Optimal Security-Aware Query Processing

Marco Guarnieri
Institute of Information Security
Department of Computer Science
ETH Zürich, Switzerland
marco.guarnieri@inf.ethz.ch

David Basin
Institute of Information Security
Department of Computer Science
ETH Zürich, Switzerland
david.basin@inf.ethz.ch

ABSTRACT

Security-Aware Query Processing is the problem of computing answers to queries in the presence of access control policies. We present general impossibility results for the existence of optimal algorithms for Security-Aware Query Processing and classify query languages for which such algorithms exist. In particular, we show that for the relational calculus there are no optimal algorithms, whereas optimal algorithms exist for some of its fragments, such as the existential fragment.

We also establish relationships between two different models of Fine-Grained Access Control, called Truman and Non-Truman models, which have been previously presented in the literature as distinct. For optimal Security-Aware Query Processing, we show that the Non-Truman model is a special case of the Truman model for boolean queries in the relational calculus, moreover the two models coincide for more powerful languages, such as the relational calculus with aggregation operators. In contrast, these two models are distinct for non-boolean queries.

1. INTRODUCTION

Databases often manage sensitive data where security is a concern. There is, however, no common terminology to refer to the problem of computing answers to queries in the presence of access control policies. For example, "secure querying" [9], "enforcing data confidentiality" [15], and "Fine-Grained Access Control" [5, 16, 19] have all been used in this context. In the following, we refer to this problem as *Security-Aware Query Processing* (SAQP). This differs from *Secure Query Processing* since the latter usually refers to querying encrypted data. Security-Aware Query Processing algorithms are implemented in commercial databases [1, 7, 18], and solutions have been proposed for other settings like XML files [9] or RDF graphs [15].

Rizvi et al. [16] identified two distinct models, called *Truman* and *Non-Truman* models [16], that capture different approaches to the SAQP problem. The term *Truman* model comes from the protagonist of the movie *The Truman Show* who is unaware that he lives in an artificial environment where everything he experiences is externally controlled. Algorithms in the Truman model transparently modify all user queries to restrict the user's access to only the data authorized by the security policy. The query's result may differ from the unrestricted one to preserve confidentiality. For example, suppose we issue a query q asking for the names of all employees, but we are authorized only to read some of their records. If the company has 1000 employees and we have access to just 800, the original query is modified, and we get just the names of the 800 employees we are authorized to read. However, such modifications may cause inconsistencies between user's expectations and the query's result. For instance, if we know that there are actually 1000 employees, then the modified result is inconsistent with our knowledge.

In contrast to the above, the *Non-Truman* model solves the problem of inconsistencies. Algorithms in this model execute a query iff it can be answered using only information the user is authorized to read under the given policy (these queries are called *conditionally valid*) and otherwise the query is rejected. In our example, in the Non-Truman model the query q is rejected because it cannot be answered using only authorized information. In contrast, a query q' asking the names of the 800 employees we are authorized to read is executed. The two models target different needs. The Truman model ensures higher data availability at the price of partial results and inconsistencies, whereas the Non-Truman model ensures consistency at the expense of lower data availability.

Wang et al. [19] analyzed query processing algorithms in the Truman model and proposed three correctness criteria: *security*, *soundness*, and *maximality*. Security requires that a query's result does not depend on sensitive information, soundness requires that an algorithm returns only correct information, and maximality requires that an algorithm returns as much information as possible without violating the given policy. In the Truman model, we say that an algorithm is *optimal* iff it satisfies all three of these criteria, whereas in the Non-Truman model, an algorithm is *optimal* iff it executes exactly the conditionally valid queries. In both models, optimal algorithms are what we ideally want as they are the only algorithms that guarantee security, correctness, and data availability. For instance, in the Non-Truman model, an algorithm that rejects all queries is secure but useless, whereas an algorithm that executes all queries is useful but insecure. Similarly, in the Truman model, an algorithm that

Proceedings of the VLDB Endowment, Vol. 7, No. 12

always returns $\emptyset$ as a query's result is secure and sound, an algorithm that always returns the original query's result is sound and maximal, and an algorithm that systematically introduces noise in the result can be secure and maximal.

Unfortunately, a thorough analysis of optimal Security-Aware Query Processing has been missing until now. For the Non-Truman model, Rizvi et al. [16] left open the decidability of the conditional validity problem. Although Zhang et al. [20] proved that it is decidable for conjunctive queries and Koutris et al. [11] extended this result to unions of conjunctive queries, there is no general characterization of the problem. For the Truman model, Wang et al. [19] claimed that "while the maximality property is desirable, it is difficult to achieve" for algorithms that are secure and sound. They presented an informal example supporting this claim but they did not give a proof.

In this paper, we study the decidability of optimal Security-Aware Query Processing for boolean and non-boolean queries in both models and we provide answers to all the above open problems. We prove possibility and impossibility results for the relational calculus (RC) and for the existential fragment of RC (ERC). RC is a query language that can express all first order queries, whereas the ERC fragment is an extension of conjunctive queries. We also establish connections between Truman and Non-Truman models.

Contributions First, we present a detailed analysis of the optimal Security-Aware Query Processing problem in the Truman model. We prove that, in general, there is no optimal SAQP algorithm in the Truman model for the relational calculus. Hence, for the query languages used in practice, such as SQL, there is no way to securely process queries without either losing some information, violating the security policy, or returning incorrect information. We classify fragments of the relational calculus for which optimal algorithms exist, and we present sufficient conditions for the decidability of the optimal Security-Aware Query Processing problem. We use these sufficient conditions to give an optimal SAQP algorithm for the existential fragment of RC. We also prove the claim of Wang et al. stated in [19].

Second, we establish connections between the Truman and Non-Truman models for the optimal SAQP problem. These two models have different, and seemingly contradictory, goals, and their relationship was previously unclear. In the past, they were sometimes considered as distinct [16,19]. We show that, for boolean RC-queries, the Non-Truman model is a special case of the Truman model. Moreover, the two models coincide for sufficiently powerful query languages, such as the relational calculus with aggregation operators. In contrast, for non-boolean queries, we prove that the two models actually are distinct.

Finally, we adapt the correctness criteria defined in [19] to the Non-Truman model. By using the resulting criteria and by exploiting the connections between optimal algorithms in the two models, we extend our decidability results to optimal algorithms in the Non-Truman model and to the conditional validity problem. Note that we consider a security model that is more expressive than the one used in previous work on the Non-Truman model [11, 16, 20]. We thereby improve upon the results in [11, 20].

Table 1 summarizes our main results, and the associated theorems. The symbol $\supset$ denotes that the Non-Truman model is a special case of the Truman model, whereas the symbol $\neq$ denotes that the two models are distinct.

Organization In Section 2 we review relevant background. In Section 3 we introduce our security model, and in Section 4 we introduce Security-Aware Query Processing. We present our results in Section 5 for boolean queries and in Section 6 for non-boolean queries. In Section 7 we compare with related work, and in Section 8 we draw conclusions.

2. BACKGROUND

In this section, we provide background on the relational model, the relational calculus, and associated decidability results. Further details can be found in [3, 6].

2.1 Relational Model

We use the definition of the relational model taken from [3]. Let **att**, **dom**, and **relname** be three countably infinite disjoint sets. The set **att** contains attribute identifiers, **relname** contains relation identifiers, and **dom** represents the underlying domain. We assume that there is a total order $\leq_{\mathbf{att}}$ on **att**. Constants are elements of **dom**, and we assume given a mapping $Dom : \mathbf{att} \to \mathbb{P}(\mathbf{dom})$ assigning to each attribute $at \in \mathbf{att}$ a set of elements from **dom**. Without loss of generality, we assume that $\mathbb{N} \subseteq \mathbf{dom}$. We associate a finite set of attributes to each relation name using the function $sort : \mathbf{relname} \to \mathbb{P}^{fin}(\mathbf{att})$, where $\mathbb{P}^{fin}(A)$ is the set of all finite subsets of the set A.

A *relation schema* is a relation name $R \in \mathbf{relname}$ with a set of attributes $sort(R)$. A *tuple* $\bar{t} := \langle t_1, \ldots, t_n \rangle$ is an n-tuple, where $t_1, \ldots, t_n \in \mathbf{dom}$ and $n \in \mathbb{N}$. We say that $\bar{t}$ is of sort $\{at_1, \ldots, at_n\}$ (ordered by $\leq_{\mathbf{att}}$) iff $t_i \in Dom(at_i)$ for $1 \leq i \leq n$. We denote by $|\bar{t}|$ the number of values in $\bar{t}$, and by $\bar{t}^i$ the i-th value in $\bar{t}$, where $i \in \{1, \ldots, |\bar{t}|\}$. A *relation instance* of a relation schema R is a (possibly empty) finite set of tuples of sort $sort(R)$.

A *database schema* D is a non-empty finite set of relation schemas, and a *state* of D is a mapping s that assigns to each relation schema $R \in D$ a relation instance of the schema R. Let D be a database schema. We denote by Ω_D the set of all possible states of D. For a relation schema R in D, we denote by $s(R)$ the instance of R in the state s.

2.2 Domain Relational Calculus

The *Domain Relational Calculus*, or simply the *Relational Calculus* (RC), is a query language defined by Codd [8] built on top of function-free First-Order Logic (FOL). Let **var** be a countably infinite set of variables.

Definition 2.1. **Relational Calculus:** Let D be a database schema. A *RC-formula over D* is inductively defined as follows.

1. $R(x_1, \ldots, x_n)$ is a RC-formula over D, where R is a relation schema in D and $x_1, \ldots, x_n \in \mathbf{var}$.
2. $x \ op \ k$ is a RC-formula over D, where $x \in \mathbf{var}$, $k \in \mathbf{dom}$, and $op \in \{=, \neq\}$.
3. $x_1 \ op \ x_2$ is a RC-formula over D, where $x_1, x_2 \in \mathbf{var}$ and $op \in \{=, \neq\}$.
4. $Q \ x.\, \phi$ is a RC-formula over D, where ϕ is a RC-formula over D, $x \in \mathbf{var}$, and $Q \in \{\exists, \forall\}$.
5. $\neg\phi$ is a RC-formula over D, where ϕ is a RC-formula over D.
6. $\phi \ op \ \psi$ is a RC-formula over D, where ϕ, ψ are RC-formulae over D and $op \in \{\wedge, \vee, \Rightarrow\}$.

