalent to our security criterion. However, it would be more common to compare the entropies of the events defined by S and V, aggregating over the answers to S and V. This would result in a criterion strictly weaker than ours.

The goal of the probabilistic relational model [15,19] is to model statistical patterns in huge amounts of data. The issues addressed are learning models from existing data, modeling statistics about a given database (e.g. to be used by a query optimizer), and inferring missing attribute values. These techniques do not provide us with means to reason about information disclosure, which is independent of a particular data instance.

For semistructured data, Yang and Li [27] study the problem of unintended disclosures in publishing. The disclosures result from the fact that semantic constraints hold over the data instances. Views that remove data items therefore does not necessarily afford secrecy. The authors propose algorithms for computing the maximal document that can be published without allowing illegal inferences.

2.4. *Theoretical connections*

A query is independent of an update if the application of the update cannot change the result of the query, for any state of the database. Detecting update independence is useful for maintenance of materialized views [6,10, 20] and efficient constraint checking [16]. Deciding whether a tuple t is critical for a query Q (a notion we define and study in Section 4.2) is equivalent to deciding whether Q is independent of the update that deletes t from the database. Update independence is undecidable for queries and updates expressed as datalog programs [20], but has been shown decidable for certain restricted classes of queries like conjunctive queries with comparisons [6,10,20]. The tight bounds shown in this paper for deciding $crit(Q)$ constitute an interesting special case for update independence.

The so-called FKG inequality [14] is a theoretical result about the correlation between events in a probability space. It is closely related to our security criterion, and can be used to show that $\mathbf{P}(V \wedge S) \geqslant \mathbf{P}(V)\mathbf{P}(S)$, for monotone boolean properties V and S. However, it says nothing about when equality holds, and its inductive proof offers little insight. Our Theorem 4.8 reproves this inequality and furthermore proves the necessary and sufficient condition for equality to hold.

Another topic that connects logic to probability theory are the 0-1 laws [12,13], which hold for a logical language if, for each formula, the probability that a formula is true converges to either 0 or 1 as the size of the domain tends to infinity. Our definition of query-view security is not related to 0-1 laws: our domain size does not grow to infinity but remains fixed and we are concerned about the effect of one formula (the view) on another (the secret query).

2.5. *Subsequent work*

Following the publication of [22], the authors, along with Nilesh Dalvi, have recently considered a relaxed notion of query-view security based on a substantially different probability distribution over databases [7]. We describe this approach in Section 6.2.

In [9] it is assumed that a set of views has been published to the adversary, regardless of disclosure about a secret query. A new view, considered for publication, is then evaluated for additional disclosure. (We consider such a scenario in Section 5.2.) The authors study a version of query-view security similar to our own, but also consider weaker variants. They provide complexity bounds for these decision problems under general probability distributions, and for more expressive integrity constraints.

3. Background and notation

As we mentioned, many disclosures do not involve an adversary computing S completely according to standard database query semantics. Instead a partial disclosure reveals to Mallory something about the likelihood of answers to a secret query S. After an overview of notation, we present our security model that allows formal statements about the probability of a database and query answer. Our discussion is based on the relational model.

3.1. Basic notations

We assume a standard relational schema consisting of several relation names $R_1, R_2, \ldots$, each with a set of attribute names. Let D be the finite domain, which includes all values that can occur in any attributes in any of the relations. For example, D may be the set of decimal numbers up to an upper bound, and all strings up to a given length. In a particular setting we may consider further restricting D, e.g. to include only valid disease names, valid people names, or valid phone numbers.

We use datalog notation to denote tuples belonging to the relations of the given schema. For example, $R_1(a, b, c)$ denotes a tuple in R_1, and $R_3(b, a, a)$ denotes a tuple in R_3. Let $tup(D)$ be the set of all tuples over all relations in the schema that can be formed with constants from the domain D. A *database instance* I is any subset of $tup(D)$, and we denote by $inst(D)$ the set of all database instances over the domain D. A *query* of arity k is a function $Q : inst(D) \to \mathcal{P}(D^k)$. For an instance I, $Q(I)$ denotes the result of applying Q to I. A boolean query is a query of arity 0. A *monotone* query has the property $I \subseteq I' \Rightarrow Q(I) \subseteq Q(I')$. Except where noted, our discussion will focus on *conjunctive queries with inequalities*, written in datalog notation. For example:

$$Q(x) :- R_1(x, a, y), R_2(y, b, c), R_3(x, -, -), \quad x < y,\ y \neq c.$$

Here x, y are variables, $-$ are anonymous variables (each occurrence of $-$ is distinct from all others) and a, b, c are constants.

3.2. The security model

We assume a probability distribution on the tuples, $\mathbf{P} : tup(D) \to [0, 1]$, s.t. for each $t_i \in tup(D)$, $\mathbf{P}(t_i) = x_i$ represents the probability that the tuple t_i will occur in a database instance. We will refer to the pair $(D, \mathbf{P})$ as a *dictionary*. A dictionary induces a probability distribution on specific instances: for any $I \in inst(D)$, the probability that the database instance is precisely I is:

$$\mathbf{P}[I] = \prod_{t_i \in I} x_i \cdot \prod_{t_j \notin I} (1 - x_j). \tag{1}$$

For example, if the schema consists of a single table Patient(*name*, *disease*) representing sensitive data in a hospital, then the domain D may consist of all possible names (e.g. those occurring in a phone book for the entire country), together with all possible diseases cataloged by the CDC. For each tuple $t_i =$ Patient(*name*, *disease*), $\mathbf{P}(t_i)$ is the (very small) probability that a person with that name and that disease is in the hospital's database. To illustrate, assuming 10^8 distinct names and 500 distinct diseases[2] there are $n = 5 \times 10^{10}$ tuples in $tup(D)$, and one possible probability distribution is $\mathbf{P}(t_i) = 200/n$ for every $t_i \in tup(D)$. This is a uniform probability distribution, for which the expected database size is 200 tuples. A more accurate, but far more complex probability distribution is one that takes into account the different risk factors of various ethnic groups and for each diseases. For example, the probability of a tuple Patient("John Johnson", "Cardiovascular Disease") will be slightly higher than the probability of the tuple Patient("Chien Li", "Cardiovascular Disease"), if Americans have a higher risk of a Cardiovascular Disease than Chinese, and the nationality of John Johnson is likely to be American while that of Chien Li is likely to be Chinese.

The probability $\mathbf{P}(t_i)$ may be too complex to compute in practice, but computing it is not our goal. Instead we will *assume* that Mallory can compute it, and can use it to derive information about the secret query S. Thus, we endow the adversary with considerable power, and study under which circumstances no information is disclosed.

[2] Fewer than 500 are listed at `http://www.cdc.gov/health/`.

Given a probability distribution over database instances, a query S attains some answer s with probability equal to the sum of the probabilities of the satisfying instances:

$$\mathbf{P}\big[S(I)=s\big]=\sum_{\{I\in inst(D)\mid S(I)=s\}}\mathbf{P}[I]. \tag{2}$$

Remark. In our model the tuples are independent probabilistic events. This is a limitation. In practice, the occurrence of tuples may be correlated due to underlying relationships in the data or integrity constraints. If tuples are positively correlated (respectively, negatively correlated) the presence of one tuple increases (decreases) the likelihood of another. For example, a key constraint introduces strong negative correlations. We will address some of these limitations in Section 5 by studying query-view security relative to *prior knowledge* that expresses a functional dependency. However, extending our results to a model that can capture arbitrary correlations between tuples remains an open problem.

4. Query-view security

In this section we formalize our notion of query-view security, describe its basic properties, and state our main theorems which result in a decision procedure for query-view security.

4.1. Definition of query-view security

Our standard for query-view security is inspired by Shannon's definition of *perfect secrecy* [24]. Let $\bar{V}=V_1,\ldots,V_k$ be a set of views, and S a "secret" query. Both the views and the query are computed over an instance I of a relational schema. We consider an adversary, Mallory, who is aware of the domain and probability distribution over instances (the dictionary), and is given $\bar{V}(I)$ (but not I). Mallory's objective is to compute $S(I)$. The definition below captures the intuition that $\bar{V}(I)$ discloses no information about $S(I)$. Below, $\bar{V}(I)=\bar{v}$ means $V_1(I)=v_1\wedge\cdots\wedge V_k(I)=v_k$.

Definition 4.1 *(Query-view security).* Let $(D,\mathbf{P})$ be a dictionary. A query S is secure with respect to a set of views $\bar{V}$ if for any possible answer s to the query, and any possible answers $\bar{v}$ to the views, the following holds:

$$\mathbf{P}\big[S(I)=s\big]=\mathbf{P}\big[S(I)=s\;\big|\;\bar{V}(I)=\bar{v}\big]. \tag{3}$$

Query-view security is denoted $S\mid_{\mathbf{P}} V$, or simply $S\mid V$ if $\mathbf{P}$ is understood from the context.

The left-hand side of Eq. (3) represents the *a priori* probability that S attains a particular answer s over the instance I, which can be computed by Mallory using $(D,\mathbf{P})$. The right-hand side is also the probability that $S(I)=s$ but conditioned on the fact that $\bar{V}(I)=\bar{v}$. The security condition asserts the equality of these two probabilities (for all possible s, $\bar{v}$) and therefore says that nothing beyond the *a priori* knowledge is provided by $\bar{V}$. Equation (3) is also the familiar definition of independence of two statistical events. Accordingly, S is secure with respect to $\bar{V}$ iff S and $\bar{V}$ are statistically independent events. We can rewrite (3) as follows:

$$\mathbf{P}\big[S(I)=s\big]\mathbf{P}\big[\bar{V}(I)=\bar{v}\big]=\mathbf{P}\big[S(I)=s\wedge\bar{V}(I)=\bar{v}\big]. \tag{4}$$

Next we apply the definition in two examples:

Example 4.2 *(Non-security).* Consider a single relation $R(X,Y)$ and domain $D=\{a,b\}$. There are 4 possible tuples $R(a,a)$, $R(a,b)$, $R(b,a)$, $R(b,b)$, and the set of instances $inst(D)$ contains the 16 subsets of these. Assume for simplicity that $\mathbf{P}(t_i)=1/2$ for each tuple t_i, and consider the following query and view:

$$V(x):-R(x,y),$$
$$S(y):-R(x,y).$$

V projects the first attribute of R while S projects the second. Although we might expect that the view provides no information about the query, it is actually not the case that $S\mid V$. Informally, the answer to V contains some information about the size of the database which impacts answers to S. Consider a particular answer $\{(a)\}$ for S. There are 3 equally-likely instances generating this answer: $\{R(a,a)\}$, $\{R(b,a)\}$, and $\{R(a,a), R(b,a)\}$. Therefore, we have *a priori* probability:

$$\mathbf{P}\big[S(I)=\big\{(a)\big\}\big]=3/16.$$

Now suppose we are given that $V(I)=\{(b)\}$. There are again 3 instances, only one of which causes $S(I)=\{(a)\}$, so because each instance is equally-likely we have:

$$\mathbf{P}\big[S(I)=\big\{(a)\big\}\mid V(I)=\big\{(b)\big\}\big]=1/3.$$

This contradicts (3) for the particular answers considered, and it follows that S and V are *not* secure for this particular probability distribution. We show in the next section that they are not secure for any distribution.

Example 4.3 *(Security).* As an example of a secure query and view, consider the same schema and dictionary, and:

$$V(x) :- R(x,b),$$
$$S(y) :- R(y,a).$$

Here S is secure with respect to V. We prove this later, but illustrate here with one example. Consider one possible output of S: $S(I)=\{(a)\}$. There are 4 instances that lead to this output, $\{R(a,a)\}$, $\{R(a,a), R(a,b)\}$, $\{R(a,a), R(b,b)\}$, and $\{R(a,a), R(a,b), R(b,b)\}$, hence:

$$\mathbf{P}\big[S(I)=\big\{(a)\big\}\big]=4/16=1/4.$$

Consider also one possible output of V, say $V(I)=\{(b)\}$. There are four instances I satisfying this constraint: $\{R(b,b)\}$, $\{R(b,b), R(a,a)\}$, $\{R(b,b), R(b,a)\}$, $\{R(b,b), R(a,a), R(b,a)\}$. Of these only one also results in $S(I)=\{(a)\}$, hence:

$$\mathbf{P}\big[S(I)=\big\{(a)\big\}\mid V(I)=\big\{(b)\big\}\big]=1/4.$$

One can manually check, for all possible combinations of outputs of S and V, that the probability of S is unchanged by publishing V. We will provide an easier criterion for checking this shortly.

4.1.1. Discussion

Several properties of query-view security follow, providing intuition and justifying our choice of definition.

Symmetry. It follows from Bayes' Theorem that security is a symmetric relation: $S \mid \bar{V}$ iff $\bar{V} \mid S$.

Security (not obscurity). We always assume that publishing the views $\bar{V}$ includes exposing both the view definitions and their answers over the hidden database. Basing the security on concealing the view and query expressions is dangerous. We thus avoid the pitfall of "security by obscurity," identified long ago by the cryptographic community as ineffective [23].

