On the Efficiency of Checking Perfect Privacy

Ashwin Machanavajjhala[*]
Department of Computer Science
Cornell University
mvnak@cs.cornell.edu

Johannes Gehrke
Department of Computer Science
Cornell University
johannes@cs.cornell.edu

ABSTRACT

Privacy-preserving query-answering systems answer queries while provably guaranteeing that sensitive information is kept secret. One very attractive notion of privacy is perfect privacy — a secret is expressed through a query Q_S, and a query Q_V is answered only if it discloses no information about the secret query Q_S. However, if Q_S and Q_V are arbitrary conjunctive queries, the problem of checking whether Q_V discloses any information about Q_S is known to be Π_2^p-complete.

In this paper, we show that for large interesting subclasses of conjunctive queries enforcing perfect privacy is tractable. Instead of giving different arguments for query classes of varying complexity, we make a connection between perfect privacy and the problem of checking query containment. We then use this connection to relate the complexity of enforcing perfect privacy to the complexity of query containment.

1. INTRODUCTION

The digitization of our daily lives has led to an explosion in the collection of personal data by governments, corporations, and individuals. Such information is stored in large databases. This has led to easy access to sensitive personal information, resulting in a dramatic increase in the disclosure of sensitive information [10]. Hence it is crucial to design database systems which can limit the disclosure of private information. To build such a system one has to answer two fundamental questions – (a) what is the sensitive information that is to be protected? and (b) what is the extent to which the database should limit the disclosure of the specified sensitive information? Usually, these two concepts – the specification of the sensitive information and the privacy guarantee – are encapsulated into a single document called a *privacy policy*.

For instance, consider a hospital which maintains patient records in two relations with schemas shown in Figure 1.

[*]Part of the work was done when the author was at IBM Almaden Research Center.

Relation P	
PID	PName

Relation D		
PID	Diagnosis	Medication

Figure 1: Relational Schema of Hospital Database

Relation P stores patient identifiers and names. Relation D stores patient identifiers, diagnosis and medication. The hospital wishes to disclose patient records to a pharmaceutical company in such a way that the pharmaceutical company cannot infer which patients have which diseases. One method to formally specify privacy policies is to express sensitive information as conjunctive queries and enforce *perfect privacy* [9, 18]. Perfect privacy guarantees that any other query Q_V answered by the database will not disclose *any* information about the answer to a sensitive query Q_S. This ensures that the adversary's belief about the answer to Q_S does not change on observing the answer to the query Q_V. In our example, the sensitive information can be expressed by the following query:

$$Q_S(name, diag) \quad :- \quad P(pid, name), D(pid, diag, med).$$

Perfect privacy has two attractive properties. First, usage of conjunctive queries enables the policy writer to express complex privacy policies. For instance, suppose the hospital is required to release as much data as possible without disclosing the names of patients diagnosed with both Hepatitis and Cancer. The sensitive information in this scenario can be easily expressed as the following conjunctive query:

$$\begin{aligned} Q_S(name) \quad :- \quad & P(pid, name), \\ & D(pid, \textsf{Hepatitis}, b_1), D(pid, \textsf{Cancer}, b_2). \end{aligned}$$

Second, perfect privacy has the following *collusion resistance* property. Consider again a secret query Q_S, and assume that there are n adversaries $\{A_1, \ldots, A_n\}$, where adversary A_i poses query Q_{V_i} to the database. Perfect privacy ensures that each query Q_{V_i} will be answered only if each query Q_{V_i} does not disclose any information about Q_S. Moreover, perfect privacy also ensures that the adversaries cannot learn any sensitive information by colluding with each other [18]. This collusion-resistance is crucial for efficient query answering when the database answers queries interactively instead of publishing views. When enforcing perfect privacy, the database does not need to keep track of all queries ever answered in order to guarantee that future query answers cannot be combined with old query answers to disclose information [13]. However, these nice properties of perfect privacy do not come for free. The problem of checking whether a conjunctive query Q_V discloses any information about another conjunctive query Q_S is Π_2^p-complete [18], even when the conjunctive queries have only equality comparisons.

In this paper, we show that there are interesting subclasses of queries for which enforcing perfect privacy is tractable (i.e., there is an algorithm which runs in time polynomial in the size of the two queries Q_S and Q_V). We first propose an alternate characterization of perfect privacy in terms of the well studied notion of query containment. We exploit this connection between perfect privacy and query containment to identify many subclasses of conjunctive queries (described in Figure 3) where checking perfect privacy is tractable.