Given a tuple $\bar{t} = \langle t_1, \ldots, t_n \rangle$, and a RC-formula $\phi(\overline{x}, \overline{y})$ with free variables $\overline{x}$ and $\overline{y}$, where $\overline{x} = \langle x_1, \ldots, x_n \rangle$, we

Query				
Boolean	Truman Model	$\supset$ (Theorem 5.4)		Non-Truman Model
	$\neg\exists$*optimal*	undecidable fragments of *RC* (Theorem 5.2)		
	$\exists$*optimal*	sufficient conditions (Lemma 5.1) *ERC* (Theorem 5.3)		
Non-Boolean	Truman Model	$\neq$ (Corollary 6.1)		Non-Truman Model
	$\neg\exists$*optimal*	undecidable fragments of *RC* (Theorem 6.2)	$\neg\exists$*optimal*	undecidable fragments of *RC* (Theorem 6.5)
	$\exists$*optimal*	sufficient conditions (Lemma 6.1) *ERC* (Theorem 6.3)	$\exists$*optimal*	sufficient conditions (Lemma 6.2) *ERC* (Theorem 6.6)

Table 1: Summary of Results

denote by $\phi[\overline{x} \mapsto \overline{t}]$ the formula obtained by replacing all the free occurrences of x_i with t_i for each $i \in \{1, \ldots, n\}$. We use a similar notation for variable substitution. We also denote by $|\overline{x}|$ the number of variables in the tuple $\overline{x}$, and by $\overline{x}^i$ the i-th element in $\overline{x}$, where $i \in \{1, \ldots, n\}$. Moreover, given a tuple of variables $\overline{x}$ without repetitions and a variable y in $\overline{x}$, we denote by $pos(\overline{x}, y)$ the position of y in $\overline{x}$, i.e., $pos(\overline{x}, y) = j$ iff $\overline{x}^j = y$. Given a formula ϕ, $free(\phi)$ denotes the set of free variables in ϕ. A *sentence* is a formula ϕ when $free(\phi) = \emptyset$.

We now introduce the existential fragment *ERC* of the relational calculus. Note that the *ERC* fragment strictly contains conjunctive queries (*CRC*).

Definition 2.2. **ERC:** Let D be a database schema. The formula $\phi(\overline{y}) := \exists\overline{x}.\, \psi(\overline{x}, \overline{y})$ is a *ERC-formula over* D iff ψ is a quantifier-free *RC*-formula over D.

For technical reasons, we also consider the Bernays Schönfinkel Ramsey fragment *BSRRC*.

Definition 2.3. **BSRRC:** Let D be a database schema. The formula $\phi(\overline{z}) := \exists\overline{x}.\, \forall\overline{y}.\, \psi(\overline{x}, \overline{y}, \overline{z})$ is a *BSRRC-formula over* D iff ψ is a quantifier-free *RC*-formula over D.

We now define boolean and non-boolean queries.

Definition 2.4. Let F be a query language and D be a database schema. A *boolean query* over D in F is an F-sentence over D. A *non-boolean query* in F is of the form $\{\overline{x} \,|\, \phi\}$, where ϕ is an F-formula over D, $\overline{x}$ is a tuple of variables, $|\overline{x}| \geq 1$, and $free(\phi) = \overline{x}$.

Without loss of generality, we will only consider non-boolean queries $\{\overline{x} \,|\, \phi\}$ where the tuple $\overline{x}$ does not contain repeated variables. The semantics of *RC* is the same as first-order logic semantics, where quantified variables range over elements of the underlying domain **dom**.

We can now define the result of a query over a state.

Definition 2.5. Let D be a database schema, $s \in \Omega_D$ be a state, and q be a query over D. We denote the evaluation of q on the state s by $[q]^s$. For a boolean query q, $[q]^s = \top$ if q is satisfied on the state s, and otherwise $[q]^s = \bot$. For a non-boolean query $q := \{\overline{x} \,|\, \phi(\overline{x})\}$, $[q]^s$ is the largest set $T \subseteq \mathbf{dom}^{|\overline{x}|}$ such that $[\phi[\overline{x} \mapsto \overline{t}]]^s = \top$ for all $\overline{t} \in T$.

In this paper, we consider only *domain independent* queries, which are queries that yield the same result on all possible underlying domains [3]. Domain independent relational calculus is as expressive as relational algebra. Although checking whether a query is domain independent is undecidable, there are various ways to obtain domain independent queries through syntactic restrictions, such as safe-range queries and range-restricted queries. Note that these syntactic approaches necessarily characterize proper subsets of the domain independent queries.

Language	$FINSAT^F$	$FINVAL^F$
RC	Undecidable	Undecidable
BSRRC	Decidable	Undecidable
ERC	Decidable	Decidable

Table 2: Decidability of $FINSAT^F$ and $FINVAL^F$

In the following, we use the term *query language* to refer to the relational calculus and its fragments, such as *ERC*. We assume that there is a unique encoding of formulae as natural numbers, and in our proofs we switch freely between formulae and their encodings.

2.2.1 Decision Problems

Here we define the finite satisfiability and finite validity decision problems for (fragments of) the relational calculus.

PROBLEM 2.1. *Let F be a query language. $FINSAT^F$ denotes the following decision problem:*
Input: *A database schema D and a sentence $\phi \in F$.*
Question: *Is there a state $s \in \Omega_D$ such that $[\phi]^s = \top$?*

PROBLEM 2.2. *Let F be a query language. $FINVAL^F$ denotes the following decision problem:*
Input: *A database schema D and a sentence $\phi \in F$.*
Question: *For all states $s \in \Omega_D$, is $[\phi]^s = \top$?*

Table 2 presents decidability results from [6] for these two problems for the query languages used in this paper.

3. SECURITY POLICY MODEL

Various Fine-Grained Access Control models for databases have been proposed in the literature [1, 7, 12, 18]. Although each has its own features, the models share common characteristics: (1) they support access control constraints at the row or column level, and (2) access control constraints are given by formulae, e.g., expressed in SQL [12], referring to the database's current state or to the user's credentials. We now describe a security policy model that captures the main features of existing fine-grained access control models.

Let F be a query language. An *F-constraint* is a pair $\langle\{\overline{x}|\psi\}, \phi\rangle$ where $\{\overline{x}|\psi\}$ is a non-boolean F-query and ϕ is an F-formula such that $free(\phi) \subseteq free(\psi)$. A *row-level F-constraint* is just an F-constraint $\langle\{\overline{x}|\psi\}, \phi\rangle$ that specifies the conditions ϕ under which we are authorized to read a tuple in the result of $\{\overline{x}|\psi\}$. A *column-level F-constraint* is a triple $\langle\{\overline{x}|\psi\}, \phi, i\rangle$ that specifies the conditions ϕ under which we are authorized to read the i-th value of a tuple in the result of $\{\overline{x}|\psi\}$, where $\langle\{\overline{x}|\psi\}, \phi\rangle$ is an F-constraint and $i \in \{1, \ldots, |\overline{x}|\}$. Note that given an F-constraint $\langle\{\overline{x}|\psi\}, \phi\rangle$, a free variable z in ϕ refers to the i-th value of the tuple on which the constraint ϕ is evaluated, where $i = pos(\overline{x}, z)$. In the following, we consider only *domain independent* F-constraints, i.e., constraints $\langle\{\overline{x}|\psi\}, \phi\rangle$ such that $\{\overline{x}|\psi \wedge \phi\}$ is domain-independent. As a result, the access control decision does not depend on the underlying domain.

The constraints are expressed using formulae that can refer to the current state and to the values of the tuple being accessed. Note that the constraints can be defined over arbitrary non-boolean queries. We can therefore restrict access not only to the data in the relation schemas but also to derived information, such as tuples in views.

A security policy in our model is defined as follows.

Definition 3.1. **Security Policy:** Let F be a query language and D be a database schema. An *F-security policy* S over D is a pair $\langle ROW, COL\rangle$, where ROW is a set of row-level F-constraints and COL is a set of column-level F-constraints, such that for any non-boolean query q: (1) there is at most one constraint in ROW associated with q, and (2) there is at most one constraint in COL associated with the i-th value of the tuples in the result of q.

Let D be a database schema, $S = \langle ROW, COL\rangle$ be a security policy over D, $\{\overline{x}|\psi\}$ be a non-boolean query, and $s \in \Omega_D$ be a state. We say that S discloses the tuple $\overline{t} \in [\{\overline{x}|\psi\}]^s$, denoted by $Disc_S(\overline{t}, \{\overline{x}|\psi\}, s)$, iff there is a constraint $\langle\{\overline{x}|\psi\}, \phi\rangle \in ROW$ such that $[\phi[\overline{x} \mapsto \overline{t}]]^s = \top$. Similarly, we say that S discloses the i-th value of the tuple $\overline{t} \in [\{\overline{x}|\psi\}]^s$, denoted by $Disc_S(\overline{t}, \{\overline{x}|\psi\}, s, i)$, iff $Disc_S(\overline{t}, \{\overline{x}|\psi\}, s)$ and there is a constraint $\langle\{\overline{x}|\psi\}, \phi, i\rangle \in COL$ such that $[\phi[\overline{x} \mapsto \overline{t}]]^s = \top$, where $i \in \{1, \ldots, |\overline{x}|\}$. Note that the authorization to read a tuple does not imply the authorization to read any of its values. Given a constraint $\langle q, \phi\rangle \in ROW$, we denote by $Auth_{S,q}(s)$ the set of tuples in $[q]^s$ disclosed by S in the state s, i.e., $Auth_{S,q}(s) := \{\overline{t} \in [q]^s | Disc_S(\overline{t}, q, s)\}$.