Instance-independence. If the query S is secure with respect to the views $\bar{V}$, it remains so even if the underlying database instance I changes: this follows from the fact that Eq. (3) must hold for any query output s and any view outputs $\bar{v}$. We say that query-view security is *instance independent*. This property is necessary in applications like message-based data exchange, were messages are exchanged continuously, even as the database changes. Once $S \mid \bar{V}$ has been checked, the views $\bar{V}(I)$ can safely be exchanged without any compromise of $S(I)$. In fact, one can prove that if successive instances are independent from one another, then even if Mallory collects snapshots of the views at various moments of time, $\bar{V}(I_1)$, $\bar{V}(I_2), \ldots, \bar{V}(I_t)$, he still cannot learn anything about any of $S(I_1), \ldots, S(I_t)$. This way of defining security is different from the standard definition in statistical databases. There the security criteria often apply to a particular database instance, and may fail if the instance is later updated. For example, one security criterion requires that the aggregate function be computed only on cells that are "large enough." One data instance may be secure, but it becomes insecure when tuples are deleted (making some cells too small), or when tuples are inserted (creating new cells, which are small).

Dictionary-independence. The definition of query-view security $S \mid \bar{V}$ is for a particular dictionary $(D, \mathbf{P})$. In practice, however, the dictionary is often ill-defined: for example, the probability distribution $\mathbf{P}$ is impossible to compute, and even the domain D may be hard to define precisely. Thus, we are interested in a stronger version of security, which is *dictionary-independent*. Our results in the next section provide necessary and sufficient conditions for dictionary-independent security. They show, surprisingly, that, in some cases, security for *some* dictionary implies security for *all* dictionaries (see Theorem 4.8 and Proposition 4.9).

Collusions. Given $\bar{V} = V_1, \ldots, V_k$, we will show in Theorem 4.5 that $S \mid \bar{V}$ if and only if $S \mid V_i$ for all $i = 1, \ldots, k$. This has the following consequence. If we send different views to different users, but have determined that the secret query S is secure with respect to each view separately, then nothing will be leaked about S even if the recipients collude, i.e. exchange the views they received and try to learn something about S by examining all the views together. This strong property is a consequence of our adoption of a notion of *perfect secrecy* to define security. Disclosure through collusion happens when each view leaks very little information when taken separately, but together may leak a lot of information about S. We will re-examine collusion in Section 6 when we discuss measuring disclosures.

Query answering. The database community has studied extensively the following *query answering* problem. Given a set of views $\bar{V} = V_1, \ldots, V_k$ and another view V' find a function f s.t. for any instance I, $V'(I) = f(\bar{V}(I))$: in this case we say that V' is answerable from $\bar{V}$. In the related *query rewriting* problem, f is restricted to be expressed in a query language. It is natural to ask about the relationship to security. Intuitively, if V' is answerable from $\bar{V}$, then the information content of V' is not more than that of $\bar{V}$, and any query S which is secure with respect to $\bar{V}$ should be also secure with respect to V'. This intuition is correct, as the following straightforward argument shows. We have

$$\mathbf{P}\big[V'(I) = v'\big] = \sum_{\bar{v}} \big\{\mathbf{P}\big[\bar{V}(I) = \bar{v}\big] \bigm| f(\bar{v}) = v'\big\}$$

and

$$\mathbf{P}\big[S(I) = s \wedge V'(I) = v'\big] = \sum_{\bar{v}} \big\{\mathbf{P}\big[S(I) = s \wedge \bar{V}(I) = \bar{v}\big] \bigm| f(\bar{v}) = v'\big\}$$

which implies that the view V' satisfies Eq. (4). In particular, if V is a boolean view, then it follows that $S \mid V$ iff $S \mid \neg V$.

A similar result holds when security fails: if $\neg(S \mid \bar{V})$ and another query S' is computable from S, then $\neg(S' \mid \bar{V})$.

Aggregates. When applied to queries with aggregates our definition of security results in a very strict condition: no query and view containing an aggregate over a common tuple are secure. Techniques from statistical databases are better-suited for the case of queries with aggregates, and are orthogonal to our discussion. We therefore omit aggregate functions from the query language we consider.

4.2. Fundamental theorems of query-view security

At this point, the only obvious procedure for deciding query-view security is to compute probabilities for each answer to the query and view. In addition to the computational complexity of this strategy, it requires re-computation for each dictionary. In this subsection we present techniques for deciding query-view security by analyzing the query and view definitions, and prove that this technique is dictionary-independent.

Definition 4.4 *(Critical tuple).* Let D be a finite domain and Q be a query. A tuple $t \in tup(D)$ is critical for Q if there exists an instance $I \in inst(D)$ such that $Q(I - \{t\}) \neq Q(I)$. The set of critical tuples of Q is denoted $crit_D(Q)$, or simply $crit(Q)$ when D is understood from the context.

The intuition is that t is critical for Q if there exists some instance where dropping t makes a difference.

For a simple illustration, consider the boolean query $Q() :- R(a_1, x)$ and let $D = \{a_1, \ldots, a_n\}$. Any tuple of the form $R(a_1, a_i)$, $i = 1, \ldots, n$, is critical for Q, because Q returns true on the database consisting of the single tuple $R(a_1, a_i)$, but if we remove that tuple then we get the empty database on which the query returns false.

We can now formulate the characterization of query-view security. The proof is in Section 4.3.

Theorem 4.5. *Let D be a domain. Let S be a query and $\bar{V}$ be a set of views. Then $S \mid_{\mathbf{P}} \bar{V}$ for every probability distribution $\mathbf{P}$ iff $crit_D(S) \cap crit_D(\bar{V}) = \emptyset$.*

Here $crit(\bar{V})$ is $crit(V_1) \cup \cdots \cup crit(V_k)$. In particular it follows that $S \mid_{\mathbf{P}} \bar{V}$ for all $\mathbf{P}$ iff $S \mid_{\mathbf{P}} V_i$ for all $i = 1, \ldots, k$ and for all $\mathbf{P}$. The theorem says that the only way a query can be insecure with respect to some views is if they have some common critical tuple. This result translates the probabilistic definition of query-view security into a purely logical statement, which does not involve probabilities. This is important, because it allows us to reason about query-view security by using traditional techniques from database theory and finite model theory.

Next we revisit the query and view examples from the last section and apply Theorem 4.5.

Example 4.6. In Example 4.2, we saw that security fails to hold for $V(x){:}-R(x, y)$ and $S(y){:}-R(x, y)$. Every tuple is critical for V: for example, $R(a, b)$ is critical for V because $V(\{\}) = \{\}$ while $V(\{R(a, b)\}) = \{(a)\}$. Similarly, every tuple is critical for S, so because $crit(V) \cap crit(S)$ is non-empty, we conclude $\neg(S \mid_{\mathbf{P}} V)$ at least for some probability distribution $\mathbf{P}$.

Example 4.7. We argued in Example 4.3 that security holds for $V(x){:}-R(x, b)$ and $S(y){:}-R(y, a)$. The critical tuples of S are $crit(S) = \{R(a, a), R(b, a)\}$, and similarly $crit(V) = \{R(a, b), R(b, b)\}$. Because $crit(S) \cap crit(V) = \emptyset$, Theorem 4.5 allows us to conclude $S \mid_{\mathbf{P}} V$ for every probability distribution $\mathbf{P}$.

So far S and $\bar{V}$ were allowed to be arbitrary queries. We now restrict S and $\bar{V}$ to be monotone queries, and will prove that the definition of query-view security is, for all practical purposes, dictionary-independent. The main step is the following theorem, whose proof is in Section 4.3.

Theorem 4.8 *(Probability-independence). Let D be a domain, and S, $\bar{V}$ be any monotone queries. Let $\mathbf{P}_0$ be a probability distribution s.t. $\forall t$, $\mathbf{P}_0(t) \neq 0$ and $\mathbf{P}_0(t) \neq 1$. If $S \mid_{\mathbf{P}_0} \bar{V}$ then for every probability distribution $\mathbf{P}$, $S \mid_{\mathbf{P}} \bar{V}$.*

This is a surprising theoretical result, which says that if a query is secure even for one probability distribution, then it is secure for all such distributions. Continuing Example 4.2, both S and V are monotone. It follows that $\neg(S \mid_{\mathbf{P}} V)$ for any probability distribution $\mathbf{P}$ which is $\neq 0$ and $\neq 1$. Notice that for the trivial distribution $\mathbf{P}(t) = 1$, $\forall t$, we have $S \mid_{\mathbf{P}} V$, because in this case the answer to both S and V are known.

We still need to show that the definition is insensitive to a particular choice of domain, and for that we will further restrict all queries to be conjunctive queries. As we vary the domain D, we will always assume that D includes all the constants occurring in S and $\bar{V}$.

Proposition 4.9 *(Domain-independence). Let n be the largest number of variables and constants occurring in any of the conjunctive queries $S, V_1, \ldots, V_k$. If there exists a domain[3] D_0 s.t. $|D_0| \geqslant n(n+1)$, and $crit_{D_0}(S) \cap crit_{D_0}(\bar{V}) = \emptyset$, then for any domain D, s.t. $|D| \geqslant n(n+1)$, $crit_D(S) \cap crit_D(\bar{V}) = \emptyset$.*

We now discuss how to decide query-view security for conjunctive queries S and $\bar{V}$. Our goal is to check dictionary-independent security, hence we need to check whether $crit_D(S) \cap crit_D(\bar{V}) = \emptyset$, and we assume that the domain D is "large enough." The previous proposition gives us an exponential time algorithm: pick a domain D_0 with $n(n+1)$ constants, then enumerate exhaustively all instances $I \subseteq D_0$ and tuples $t \in I$, checking whether t is a critical tuple for S, and for $\bar{V}$. This also shows that the query-view security problem is in complexity class Π_2^p.[4]

[3] For conjunctive queries without order predicates it suffices to pick the domains D_0, D with size $\geqslant n$. When order predicates are allowed, then we need n fresh constants between any two constants mentioned in the queries, which leads to $n(n+1)$.

[4] Recall that NP is the class of problems that can be expressed as $\{z \mid \exists y\ \phi(y, z)\}$ where the "certificate" y has length polynomial in z and ϕ is PTIME computable. Complexity class Π_2^p consists of problems that can be expressed as $\{z \mid \forall x\ \exists y\ \phi(x, y, z)\}$ where x, y are polynomial in z and ϕ is PTIME computable.

Theorem 4.10. *The problem of deciding, for conjunctive query Q, whether a tuple $t \notin crit(Q)$ is Π_2^p-hard* (*query complexity*).

This is a non-trivial result, whose proof uses a lengthy reduction from the ∀∃3-CNF problem, and is included in Appendix A. We illustrate with an example why computing $crit(Q)$ is non-obvious. Clearly, any critical tuple t must be an homomorphic image of some subgoal of Q. But the following example shows the converse is not true:

$$Q() :- R(x, y, z, z, u), R(x, x, x, y, y).$$

Consider the tuple $t = R(a, a, b, b, c)$, which is a homomorphic image of the first subgoal. Yet t is not critical. Indeed, let I be any database s.t. $Q(I) = true$. Then the first subgoal must be mapped to t. But that means that both x and y are mapped to a. Thus the second subgoal must be mapped to the tuple $t' = R(a, a, a, a, a)$ and then $t' \in I$. Then the first subgoal can also be mapped to t', hence t is not critical.

Next, we show that deciding whether $crit(S) \cap crit(V) = \emptyset$ is at least as hard as deciding whether a tuple t is not critical for a query Q. Indeed, if we define $V = Q$, and $S() :- t$ (i.e. S simply checks for the presence of the tuple t), then $t \notin crit(Q)$ iff $crit(S) \cap crit(V) = \emptyset$. For the former we have claimed that it is Π_2^p-hard. In summary:

Theorem 4.11. *The problem of deciding whether a conjunctive query S is secure with respect to a set of conjunctive views $V_1, \ldots, V_k$ is Π_2^p-complete* (*query complexity*).

A practical algorithm. For practical purposes, one can check $crit(S) \cap crit(\bar{V}) = \emptyset$ and hence $S \mid \bar{V}$ quite efficiently. Simply compare all pairs of subgoals from S and from $\bar{V}$. If any pair of subgoals unify, then $\neg S \mid \bar{V}$. While false positives are possible, they are rare: this simple algorithm would correctly classify all examples in this paper.

4.3. Proof of the fundamental theorems

The technique used in the proof of Theorems 4.5 and 4.8 is of independent interest, and we present it here. Throughout this subsection we will fix the domain D and denote the set of tuples with $tup(D) = \{t_1, \ldots, t_n\}$. Recall our notation from Section 3: $x_1 = \mathbf{P}[t_1], \ldots, x_n = \mathbf{P}[t_n]$. Hence, a probability distribution $\mathbf{P}$ is given by a set of numbers $\bar{x} \in [0, 1]^n$.