The rest of the paper is organized as follows. In Section 2, we introduce notation and give intuition behind the ideas in the paper. Section 3 gives an alternative characterization of critical tuples, a core property of perfect privacy. We use this new characterization to connect perfect privacy with query containment in Section 4. In Section 5 we use this connection to prove tractability results for checking perfect privacy for different query classes. We discuss related work in Section 6 and then conclude in Section 7.

2. OVERVIEW

2.1 Preliminaries

We start with some basic notation following [18].

Relations and Queries: A *relation* R consists of a *relation schema* $R(A_1, \ldots, A_r)$ and an associated relation instance which is a finite two-dimensional table with r columns; we call r the *arity* of relation R. The columns are called *attributes* and the rows are called *tuples*. Attribute A_i has domain $D(A_i)$. We define D_R as the domain of tuples in the relation R, i.e., $D_R = D(A_1) \times \ldots \times D(A_r)$. We assume that all the domains are finite. Each tuple t in the relation is an element $(c_1, \ldots, c_r) \in D_R$ and is denoted by $t = R(c_1, \ldots, c_r)$; we call R the *relational symbol* of t. We denote by $t[A_i]$ or short $t[i]$ the value of attribute A_i in tuple t.

A *database schema* S is a finite set of relational schemas $\{R_1, \ldots, R_m\}$. $Tup(S)$ denotes the set of all tuples over all relations in the schema that can be formed from the constants in the respective domains. A *database instance* I is a subset of $Tup(S)$ and let $\mathcal{I}$ denote the set of all database instances. We henceforth use $Tup(\mathcal{I})$ to mean $Tup(S)$. We assume that the only constraints on the database are key constraints. A *key* of a relation R is a minimal set of attributes $\mathcal{A}$ such that for every two tuples $t_1, t_2 \in D_R$, $t_1[\mathcal{A}] = t_2[\mathcal{A}] \Rightarrow t_1 = t_2$. Let $\mathcal{I}^{\mathcal{K}}$ denote the set of all possible database instances satisfying the key constraints in $\mathcal{K}$. Let $\mathcal{K}_R$ denote the set of key constraints on relation R.

A *conjunctive query with inequalities* Q has the form

$$Q(a_1, a_2, \ldots, a_m) \quad :- \quad G_1 \wedge G_2 \wedge \ldots \wedge G_n \wedge \mathcal{C}_Q;$$

where $\mathcal{C}_Q$ is a set of inequality constraints.

We call G_ℓ a *subgoal*, $Q(a_1, a_2, \ldots, a_n)$ the *goal*. Each subgoal G_ℓ is of the form $R_\ell(x_1, x_2, \ldots, x_{k_\ell})$, where R_ℓ is a relation. Similar to a tuple, we call R_ℓ the *relational symbol of subgoal* G_ℓ. Each $x_i = G_\ell[i]$ is either a constant in $D(R_\ell.A_i)$ or a variable ranging over $D(R_\ell.A_i)$. Note that while each field in a tuple is a constant, a field in a subgoal can be a variable. Each a_i in the goal is either a variable appearing in one of the subgoals, or a constant in the domain of one of the attributes of a relation appearing in the query. We denote the set of variables appearing in the goal by A_Q and we also refer to it as the set of *distinguished variables*. The set of variables which appear in the subgoals but not in the goal is denoted by B_Q and called the set of *non-distinguished variables*. Let K_Q denote the set of all constants appearing in the query. We abuse the notation Q to also denote the set of subgoals in Q; i.e., $Q = \{G_1, G_2, \ldots, G_n\}$. The inequality constraints in the set $\mathcal{C}_Q$ are of the form $x_i \theta x_j$, where $\theta \in \{\leq, <, \geq, >, \neq\}$, x_i is a variable in $A_Q \cup B_Q$ and x_j is either a variable in $A_Q \cup B_Q$ or a constant in the domain that x_i ranges over.

Semantically, a query is a function from database instances to relation instances. We define the output of a query on a database instance using the idea of a *valuation*. A *valuation* ρ on a conjunctive query Q is a function that maps variables in $A_Q \cup B_Q$ to constants in the respective domains and the identity function for constants in K_Q. A *constraint preserving valuation* ρ on Q is a valuation such that $(x_i \, \theta \, x_j) \in \mathcal{C}_Q \Rightarrow \rho(x_i) \, \theta \, \rho(x_j)$. Unless otherwise specified, all valuations in the remainder of the paper are constraint preserving. A subgoal is said to be *compatible* with a tuple t if the subgoal and the tuple have the same relational symbol and there is a valuation $|rho$ mapping the subgoal to t. A set of subgoals $\{G_1, \ldots G_n\}$ is *compatible* if there is a tuple t that $\forall i \exists \rho : \rho$ is a valuation mapping G_i to t.