Example 3.1. Let D be a database schema containing only one relation schema *Employee* with *sort(Employee)*= $\{name, age, job\}$, where *name* and *job* range over strings and *age* ranges over $\mathbb{N}$. We want to prevent the access to tuples representing secret agents. Although *name* is not protected, we want to disclose only the *age* of employees over 18, and only the *job* of the *Clerks*. We also want to disclose the set of the jobs in the database. Let q be the query $\{x, y, z |\ Employee(x, y, z)\}$, and r be the query $\{z |\ \exists x, y.\ Employee(x, y, z)\}$. The security policy $S = \langle ROW, COL\rangle$ is as follows: $ROW = \{\langle q, z \neq SecretAgent\rangle, \langle r, \top\rangle\}$, and $COL = \{\langle q, \top, 1\rangle, \langle q, \bigwedge_{i\in\{1,\ldots,17\}} y \neq i, 2\rangle, \langle q, z = Clerk, 3\rangle, \langle r, \top, 1\rangle\}$. ■

Note that a policy specifies which tuples and values we are authorized to read; it does not directly specify the result of a given query. The semantics of Security-Aware Query Processing will be introduced in the following sections.

Employee		
name	age	job
John	25	Clerk
Frank	17	Admin
Jack	36	SecretAgent

S_1

Employee		
name	age	job
John	25	Clerk
Frank	16	Admin
Jack	36	SecretAgent
Carl	26	SecretAgent

S_2

Employee		
name	age	job
John	25	Clerk
Frank	19	Admin
Jack	36	SecretAgent
Carl	26	SecretAgent

S_3

Employee		
name	age	job
John	25	Clerk
Frank	17	Admin

S_4

Figure 1: Some states

We now introduce the concept of masked tuple from [19]. A *masked tuple* is a tuple where some of its values are replaced by the distinguished value †. This value prevents the disclosure of data that may be sensitive. A tuple $\overline{v}$ is *subsumed* by another tuple $\overline{t}$, written $\overline{v} \sqsubseteq \overline{t}$, iff $|\overline{v}| = |\overline{t}|$, and for all $i \in \{1, \ldots, |\overline{t}|\}$, $\overline{v}^i = \overline{t}^i$ or $\overline{v}^i = \dagger$.

Example 3.2. Let q be the query in Example 3.1. The result of q in the state s_1 in Figure 1 contains three tuples: $\overline{t} = \langle John, 25, Clerk\rangle$, $\overline{v} = \langle Frank, 17, Admin\rangle$, and $\overline{u} = \langle Jack, 36, SecretAgent\rangle$. The policy S discloses $\overline{t}$ and $\overline{v}$ but not $\overline{u}$ because Jack is a secret agent. Moreover, the tuple $\overline{v}$ is only partially disclosed. Indeed, we are authorized to read the name, but not the age or the job. The policy S also fully discloses the result of the query r, i.e., the set $\{\langle Clerk\rangle, \langle Admin\rangle, \langle SecretAgent\rangle\}$. Note that the tuple $\overline{z} = \langle Frank, \dagger, \dagger\rangle$ is subsumed by the tuple $\overline{v}$, i.e., $\overline{z} \sqsubseteq \overline{v}$. ■

Note that our security policy model uses a set semantics derived from *RC*, whereas related approaches use a multi-set semantics derived from SQL. The models in [1, 18] support only row-level constraints, those in [7, 12] support only column-level constraints, and the model in [5] supports both. Our security policy model subsumes all these approaches as well as approaches where the security policy is expressed using views. Note that our security policies can represent, by combining column-level and row-level constraints, cell-level disclosure policies [5, 12, 19].

For simplicity, we ignore users and their credentials. This neither restricts nor limits our theoretical results. Users' credentials can be modeled as a mapping *cred* from the set $\mathcal{U}$ of users to the set $\mathcal{S}$ of security policies that assigns to each user $u \in \mathcal{U}$ a security policy $S \in \mathcal{S}$ where the users' credentials are hard-coded in S using constant values.

4. SECURITY-AWARE QUERY PROCESSING

In this section, we introduce security-aware query processors, indistinguishable states, and correctness criteria for boolean queries in the Truman model.

Definition 4.1. Let D be a database schema, F be a query language, U be the set of all possible results, and $\mathcal{S}$ be the set of F-policies. An *F-Security-Aware Query Processor* (F-SAQP) is a function $\mathcal{M} : F \times \mathcal{S} \times \Omega_D \to U$.

Note that the set U depends on the type of query, i.e., boolean or non-boolean. Although boolean queries can be only true ($\top$) or false ($\bot$) in a given state, a security-aware query processor may also return the third value †, i.e., $U = \{\top, \bot, \dagger\}$. This value is used to prevent the leakage of sensitive information, which is in contrast to [19] where † is used only to mask tuples in the query's result.

$$\mathcal{M}_{opt}(\phi, S, s) := \begin{cases} \top & \text{if } \forall s' \in [\![s]\!]_{\cong_S}. [\phi]^{s'} = \top \\ \bot & \text{if } \forall s' \in [\![s]\!]_{\cong_S}. [\phi]^{s'} = \bot \\ \dagger & \text{otherwise} \end{cases}$$

Figure 2: An optimal SAQP for boolean queries

We now define indistinguishability, adapted from [19] to our setting.

Definition 4.2. Let D be a database schema, F be a query language, and $S = \langle ROW, COL \rangle$ be an F-policy over D. We say that two states s_1 and s_2 in Ω_D are *indistinguishable* according to S, written $s_1 \cong_S s_2$, iff for all $\langle q, \phi \rangle \in ROW$, there is a bijection f from $Auth_{S,q}(s_1)$ to $Auth_{S,q}(s_2)$ such that for all $\bar{t} \in Auth_{S,q}(s_1)$ and all $i \in \{1, \ldots, |\bar{t}|\}$, (1) $Disc_S(\bar{t}, q, s_1, i)$ iff $Disc_S(f(\bar{t}), q, s_2, i)$, and (2) if $Disc_S(\bar{t}, q, s_1, i)$, then $\bar{t}^i = f(\bar{t})^i$.

Example 4.1. The states s_1 and s_2 in Figure 1 are indistinguishable according to S. In contrast, $s_1 \ncong_S s_3$ because the value of the attribute *age* for *Frank* is protected in state s_1 but not in state s_3, and $s_1 \ncong_S s_4$ because the result of the query r is different in the two states. ■

Note that all the states are indistinguishable according to the empty policy $S = \langle \emptyset, \emptyset \rangle$. Given a database schema D and a policy S, the indistinguishability relation $\cong_S$ is an equivalence relation over Ω_D. Given a state $s \in \Omega_D$, the equivalence class of s defined by $\cong_S$ is denoted by $[\![s]\!]_{\cong_S}$.

We now adapt the correctness criteria given in [19] to boolean queries. A secure SAQP $\mathcal{M}$ is not influenced by data protected by a given security policy, i.e., $\mathcal{M}$'s result does not depend on undisclosed data.

Definition 4.3. Let D be a database schema and F be a query language. An F-security-aware query processor $\mathcal{M}$ is *secure* iff for all F-policies S over D, all F-queries q, and all $s, s' \in \Omega_D$, if $s \cong_S s'$, then $\mathcal{M}(q, S, s) = \mathcal{M}(q, S, s')$.

A sound SAQP $\mathcal{M}$ returns only correct information, i.e., information already returned by the original query. Note that $\mathcal{M}(q, S, s)$ may return less information than $[q]^s$, but never more information. For boolean queries, $\mathcal{M}(q, S, s)$ can return either $[q]^s$ or $\dagger$.

Definition 4.4. Let D be a database schema and F be a query language. An F-security-aware query processor $\mathcal{M}$ is *sound* iff for all queries $q \in F$, all F-policies S over D, and all $s \in \Omega_D$, $\mathcal{M}(q, S, s) = [q]^s$ or $\mathcal{M}(q, S, s) = \dagger$.

Finally, maximality formalizes that $\mathcal{M}$ returns as much information as possible. For boolean queries, a maximal SAQP, given a query q, returns $[q]^s$ for all states s where it is secure to return $[q]^s$.

Definition 4.5. Let D be a database schema and F be a query language. An F-security-aware query processor $\mathcal{M}$ is *maximal* iff for all F-security policies S over D, all F-queries q, and all $s \in \Omega_D$, if $[q]^s = [q]^{s'}$ for all $s' \in [\![s]\!]_{\cong_S}$, then $\mathcal{M}(q, S, s) = [q]^s$.

We are now ready to define optimal algorithms for boolean queries in the Truman model.

Definition 4.6. Let F be a query language. An F-SAQP for boolean queries is *optimal in the Truman model* iff it satisfies the Definitions 4.3–4.5.

Optimal algorithms are those algorithms that return as much information as possible without returning incorrect information or violating the security policy. Figure 2 describes an optimal SAQP for boolean queries. Depending on the query language, $\mathcal{M}_{opt}$ may not be computable.

Optimal algorithms are important because satisfying only two out of three correctness criteria is usually not enough. Indeed, a function $\mathcal{M}_1(q, S, s) = \dagger$ for all q, S, and s is secure and sound but is completely useless. Similarly, a function $\mathcal{M}_2(q, S, s) = [q]^s$ for all q, S, and s is sound and maximal but leaks sensitive information. Finally, a function $\mathcal{M}_3$ that satisfies the security and maximality criteria can systematically return arbitrary results without violating security and maximality. Indeed, let E be one of the partitions of Ω_D defined by $\cong_S$ such that $[q]^s \neq [q]^{s'}$ for some $s, s' \in E$, then $\mathcal{M}_3$ can return an arbitrary value for all states in E.

5. BOOLEAN QUERIES

In this section, we study the existence of optimal Security-Aware Query Processing algorithms for boolean queries.

5.1 Preliminaries

We first define the query agreement decision problem. Afterwards, we show how this problem is related to optimal Security-Aware Query Processing.

PROBLEM 5.1. *Let F be a query language. We denote by $AGREE^F$ the following decision problem:*
Input: *A database schema D, an F-security policy S over D, a boolean query $q \in F$, and a state $s \in \Omega_D$.*
Question: *For all states $s' \in [\![s]\!]_{\cong_S}$, is $[q]^{s'} = [q]^s$?*

Theorem 5.1 establishes that the decidability of $AGREE^F$ is related to the existence of optimal SAQP algorithms for the query language F.