The boolean case, single view. We first prove both theorems for the case of boolean queries; moreover, we will consider a single view, rather than a set of views. Given a boolean query Q, we denote by $\mathbf{P}[Q]$ the probability that Q is true on a randomly chosen database instance. Recall from Eqs. (1) and (2) that this probability is given by:

$$\mathbf{P}[Q] = \sum_{\{I \in inst(D) \mid Q(I) = true\}} \mathbf{P}[I],$$
$$\mathbf{P}[I] = \prod_{t_i \in I} x_i \cdot \prod_{t_j \notin I} (1 - x_j). \tag{5}$$

Therefore $\mathbf{P}[Q]$ is given by a polynomial in the variables $x_1, \ldots, x_n$, which we denote $f_Q(x_1, \ldots, x_n)$ or $f_Q(\bar{x})$.

Example 4.12. Let $D = \{a, b\}$, and consider the boolean query:

$$Q() :- R(a, x), R(x, x).$$

In this case $tup(D) = \{t_1, t_2, t_3, t_4\}$, where $t_1 = R(a, a)$, $t_2 = R(a, b)$, $t_3 = R(b, a)$, and $t_4 = R(b, b)$. Then Q can be written as the following DNF formula:

$$Q = t_1 \vee (t_2 \wedge t_4).$$

To compute f_Q one enumerates all 16 database instances $I \subseteq tup(D)$. Q is true on 12 of them: $\{t_2, t_4\}, \{t_2, t_3, t_4\}, \ldots.$ For each of them we apply Eq. (5). This results in a sum of 12 expressions:

$$f_Q = (1 - x_1)x_2(1 - x_3)x_4 + (1 - x_1)x_2x_3x_4 + \cdots.$$

After simplification we obtain: $f_Q = x_1 + x_2x_4 - x_1x_2x_4$. Let $Q' :- R(b, a)$ (so that $f_{Q'} = x_3$), and consider the boolean formula $Q \wedge Q'$. The polynomial $f_{Q \wedge Q'}$ is equal to $f_Q \times f_{Q'}$, i.e. $(x_1 + x_2x_4 - x_1x_2x_4)x_3$ because Q and Q' depend on disjoint sets of tuples.

Before we prove the two theorems we notice that query-view security for boolean queries can be restated as follows. Given boolean queries S and V, $S\,|_{\mathbf{P}}\,V$ iff:

$$f_{S\wedge V}(\bar{x}) = f_S(\bar{x}) \times f_V(\bar{x}) \qquad (6)$$

where $\bar{x}$ corresponds to $\mathbf{P}$. Indeed, this represents precisely Eq. (4) for one specific choice of s and v, namely $s = \mathit{true}$ and $v = \mathit{true}$. One can show that if Eq. (4) holds for $(\mathit{true}, \mathit{true})$, then it also holds for the other three combinations, $(\mathit{false}, \mathit{true})$, $(\mathit{true}, \mathit{false})$, $(\mathit{false}, \mathit{false})$. Thus, $S\,|_{\mathbf{P}}\,V$ holds precisely if (6) holds.

We now restate the two theorems for the boolean case:

Theorem 4.5. *Let D be a domain, S a query, and V a view. Then: $\forall \bar{x} \in [0,1]^n . f_{S\wedge V}(\bar{x}) = f_S(\bar{x}) \times f_V(\bar{x})$ iff $\mathit{crit}_D(S) \cap \mathit{crit}_D(V) = \emptyset$.*

Theorem 4.8. *Let D be a domain. If S and V are monotone boolean queries, then: $\exists \bar{x} \in (0,1)^n . f_{S\wedge V}(\bar{x}) = f_S(\bar{x}) \times f_V(\bar{x})$ implies $\forall \bar{x} \in [0,1]^n . f_{S\wedge V}(\bar{x}) = f_S(\bar{x}) \times f_V(\bar{x})$.*

The crux of the proof relies on a close examination of the polynomials f_Q. The properties we need are summarized below. Their proofs are straightforward and are omitted:

Proposition 4.13. *Let $f_Q = \mathbf{P}[Q]$, where Q is a boolean formula in $t_1, \ldots, t_n$. Then f_Q is a polynomial in the variables $x_1, \ldots, x_n$ with the following properties:*

(1) *For each $i = 1, \ldots, n$, the degree of x_i is $\leqslant 1$.*
(2) *For each $i = 1, \ldots, n$, the degree of x_i is 1 iff $t_i \in \mathit{crit}_D(Q)$. (In Example 4.12, $\mathit{crit}_D(Q) = \{t_1, t_2, t_4\}$ and indeed x_1, x_2, x_4 have degree 1, while x_3 has degree 0.)*
(3) *If $\mathit{crit}_D(Q_1) \cap \mathit{crit}_D(Q_2) = \emptyset$ then $f_{Q_1 \wedge Q_2} = f_{Q_1} \times f_{Q_2}$.*
(4) *Choose values in $[0,1]^{n-1}$ for all variables except for one, x_i: f_Q becomes a polynomial of degree $\leqslant 1$ in x_i. Then, if Q is a monotone boolean formula, the coefficient of x_i is $\geqslant 0$. In Example 4.12, the coefficient of x_4 in f_Q is $x_2 - x_1 x_2$, which is always $\geqslant 0$ when $x_1, x_2 \in [0,1]^2$.*
(5) *Let Q_0 be the boolean formula obtained from Q by setting $t_n = \mathit{false}$, and Q_1 be the boolean formula obtained by setting $t_n = \mathit{true}$. Then $f_{Q_0} = f_Q[x_n = 0]$ and $f_{Q_1} = f_Q[x_n = 1]$. In Example 4.12, $Q_0 = t_1$ and $f_Q[x_4 = 0] = x_1$; similarly $Q_1 = t_1 \vee t_2$ and $f_Q[x_4 = 1] = x_1 + x_2 - x_1 x_2$.*

We prove now Theorem 4.5 for the boolean case.

Proof. Assume first that $\mathit{crit}_D(S) \cap \mathit{crit}_D(V) = \emptyset$. Then $f_{S\wedge V} = f_S \times f_V$, by Proposition 4.13, item (3). Assume now that $\forall \bar{x} \in [0,1]^n . f_{S\wedge V}(\bar{x}) = f_S(\bar{x}) \times f_V(\bar{x})$ holds. Then the polynomials $f_{S\wedge V}$ and $f_S \times f_V$ must be identical. In particular, f_S and f_V cannot have a common variable x_i, otherwise its degree would be 2. Hence $\mathit{crit}_D(S)$ and $\mathit{crit}_D(V)$ cannot have a common tuple (by Proposition 4.13 item (2)). □

Next we prove Theorem 4.8 for the boolean case.

Proof. Consider the polynomial $g_{S,V} = f_{S\wedge V} - f_S \times f_V$. We show by induction on the number n of tuples in $\mathit{tup}(D)$ that $\forall \bar{x} \in [0,1]^n$, $g_{S,V}(\bar{x}) \geqslant 0$. It holds trivially for $n = 0$. For $n > 0$, $g_{S,V}$ is a polynomial of degree $\leqslant 2$ in x_n, and the coefficient of x_n^2 is negative: this follows from Proposition 4.13 item (4) and the fact that S, V are monotone. For $x_n = 0$, the polynomial in $n-1$ variables $g_{S,V}[x_n = 0]$ corresponds to the boolean formulas $S[t_n = \mathit{false}]$, $V[t_n = \mathit{false}]$ (item (5) of the proposition), hence we can apply the induction hypothesis and obtain that $g_{S,V} \geqslant 0$ for $x_n = 0$. Similarly, $g_{S,V} \geqslant 0$ for $x_n = 1$, since now it corresponds to the boolean formulas $S[t_n = \mathit{true}]$, $V[t_n = \mathit{true}]$. Furthermore, since $g_{S,V}$ has degree $\leqslant 2$ and the coefficient of x_n^2 is $\leqslant 0$, it follows that $g_{S,V} \geqslant 0$ for every $x_n \in [0,1]$. This completes the inductive proof. Now assume that for some $\bar{x} \in (0,1)^n$, $g_{S,V}(\bar{x}) = 0$. We will prove that $\mathit{crit}_D(S) \cap \mathit{crit}_D(V) = \emptyset$. Assume by contradiction that $t_i \in \mathit{crit}_D(S) \cap \mathit{crit}_D(V)$ for some tuple t_i. Then $g_{S,V}$ is a polynomial of degree 2 in x_i, with a negative coefficient for x_i^2, which has at least one root in $(0,1)$. It follows that $g_{S,V}$ must be < 0 either in $x_i = 0$, or in $x_i = 1$, contradicting the fact that $g_{S,V} \geqslant 0$ for all $\bar{x} \in [0,1]^n$. □

The boolean case, multiple views. Let $\bar{V} = V_1, \dots, V_n$ be n boolean views. The next step is to show that $S \mid_{\mathbf{P}} \bar{V}$ for all $\mathbf{P}$ iff $S \mid_{\mathbf{P}} V_i$ for all $i = 1, \dots, n$ and all $\mathbf{P}$. For simplicity we prove this for $n = 2$.

Proof. For the 'only if' direction we prove Eq. (4) directly. To show $\mathbf{P}[S(I) = s \wedge V_2(I) = v_2] = \mathbf{P}[S(I) = s] \times \mathbf{P}[V_2(I) = v_2]$ we notice:

$$\mathbf{P}\big[S(I) = s \wedge V_2(I) = v_2\big] = \sum_{v_1} \mathbf{P}\big[S(I) = s \wedge V_1(I) = v_1 \wedge V_2(I) = v_2\big],$$

$$\mathbf{P}\big[V_2(I) = v_2\big] = \sum_{v_1} \mathbf{P}\big[V_1(I) = v_1 \wedge V_2(I) = v_2\big]$$

then we use the fact that $S \mid_{\mathbf{P}} (V_1, V_2)$ to complete the argument. For the 'if' direction, we need to check $\mathbf{P}[S(I) = s \wedge V_1(I) = v_1 \wedge V_2(I) = v_2] = \mathbf{P}[S(I) = s] \times \mathbf{P}[V_1(I) = v_1 \wedge V_2(I) = v_2]$. Using Theorem 4.5 for the boolean, single view case, it suffices to check $crit_D(S) \cap crit_D(V) = \emptyset$ where $V(I)$ is the boolean query $V_1(I) = v_1 \wedge V_2(I) = v_2$. This follows from $crit_D(V) \subseteq crit_D(V_1) \cup crit_D(V_2)$, and the assumption, $crit_D(S) \cap crit_D(V_i) = \emptyset$ for $i = 1, 2$. □

The non-boolean case. We now generalize to non-boolean queries. Given a k-ary query Q, let $t_1, \dots, t_m$ be all k-tuples over the domain D ($m = |D|^k$). For each $i = 1, \dots, m$, define Q_i^b the boolean query $Q_i^b(I) = (t_i \in Q(I))$; that is, it checks whether t_i is in Q. Notice that $crit_D(Q) = \bigcup_i crit_D(Q_i^b)$, and if Q is monotone then Q_i^b is monotone for $i = 1, \dots, m$.

Given a domain D and probability distribution, the following is easy to check, by applying directly Definition 4.1. For any query S and views $\bar{V} = V_1, \dots, V_k$:

$$S \mid_{\mathbf{P}} \bar{V} \quad \text{iff} \quad \forall i, j, l. S_i^b \mid_{\mathbf{P}} V_{j,l}^b.$$

Here $V_{j,l}^b$ denotes $(V_j)_l^b$. This immediately reduces both theorems to the boolean case.

5. Modeling prior knowledge

So far we have assumed that the adversary has no knowledge about the data other than the domain D and the probability distribution $\mathbf{P}$ provided by the dictionary. Next we consider security in the presence of prior knowledge, which we denote with K. Our standard for security compares Mallory's knowledge about the secret query S before and after publishing the views $\bar{V}$, but always assuming he knows K. In the most general case K is any boolean statement on the database instance I. For example, it can be a key or foreign-key constraint, some previously published views, or some general knowledge about the domain. K is thus any boolean predicate on the instance I, and we write $K(I)$ whenever I satisfies K. To avoid introducing new terminology, we will continue to call K a *boolean query*. We do not however restrict K by requiring that it be expressed in a particular query language.

5.1. Definition and main theorem

As before we assume domain D to be fixed. K is a boolean query, while S and $\bar{V}$ are arbitrary queries.

Definition 5.1 *(Prior knowledge security).* Let $\mathbf{P}$ be a probability distribution on the tuples. We say that S is secure with respect to $\bar{V}$ under prior knowledge K if for every $s, \bar{v}$:

$$\mathbf{P}\big[S(I) = s \mid K(I)\big] = \mathbf{P}\big[S(I) = s \mid \bar{V}(I) = \bar{v} \wedge K(I)\big].$$

We denote prior knowledge security by $K : S \mid_{\mathbf{P}} \bar{V}$.