Given a conjunctive query with inequalities Q and a database instance I, a tuple $t^o = Q(c_1, c_2, \ldots, c_m)$ is said to be in the output of $Q(I)$ if there is a valuation ρ such that $\rho(a_i) = c_i$, i.e., each distinguished variable a_i is mapped to a constant c_i; and $\rho(R(x_1, \ldots, x_r)) \in I$, i.e., subgoal $R(x_1, \ldots, x_r)$ is mapped to one of the tuples in relation instance R of the database instance I.

Query Containment: Query Q_1 is said to be *contained* in query Q_2 on the set of databases in $\mathcal{I}$ and is denoted by $Q_1 \subseteq Q_2$, if

$$\forall I \in \mathcal{I}, \, Q_1(I) \subseteq Q_2(I).$$

Throughout the paper, we use the term *conjunctive query* to denote a *conjunctive query without inequalities*, unless otherwise specified.

Perfect Privacy [18]: We give a quick recap of the basic notions of perfect privacy from Miklau et al. [18]. Let $\mathbf{P}$ be a probability distribution over all the possible tuples $\mathbf{P} : Tup(\mathcal{I}) \rightarrow [0, 1]$, where $\mathbf{P}(t) = p$ denotes the probability that tuple t will occur in a database instance. Miklau et al. call the pair $(Tup(\mathcal{I}), \mathbf{P})$ a *dictionary*. Let $\mathbf{P}[Q_S = s]$ denote the probability that the query Q_S outputs s. Then query Q_S is perfectly private with respect to Q_V if for every probability distribution and for all answers s, v to the queries Q_S, Q_V respectively,

$$\mathbf{P}[Q_S = s] = \mathbf{P}[Q_S = s \,|\, Q_V = v].$$

In other words, the query Q_S is perfectly private with respect to Q_V if the adversary's belief about the answer to Q_S does not change even after seeing the answer to Q_V.

When the only constraints in the database are key constraints, the above condition can be rephrased in terms of critical tuples. A tuple t is *critical* to a query Q, denoted by $t \in crit(Q)$, if $\exists I \in \mathcal{I}, \; Q(I \cup \{t\}) \neq Q(I)$. Let us now state one of the main results from Miklau et al. [18].

THEOREM 2.1 (PERFECT PRIVACY [18]). *Let $(Tup(\mathcal{I}), \mathbf{P})$ be a dictionary. Two queries Q_S and Q_V are perfectly pri-*

Relation P	
PID	PName
123	John
135	Daisy
146	Chris

Relation D		
PID	Diagnosis	Medication
123	Hepatitis	RX1
135	Cancer	RX2
135	Hepatitis	RX1

Figure 2: A database instance $I \in \mathcal{I}$

vate with respect to each other if

$$crit(Q_S) \cap crit(Q_V) = \emptyset.$$

Let $\mathcal{K} = \{\mathcal{A}_1, \ldots, \mathcal{A}_m\}$ be a set of key constraints over the relations involved in Q_V and Q_S. Then Query Q_V and Q_S are perfectly private with respect to each other if

$$\forall t_S \in crit(Q_S), t_V \in crit(Q_V), \forall \mathcal{A}_i \in \mathcal{K},\ t_S[\mathcal{A}_i] \neq t_V[\mathcal{A}_i].$$

Miklau et al. also proved the following complexity result.

THEOREM 2.2 (INTRACTABILITY OF PERFECT PRIVACY[18]) *The problem of checking whether a query Q_V is perfectly private with respect to query Q_S is Π_2^p-complete.*

2.2 Intuition

In this section we provide some intuition behind the rest of the paper. We are interested in identifying subclasses of conjunctive queries which allow specification of interesting privacy policies while permitting tractable enforcement of perfect privacy. We first take a very simple class of queries – queries without self-joins – and illustrate that enforcing perfect privacy is indeed tractable (Section 2.2.1). In order to determine the complexity of other query classes, we could have laboriously enumerated every interesting query class and investigated the hardness of checking perfect privacy for queries in that class. Instead, we show a connection between the problem of checking perfect privacy and the problem of query containment (Section 2.2.2). We use this connection and the vast literature in query containment to identify interesting query classes (Section 2.2.3).