THEOREM 5.1. *Let F be a query language. There is a computable optimal F-SAQP algorithm $\mathcal{M}$ for boolean queries iff $AGREE^F$ is decidable.*

PROOF. ($\Rightarrow$) Let $\mathcal{M}$ be a computable optimal F-SAQP algorithm for boolean queries. We use $\mathcal{M}$ as a subroutine in a decision procedure for $AGREE^F$. $AGREE^F$ takes as input a database schema D, a policy S over D, a boolean F-query q, and a state s. If $\mathcal{M}(q, S, s) = \dagger$, then $AGREE^F(D, S, q, s) = \bot$, otherwise $AGREE^F(D, S, q, s) = \top$.
($\Leftarrow$) We use $AGREE^F$ to build an optimal F-SAQP $\mathcal{M}$. Let D be a database schema. $\mathcal{M}$ takes as input an F-policy S over D, a boolean F-query q, and a state $s \in \Omega_D$. If $AGREE^F(D, S, q, s) = \top$, then $\mathcal{M}(q, S, s) = [q]^s$, otherwise $\mathcal{M}(q, S, s) = \dagger$. It easy to see that $\mathcal{M}$ is optimal and computable. □

We next study the decidability of $AGREE^F$ for RC and ERC, and extend the results to the existence of optimal algorithms for these query languages. In our theorem statements and proofs, we will ignore subrecursive complexity bounds and all reductions will be Turing reductions.

5.2 Impossibility Results

We now show that there are no optimal SAQP algorithms for boolean RC queries. Since all query languages used in practice, such as SQL, are Codd-complete, i.e., they are at least as expressive as RC, it follows from our impossibility result that there are no optimal SAQP algorithms for the boolean query languages currently in use.

THEOREM 5.2. *$AGREE^{RC}$ is undecidable.*

Proof Sketch. Let S be the empty policy $\langle \emptyset, \emptyset \rangle$ and D be a database schema. Then, $\cong_S$ defines just one equivalence class containing every state. An RC-sentence ϕ is finitely satisfiable iff $AGREE^{RC}(D, S, \phi, s) = \bot$ or $[\phi]^s = \top$, where s is the empty state. The reduction can be implemented by a total Turing machine. Since $FINSAT^{RC}$ is undecidable, so is $AGREE^{RC}$. $\square$

From Theorem 5.2, it follows that there are no optimal SAQP algorithms for boolean RC-queries. Similarly to Theorem 5.2, we can prove an even stronger result: for any fragment F of RC such that $FINSAT^F$ (or $FINVAL^F$) is undecidable, then $AGREE^F$ is also undecidable. Therefore, from well-known undecidability results for fragments of RC, one can identify fragments for which there are no optimal SAQP algorithms. Note also that considering integrity constraints or functional dependencies in a particular fragment of RC might cause the undecidability of the $AGREE$ problem, and therefore the impossibility of optimal SAQP.

5.3 Possibility Results

Although $AGREE^{RC}$ is undecidable, there are fragments of RC for which the problem is decidable. In this section, we present sufficient conditions for the decidability of $AGREE^F$, and we use these conditions to prove the decidability of $AGREE^{ERC}$. Note that we do not provide a complete classification of the fragments of RC for which $AGREE^F$ is decidable and there are fragments that meet neither the conditions for undecidability stated above nor the conditions of Lemma 5.1 below. Moreover, although we prove decidability, we do not derive complexity bounds for optimal SAQP algorithms.

We first introduce the notion of encoding the indistinguishability relation in a formula. Let D be a database schema, S be a security policy over D, and s be a state in Ω_D. We say that a sentence $\phi_{INDIST(S,s)}$ *encodes the indistinguishability relation* defined by the policy S on the state s iff $\phi_{INDIST(S,s)}$ is domain independent and for all $s' \in \Omega_D$, $[\phi_{INDIST(S,s)}]^{s'} = \top$ iff $s \cong_S s'$.

We now present sufficient conditions for the decidability of the $AGREE^F$ decision problem.

LEMMA 5.1. *Let F be a query language. $AGREE^F$ is decidable if there is a query language F' such that:*

1. *$FINSAT^{F'}$ is decidable.*
2. *For any $s \in \Omega_D$ and any F-policy S, we can compute a sentence $\phi_{INDIST(S,s)}$ that encodes the indistinguishability relation.*
3. *We can compute sentences $\gamma, \gamma' \in F'$ equivalent to $\phi_{INDIST(S,s)} \wedge \psi$ and $\phi_{INDIST(S,s)} \wedge \neg\psi$ respectively, for any $s \in \Omega_D$, any F-policy S, and any sentence $\psi \in F$.*

Proof Sketch. Let D be a database schema, $s \in \Omega_D$ be a state, ψ be an F-sentence, and S be an F-policy. Let $\gamma_\top(S, s, \psi)$ and $\gamma_\bot(S, s, \psi)$ be the two F'-sentences respectively equivalent to $\phi_{INDIST(S,s)} \wedge \neg\psi$ and $\phi_{INDIST(S,s)} \wedge \psi$. The algorithm for $AGREE^F$ computes $k = [\psi]^s$. Then, $AGREE^F(D, S, \psi, s) = \top$ iff $FINSAT^F(\gamma_k(S, s, \phi)) = \bot$. $\square$

There are fragments of RC that are not expressive enough to encode the indistinguishability relation. For this reason, in Lemma 5.1, we introduced another fragment F' that is more expressive and allows such an encoding. For instance, in the ERC fragment there is no encoding of the indistinguishability relation, but it can be encoded in the more expressive $BSRRC$ fragment.

There is a similar set of preconditions for solving $AGREE^F$ using $FINVAL^{F'}$; we just need to consider the formulae $\phi_{INDIST(S,s)} \Rightarrow \psi$ and $\phi_{INDIST(S,s)} \Rightarrow \neg\psi$ instead of the formulae $\phi_{INDIST(S,s)} \wedge \psi$ and $\phi_{INDIST(S,s)} \wedge \neg\psi$.

Before proving the decidability of $AGREE^{ERC}$, we introduce some notation. Given a policy $S = \langle ROW, COL \rangle$, a state s, a constraint $\langle q, \phi \rangle \in ROW$, and a tuple $\bar{t} \in Auth_{S,q}(s)$, we denote by $mask_{S,s,q}(\bar{t})$ the masked tuple $\overline{v}$ obtained from $\bar{t}$ by replacing all the undisclosed values with †, i.e., $|\overline{v}| = |\bar{t}|$ and for all $i \in \{1, \ldots, |\bar{t}|\}$, if $Disc_S(\bar{t}, q, s, i)$, then $\overline{v}^i = \bar{t}^i$, otherwise $\overline{v}^i = \dagger$.

Let D be a database schema, $S = \langle ROW, COL \rangle$ be a security policy over D, $\langle q, \phi \rangle$ be a constraint in ROW, and s be a state in Ω_D. The set $Ind_{S,q}(s)$ contains all the masked tuples $\bar{t}$ that can be obtained from tuples in $Auth_{S,q}(s)$ using the function $mask_{S,s,q}(\bar{t})$, i.e., $Ind_{S,q}(s) := \{mask_{S,s,q}(\bar{t}) \mid \bar{t} \in Auth_{S,q}(s)\}$. Given a tuple $\bar{t} \in Ind_{S,q}(s)$, we denote by $card_{S,s,q}(\bar{t})$ the number of tuples $\bar{t}' \in Auth_{S,q}(s)$ that cannot be distinguished from $\bar{t}$ according to the security policy S, i.e., $card_{S,s,q}(\bar{t}) := |\{\bar{t}' \in Auth_{S,q}(s) \mid mask_{S,s,q}(\bar{t}') = \bar{t}\}|$.

Let $S = \langle ROW, COL \rangle$ be a security policy, $i \in \mathbb{N}$, and $q := \{\overline{x} | \phi\}$ be a non-boolean query. We denote by ψ^S_q the condition in the constraint associated with q, i.e., $\langle q, \psi^S_q \rangle \in ROW$. Similarly, we denote by $\psi^S_{q,i}$ the condition associated with the i-th value of q, i.e., $\langle q, \psi^S_{q,i}, i \rangle \in COL$. If there is no such a constraint, then $\psi^S_q = \bot$ (respectively, $\psi^S_{q,i} = \bot$).

We now use Lemma 5.1 to prove that $AGREE^{ERC}$ is decidable. Due to the limited expressiveness of the ERC fragment, we cannot encode the indistinguishability relation in it. However, we can prove the decidability of $AGREE^{ERC}$ using the more expressive $BSRRC$ fragment.

THEOREM 5.3. *$AGREE^{ERC}$ is decidable.*

Proof Sketch. Let $S = \langle ROW, COL \rangle$ be an ERC-policy, s be a state, and $q := \{\overline{x} | \psi\}$ be a non-boolean ERC-query such that there is a constraint for q in S. The formula $\theta_{q,S}(\overline{y}, \bar{t})$, where $\overline{y}$ is a tuple of variables and $\bar{t}$ is a possibly masked tuple of values in **dom** such that $|\overline{y}| = |\bar{t}|$, is as follows:

$$\theta_{\{\overline{x}|\psi\},S}(\overline{y}, \bar{t}) := \bigwedge_{\substack{i \in \{1,\ldots,|\bar{t}|\} \\ \wedge \bar{t}^i \neq \dagger}} (\psi^S_{q,i}[\overline{x} \mapsto \overline{y}] \wedge \overline{y}^i = \bar{t}^i) \wedge \bigwedge_{\substack{i \in \{1,\ldots,|\bar{t}|\} \\ \wedge \bar{t}^i = \dagger}} \neg\psi^S_{q,i}[\overline{x} \mapsto \overline{y}].$$

The formula $\theta_{\{\overline{x}|\psi\},S}(\overline{y}, \bar{t})$ forces the masked version of the tuple associated with the values of the variables $\overline{y}$ to be $\bar{t}$.

In our encoding $\phi'_{INDIST(S,s)}$, we use counting quantifiers. A counting quantifier $\exists^{op\ m}\overline{x}.\,\phi(\overline{x})$, where $op \in \{=, \leq, \geq, <, >\}$ and $m \in \mathbb{N}$, is a quantifier that specifies the number of tuples that can be mapped to $\overline{x}$ and satisfy $\phi(\overline{x})$. For instance, the formula $\exists^{=2}x_1.\,\phi(x_1)$ is true iff there are exactly two distinct elements x_1 satisfying $\phi(x_1)$. Note that counting quantifiers do not add expressive power to RC.