Applying Bayes' theorem reduces the above to:

$$\mathbf{P}\big[S(I) = s \wedge \bar{V}(I) = \bar{v} \wedge K(I)\big] \times \mathbf{P}\big[K(I)\big] = \mathbf{P}\big[S(I) = s \wedge K(I)\big] \times \mathbf{P}\big[\bar{V}(I) = \bar{v} \wedge K(I)\big]. \tag{7}$$

Both the *prior knowledge* and the *relative security* applications mentioned in Section 1 are modeled as a security problem with prior knowledge. In the case of relative security, we take K to be the knowledge that the prior view has some given answer.

Theorem 4.5, which showed query-view security is equivalent to disjointness of the critical tuples, can be generalized for security with prior knowledge. We state the theorem for the boolean case, and will discuss specific generalizations to non-boolean queries and views. The proof is deferred to Section 5.3, after a discussion of some applications of the theorem.

Theorem 5.2. *Let D be a domain, $T = tup(D)$, and K, S, V be arbitrary boolean queries. Then $K : S \mid_{\mathbf{P}} V$ for all probability distributions $\mathbf{P}$ iff the following holds*:

COND-K. *There exists sets of tuples T_1, T_2 and boolean queries K_1, K_2, V_1, S_2 s.t.*:

$$\begin{aligned}
&T_1 \cap T_2 = \emptyset,\\
&K = K_1 \wedge K_2,\\
&S \wedge K = K_1 \wedge S_2,\\
&V \wedge K = V_1 \wedge K_2,\\
&crit_D(K_1) \subseteq T_1, \qquad crit_D(K_2) \subseteq T_2,\\
&crit_D(V_1) \subseteq T_1, \qquad crit_D(S_2) \subseteq T_2.
\end{aligned}$$

Informally, the theorem says that the space of tuples can be partitioned into T_1 and T_2 such that property K is the conjunction of two independent properties, K_1 over T_1 and K_2 over T_2. In addition, assuming K holds, S just says something about the tuples in T_2 (and nothing more about T_1). Similarly, when K holds, V just says something about T_1 (and nothing more about T_2).

By itself, this theorem does not result in a practical decision procedure, because it is too general. We show, however, how it can be applied to specific applications, and in particular derive decision procedures.

5.2. Applying prior knowledge

Application 1: No prior knowledge. As a baseline check, let us see what happens if there is no prior knowledge. Then $K = true$ and condition **COND-K** says that there are two disjoint sets of tuples T_1 and T_2 such that $crit_D(S) \subseteq T_2$ and $crit_D(V) \subseteq T_1$. This is equivalent to saying $crit_D(S) \cap crit_D(V) = \emptyset$, thus we recover Theorem 4.5 for boolean queries.

Application 2: Keys and foreign keys. The notion of query-view secrecy is affected by keys and foreign-keys constraints K. For an illustration, consider the boolean query: $S() : -R(a, b)$, and the boolean view $V() : -R(a, c)$. Here a, b, c are distinct constants. We have $S \mid_{\mathbf{P}} V$ for any $\mathbf{P}$, because $crit_D(S) = \{R(a, b)\}$ and $crit_D(V) = \{R(a, c)\}$ are disjoint. But now suppose that the first attribute of R is a key. Then by knowing V we know immediately that S is false, which is a total information disclosure, hence $K : S \mid V$ does not hold.

We apply now Theorem 5.2 to derive a general criterion for query-view secrecy in the presence of key constraints K. Given a domain D, define the following equivalence relation on $tup(D)$: $t \equiv_K t'$ if t and t' are tuples over the same relation, and they have the same key. In the example above, we have $R(a, b) \equiv_K R(a, c)$, and $R(a, b) \not\equiv_K R(d, b)$ for a new constant d. Given a query Q, denote $crit_D(Q, K)$ the set of tuples t s.t. there exists a database instance I that satisfies the key constraints K and $Q(I) \neq Q(I - \{t\})$. The following criterion can be proven from Theorem 5.2 and shows how to check $K : S \mid \bar{V}$.

Corollary 5.3. *Let K be a set of key constraints, D a domain, and S, $\bar{V}$ conjunctive queries. Then $S \mid_{\mathbf{P}} \bar{V}$ for any $\mathbf{P}$ iff $\forall t \in crit_D(S, K)$, $\forall t' \in crit_D(\bar{V}, K)$, $t \not\equiv_K t'$. In particular, the problem whether $K : S \mid_{\mathbf{P}} V$ for all $\mathbf{P}$ is decidable, and Π_2^p-complete.*

As a simple illustration, in the previous example, we have $crit_D(S, K) = \{R(a, b)\}$, $crit_D(V, K) = \{R(a, c)\}$, and $R(a, b) \equiv_K R(a, c)$, hence it is not the case that $K : S \mid_{\mathbf{P}} V$ for all $\mathbf{P}$. Foreign keys can be handled similarly, however the corresponding decidability and complexity result holds only when the foreign keys introduce no cycles.

Proof *(sketch).* We give the proof for the boolean case only: the generalization to non-boolean queries is straightforward. The proof relies on two observations. Let $U_1, U_2, \ldots$ be the $\equiv_K$ equivalence classes on $tup(D)$. For each U_i, denote L_i the predicate saying "at most one tuple from U_i can be in the instance I." For example, if $U_i = \{t_1, t_2, t_3\}$ set

$$L_i = (\neg t_1 \wedge \neg t_2 \neg t_3) \vee (t_1 \wedge \neg t_2 \wedge \neg t_3) \vee (\neg t_1 \wedge t_2 \wedge \neg t_3) \vee (\neg t_1 \wedge \neg t_2 \wedge t_3).$$

Then K is $L_1 \wedge L_2 \wedge \cdots$. The first observation is that whenever we split K into $K_1 \wedge K_2$ as in **COND-K** each of the two subexpressions must be a conjunction of some L_i's. It follows that every equivalence class U_i intersects either $crit_D(K_1)$ or $crit_D(K_2)$ but not both. The second observation is that there exists S_2 such that $crit_D(S_2) \subseteq T_2$ and $S \wedge K = K_1 \wedge S_2$ iff $crit_D(S, K) \subseteq T_2$. Indeed, in one direction, we notice first that if I satisfies K, then $S(I) = K_1(I \cap T_1) \wedge S_2(I \cap T_2) = S_2(I \cap T_2)$. Then for any $t \in crit_D(S, K)$, there exists I satisfying K s.t. $S(I) \neq S(I - \{t\})$, hence $S_2(I \cap T_2) \neq S_2((I - \{t\}) \cap T_2)$, which implies $t \in T_2$. For the other direction, define $S_2(I) = K_2(I) \wedge S(I \cap T_2)$ (obviously $crit_D(S_2) \subseteq T_2$) and let us show that $S \wedge K = K_1 \wedge S_2$. Let I be an instance s.t. $(S \wedge K)(I)$ is true; in particular I satisfies K, hence $crit_D(S, K) \subseteq T_2$ which means $S(I) = S(I \cap T_2)$. The claim follows from the fact that $K_2(I)$ is true. With these two observations the proof of the corollary is straightforward and omitted. The decidability and complexity is shown with an argument similar to that used in Theorem 4.11. □

Application 3: Cardinality constraint. What happens if Mallory has some partial knowledge about the cardinality of the secret database? This is quite common in practice. For example, the number of patients in a hospital is likely to be between 100 or 1000, but not 2 and not 1,000,000. In this case K is a cardinality constraint, such as "there are exactly n tuples in I" or "there are at most n tuples" or "at least n tuples." Surprisingly, there are no secure queries when the prior knowledge involves any cardinality constraints! This follows from Theorem 5.2 since K cannot be expressed as $K_1 \wedge K_2$ over disjoint sets of tuples, by a simple counting argument, except for the trivial case when $T_1 = \emptyset$ or $T_2 = \emptyset$. Hence, no query is perfectly secret with respect to any view in this case, except if one of them (S or V) is trivially true or false.

Application 4: Protecting secrets with knowledge. Sometimes prior knowledge can protect secrets! Take any queries S, $\bar{V}$, and assume that S is not secure with respect to $\bar{V}$. Suppose now that we disclose publicly the status of every tuple in $crit_D(S) \cap crit_D(\bar{V})$. That is, for each common critical tuple t we announce whether $t \in I$ or $t \notin I$. If we denote with K this knowledge about all common critical tuples, then Theorem 5.2 implies that $K : S \,|_{\mathbf{P}}\, \bar{V}$ for any $\mathbf{P}$, as we show below. For a simple illustration, assume $S()\colon -R(a, -)$ and $V()\colon -R(-, b)$. They are not secure because $crit_D(S) \cap crit_D(V) = \{R(a, b)\}$. But now suppose we disclose that the pair (a, b) is not in the database, $R(a, b) \notin I$, and call this knowledge K. Then $K : S \,|_{\mathbf{P}}\, V$. The same is true if we publicly announce that $R(a, b)$ *is* in the database instance. We prove this formally next:

Corollary 5.4. *Let K be such that $\forall t \in crit_D(S) \cap crit_D(\bar{V})$, either $K \models t \in I$, or $K \models t \notin I$. Then, for every* $\mathbf{P}$, *$K : S \,|_{\mathbf{P}}\, \bar{V}$.*

Proof. We will prove this for two boolean queries S, V only: the general case follows easily. Let $T_1 = crit_D(S) \cap crit_D(V)$, and $T_2 = tup(D) - T_1$. Let $K_1 = K$, $K_2 = true$, $S_2 = S$, $V_1 = V \wedge K$. Then the conditions of Theorem 5.2 are satisfied, hence $K : S \,|_{\mathbf{P}}\, V$ for any $\mathbf{P}$. □

Application 5: Prior views. Suppose Alice already published a view U (there may have been leakage about S, but she decided the risk was acceptable). Now she wants to publish another view V, and she wonders: will I leak any *more* information about S?

Using Theorem 5.2 we give below a decision procedure for the case of conjunctive queries, but only when U is a boolean query. This is a limitation, and due to the fact that both sides of the formula (7) are linear in S and $\bar{V}$, but not in K: this made it possible to generalize statements from boolean queries S, V to arbitrary ones, but not for K. To simplify the statement, we also restrict S and V to be boolean: these, however, can be generalized to arbitrary conjunctive queries.

Corollary 5.5. *Let U, S, V be boolean conjunctive queries. Then $U : S \,|_{\mathbf{P}}\, V$ for every probability distribution $\mathbf{P}$ iff each of the queries can be split as follows*:

$$U = U_1 \wedge U_2,$$
$$S = S_1 \wedge S_2,$$
$$V = V_1 \wedge V_2.$$

Such that the sets $crit_D(U_1) \cup crit_D(S_1) \cup crit_D(V_1)$ and $crit_D(V_2) \cup crit_D(S_2) \cup crit_D(V_2)$ are disjoint, and $U_1 \Rightarrow S_1$ and $U_2 \Rightarrow V_2$. Hence, $U : S \,|_{\mathbf{P}}\, V$ is decidable.

The proof follows rather directly from Theorem 5.2 and is omitted. For a simple illustration consider:

$$U{:}-R_1(a, b, -, -), R_2(d, e, -, -),$$
$$S{:}-R_1(a, -, -, -), R_2(d, e, f, -),$$
$$V{:}-R_1(a, b, c, -), R_2(d, -, -, -).$$

Here S is not secure with respect to either U or V. However, $U : S \mid V$. By giving out U we already disclosed something about S, namely $R_1(a, -, -, -)$. By publishing V in addition we do not further disclose any information.

5.3. *Proof of Theorem 5.2*

Proof *(sketch).* For boolean queries, $K : S \,|_{\mathbf{P}}\, V$ can be expressed as follows:

$$\mathbf{P}[S \wedge V \wedge K] \times \mathbf{P}[K] = \mathbf{P}[S \wedge K] \times \mathbf{P}[V \wedge K].$$

Using the notation f_Q for a boolean query Q (see Section 4.3), this becomes:

$$f_{S\wedge V\wedge K}(\bar{x}) \times f_K(\bar{x}) = f_{S\wedge K}(\bar{x}) \times f_{V\wedge K}(\bar{x}). \tag{8}$$

We need to prove that (8) holds for any $\bar{x} \in [0, 1]^n$ iff **COND-K** holds. For that we need the properties of f_Q in Proposition 4.13 plus three more. Call any multi-variable polynomial $g(\bar{x})$ of degree $\leqslant 1$ in each variable a *boolean* polynomial if $\forall \bar{x} \in \{0, 1\}^n$, $g(\bar{x})$ is either 0 or 1. Clearly, any polynomial f_Q is a boolean polynomial.

Proposition 5.6.

(1) *If g is a boolean polynomial then there exists a unique boolean formula Q s.t. $g = f_Q$.*
(2) *Let Q be a boolean formula, and suppose f_Q is the product of two polynomials $f_Q = g \times h$. Then there exists a constant $c \neq 0$ s.t. both cg and $\frac{1}{c}h$ are boolean polynomials.*
(3) *If $f_Q = f_{Q_1} \times f_{Q_2}$ then $crit_D(Q_1) \cap crit_D(Q_2) = \emptyset$.*

We can now prove the equivalence of (8) to **COND-K**. Assume (8) holds for every $\bar{x} \in [0, 1]^n$, i.e. this is an identity of polynomials. Then f_K divides $f_{S\wedge K} \times f_{V\wedge K}$. Hence $f_K = g \times h$ where g divides $f_{S\wedge K}$ and h divides $f_{V\wedge K}$. By Proposition 5.6 we can assume that g, h are boolean, hence $f_K = f_{K_1} \times f_{K_2}$ for some boolean formulas K_1, K_2, and moreover we have $K = K_1 \wedge K_2$ and $crit_D(K_1) \cap crit_D(K_2) = \emptyset$. Since f_{K_1} divides $f_{S\wedge K}$, we can write the latter as $f_{S\wedge K} = f_{K_1} \times f_{S_2}$, for some boolean query S_2, which implies $S \wedge K = K_1 \wedge S_2$. Similarly, f_{K_2} divides $f_{V\wedge K}$, hence we can write the latter as $f_{V\wedge K} = f_{V_1} \times f_{K_2}$ for some query V_1. Finally, substituting in (8) and simplifying with $f_{K_1} \times f_{K_2}$ we get $f_{S\wedge V\wedge K} = f_{V_1} \times f_{S_2}$. It follows that f_{V_1} and f_{S_2} have no common variables, hence $crit_D(V_1) \cap crit_D(S_2) = \emptyset$. Define $T_1 = crit_D(K_1) \cup crit_D(V_1)$ and $T_2 = tup(D) - T_1$. Then it follows that $crit_D(K_2) \subseteq T_2$ and $crit_D(S_2) \subseteq T_2$, completing the proof of **COND-K**.

For the other direction, assume **COND-K** is satisfied and let us prove (8). We have:

$$f_{S\wedge V\wedge K} = f_{(K_1\wedge V_1)\wedge(K_2\wedge S_2)} = f_{K_1\wedge V_1} \times f_{K_2\wedge S_2},$$
$$f_K = f_{K_1} \times f_{K_2},$$
$$f_{S\wedge K} = f_{K_1} \times f_{S_2\wedge K_2},$$
$$f_{V\wedge K} = f_{V_1\wedge K_1} \times f_{K_2}$$

and (8) follows immediately. □

5.4. Encrypted views

Encryption is increasingly being used to protect both published data and data stored in the DBMS. In many scenarios [2,17,21], data is encrypted at the attribute level, and analyzing the disclosure of these somewhat unusual "views" is an important open research question.

Encrypted views can be modeled in our framework. One way is to model an encrypted view of a relation as the result of applying a perfect one-way function f to each attribute. Because our goal is to study the logical security of a view, we assume idealized properties of the encryption primitive: namely that given $f(x)$ it is impossible to recover x, and that f is collision free. Under these assumptions an encrypted view is essentially an isomorphic copy of the original relation. Clearly, such a view provides information that can be used to answer queries. For example, $Q_1() :- R(x, y), R(y, z), x \neq z$, is answerable using such a view. Query $Q_2() :- R(a, x)$ is not answerable using the view, but substantial information is nevertheless leaked.

The encrypted view reveals the cardinality of the relation R. It follows from this fact and results in Section 4.1.1 on query-view security and answerability that no secret query S is secure with respect to an encrypted view V. However, the definition of leakage given in Section 6 can be applied to encrypted views too, and can be used to distinguish between cases where the information disclosure is minute and cases where leakage is substantial.

6. Relaxing the definition of security

Our standard for query-view security is very strong. It classifies as insecure query-view pairs that are considered secure in practice. In many applications we can tolerate deviations from this strong standard, as long as the deviations are not too large. We discuss briefly two directions for a more practical definition of security. The first strategy is a numerical measure of information disclosure, and the second, based on [7], uses a substantially different assumption of database probabilities which effectively ignores certain minute disclosures.

6.1. Measuring disclosures

We discuss here a measure of information disclosure that attempts to quantify the amount by which a query and view depart from our definition of security. Ours is one possible choice of measure; others are definitely possible. The main objective is to show that the theoretical concepts and results presented in this work can be employed to evaluate information disclosure in practical settings. We restrict our discussion to the case of no prior knowledge.

We will define a measure of positive information disclosure. This is easier to analyze, and far more important in practice than negative information disclosure. An example of the former is whether "John Johnson" has "cardiovascular disease"; and example of the latter is whether "John Johnson" does not have "cardiovascular disease." We will restrict the queries S and $\bar{V}$ to be monotone queries, and will study atomic statements given by inclusions $s \subseteq S(I)$ and $\bar{v} \subseteq \bar{V}(I)$, which are monotone in I.

Our definition of leakage is the following:

$$leak(S, \bar{V}) = \sup_{s,\bar{v}} \frac{\mathbf{P}[s \subseteq S(I) \mid \bar{v} \subseteq \bar{V}] - \mathbf{P}[s \subseteq S(I)]}{\mathbf{P}[s \subseteq S(I)]}. \tag{9}$$

The goal of a user wishing to publish $\bar{V}$ while not disclosing S is to ensure that $leak(S, \bar{V}) \ll 1$. This will ensure that $\mathbf{P}[s \subseteq S(I)]$ can increase only very little after publishing the view, giving Mallory a negligible amount of positive information. $S \mid_{\mathbf{P}} \bar{V}$ iff $leak(S, \bar{V}) = 0$.

For a given $s, \bar{v}$, denote $S_s(I)$ and $V_{\bar{v}}$ the boolean queries $s \subseteq S(I)$ and $\bar{v} \subseteq \bar{V}(I)$. Let $T_{s,\bar{v}} = crit_D(S_s) \cap crit_D(V_{\bar{v}})$. We know from Theorem 4.5 that, when $T_{s,\bar{v}} = \emptyset$, then the difference in Eq. (9) is 0 for this pair of s and $\bar{v}$. Our analysis of the leakage is based on the probability of I having some tuple in $T_{s,\bar{v}}$. Let us denote $L_{s,\bar{v}}(I)$ the predicate $I \cap T_{s,\bar{v}} \neq \emptyset$, and $K_{s,\bar{v}} = \neg L_{s,\bar{v}}$. Then by Corollary 5.4, $K : S_s \mid_{\mathbf{P}} V_{\bar{v}}$, and we can prove:

Theorem 6.1. *Suppose that there exists some $\varepsilon < 1$ such that for all s and for all $\bar{v}$:*

$$\mathbf{P}\big[L_{s,\bar{v}}(I) \bigm| S_s(I) \wedge V_{\bar{v}}(I)\big] < \varepsilon.$$

Then:

$$leak(S, \bar{V}) \leqslant \frac{\varepsilon^2}{1-\varepsilon^2}.$$

Example 6.2 *(Minute leakage).* We will illustrate with the example in Table 1. We abbreviate here with $E(n, d, p)$ the table Employee(*name*, *department*, *phone*). Consider the view $V(d){:}-E(n, d, p)$ and the query $S(n, p){:}-E(n, d, p)$ (it corresponds to S_2 in the table). To simplify the discussion we consider the case when s and v consists of a single tuple. First, $s = (d_{name}, d_{phone})$ is a secret name–phone pair, and $v = (d_{dept})$ is some department that we publish. The boolean queries are $V_v(I) = (v \in V(I))$ and $S_s(I) = (s \in S(I))$. We have $crit_D(V_v) = \{(-, d_{dept}, -)\}$, $crit_D(S_s) = \{(d_{name}, -, d_{phone})\}$, and $T_{s,v} = \{(d_{name}, d_{dept}, d_{phone})\}$. The conditional probability $\mathbf{P}[L \mid S_s \wedge V_v]$ is in this case $\mathbf{P}[L \wedge S_s \wedge V_v]/\mathbf{P}[S_s \wedge V_v] = \mathbf{P}[L]/\mathbf{P}[S_s \wedge V_v]$. This corresponds to ε in Theorem 6.1. For example, assuming a uniform probability distribution $\mathbf{P}[t] = p$ for all $t \in tup(D)$, and denoting n_1, n_2, n_3 the number of names, departments, and phone numbers in the dictionary, we have $\mathbf{P}[L] = p$ and $\mathbf{P}[S_s \wedge V_v] = p + (1-p)(1-(1-p)^{n^2-1}) \times (1-(1-p)^{n_1 n_3 - 1}) \approx p^2 n_1 n_2 n_3$. It follows that $\mathbf{P}[L \mid S_s \wedge V_v] \approx 1/(pn_1 n_2 n_3) = 1/m$, where m is the expected cardinality of an instance I. It can be shown that $\mathbf{P}[L \mid S_s \wedge V_v]$ is small for *any sets* s, v, implying that ε in Theorem 6.1 is small. Hence the information disclosure about S by publishing V is minute. Example (3) in Table 1 can be explained similarly.

Example 6.3 *(Collusions).* Continuing the example, consider now the view $V(n, d){:}-E(n, d, p)$ and the query $S(n, p){:}-E(n, d, p)$ (they correspond to S_2, V_2 in the table). We will still restrict s and v to a single tuple. The interesting case here is when $s = (d_{name}, d_{phone})$ and $v = (d_{name}, d_{dept})$. As before we obtain $T_{s,v} = \{(d_{name}, d_{dept}, d_{phone})\}$. The conditional probability $\mathbf{P}[L \mid S_s \wedge V_v] = \mathbf{P}[L]/\mathbf{P}[S_s \wedge V_v]$ is slightly larger than in the previous example, because $\mathbf{P}[S_s \wedge V_v]$ has decreased. The ε in Theorem 6.1 increases, suggesting more information disclosure. This is to be expected, since now the view discloses information about the names in the secret query.

Consider now the effect of the collusion between the view V and the view $V'(d, p){:}-E(n, d, p)$ (this is V'_2 in Table 1). The interesting case to consider here is $s = (d_{name}, d_{phone})$, $v = (d_{name}, d_{dept})$, $v' = (d_{dept}, d_{phone})$. We still have $T_{s,v,v'} = \{(d_{name}, d_{dept}, d_{phone})\}$. Now, $\mathbf{P}[L \mid S_s \wedge V_v \wedge V'_{v'}] = \mathbf{P}[L]/\mathbf{P}[S_s \wedge V_v \wedge V'_{v'}]$ is even smaller, because $\mathbf{P}[S_s \wedge V_v \wedge V'_{v'}]$ is smaller. The amount of leakage given by Theorem 6.1 is now larger.

6.2. Subsequent work on practical query-view security

Following the original publication of this work [22], the authors, along with Nilesh Dalvi, analyzed query-view security under a substantially different probabilistic model which can permit a relaxed notion of security termed *practical* security. For comparison purposes, we provide here a brief overview of the setting and main results for this approach, referring the reader to [7] for a full treatment of the topic.

To capture practical query-view security we adopt a new probability distribution over databases. In this model, individual tuples have a uniform probability of occurring in the database, but the probability of each tuple t is now such that the expected size of the relation instance R is a given constant S (different constants may be used for different relation names). As the domain size n grows to ∞, the expected database size remains constant. Hence, in the case of directed graphs (i.e. a single, binary relation R), the probability that two given nodes are connected by an edge is S/n^2. Denoting by $\mu_n[Q]$ the probability that a boolean query Q is true on a domain of size n, our goal is to compute $\mu_n[Q \mid V]$ as $n \to \infty$.

We propose as a definition of practical security $\lim_n \mu_n[Q \mid V] = 0$. This is justified as follows. The adversary faces a large domain. For example, if he is trying to guess whether "*John Smith*" is an employee, then he has only a tiny probability of success: $1/n$ where n is the size of the domain. On the other hand, the size of the database is much smaller, and the adversary often knows a good approximation. This definition relaxes the previous definition of security for sensitive queries Q.

In [7] we show that $\lim_n \mu_n[Q \mid V]$ for *conjunctive queries* Q and V always exists and to provide an algorithm for computing it. The key technical lemma is to show that, for each conjunctive query Q there exists two numbers c, d s.t. $\mu_n[Q] = c/n^d + O(1/n^{d+1})$. Moreover, both d and c can be computed algorithmically. Since $\mu_n[Q \mid V] = \mu_n[QV]/\mu_n[V]$, the main result follows easily.

With this condition of practical security in mind we distinguish the following cases:

Perfect query-view security. This is the condition analyzed in this paper. $V \mid Q$ can be rewritten as $\mu_n[Q \mid V] = \mu_n[Q]$ for all n large enough. Here V provides no information about Q.

Practical query-view security. $\lim_{n\to\infty} \mu_n[Q \mid V] = 0$. This implies that the difference of probabilities is zero in the limit (since $\lim_n \mu_n[Q] = 0$ for all practical purposes). For finite n, V may in fact contain some information for answering Q, but it is considered negligible in this model.