Given a tuple of variables $\overline{x}$, we denote by $\overline{y}_{\overline{x}}$ the tuple of variables $y_1, \ldots, y_{|\overline{x}|}$. The encoding $\phi'_{INDIST(S,s)}$ is:

$$\psi_{S,\{\overline{x}|\psi\},s} := \exists^{=|Auth_{S,\{\overline{x}|\psi\}}(s)|}\overline{y}_{\overline{x}}.\,(\psi[\overline{x} \mapsto \overline{y}_{\overline{x}}] \wedge \psi^{S}_{\{\overline{x}|\psi\}}[\overline{x} \mapsto \overline{y}_{\overline{x}}])$$

$$\gamma_{\{\overline{x}|\psi\},S,s,\overline{t}} := \exists^{\geq card_{S,s,\{\overline{x}|\psi\}}(\overline{t})}\overline{y}_{\overline{x}}.\,(\psi[\overline{x} \mapsto \overline{y}_{\overline{x}}] \wedge \psi^{S}_{\{\overline{x}|\psi\}}[\overline{x} \mapsto \overline{y}_{\overline{x}}] \wedge \theta_{\{\overline{x}|\psi\},S}(\overline{y}_{\overline{x}}, \overline{t}))$$

$$\phi'_{INDIST(S,s)} := \bigwedge_{\langle q,\phi\rangle \in ROW} (\psi_{S,q,s} \wedge \bigwedge_{\overline{t} \in Ind_{S,q}(s)} \gamma_{q,S,s,\overline{t}}).$$

The sentence $\psi_{S,\{\overline{x}|\psi\},s}$ states that, for any state s' such that $[\psi_{S,\{\overline{x}|\psi\},s}]^{s'} = \top$, $|Auth_{S,\{\overline{x}|\psi\}}(s')| = |Auth_{S,\{\overline{x}|\psi\}}(s)|$. The sentence $\gamma_{\{\overline{x}|\psi\},S,s,\overline{t}}$ states that, for any state s' such that $[\gamma_{\{\overline{x}|\psi\},S,s,\overline{t}}]^{s'} = \top$, there are at least $card_{S,s,\{\overline{x}|\psi\}}(\overline{t})$ tuples $\overline{v}$ in $Auth_{S,\{\overline{x}|\psi\}}(s')$ such that $mask_{S,s',\{\overline{x}|\psi\}}(\overline{v}) = \overline{t}$. The sentence $\phi'_{INDIST(S,s)}$ encodes the indistinguishability relation for the ERC-policy S and the state s.

Although $\phi'_{INDIST(S,s)}$ is not in $BSRRC$, it can be rewritten as a $BSRRC$-sentence. Furthermore, there are $BSRRC$-sentences equivalent to $\phi'_{INDIST(S,s)} \wedge \psi$ and $\phi'_{INDIST(S,s)} \wedge \neg\psi$ for any ERC-sentence ψ and any ERC-policy S. Therefore, $AGREE^{RC}$ is decidable by Lemma 5.1. □

From Theorem 5.3 and $CRC \subset ERC$, it follows that $AGREE^{CRC}$ is decidable. Therefore, there are optimal SAQP algorithms for boolean conjunctive queries.

5.4 Truman and Non-Truman models

In this section, we study the connections between Truman and Non-Truman models as defined in [16]. In the Non-Truman model, the security policy is expressed using a set of authorization views. Authorization views are standard database views extended with parameters referring to users' credentials. They can be used to specify the data a user is authorized to read. Authorization views in [16] are expressed in SQL, whereas in this paper we study authorization views in the relational calculus. In the following, we ignore users' credentials, which does not limit our results.

Definition 5.1. Let D be a database schema and F be a query language. An *F-authorization view* is defined by assigning an identifier $V \in \mathbf{relname} \setminus D$ to a non-boolean F-query $\{\overline{x}|\phi(\overline{x})\}$, i.e., $V := \{\overline{x}|\phi(\overline{x})\}$.

Let $s \in \Omega_D$ be a state and $V := \{\overline{x}|\phi(\overline{x})\}$ be a view. The *materialization* of V in s, denoted by V_s, is $[\{\overline{x}|\phi(\overline{x})\}]^s$. Views can be used in RC-formulae in the same way as relation schemas, but in this case we consider the materialized views instead of the relation instances.

We now introduce the notion of equivalence with respect to a set of authorization views AV (AV-equivalence) defined in [16]. Two states are *AV-equivalent* iff their views' materializations are the same.

Definition 5.2. Let D be a database schema, AV be a set of authorization views over D, and $s, s' \in \Omega_D$ be two states. Then, s and s' are *AV-equivalent*, written $s \cong_{AV} s'$, iff for all $V \in AV$, $V_s = V_{s'}$.

For any database schema D and any set of views AV, the relation $\cong_{AV}$ is an equivalence relation over Ω_D.

We now introduce *row-level policies*. Intuitively, a row-level policy does not disclose partial tuples.

Definition 5.3. Let D be a database schema and $S = \langle ROW, COL\rangle$ be a security policy over D. S is a *row-level policy* iff for all $\langle q, \phi\rangle \in ROW$, all $s \in \Omega_D$, and all $\overline{t} \in [q]^s$, if $Disc_S(\overline{t}, q, s)$, then $Disc_S(\overline{t}, q, s, i)$ for all $i \in \{1, \ldots, |\overline{t}|\}$.

We say that a security policy S and a set of views AV are *equivalent* iff $\cong_{AV} = \cong_S$. We prove below that RC-authorization views are as expressive as row-level RC-policies. It is easy to see that row-level RC-policies are strictly less expressive than RC-policies, and therefore RC-authorization views are strictly less expressive than RC-policies.

PROPOSITION 5.1. *Let D be a database schema. For each set of RC-authorization views over D, there is an equivalent row-level RC-security policy over D and vice versa.*

Proof Sketch. We can use the formulae defining the views to define an equivalent row-level policy. Conversely, we can use the constraints in a row-level policy to define an equivalent set of views. □

Example 5.1. Let AV be the set of views $\{q_1, q_2\}$, where q_1 is the query $\{z \mid \exists x, y.\, Employee(x, y, z)\}$ and q_2 is the query $\{y \mid \exists x, z.\, Employee(x, y, z)\}$. The equivalent row-level policy is $S := \langle\{\langle q_1, \top\rangle, \langle q_2, \top\rangle\}, \{\langle q_1, \top, 1\rangle, \langle q_2, \top, 1\rangle\}\rangle$.

Let S be the row-level policy $\langle\{\langle q_1, \phi\rangle, \langle q_2, \psi\rangle\}, \{\langle q_1, \top, 1\rangle, \langle q_2, \top, 1\rangle\}\rangle$, where ϕ and ψ are RC-formulae. The equivalent set of views is $\{q'_1, q'_2\}$, where $q'_1 := \{z \mid \exists x, y.\, Employee(x, y, z) \wedge \phi\}$ and $q'_2 := \{y \mid \exists x, z.\, Employee(x, y, z) \wedge \psi\}$. ■

In the relational calculus, authorization views are strictly less expressive than security policies. This is no longer the case for sufficiently powerful query languages. For instance, in the relational calculus extended with the **count** aggregation operator, authorization views are as expressive as security policies, as shown in Example 5.2. This also implies that, in contrast to the case of RC, for sufficiently powerful languages, row-level policies are as expressive as policies that combine both row-level and column-level constraints.

Example 5.2. In this example, we use the aggregation operator **count** [3]. Intuitively, $\mathbf{count}[\overline{x}|\psi(\overline{x})]$ returns the number of tuples in the result of the query $\{\overline{x}|\psi(\overline{x})\}$.

Let q be the query $\{x, z|\,\alpha\}$, where α is the RC-formula $\exists y.\, Employee(x, y, z)$, and let S be the policy $\langle\{\langle q, \phi\rangle\}, \{\langle q, \psi, 1\rangle, \langle q, \gamma, 2\rangle\}\rangle$, where ϕ, ψ, and γ are RC-formulae. The set of authorization views AV that is equivalent to S is:

$$\{x, z, s \mid \alpha \wedge \phi \wedge \psi \wedge \gamma \wedge s = 1\},$$
$$\{x, s \mid \exists z.\,(\alpha \wedge \phi \wedge \psi \wedge \neg\gamma) \wedge s = \mathbf{count}[z|\alpha \wedge \phi \wedge \psi \wedge \neg\gamma]\},$$
$$\{z, s \mid \exists x.\,(\alpha \wedge \phi \wedge \gamma \wedge \neg\psi) \wedge s = \mathbf{count}[x|\alpha \wedge \phi \wedge \gamma \wedge \neg\psi]\},$$
$$\{s \mid \exists x, z.\,(\alpha \wedge \phi \wedge \neg\psi \wedge \neg\gamma) \wedge s = \mathbf{count}[x, z|\alpha \wedge \phi \wedge \neg\psi \wedge \neg\gamma]\}.$$

Let z be a state. The materialization of the views in AV in the state z discloses, for each $\overline{t} \in Ind_{S,q}(z)$, the values in $\overline{t}$

different from † and the value of $card_{S,z,q}(\bar{t})$, which is stored in the variable s. Therefore, $\cong_{AV} = \cong_S$. ■

From Proposition 5.1, it follows that, given a set of authorization views AV, two states are AV-equivalent iff they are indistinguishable according to the equivalent security policy and vice versa.

We now introduce the concept of conditional validity [16], which is different from validity in *FOL*. Afterwards, we define optimal algorithms in the Non-Truman model.

Definition 5.4. Let D be a database schema, $s \in \Omega_D$ be a state, and AV be a set of authorization views over D. An RC-query q is *conditionally valid* in s iff there is another RC-query q' written only in terms of the views in AV that is equivalent to q on all the states s' such that $s \cong_{AV} s'$.

Definition 5.5. Let F be a query language. An F-SAQP $\mathcal{M}$ is *optimal in the Non-Truman model* iff it executes exactly the conditionally valid queries.

Before reconciling Truman and Non-Truman models, we state the following lemma.