Practical disclosure. $0 < \lim_{n\to\infty} \mu_n[Q \mid V] < 1$. Disclosure is non-negligible in this case. Our main result allows us to compute this quantity in terms of expected database size S.

7. Conclusion

We have presented a novel definition of security for analyzing the information disclosure between relational views and queries. This definition of security captures very subtle partial disclosures, and has some surprising consequences, highlighting disclosures in query-view pairs commonly considered safe. Our analysis includes tight complexity bounds for deciding query-view security of conjunctive queries and views under tuple-independent probability distributions. We also provide useful results for disclosure relative to pre-existing knowledge an adversary may possess, and we adapt our results to a number of practical scenarios.

This security standard is best viewed as a theoretical ideal, and is probably too strong a criterion for many practical applications. Therefore, an important future direction is too extend the present investigation to a relaxed security standard. We have proposed two initial directions towards this goal in Section 6. The other major direction for extending this work is to accommodate more complex probability distributions.

Acknowledgments

Ron Fagin, Venkatesan Guruswami, Daniele Micciancio, and Victor Vianu provided helpful comments on this work. This work was partially supported by Suciu's CAREER grant IIS-00992955 and by NSF grants IIS-0428168 and 61-2252.

Appendix A. Proof of Theorem 4.10

Here we prove Theorem 4.10: The problem of deciding, for conjunctive query Q, whether a tuple $t \notin crit(Q)$ is Π_2^p-hard.

A.1. Some notations

If $Q(I)$ is true, then there exists a homomorphism $h: Q \to I$. We consider only instances I s.t. $Q(I)$ is true, and we will assimilate such an instance with the pair (I, h). We say that t is not necessary for (I, h) if there exists a homomorphism $h_{new}: Q \to I - \{t\}$. The condition "t is not necessary for Q" is equivalent to:

$$\forall (I, h).\exists h_{new}.\big(h_{new}: Q \to I - \{t\}\big) \text{ is a homomorphism.} \tag{A.1}$$

Given an instance (I, h), we say that t is necessary for (I, h) if there exists a homomorphism $h_{new}: Q \to I - \{t\}$. Condition (A.1) says that t is necessary for every instance (I, h). We will only consider in the sequel only instances (I, h) where I contains the tuple t: if I does not contain t, then t is trivially necessary for I (take $h_{new} = h$).

A.2. Restricting the search

We show here that it suffices to check (A.1) on specific instances (I, h): *minimal* and *fine*. Moreover, for Π_2^p completeness it suffices to consider certain *multigoal* queries Q.

Minimal instances. In (A.1) is suffices to range (I, h) only over surjective homomorphisms h. We say that (I, h) is minimal.

Fine instances. It suffices to restrict the range to *fine* (I,h)'s only, defined as follows. Given (I,h), denote $G=\{g_1,\dots,g_k\}=h^{-1}(t)$ (the set of subgoals mapped to t), $V=Var(G)$, and $W=Var(Q)-V$. Then (I,h) is called *fine* if $\forall x,y\in W, x\neq y\Rightarrow h(x)\neq h(y)$. In other words, in a fine database each variable x in Q is mapped to a distinct constant, unless x is part of a subgoal mapped to t. We denote c_x the constant corresponding to the variable x.

Proposition A.1. *If t is necessary for any instance, then t it is necessary for some fine instance. In other words, in* (A.1) *it suffices to restrict (I,h) to fine instances.*

Proof. We prove the counterpositive: if it is not necessary for fine instances then it is not necessary. Let (I,h) be an instance, G, V, and W defined as above. Define the following instance I': $dom(I')=dom(t)\cup W$, $h': Var(Q)\to dom(I')$ is $h'(x)=h(x)$ for $x\in V$ and $h'(x)=x$ for $x\in W$, and the tuples in I' are precisely $h'(Q)$. Clearly I' is fine, hence t is not necessary for I': there exists a homomorphisms $h'_{new}: Q\to I'-\{t\}$. Define the function $f: dom(I')\to dom(I)$ by $f(c)=c$ if $x\in dom(t)$ and $f(x)=h(x)$ if $x\in W$. Clearly f is a homomorphism and $f\circ h'=h$. To prove that t is not necessary for I, we use $f\circ h'_{new}$, which is a homomorphism from Q to $f(I'-\{t\})$. It remains to show that $f(I'-\{t\})=f(I')-\{t\}$, because this implies that $f\circ h'_{new}$ is a homomorphism from Q to $I-\{t\}$. For that we need to prove that $f(t')=t\Rightarrow t'=t$. Suppose $f(t')=t$, and $t'\neq t$. The t is (the image under h' of) a subgoal g in Q, and $t=f(t')=h(g)$. But then $g\in G$, hence $t'=h'(g)=h(g)=t$, contradiction; hence $f^{-1}(t)=\{t\}$. $\square$

If (I,h) is a fine instance then for each variable x, $h(x)$ is either some constant in t (and this happens only if x occurs in a subgoal g that is mapped to t), or $h(x)$ is some unique constant c_x.

Multigoal queries. Let k be a fixed number. Consider a query Q and tuple t s.t. the number of subgoals in Q that unify with t is at most k. Then the problem whether t is not necessary for Q is in NP. Indeed, it suffices to do the following, for every non-empty set of goals G that unify with t: define h to map all subgoals in G to t (h is unique, when it exists); extend h to the other variables x by defining $h(x)=c_x$, where each c_x is a distinct constant; check that $h^{-1}(t)$ is indeed G (it might be larger); if so, then check that there exists $h_{new}: Q\to I-\{t\}$ (this test is in NP). If all answers are 'yes,' then t is not necessary for Q. This procedure is in NP because there are only a constant number (2^k-1) sets G.

It follows that in the proof of Theorem 4.10 we need to ensure that many subgoals in Q may be mapped to t.

A.3. The reduction

To prove Theorem 4.10 we reduce the ∀∃-3CNF problem to (A.1). Recall that this problem consists of checking whether a formula of the following form is true:

$$\Phi=\forall X_1\dots\forall X_m\exists Y_1\dots\exists Y_n.C$$

where:

$$C=C_1\wedge\cdots\wedge C_p.$$

Here $X_1,\dots,X_m,Y_1,\dots,Y_n$ are propositional variables, and $C_1,\dots,C_p$ are disjunctive clauses involving at most three variables. Examples of clauses are $X_3\vee\neg X_7\vee Y_2$, and $\neg X_1\vee Y_8\vee\neg Y_9$.

Formally, denote truth assignments with $\theta_X:\{X_1,\dots,X_m\}\to\{0,1\}$ and $\theta_Y:\{Y_1,\dots,Y_n\}\to\{0,1\}$. Then the validity problem is expressed by:

$$\forall\theta_X.\exists\theta_Y.\big((\theta_X\cup\theta_Y)(C)\text{ is true}\big). \tag{A.2}$$

Notice that when $m=0$ then this is the 3CNF satisfiability problem.

An inspection of (A.1) and (A.2) imposes the following correspondences.

- Truth assignments θ_X correspond to instances (I,h).
- Truth assignments θ_Y correspond to h_{new}.
- The fact that $(\theta_X\cup\theta_Y)(C)$ is true corresponds to h_{new} being a homomorphism.

In the sequel we assume Φ to be given and construct Q, t s.t. (A.2) $\Leftrightarrow$ (A.1).

A.4. *The tuple*

The tuple is:

$$t = R(0, 1, 2, 3, 3).$$

Here 0, 1, 2, 3 are four arbitrary, distinct constants; 3 occurs twice.

A.5. *The query*

The query is:

$$Q{:}-\mathit{XSubgoals}, \mathit{BackupSubgoals}, \mathit{YSubgoals}, \mathit{CSubgoals}.$$

All four subqueries have many subgoals, and share several variables, but they have disjoint sets of relation names, which allows us to reason about them independently. They will be described in the following sections. Recall that for any instance (I, h) and any variable $x \in \mathit{Var}(Q)$, either $h(x) \in \{0, 1, 2, 3\}$ or $h(x) = c_x$, where c_x is a unique constant (i.e. distinct from 0, 1, 2, 3 and from all other constants c_y).

We call some of the variables in Q *backup* variables, and denote them v^b. In general there is a 1-to-1 correspondence between a normal variable v and a backup variable v^b, but there are exceptions.

A.6. *XSubgoals*

There is a variable z ('zero') and m variables $x_1, \ldots, x_m$, plus many anonymous variables, denoted $-$: each anonymous variable occurs exactly once. The preamble is (we also include the tuple t for readability):

$$\begin{aligned}&\mathit{XSubgoals}{:}-R(z, u, v, e_1, e_2), R(x_1, -, -, -, -), \ldots, R(x_m, -, -, -, -),\\ &\qquad R\bigl(z^b, w^b, w^b, e_1^b, e_2^b\bigr), R\bigl(x_1^b, w_1^b, w_1^b, -, -\bigr), \ldots, R\bigl(x_m^b, w_m^b, w_m^b, -, -\bigr),\\ &t = R(0, 1, 2, 3, 3).\end{aligned}$$

Every variable in the second row is the backup variable of the corresponding variable in the first row. This defines a 1-to-1 correspondence except for $u \leftrightarrow w^b$ and $v \leftrightarrow w^b$.

We classify an instance (I, h) for Q as follows. Recall that we restrict our discussion to instances that are minimal and fine.

Good instances. We call (I, h) *good* if $h(z) = 0$.
Bad instances. We call (I, h) *bad* if $h(z) \neq 0$.

The relation R only occurs in *XSubgoals*, and not in any of the other parts of the query. The purpose of *XSubgoals* is to establish a correspondence between good instances (I, h) and truth assignments to the variables $X_1, \ldots, X_m$. The correspondence is as follows

- $X_i = \mathit{false}$ iff $h(x_i) = 0$,
- $X_i = \mathit{true}$ iff $h(x_i) \neq 0$.

Moreover, the truth assignment to $X_1, \ldots, X_m$ determines a good instance (I, h) up to isomorphism. To see that, let us examine all tuples in R^I, i.e. the relation R in the instance I. Since h is surjective, all these tuples are images of subgoals in *XSubgoals*, i.e. R^I has at most as many tuples as subgoals there are in *XSubgoals*, the question is which subgoals are mapped to the same tuple(s). Start by examining the second row in the definition of *XSubgoals*. $R(z^b, w^b, w^b, e_1^b, e_2^b)$ cannot be mapped to the tuple $t = R(0, 1, 2, 3, 3)$ because it has the same variable w^b on positions 2 and 3. It follows that its variables are mapped to distinct constants, i.e. $h(z^b) = c_{z^b}$ etc., because (I, h) is fine. Similarly, all the other subgoals in the second row are mapped to distinct tuples in R^I. This results in $m + 1$ tuples in R^I, independently of the truth assignment for $X_1, \ldots, X_m$. Now consider the first row. We have $h(z) = 0$ (because (I, h) is good), and this can only be allowed if h maps $R(z, u, v, e_1, e_2)$ to $R(0, 1, 2, 3, 3)$ (since (I, h) is fine); again, this still does not depend on the truth assignment. Finally, each subgoal $R(x_i, -, -, -, -)$ is mapped as follows: if $X_i = \mathit{false}$, i.e. $h(x_i) = 0$, then the subgoal must be mapped to $R(0, 1, 2, 3, 3)$, otherwise, it is mapped into a fresh tuple, distinct from all others. Thus, if the instance is good ($h(z) = 0$), then it is determined up to isomorphism by the set of variables x_i s.t. $h(x_i) = 0$, i.e. it is determined up to isomorphism by the truth assignment to $X_1, \dots, X_m$.

Notice that, except for the tuple $t = R(0, 1, 2, 3, 3)$, all the tuples in R^I have distinct constants on the last two positions, i.e. are of the form $R(-, -, -, a, b)$ with a, b distinct constants. Examine now the variables e_1, e_2, in the first subgoal. For a good instance we have $h(e_1) = h(e_2) = 3$, and, whenever h_{new} exists, then $h_{new}(e_1) \neq h_{new}(e_2)$, since now the tuple t is removed.

Before proving any formal properties for *XSubgoals* we define *BackupSubgoals*.

A.7. BackupSubgoals

BackupSubgoals contains three kinds of subgoals.

First, there is one subgoal for each backup variable v^b:

$$\mathit{BackupSubgoals}{:}-\dots, R_{v^b}\big(v^b\big), \dots.$$

The relation name R_{v^b} is unique for the variable v^b and not used anywhere else. It follows that for every instance (I, h) and homomorphism $h_{new} : Q \to I - \{t\}$ the following holds (since (I, h) is minimal, i.e. h is surjective):

$$h_{new}\big(v^b\big) = h\big(v^b\big).$$

That is, backup variables remain unchanged.

Second, there are two subgoals for every variable v that has a unique backup variable v^b:

$$\mathit{BackupSubgoals}{:}-\dots, R_v(v), R_v\big(v^b\big), \dots.$$

Again, the relation name R_v is unique. If two variables v, v' have the same backup variable v^b then we have two subgoals for each of the pairs (v, v^b) and (v', v^b). But we do not include the y variables (to be introduced later), which have two backup variables y^f, y^t.

It follows that for every instance (I, h) and homomorphism $h_{new} : Q \to I - \{t\}$ the following holds:

$$h_{new}(v) = h(v) \vee h_{new}(v) = h\big(v^b\big).$$

Third, we treat the variables $y_1, \dots, y_n$ special, since each variable y_i that has two backup variables y_i^f, y_i^t, $i = 1, \dots, n$. Namely, for each y_i we introduce the following subgoals:

$$\mathit{BackupSubgoals}{:}-\dots, R_{y_i}(y_i), R_{y_i}\big(y_i^t\big), R_{y_i}\big(y_i^f\big), \dots.$$

It follows:

$$h_{new}(y_i) = h(y_i) \vee h_{new}(y_i) = h\big(y_i^f\big) \vee h(y_i) = h\big(y_i^t\big).$$

We will show, however, that for any good instance (I, h) we have $h_{new}(y_i) \neq h(y_i)$, hence one of the latter two cases must hold.

The following two lemmas state the main property about *XSubgoals*.

Lemma A.2. *Let (I, h) be a good instance. Then:*

- *There exists a homomorphism $h_{new} : \mathit{XSubgoals} \cup \mathit{BackupSubgoals} \to I - \{t\}$ s.t.:*
  - *If $h(x_i) = 0$ then $h_{new}(x_i) = h(x_i^b)$.*
  - *If $h(x_i) \neq 0$ then $h_{new}(x_i) = h(x_i)$.*
- *Any homomorphism $h_{new} : \mathit{XSubgoals} \cup \mathit{BackupSubgoals} \to I - \{t\}$ satisfies the following:*
  - *$h_{new}(z) = h(z^b) \neq h(z)$.*
  - *$h_{new}(e_1) \neq h_{new}(e_2)$.*
  - *If $h(x_i) = 0$ then $h_{new}(x_i) = h(x_i^b)$.*
  - *If $h(x_i) \neq 0$ then $h_{new}(x_i) = h(x_i)$ or $h_{new}(x_i) = h(x_i^b)$.*

Notice that we cannot prevent $h_{new}(x_i)$ to be $h(x_i^b)$ in the last case. Finally, we have:

Lemma A.3. *If (I, h) is a bad instance, then there exists a homomorphism $h_{new}: XSubgoals \cup BackupSubgoals \to I - \{t\}$ s.t.*

- *If $h(x_i) = 0$ then h_{new} maps the subgoal $R(x_i, -, -, -, -)$ to $h(R(x_i^b, -, -, -, -))$. In particular, $h_{new}(x_i) = h(x_i^b)$.*
- *If $h(x_i) \neq 0$ then h_{new} maps the subgoal $R(x_i, -, -, -, -)$ to $h(R(x_i, -, -, -, -))$. In particular, $h_{new}(x_i) = h(x_i)$.*
- *If g is any other subgoal than the two above, then $h_{new}(g) = h(g)$.*

A.8. YSubgoals

For each $i = 1, \dots, n$ there are 3 variables corresponding to Y_i: y_i and two backup variables y_i^f, y_i^t. *YSubgoals* is:

$$\begin{aligned}\textit{YSubgoals}: -&R_{y_1}(z, y_1), \dots, R_{y_n}(z, y_n),\\ &R_{y_1}\big(z^b, y_1^f\big), \dots, R_{y_n}\big(z^b, y_n^f\big),\\ &R_{y_1}\big(z^b, y_1^t\big), \dots, R_{y_n}\big(z^b, y_n^t\big).\end{aligned}$$

Here z is the "zero" variable from the *XSubgoals*. This piece of the query ensures the correspondence between h_{new} and truth assignments θ_Y for $Y_1, \dots, Y_n$, as follows. Consider a good instance (I, h). Then none of $h(y_1), \dots, h(y_n)$ can be any of the constants 0, 1, 2, or 3 (since we assume that I is a fine instance), hence they are fresh, distinguished constants. Consider now a homomorphism $h_{new}: Q \to I - \{t\}$. The first subgoal $R_{y_1}(z, y_1)$: h_{new} cannot map it to the same tuple as h, because $h_{new}(z) \neq h(z)$, hence it has to map it to (the image under h of) either $R_{y_1}(z^b, y_1^f)$ or $R_{y_1}(z^b, y_1^t)$. Hence, we have either $h_{new}(y_1) = h(y_1^f)$ or $h_{new}(y_1) = h(y_1^t)$. In the first case we set $\theta_Y(Y_1) = 0$; in the second, $\theta_Y(Y_1) = 1$. Similarly for $y_2, \dots, y_n$. Conversely, to any truth assignment θ_Y we associate an h_{new} defined on y_1 to be $h_{new}(y_1) = h(y_1^f)$ if $\theta_Y(Y_1) = 0$ and $h_{new}(y_1) = h(y_1^t)$ if $\theta_Y(Y_1) = 1$.

Consider now a bad instance (I, h), i.e. $h(z) = c_z$. Then we may define $h_{new}(z) = h(z)$, and also $h_{new}(y_i) = h(y_i)$ for every $i = 1, \dots, n$.

A.9. CSubgoals

Recall that $C = C_1 \wedge \dots \wedge C_p$ is a 3CNF formula. Then:

$$\textit{CSubgoals}: -G_1, \dots, G_p.$$

Each subquery G_i corresponds to a clause C_i, for $i = 1, \dots, p$. We describe them next. To reduce notation clutter, we illustrate G_1 for C_1, and will do this on several examples: the general case follows then easily. We write C_1 as an implication, with the $\bar{X}$ variables on the right and the $\bar{Y}$ variables on the left, as in the example below:

$$C_1 = \neg X_2 \wedge \neg X_4 \wedge X_6 \wedge X_8 \quad\Longrightarrow\quad \neg Y_3 \vee Y_5 \vee \neg Y_7.$$

Normally C_1 must have at most 3 variables. We will use more than 3 variables for the examples in the sequel, for illustration purposes only. The case with at most 3 variables follows easily.

Each clause must have at least one Y variable: otherwise Φ is false. It follows that each clause has at most 2 X variables.

A.10. Clauses without X's

Here:

$$C_1 = \neg Y_3 \vee Y_5 \vee \neg Y_7$$

and the corresponding subgoal is:

$$\begin{aligned}G_1 :- {} & S_1(z, y_3, y_5, y_7),\\ & S_1\big(z^b, y_3^f, y_5^f, y_7^f\big)\\ & \ldots \text{ only the subgoal } S_1\big(z^b, y_3^t, y_5^f, y_7^t\big) \text{ is missing}\\ & S_1\big(z^b, y_3^t, y_5^t, y_7^t\big).\end{aligned}$$

Let (I, h) be a good instance. Consider a truth assignment θ_Y for $Y_1, \ldots, Y_n$ and the corresponding $h_{new} : Q \to I - \{t\}$. Then h_{new} is a homomorphism iff $\theta_Y(C_1)$ is true. This is because h_{new} cannot map the first subgoal to the same tuple as h, since $h_{new}(z) \neq h(z)$. Instead it has to map it to one of the seven tuples corresponding to the other seven subgoals, under h. This is possible if and only if $\theta_Y(C_1)$ is true. In addition it is easy to see that h_{new} can map the other seven subgoals to the same tuples as h.

Let (I, h) be a bad instance. Then h_{new} can be constructed to be identical to h on G_1.

A.11. Clauses with X variables

In this case we write C_1 like this:

$$C_1 = (A \Longrightarrow \neg Y_3 \vee Y_5 \vee \neg Y_7)$$

and define:

$$C_{1Y} = \neg Y_3 \vee Y_5 \vee \neg Y_7.$$

Then the subquery has the following form:

$$G_1 :- H_1, K_1.$$

Here H_1 depends on C_{1Y} and is somewhat similar to the subquery G_1 for the case without $\bar{X}$ variables, while K_1 depends on A. Both will be explained below. The role of K_1 is to define a certain variable, a, and its backup, a^b, with the following two properties. Denote S the following set of subgoals: $S = \mathit{XSubgoals} \cup \mathit{BackupSubgoals} \cup \mathit{YSubgoals} \cup K$. Below, an instance (I, h) refers to a homomorphism $h : S \to I$. Then K will be such that the following holds:

- If (I, h) is good and corresponds to a truth assignment θ_X then:
  - If $\theta_X(A) = \mathit{false}$, then there exists a homomorphism $h_{new} : S \to I - \{t\}$ s.t. $h_{new}(a) = h(a)$.
  - If $\theta_X(A) = \mathit{true}$, then there exists a homomorphism $h_{new} : S \to I - \{t\}$ s.t. $h_{new}(a) = h(a^b)$; conversely, for any homomorphism $h_{new} : S \to I - \{t\}$, we have $h_{new}(a) = h(a^b) \neq h(a)$. That is, $h(a^b)$ is in fact the only option for $h_{new}(a)$.
- If (I, h) is bad, then there exists a homomorphism $h_{new} : S \to I - \{t\}$ s.t. $h_{new}(a) = h(a)$.

We call a variable a with these properties a trigger for A. The subquery K will introduce a variable a that is a trigger. The subquery H will use the trigger variable a. We define H and K next.

A.12. H

A subgoal H is associated to a clause C_Y of Y variables such that the following property holds. Recall that a is a query variable, which we later will ensure is a trigger.

Lemma A.4. *Denote $S = \mathit{XSubgoals} \cup \mathit{BackupSubgoals} \cup \mathit{YSubgoals}$. We will assume that S includes the trigger variable a; H has one additional variable t_1 (not to be confused with the tuple t).*

- *Let (I, h) be a good instance, where h is a homomorphism $S \to I$.*
  - *Any homomorphism $h_{new} : S \to I - \{t\}$ s.t. $h_{new}(a) = h(a)$ can be extended to a homomorphism $h_{new} : S \cup H \to I - \{t\}$.*

- ∘ *Let $h_{new} : S \to I - \{t\}$ be a homomorphism s.t. $h_{new}(a) = h(a^b)$, and let θ_Y be the corresponding truth assignment to the variables $Y_1, \ldots, Y_n$. Then h_{new} can be extended to a homomorphism $h_{new} : S \cup H \to I - \{t\}$ iff $\theta_Y(C_Y)$ is true.*
- *Let (I, h) be a bad instance, where $h : S \to I$. Then any homomorphism $h_{new} : S \to I - \{t\}$ s.t. $h_{new}(a) = h(a)$, and $h_{new}(y_i) = h(y_i)$, for $i = 1, n$, can be extended to a homomorphism $h_{new} : S \cup H \to I - \{t\}$.*

The lemma establishes the desired connection between the boolean formula and the query. If $\theta_X(A) = \mathit{false}$ then the implication $A \Rightarrow C$ holds vacuously: in this case there exists a homomorphism h_{new} s.t. $h_{new}(a) = h(a)$ (the first trigger property), hence, by the lemma, it can be extended to H regardless of whether C is satisfiable or not. If $\theta_X(A) = \mathit{true}$, then the implication $A \Rightarrow C$ is logically equivalent to C; in this case the only possible homomorphism h_{new} is such that $h_{new}(a) = h(a^b)$, and it can be extended to H if and only if C is satisfiable (and the extension corresponds to a truth assignment making C true).

Instead of a general construction, we illustrate H_1 for our example clause C_{1Y}:

$H_1 : -S_1(a, y_3, y_5, y_7, t_1, t_1),$

$S_1(a^b, y_3^f, y_5^f, y_7^f, e_1, e_2)$

$\ldots$ Group 1: seven subgoals, corresponding to truth assignments making C_{1Y} true

$S_1(a^b, y_3^t, y_5^t, y_7^t, e_1, e_2),$

$S_1(a, y_3^f, y_5^f, y_7^f, e_1, e_2)$

$\ldots$ Group 2: eight subgoals, corresponding to all truth assignments

$S_1(a, y_3^t, y_5^t, y_7^t, e_1, e_2),$

$S_1(a, y_3^f, y_5^f, y_7^f, e_1^b, e_2^b)$

$\ldots$ Group 3: eight subgoals, corresponding to all truth assignments

$S_1(a, y_3^t, y_5^t, y_7^t, e_1^b, e_2^b),$

$S_1(a^b, y_3^f, y_5^f, y_7^f, e_1^b, e_2^b)$

$\ldots$Group 4: eight subgoals, corresponding to all truth assignments

$S_1(a^b, y_3^t, y_5^t, y_7^t, e_1^b, e_2^b).$

Recall the variables e_1, e_2 from *XSubgoals*. They have the property that in any good instance (I, h) $h(e_1) = h(e_2) = 3$, but for any h_{new}, $h_{new}(e_1) \neq h_{new}(e_2)$. t_1 is a fresh variable, used only in the first subgoal of H_1 (where it occurs twice). Denote the first subgoal with $g = S_1(a, y_3, y_5, y_7, t_1, t_1)$.