LEMMA 5.2. *Let D be a database schema, F be a query language, AV be a set of F-authorization views, $s \in \Omega_D$ be a state, and ϕ be a boolean F-query. Moreover, let S be the F-security policy equivalent to AV. ϕ is conditionally valid in the state s with respect to AV iff $AGREE^F(D, S, \phi, s) = \top$.*

From Theorem 5.1, Proposition 5.1, and Lemma 5.2, it follows:

THEOREM 5.4. *Let F be a query language. An optimal F-SAQP for boolean queries in the Truman model is an optimal F-SAQP for boolean queries in the Non-Truman model when † is interpreted as rejecting the query.*

In the past, Truman and Non-Truman models have been presented as two distinct and alternative approaches: the former concerned with returning as much information as possible and the latter concerned with avoiding inconsistencies. For boolean RC-queries, Theorem 5.4 shows that an optimal SAQP in the Truman model can be used as an optimal SAQP in the Non-Truman model. The reason is that, for boolean queries, the only way to protect sensitive information is to return † and there is no way to return partial results. Therefore, for boolean RC-queries, the Non-Truman model is a special case of the Truman model. Indeed, RC-authorization views are strictly less expressive than RC-policies. For sufficiently powerful query languages, such as the relational calculus extended with the **count** operator, the two models coincide for boolean queries because authorization views are as expressive as security policies.

The practical consequence of Theorem 5.4 is that we can straightforwardly use an optimal algorithm in the Truman model to compute the answer to a query in the Non-Truman model. It also follows that the results in Sections 5.2 and 5.3 apply to the Non-Truman model and to the conditional validity problem.

6. NON-BOOLEAN QUERIES

In this section, we study the existence of optimal SAQP algorithms for non-boolean queries and the connections between optimal SAQP algorithms in the Truman and Non-Truman models.

6.1 Correctness Criteria

The answer to a non-boolean query under a given security policy is a multiset of masked tuples and not a set of tuples as in standard database theory. Indeed, some values may be set to † because we are not authorized to read them. However, not all multisets of tuples are valid results for SAQP algorithms. We are interested only in those multisets that can be obtained from a set of unmasked tuples by replacing values with †. The set M is the set of all multisets of tuples V such that there is a set of unmasked tuples T and a bijection f from V to T such that $\bar{t} \sqsubseteq f(\bar{t})$ for all $\bar{t} \in V$. In the following, we always refer just to multisets in M.

Following [19], we define a subsumption relation $\preceq$ on multisets, which is a partial order on M. A multiset $K \in M$ is subsumed by another multiset $K' \in M$, written $K \preceq K'$, iff there is an injective mapping $f : K \to K'$ such that for all $\bar{t} \in K$, $\bar{t} \sqsubseteq f(\bar{t})$.

Example 6.1. Let $\bar{t}$ and $\bar{z}$ be the tuples $\langle John, 25, Clerk\rangle$ and $\langle Frank, \dagger, \dagger\rangle$. The multiset $\{\bar{t}, \bar{t}, \bar{z}\}$ is not in M because there are two occurrences of the unmasked tuple $\bar{t}$. In contrast, the multiset $J := \{\bar{z}, \bar{z}, \bar{t}\}$ is in M. Let T be the set $\{\langle Frank, 46, Clerk\rangle, \langle Frank, 27, SecretAgent\rangle, \langle John, 25, Clerk\rangle\}$. Then, $J \preceq T$. ■

We now introduce the correctness criteria for non-boolean queries. Security is the same as in Definition 4.3 and we state here only soundness and maximality. Wang et al. [19] studied only secure and sound algorithms and considered only algorithms that return a unique multiset of masked tuples as a query's result. However, there are cases in which the optimal answer is not unique, i.e., there are finitely many different multisets of masked tuples that are all optimal and incomparable. In the following, we consider algorithms that might return as a query's result a set of multisets. Therefore, for non-boolean queries, the set U, which contains all possible results of optimal SAQPs, is $\mathbb{P}(M)$. For this reason, we must modify the criteria given in [19].

For non-boolean queries, a sound algorithm must return a result subsumed by the original query's result.

Definition 6.1. Let D be a database schema and F be a query language. An F-security-aware query processor $\mathcal{M}$ is *sound* iff for all non-boolean queries $q \in F$, all F-policies S over D, all $s \in \Omega_D$, and all $V \in \mathcal{M}(q, S, s)$, $V \preceq [q]^s$.

A maximal algorithm must return a result that subsumes any multiset $T \in M$ that is subsumed by the original query's result in all indistinguishable states.

Definition 6.2. Let D be a database schema and F be a query language. An F-security-aware query processor $\mathcal{M}$ is *maximal* iff for all F-policies S over D, all non-boolean queries $q \in F$, all $s \in \Omega_D$, and all $T \in M$, if $T \preceq [q]^{s'}$ for all $s' \in [\![s]\!]_{\cong_S}$, then there is a $V \in \mathcal{M}(q, S, s)$ such that $T \preceq V$.

We now define optimal algorithms for non-boolean queries in the Truman model.

Definition 6.3. Let F be a query language. An F-SAQP for non-boolean queries is *optimal in the Truman model* iff it satisfies the Definitions 4.3, 6.1, and 6.2.

Figure 3 describes an optimal SAQP for non-boolean queries. Depending on the query language, this function may not be computable.

$$\mathcal{M}_{opt}(q,S,s) := \{V \in M \mid \forall s' \in [\![s]\!]_{\cong_S}.\ (V \preceq [q]^{s'})\}$$

Figure 3: An optimal SAQP for non-boolean queries

Example 6.2. Let D be a database schema with only one relation schema R with two attributes a and b that range over $\mathbb{N}$. Moreover, let $\phi(n,m)$, where $n,m \in \mathbb{N}$, be the formula $x = n \wedge y = m \wedge \neg\exists y.\,(R(x,y) \wedge y \neq m)$. Let q be the query $\{a,b \mid R(a,b)\}$ and S be the security policy $\langle\{\langle q,\top\rangle\},\{\langle q,\psi,1\rangle,\langle q,\psi,2\rangle\}\rangle$, where ψ is the formula $\neg\phi(1,3) \wedge \neg\phi(1,4) \wedge \neg\phi(2,3) \wedge \neg\phi(2,4)$. One of the equivalence classes defined by S contains exactly the two states s and s' such that $s(R) = \{\langle 1,3\rangle,\langle 2,4\rangle\}$ and $s'(R) = \{\langle 1,4\rangle,\langle 2,3\rangle\}$. In this case, $\mathcal{M}_{opt}(q,S,s) = \mathcal{M}_{opt}(q,S,s') = K$ and $K = \{V \in M \mid V \preceq \{\langle 1,\dagger\rangle,\langle 2,\dagger\rangle\} \vee V \preceq \{\langle\dagger,3\rangle,\langle\dagger,4\rangle\}\}$. Therefore, in s and s', we are authorized to read the values of the attributes a and b separately, but not together. ■

In the Truman model, boolean queries are not a special case of non-boolean queries. For fragments that are not closed under negation, we cannot use an optimal SAQP $\mathcal{M}$ for non-boolean queries to distinguish whether a sentence returns $\bot$ or $\dagger$ because $\mathcal{M}$ returns $\{\emptyset\}$ in both cases.

We now define a decision problem, denoted as *SUBSUME*, that will be used in a way similar to $AGREE^F$.

PROBLEM 6.1. *Let F be a query language. We denote by $SUBSUME^F$ the problem:*

Input: *A database schema D, an F-security policy S, a non-boolean F-query $q := \{\overline{x} \mid \phi(\overline{x})\}$, a multiset $T \in M$ such that $|\bar{t}| = |\overline{x}|$ for all $\bar{t} \in T$, and a state $s \in \Omega_D$.*

Question: *For all states $s' \in [\![s]\!]_{\cong_S}$, is $T \preceq [q]^{s'}$?*

The $SUBSUME^F$ decision problem is related to the existence of optimal SAQP algorithms.

THEOREM 6.1. *Let F be a query language. There is a computable optimal F-SAQP algorithm $\mathcal{M}$ for non-boolean queries iff $SUBSUME^F$ is decidable.*

PROOF. ($\Rightarrow$) Let $\mathcal{M}$ be a computable optimal F-SAQP algorithm for non-boolean queries. We use $\mathcal{M}$ as a subroutine in a decision procedure for $SUBSUME^F$. Let D, S, q, T, and s be the inputs of the $SUBSUME^F$ problem. Then, $SUBSUME^F(D,S,q,T,s) = \top$ iff there is a $V \in \mathcal{M}(q,S,s)$ such that $T \preceq V$.

($\Leftarrow$) We use $SUBSUME^F$ to build an optimal F-SAQP $\mathcal{M}$. Let D be a database schema, S be an F-security policy over D, $s \in \Omega_D$ be a state, and q be a non-boolean F-query. $\mathcal{M}(q,S,s)$ is $\{T \in M \mid T \preceq [q]^s \wedge SUBSUME^F(D,S,q,T,s) = \top\}$. It is easy to see that $\mathcal{M}$ is optimal and computable. □

The $SUBSUME^F$ decision problem plays the same role for optimal algorithms for non-boolean queries as $AGREE^F$ does for boolean queries. We therefore now analyze the decidability of $SUBSUME^F$.

6.2 Impossibility Results

In this section, we extend the impossibility results in Section 5.2 from boolean RC-queries to non-boolean RC-queries.

THEOREM 6.2. *$SUBSUME^{RC}$ is undecidable.*

Proof Sketch. Let D be a database schema. We define a new schema D' obtained from D by adding a new relation schema V with one attribute at. Let v be a value in $Dom(at)$, s be the state in $\Omega_{D'}$ such that $s(V) = \{\langle v\rangle\}$ and $s(R) = \emptyset$ for all $R \in D$, and S be the RC-policy $\langle\{\langle\{x|V(x)\},\top\rangle\},\{\langle\{x|V(x)\},\top,1\rangle\}\rangle$. Then, a RC-sentence ϕ over D is finitely valid iff $SUBSUME^{RC}(D',S,\{x|V(x) \wedge \phi\},\{\langle v\rangle\},s) = \top$. This reduction can be implemented by a total Turing machine. Therefore, since $FINVAL^{RC}$ is undecidable, so is $SUBSUME^{RC}$. □

From Theorem 6.2, it follows that, for non-boolean queries, there are no computable optimal SAQP algorithms in the Truman model for the relational calculus. Therefore, it is impossible to securely process queries without either violating the security policy, losing some information, or returning incorrect results.