To check the lemma, let (I, h) be a good instance, and let $h : S \to I - \{t\}$ be a homomorphism. If we have $h_{new}(a) = h(a)$, then simply define h_{new} to map the goal g to (the image under h of) one of the eight subgoals in Group 2. This is possible because these eight images have all $h(e_1) = h(e_2)$ on the last two positions allowing h_{new} to map the two occurrences of t_1 to the same constant. All the other subgoals in H_1 are mapped by h_{new} as follows: Group 1 goes to (the image under h of) Group 4, Group 2 goes to Group 3, while Groups 3 and 4 remain unchanged (i.e. h_{new} is equal to h here).

Suppose now that we have $h_{new}(a) = h(a^b)$. Then we cannot map g anywhere to (the image under h of) Group 2, because these eight tuples have on the first position $h(a)$, while we need $h(a^b)$. Group 3 is excluded for the same reason. Group 4 is excluded for a different reason: the image under h of this group forms eight tuples that have $h(e_1^b) \neq h(e_2^b)$ on the last two positions, while we need the same constant (because of t_1, t_1). Hence, our only choice is to map g to one of the seven tuples that form the image of Group 1 under h. This is possible iff h corresponds to one of the seven satisfying assignments θ_Y for C_{1Y}.

Now let (I, h) be a bad instance and consider a $h_{new} : S \to I - \{t\}$ s.t. $h_{new}(a) = h(a)$ and $h_{new}(y_i) = h(y_i)$, $i = 1, 2, \ldots$. Then we simply extend h_{new} by defining $h_{new}(t_1) = h(t_1)$: thus, h_{new} maps the first subgoal to the same tuple as h. For all other subgoals, h_{new} either maps the to the same tuple as h, or to the corresponding backup tuple, depending on whether $h_{new}(e_1)$, $h_{new}(e_2)$ are the same as $h(e_1)$, $h(e_2)$, or the same as $h(e_1^b)$, $h(e_2^b)$.

This completes the construction of H_1. It remains to show how to enforce the semantics of the trigger a.

A.13. K

We consider three cases, depending on A.

$\boldsymbol{A = \neg X_2}$. Before giving the definition of K, we note that the variable x_2 almost plays the role of a trigger variable in this case. More precisely, denote $S = \mathit{XSubgoals} \cup \mathit{BackupSubgoals}$. Then let (I, h) be good, $h : S \to I$, and $\theta_X(X_2)$ the corresponding truth assignment. Then the following two trigger conditions are satisfied:

- If $\theta_X(X_2) = \mathit{false}$, then $h(x_2) \neq 0$. Then we know that there exists a homomorphism $h_{new} : S \to I - \{t\}$ s.t. $h_{new}(x_2) = h(x_2)$.
- If $\theta_X(X_2) = \mathit{true}$, then $h(x_2) = 0$. Then we know that there exists a homomorphism $h_{new} : S \to I - \{t\}$ s.t. $h_{new}(x_2) = h(x_2^b)$.

Unfortunately, x_2 does not satisfy the third condition: when (I, h) is bad, it is not always possible to define a homomorphism $h_{new} : S \to I - \{t\}$ s.t. $h_{new}(x_2) = h(x_2)$. Namely when $h(x_2) = 0$ then we cannot set $h_{new}(x_2) = 0$, since the only tuple having the value 0 is t.

To define the trigger variable a, we need to construct a much more complex set K of subgoals, where we use the variable x_2 and the properties above.

$$\begin{array}{llllllr}
K\!:- & N(a, & x_2, & z, & s, & s) & = g \\
 & N(a, & x_2 \mid x_2^b, & z, & e_1, & e_2) & = g_{bad} \\
 & N(a, & x_2, & z^b, & e_1, & e_2) & = g_{X_2=true} \\
 & N(a^b, & x_2^b, & z^b, & e_1, & e_2) & = g_{X_2=false} \\
 & \multicolumn{6}{l}{\text{closure of all rows containing } e_1, e_2, \text{ with } e_1^b, e_2^b \text{ instead.}}
\end{array}$$

The notation $x_2 \mid x_2^b$ in the second "subgoal" is an abbreviation and, in fact, denotes two subgoals one with x_2, the other with x_2^b. Consider a bad instance: then h_{new} must send g to $h(g)$ (all other subgoals have e_1, e_2 on the last two positions, and they are mapped by h to distinct constants), hence $h_{new}(a) = h(a)$. For a good database where X_2 is true, map g to $h(g_{X_2=true})$: this is possible since $h_{new}(x_2)$ can be defined to be $h(x_2)$. So $h_{new}(a) = h(a)$ in this case. In this case we cannot prevent g to be mapped to some other subgoal, hence it is possible that $h_{new}(a) = h(a^b) \neq h(a)$, but we do not care. If X_2 is false then map g to $h(g_{X_2=false})$: this is possible since now $h_{new}(x_2) = h(x_2)$. In this case it is not possible to map g anywhere else: g_{bad} is excluded because $h_{new}(z) \neq h(z)$, g_{true} is excluded because $h_{new}(x_2) \neq h(x_2)$, and the closure subgoals are excluded because they all end in e_1^b, e_2^b and $h(e_1^b) \neq h(e_2^b)$, so we do not have a destination for s.

Finally, in each case we can extend the homomorphism h_{new} to the other subgoals, because they are closed under variables/backup-variables combinations (see below more detailed examples of closures).

$\boldsymbol{A = X_2}$. Define:

$$\begin{array}{llllllll}
K\!:- & N(a, & z, & s, & s, & t, & t) & = g \\
 & N(a, & z^b, & x_2, & z, & e_1, & e_2) & = g_{X_2=false} \\
 & N(a^b, & z^b, & s^b, & s^b, & e_1, & e_2) & = g_{X_2=true} \\
 & N(a \mid a^b, & z^b, & x_2 \mid x_2^b, & z \mid z^b, & e_1^b, & e_2^b) & = \mathit{closure}(g_{X_2=false}) \\
 & N(a^b, & z^b, & s^b, & s^b, & e_1^b, & e_2^b) & = \mathit{closure}(g_{X_2=true}).
\end{array}$$

Here N is a fresh relation symbol, s, t are fresh variables with their backups, and e_1, e_2 are the variables introduced by *XSubgoals*. The main subgoal is g, and the next two are denoted $g_{X_2=false}$ and $g_{X_2=true}$, respectively. The fourth row denotes eight subgoals, obtained by considering all possible combinations: we call this the "closure" of the subgoal in row two. Similarly the last subgoal is the closure of the subgoal $g_{X_2=true}$. Notice that the closure subgoals end in e_1^b, e_2^b rather than e_1, e_2.

We check the conditions for a to be a trigger. When (I, h) is a bad instance then we can define h_{new} to send g to itself: this is possible because we can take $h_{new}(z) = h(z)$. In this case $h_{new}(a) = h(a)$. Other mappings may be possible, but we do not care.

Let (I, h) be a good instance corresponding to a truth assignment making X_2 false. Then define h_{new} to send g to $g_{X_2=false}$. This is possible because $h(x_2) = 0 = h(z)$, so it suffices to define $h_{new}(a) = h(a)$, $h_{new}(s) = 0$, $h_{new}(t) = 3$ $(= h(e_1) = h(e_3))$. Notice that here we cannot prevent $h_{new}(a)$ to have some other value (namely $h(a^b)$).

Finally, let (I, h) be a good instance corresponding to a truth assignment making X_2 true. Here define h_{new} to map g to $g_{X_2=true}$. This means $h_{new}(a) = h(a^b)$ and $h_{new}(s) = h(s^b)$. In this case we must also prove that for any h_{new} we have $h_{new}(a)! = h(a)$. This can be seen by examining other possible targets for $h_{new}(g)$. It cannot be $g_{X_2=false}$ because we cannot map s (since $h(x_2)! = h(z)$). It cannot be any of the closure subgoals, because we cannot map t (since $h(e_1^b)! = h(e_2^b)$).

It remains to show that in each case h_{new} can also map all the other subgoals, except g. This is trivial because these sets of subgoals are closed under all combinations of variable/backup-variable.

$\boldsymbol{A = A_1 \wedge A_2}$. Here we show how to construct a trigger variable a from a_1, a_2:

$$\begin{aligned} K :- & N(a, a_1, a_2), \\ & N\big(a, a_1^b, a_2\big), \\ & N\big(a, a_1, a_2^b\big), \\ & N\big(a^b, a_1^b, a_2^b\big). \end{aligned}$$

If $h_{new}(a_1) \neq h(a_1)$ and $h_{new}(a_2) \neq h(a_2)$ then we must define $h_{new}(a) = h(a^b)$, hence $h_{new}(a) \neq h(a)$. In all other cases we can define $h_{new}(a) = h(a)$.

References

[1] Nabil R. Adam, John C. Wortmann, Security-control methods for statistical databases, ACM Comput. Surv. 21 (4) (1989) 515–556.

[2] Rakesh Agrawal, Alexandre Evfimievski, Ramakrishnan Srikant, Information sharing across private databases, in: Conference on Management of Data (SIGMOD), 2003, pp. 86–97.

[3] François Bancilhon, Nicolas Spyratos, Protection of information in relational data bases, in: Conference on Very Large Databases (VLDB), 1977, pp. 494–500.

[4] François Bancilhon, Nicolas Spyratos, Algebraic versus probabilistic independence in data bases, in: Principles of Database Systems (PODS), 1985, pp. 149–153.

[5] Elisa Bertino, Sushil Jajodia, Pierangela Samarati, Database security: Research and practice, Inform. Syst. 20 (7) (1995) 537–556.

[6] José A. Blakeley, Neil Coburn, Per-Åke Larson, Updating derived relations: Detecting irrelevant and autonomously computable updates, ACM Trans. Database Systems (TODS) 14 (3) (1989) 369–400.

[7] Nilesh Dalvi, Gerome Miklau, Dan Suciu, Asymptotic conditional probabilities for conjunctive queries, in: Proceedings of the International Conference on Database Theory (ICDT), 2005.

[8] Dorothy Denning, Cryptography and Data Security, Addison–Wesley, 1982.

[9] Alin Deutsch, Yannis Papakonstantinou, Privacy in database publishing, in: Proceedings of the International Conference on Database Theory (ICDT), 2005.

[10] Charles Elkan, Independence of logic database queries and update, in: Principles of Database Systems (PODS), ACM Press, 1990, pp. 154–160.

[11] Alexandre Evfimievski, Johannes Gehrke, Ramakrishnan Srikant, Limiting privacy breaches in privacy preserving data mining, in: Principles of Database Systems (PODS), ACM Press, 2003, pp. 211–222.

[12] R. Fagin, Probabilities on finite models, Notices Amer. Math. Soc. October (1972) A714.

[13] R. Fagin, Probabilities on finite models, J. Symbolic Logic 41 (1) (1976).

[14] C.M. Fortuin, P.W. Kasteleyn, J. Ginibre, Correlation inequalities on some partially ordered sets, Comm. Math. Phys. 22 (1971) 89–103.

[15] Lise Getoor, Benjamin Taskar, Daphne Koller, Selectivity estimation using probabilistic models, in: SIGMOD, 2001.

[16] Ashish Gupta, Yehoshua Sagiv, Jeffrey D. Ullman, Jennifer Widom, Constraint checking with partial information, in: Principles of Database Systems (PODS), 1994, pp. 45–55.

[17] Hakan Hacigumus, Balakrishna R. Iyer, Chen Li, Sharad Mehrotra, Executing SQL over encrypted data in the database service provider model, in: SIGMOD Conference, 2002.

[18] Alon Halevy, Answering queries using views: A survey, VLDB J. 10 (4) (2001) 270–294.

[19] D. Koller, A. Pfeffer, Probabilistic frame-based systems, in: Conference on Artificial Intelligence, 1998, pp. 580–587.

[20] Alon Y. Levy, Yehoshua Sagiv, Queries independent of updates, in: Conference on Very Large Data Bases (VLDB), 1993, pp. 171–181.

[21] Gerome Miklau, Dan Suciu, Controlling access to published data using cryptography, in: Conference on Very Large Databases (VLDB), September 2003, pp. 898–909.

[22] Gerome Miklau, Dan Suciu, A formal analysis of information disclosure in data exchange, in: Conference on Management of Data (SIGMOD), ACM Press, 2004, pp. 575–586.

[23] Bruce Schneier, Applied Cryptography, second ed., Wiley, 1996.

[24] C.E. Shannon, Communication theory of secrecy systems, Bell Syst. Techn. J. 28 (1949) 656–715.
[25] Claude E. Shannon, A mathematical theory of communication, Bell Syst. Techn. J. 27 (3) (1948) 379–423.
[26] Latanya Sweeney, k-Anonymity: A model for protecting privacy, Internat. J. Uncertain. Fuzziness Knowledge-Based Systems 10 (5) (2002) 557–570.
[27] Xiaochun Yang, Chen Li, Secure XML publishing without information leakage in the presence of data inference, in: Conference on Very Large Databases (VLDB), 2004, pp. 96–107.