Note that, following the proof of Theorem 6.2, we can prove an even stronger result: for any fragment F of RC that is closed under conjunction such that $FINVAL^F$ is undecidable, then $SUBSUME^F$ is also undecidable. Therefore, from well-known undecidability results for fragments of first-order logic, we can identify fragments of RC for which there are no optimal algorithms for non-boolean queries, such as the $BSRRC$ fragment.

6.3 Possibility Results

We now present a general criterion for identifying fragments of RC where $SUBSUME^F$ is decidable.

We first introduce the notion of encoding the subsumption relation $\preceq$ in a formula. Let D be a database schema, $\psi(\overline{x})$ be a formula with free variables $\overline{x}$, and $T \in M$ be a multiset such that $|\bar{t}| = |\overline{x}|$ for all $\bar{t} \in T$. We say that a sentence $\phi_{T,\psi(\overline{x})}$ *encodes the subsumption relation between T and $\psi(\overline{x})$* iff $\phi_{T,\psi(\overline{x})}$ is domain independent and for all $s \in \Omega_D$, $[\phi_{T,\psi(\overline{x})}]^s = \top$ iff $T \preceq [\{\overline{x} \mid \psi(\overline{x})\}]^s$.

In the following lemma, we present sufficient conditions for the decidability of the $SUBSUME^F$ decision problem.

LEMMA 6.1. *Let F be a query language. $SUBSUME^F$ is decidable if there is a query language F' such that:*

1. *$F \subseteq F'$,*
2. *$AGREE^{F'}$ is decidable for all F-policies, and*
3. *for any multiset $T \in M$ and any F-formula $\psi(\overline{x})$, we can compute a sentence $\phi_{T,\psi(\overline{x})} \in F'$ that encodes the subsumption relation between T and $\psi(\overline{x})$.*

Proof Sketch. Let D be a database schema, $s \in \Omega_D$ be a state, $\{\overline{x} \mid \psi\}$ be an F-query, S be an F-policy, and $T \in M$ be a multiset of tuples. Let $\phi_{T,\psi}$ be the F'-sentence encoding the subsumption relation. Then, $SUBSUME^F(D,S,\{\overline{x} \mid \psi\},T,s) = \top$ iff $[\phi_{T,\psi}]^s = \top$ and $AGREE^{F'}(D,S,\phi_{T,\psi},s) = \top$. □

We now use Theorem 5.3 and Lemma 6.1 to prove the decidability of $SUBSUME^{ERC}$. From Theorem 6.3, it follows that there are optimal SAQP algorithms for the existential fragment of the relational calculus.

THEOREM 6.3. *$SUBSUME^{ERC}$ is decidable.*

Proof Sketch. Given a multiset of tuples T in M, and a tuple $\bar{t} \in T$, let $K_{\bar{t},T}$ be the multiset $\{\bar{t}' \mid \bar{t}' \in T \wedge \bar{t} \sqsubseteq \bar{t}'\}$.

$$\mathcal{M}_{s\text{-}opt}(q,S,s) := \begin{cases} [q]^s & \text{if } \forall s' \in [\![s]\!]_{\cong_S}.\, [q]^{s'} = [q]^s \\ \dagger & \text{otherwise} \end{cases}$$

Figure 4: A strongly-optimal SAQP for non-boolean queries

The encoding is given by:

$$\phi_{T,\psi(\overline{x})} := \bigwedge_{\overline{t}\in T} \exists^{\geq |K_{\overline{t},T}|} \overline{x}.\, (\psi(\overline{x}) \wedge \bigwedge_{i\in\{1,\ldots,|\overline{x}|\}\wedge \overline{t}^i \neq \dagger} \overline{x}^i = \overline{t}^i).$$

$\phi_{T,\psi(\overline{x})}$ can be equivalently rewritten as an ERC-sentence. □

Since ERC strictly contains conjunctive queries, it follows that $SUBSUME^{CRC}$ is decidable.

6.4 Truman and Non-Truman models

We now study the connections between Truman and Non-Truman models for non-boolean queries. We first introduce the notion of *strongly-optimal* security-aware query processor. Using strongly-optimal SAQPs, we generalize the Non-Truman model approach for non-boolean queries [16] to our security policies, which are more expressive than authorization views. Afterwards, we study the relationships between strongly-optimal SAQPs and optimal SAQPs in the Non-Truman model and extend our decidability results.

A strongly-optimal SAQP for non-boolean queries returns the original query's result whenever it is secure to do so, and otherwise returns †. Let L be the set of all finite sets of unmasked tuples. A security-aware query processor for non-boolean queries is called *strongly-optimal* if it satisfies the Definitions 4.3–4.5 and the set U, containing all the possible results, is $L \cup \{\dagger\}$. Figure 4 describes a strongly optimal SAQP for non-boolean queries. Depending on the query language, this function may not be computable.

Optimal SAQPs and strongly-optimal SAQPs are distinct. For instance, optimal SAQPs can return partial results while strongly-optimal SAQPs cannot. This holds even if we consider just row-level policies. Note too that, in the Non-Truman model, boolean queries are a special case of non-boolean queries.

We first define a new decision problem corresponding to $SUBSUME^F$.

PROBLEM 6.2. *Let F be a query language. $EQUAL^F$ denotes the problem:*

Input: *A database schema D, an F-security policy S, a non-boolean F-query $q := \{\overline{x}|\phi(\overline{x})\}$, a set of tuples $T \in L$ such that $|\overline{t}| = |\overline{x}|$ for all $\overline{t} \in T$, and a state $s \in \Omega_D$.*

Question: *For all states $s' \in [\![s]\!]_{\cong_S}$, is $T = [q]^{s'}$?*

Similarly to Theorem 5.1 and Theorem 6.1, we can prove:

THEOREM 6.4. *Let F be a query language. There is a computable strongly-optimal F-SAQP algorithm $\mathcal{M}$ for non-boolean queries iff $EQUAL^F$ is decidable.*

We can easily adapt the proof in Section 6.2 to obtain the following undecidability result:

THEOREM 6.5. *$EQUAL^{RC}$ is undecidable.*

Therefore, even strongly-optimal Security-Aware Query Processing is impossible for the relational calculus. Furthermore, $EQUAL^F$ is undecidable for any fragment F such that either $FINVAL^F$, $FINSAT^F$, or $AGREE^F$ are undecidable.

We can also provide some decidability results. We first define a new encoding. Let D be a database schema, $\psi(\overline{x})$ be a formula with free variables $\overline{x}$, and $T \in L$ be a finite set of tuples such that $|\overline{t}| = |\overline{x}|$ for all $\overline{t} \in T$. We say that a sentence $\phi_{T,\psi(\overline{x})}$ *encodes the property $\psi(\overline{x})$ satisfied by the set T* iff $\phi_{T,\psi(\overline{x})}$ is domain independent and for all $s \in \Omega_D$, $[\phi_{T,\psi(\overline{x})}]^s = \top$ iff $T = [\{\overline{x}|\psi(\overline{x})\}]^s$.

We can now give sufficient conditions for the decidability of the $EQUAL^F$ problem.

LEMMA 6.2. *Let F be a query language. $EQUAL^F$ is decidable if there is a query language F' such that:*

1. *$F \subseteq F'$,*
2. *$AGREE^{F'}$ is decidable for all F-policies, and*
3. *for any finite set $T \in L$ and any formula $\psi(\overline{x}) \in F$, we can compute a sentence $\phi_{T,\psi(\overline{x})} \in F'$ that encodes the property $\psi(\overline{x})$ satisfied by the set T.*

Using Lemma 6.2, we prove the following result.

THEOREM 6.6. *$EQUAL^{ERC}$ is decidable.*

Similarly to Section 5.4, we have:

LEMMA 6.3. *Let D be a database schema, F be a query language, AV be a set of F-authorization views, $s \in \Omega_D$ be a state, and q be a non-boolean F-query. Moreover, let S be the F-security policy equivalent to AV. The query q is conditionally valid in the state s with respect to AV iff $EQUAL^F(D, S, q, [q]^s, s) = \top$.*

THEOREM 6.7. *Let F be a query language. A strongly-optimal F-SAQP for non-boolean queries is an optimal F-SAQP for non-boolean queries in the Non-Truman model when † is interpreted as rejecting the query.*

For non-boolean queries, strongly-optimal algorithms and optimal algorithms are distinct. Therefore, from Theorem 6.7, we have:

COROLLARY 6.1. *There is a non-boolean RC-query q, a state s, a set AV of RC-views, and the equivalent row-level RC-policy S, such that the result of an optimal SAQP for non-boolean queries in the Non-Truman model is different from the result of an optimal SAQP for non-boolean queries in the Truman model.*

For boolean RC-queries, optimal algorithms in the Non-Truman model are a special case of optimal algorithms in the Truman model. Corollary 6.1 shows that this result does not hold for non-boolean queries. In this case, the two models are distinct.

The reason for this difference is that in the Truman model we can return partial results, whereas in the Non-Truman model we either return the query's result or reject the query. Moreover, given the same inputs, the result of an optimal SAQP in one model does not provide any insights about the result of an optimal SAQP in the other model. For instance, if an optimal SAQP in the Non-Truman model rejects a query q, we do not know anything about the result in the Truman model. Similarly, if an optimal SAQP in

the Truman model returns $\{\emptyset\}$ as a query's result, we do not know whether an optimal SAQP in the Non-Truman model accepts the query or rejects it. Hence, we cannot use optimal SAQPs in the Truman model as optimal SAQPs in the Non-Truman model and vice versa.

Theorem 6.7 shows that, for RC, optimal SAQPs in the Non-Truman model are a special case of strongly-optimal SAQPs. From this, it follows that Lemma 6.2 and Theorem 6.6 apply to optimal SAQPs in the Non-Truman model, and to the conditional validity problem. Note too that, for sufficiently powerful query languages, optimal SAQPs in the Non-Truman model and strongly-optimal SAQPs are equivalent modulo the interpretation of †.

7. RELATED WORK

In this section we review previous work on Fine-Grained Access Control and on other relevant topics.

Security-Aware Query Processing Security-Aware Query Processing algorithms are implemented in commercial databases [1, 7, 18]. Despite that, only limited work has been done on theoretical aspects of this problem. In [16], Rizvi et al. proposed the notions of Truman and Non-Truman models. They provided inference rules, which are sound but not complete, for determining whether a query is conditionally valid. The undecidability of the unconditional validity problem follows from well-known results on query rewriting using views [14]. Zhang et al. [20] studied the conditional validity problem for conjunctive queries and showed that it is decidable. We improve these results in that we provide sufficient conditions for the decidability of conditional validity. We also show that this problem is decidable for the existential fragment of the relational calculus, which contains conjunctive queries, and for our security policies, which are more expressive than authorization views.

Wang et al. [19] were the first to propose correctness criteria for algorithms in the Truman model. They proposed a secure and sound SAQP algorithm. Other secure and sound algorithms have been proposed since then, such as [10, 17]. Our results prove the claim of Wang et al. that optimal Security-Aware Query Processing is difficult. We also prove that optimal SAQP in the Truman model is possible for the existential fragment of the relational calculus.

Instance-based Determinacy The instance-based determinacy problem [11] consists of checking whether, given a database state s, a set of views V, and a query q, the materialization of the views in V in the state s fully determines the result of q in s. Koutris et al. [11] proved that the problem of instance-based determinacy for unions of conjunctive queries under the set semantics is decidable and is *coNP-complete* in terms of data complexity.

When restricted to row-level policies, which are equivalent to RC-views, the $AGREE$ and $EQUAL$ problems are equivalent to the instance-based determinacy problem for boolean and non-boolean queries respectively. In general, these equivalences do not hold because security policies are more expressive than RC-views and, therefore, $AGREE$ and $EQUAL$ are more general than instance-based determinacy under the set semantics. Indeed, masked tuples introduce a kind of bag semantics that cannot be captured using RC-views under the set semantics. We are not aware of any work exploring instance-based determinacy under the bag semantics.

Certain Answers The problem of computing *certain answers* using views under the closed world assumption [2] shares similarities with optimal SAQP. But they have important differences. For instance, the result of a boolean query ϕ according to optimal SAQP is one of $\{\top, \bot, \dagger\}$ whereas the certain answer is one of $\{\top, \bot\}$. Indeed, one must compute both the certain answer and the possible answer to compute the result of boolean optimal SAQP.

For non-boolean queries, optimal algorithms in the Truman model return a set of results, whereas the certain answer is unique. Moreover, while optimal SAQP in the Truman model considers masked tuples, the certain answer problem considers only unmasked tuples. This seriously limits Fine-Grained Access Control. For example, suppose we have a policy with a constraint on the i-th value of a query $q := \{\overline{x}|\psi\}$, for some $i \in \{1, \ldots, |\overline{x}|\}$. The certain answer of q will be $\emptyset$, whereas the result of an optimal SAQP will be a multiset of masked tuples where the i-th value is replaced with †. In the Non-Truman model, an algorithm either returns the query's result or rejects the query. In contrast, the certain answer to a query is generally different from the query's result.

The main difference between optimal SAQP and the certain answer problem is that security policies are more expressive than views in the relational calculus. As we previously noted, the presence of masked tuples introduces a kind of bag semantics. However, while the problem of querying views has been studied extensively under the set semantics [2, 14], only limited work have been done under the bag-set and bag semantics [4]. Note that our work can be viewed in the general setting of querying views under the bag-set semantics. Despite that, we decided to keep the terminology of Security-Aware Query Processing for consistency with previous works on Fine-Grained Access Control in databases, e.g., [16, 19].

Another related problem is computing the certain answer to a query in an incomplete database [13]. The same considerations for certain answers using views apply to this case. Moreover, while the notion of indistinguishability appears related to the semantics of incomplete databases [13], in general this is not the case.

8. CONCLUSIONS

We presented the first analysis of optimal Security-Aware Query Processing. Our results show that (1) it is impossible to build optimal SAQP algorithms for Codd-complete query languages, such as SQL, and (2) this impossibility is not due to specific characteristics of these query languages but rather to the undecidability of the relational calculus and several of its fragments. Note that our results also do not depend on the particular characteristics of our security model. We showed that there are interesting fragments of RC for which optimal Security-Aware Query Processing is possible, such as the existential fragment of RC. Our results may be used to prove the decidability of optimal Security-Aware Query Processing for other fragments of RC, such as the monadic fragment and the guarded fragment, and other query languages.

For boolean queries, we showed that, for the relational calculus, optimal SAQP in the Non-Truman model is a special case of optimal SAQP in the Truman model, and that optimal algorithms in the two models coincide for the relational calculus extended with aggregation operators. In

contrast, for non-boolean queries, optimal algorithms in the two models are distinct. This has direct consequences for developing algorithms for those models.

Optimal Security-Aware Query Processing is a difficult problem: it is intractable even for conjunctive queries. Indeed, it is *coNP-complete* in terms of data complexity for boolean conjunctive queries and row-level security policies [11]. Despite that, optimal SAQP can still have practical applications. For instance, there are fragments of conjunctive queries for which optimal SAQP in the Non-Truman model is in *PTIME* in terms of data complexity for row-level policies [11]. This suggests that there might be other fragments of conjunctive queries for which optimal SAQP is tractable.

Non-optimal SAQP algorithms can benefit from efficient optimal algorithms for some special cases. In this way, we can use tractable optimal algorithms when it is possible, and fall back to efficient non-optimal algorithms for the cases where optimal SAQP is undecidable or intractable. Furthermore, the study of optimal SAQP may shed some light on the trade-offs between efficiency and optimality, and can therefore lead to improvements for non-optimal algorithms.

In the future, we plan to study the complexity of optimal Security-Aware Query Processing. We also aim to identify fragments of the relational calculus with efficient optimal algorithms and to implement a prototype of an optimal Security-Aware Query Processor for these fragments.

Acknowledgments. The authors would like to thank Jannik Dreier, Christoph Sprenger, Mohammad Torabi Dashti, Eugen Zalinescu, as well as the anonymous reviewers for their comments and suggestions. This work is partially supported by the EU FP7-ICT-2009.1.4 Project No. 256980, NESSoS: Network of Excellence on Engineering Secure Future Internet Software Services and Systems.

9. REFERENCES

[1] New Security Features in Sybase Adaptive Server Enterprise. *Sybase Technical White Paper*, 2003.

[2] S. Abiteboul and O. M. Duschka. Complexity of answering queries using materialized views. In *Proceedings of the 17th Symposium on Principles of Database Systems*, pages 254–263. ACM, 1998.

[3] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of databases*, volume 8. Addison-Wesley, 1995.

[4] F. Afrati, R. Chirkova, M. Gergatsoulis, and V. Pavlaki. View selection for real conjunctive queries. *Acta Inf.*, 44(5):289–321, Aug. 2007.

[5] R. Agrawal, P. Bird, T. Grandison, J. Kiernan, S. Logan, and W. Rjaibi. Extending relational database systems to automatically enforce privacy policies. In *Proceedings of the 21st International Conference on Data Engineering*, pages 1013–1022. IEEE, 2005.

[6] E. Börger, E. Grädel, and Y. Gurevich. *The classical decision problem*. Springer Verlag, 2001.

[7] K. Browder and M. Davidson. The virtual private database in Oracle9iR2. *Oracle Technical White Paper, Oracle Corporation*, 500, 2002.

[8] E. F. Codd. *Relational completeness of data base sublanguages*. IBM Corporation, 1972.

[9] E. Damiani, M. Fansi, A. Gabillon, and S. Marrara. A general approach to securely querying XML. *Computer standards & interfaces*, 30(6):379–389, 2008.

[10] R. Halder and A. Cortesi. Fine grained access control for relational databases by abstract interpretation. In *Software and Data Technologies*, volume 170, pages 235–249. Springer, 2013.

[11] P. Koutris, P. Upadhyaya, M. Balazinska, B. Howe, and D. Suciu. Query-based data pricing. In *Proceedings of the 31st Symposium on Principles of Database Systems*, pages 167–178. ACM, 2012.

[12] K. LeFevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan, Y. Xu, and D. DeWitt. Limiting disclosure in hippocratic databases. In *Proceedings of the 30th International Conference on Very Large Data Bases*, pages 108–119. VLDB Endowment, 2004.

[13] L. Libkin. Incomplete information and certain answers in general data models. In *Proceedings of the 30th Symposium on Principles of Database Systems*, pages 59–70. ACM, 2011.

[14] A. Nash, L. Segoufin, and V. Vianu. Views and queries: Determinacy and rewriting. *ACM Transactions on Database Systems*, 35(3):21, 2010.

[15] S. Oulmakhzoune, N. Cuppens-Boulahia, F. Cuppens, and S. Morucci. fQuery: SPARQL query rewriting to enforce data confidentiality. In *Data and Applications Security and Privacy*, pages 146–161. Springer, 2010.

[16] S. Rizvi, A. Mendelzon, S. Sudarshan, and P. Roy. Extending query rewriting techniques for fine-grained access control. In *Proceedings of the 31st International Conference on Management of Data*, pages 551–562. ACM, 2004.

[17] J. Shi, H. Zhu, G. Fu, and T. Jiang. On the Soundness Property for SQL Queries of Fine-grained Access Control in DBMSs. In *8th IEEE/ACIS International Conference on Computer and Information Science*, pages 469–474, 2009.

[18] M. Stonebraker and E. Wong. Access control in a relational data base management system by query modification. In *Proceedings of the 1974 Annual Conference - Volume 1*, pages 180–186. ACM, 1974.

[19] Q. Wang, T. Yu, N. Li, J. Lobo, E. Bertino, K. Irwin, and J.-W. Byun. On the correctness criteria of fine-grained access control in relational databases. In *Proceedings of the 33rd International Conference on Very large data bases*, pages 555–566. VLDB Endowment, 2007.

[20] Z. Zhang and A. Mendelzon. Authorization views and conditional query containment. In *Proceedings of International Conference on Database Theory*, volume 3363, pages 259–273. Springer, 2005.