2.2.1 A Simple Tractable Case

EXAMPLE 2.1. *Consider the hospital database from Figure 1. Let the sensitive information be specified by the following query Q_S:*

$$Q_S(name) \quad :- \quad P(pid, name), D(pid, \mathsf{Cancer}, b_1)$$

Query Q_S has no self joins, i.e., no two subgoals have the same relational symbol. Which tuples in the domain of relation D are critical to this query? Clearly, any tuple t having the diagnosis attribute valued Cancer*, for example $t = D($*123, Cancer, RX2*$)$, is critical to Q_S. In particular, for $t = D($*123, Cancer, RX2*$)$, the instance I in Figure 2 is such that $Q_S(I \cup \{t\}) \neq Q_S(I)$. Also, it is easy to see that any tuple not having* Cancer *as the value for the diagnosis attribute, such as $t' = D($*123, Hepatitis, RX1*$)$, is not critical to Q_S. For any instance $I' \in \mathcal{I}$, adding the tuple t' will not lead to a new output tuple being created since t' is not compatible with any subgoal. Hence, a tuple in the domain of relation D is critical to Q_S if and only if the value for the diagnosis attribute is* Cancer. ■

As illustrated in this example, if Q_S has no self-joins, a tuple t is critical to Q_S if and only if t is compatible with some subgoal in Q_S. Using this simple check for critical tuples, the condition for perfect privacy can be simplified as follows. Queries Q_S and Q_V, both without self-joins, share a critical tuple if and only if there are subgoals $G_S \in Q_S$ and $G_V \in Q_V$ both of which are compatible with some common tuple $t \in D_R$. Recall that subgoals are compatible if they have the same relation symbol and the subgoals do not have differing constants for any attribute. Checking this condition is linear in the size of the queries and hence the problem is tractable.

2.2.2 Connecting Privacy and Query Containment

Let us now give an intuition behind our alternate characterization of perfect privacy. Suppose a tuple t is critical to Q. By definition, there is some database instance I, such that $Q(I \cup \{t\}) \neq Q(I)$. Since Q is a conjunctive query, it is monotone, i.e., $Q(I) \subseteq Q(I \cup \{t\})$. Therefore, there is an output tuple t^o, such that $t^o \in Q(I \cup \{t\})$ and $t^o \notin Q(I)$. Consider a valuation ρ on Q which creates this output tuple $t^o \in Q(I \cup \{t\})$. Recall that ρ should map Q's goal to t^o and the subgoals in Q to tuples in $(I \cup \{t\})$. Since, $t^o \notin Q(I)$, ρ cannot map all the subgoals in Q to tuples in I. Hence, ρ should map some set of subgoals from Q, say $\mathcal{G}$, to the critical tuple t.

Consider a query Q' which is constructed from Q by replacing all of the variables x_i appearing in Q by $\rho(x_i)$. Since a valuation is an identity function on constants, ρ is still a valuation from Q' to $(I \cup \{t\})$ which outputs the tuple t^o. So we still have $t^o \in Q'(I \cup \{t\})$ and $t^o \notin Q(I)$.

Remove from Q' the subgoals in $\mathcal{G}$. Call this new query $Q|_{(\mathcal{G},t)}$ (see Section 3 for a formal definition). The valuation ρ now maps all the subgoals in $Q|_{(\mathcal{G},t)}$ to I and the goal of $Q|_{(\mathcal{G},\sqcup)}$ to t^o. Thus we have, $t^o \in Q|_{(\mathcal{G},t)}(I)$ and $t^o \notin Q(I)$; i.e., $Q|_{(\mathcal{G},t)} \not\subseteq Q$. This gives the following condition:

$$t \in crit(Q) \Rightarrow \exists \mathcal{G} \text{ compatible with } t,\ Q|_{(\mathcal{G},t)} \not\subseteq Q \qquad (1)$$

Interestingly, whenever $\mathcal{G}$ is a set of subgoals compatible with t, the query $Q|_{(\mathcal{G},t)}$ is such that $Q|_{(\mathcal{G},t)}(I) \subseteq Q(I \cup \{t\})$ (see Section 3). This is because any valuation ρ from $Q|_{(\mathcal{G},t)}$ to I can be extended to a valuation ρ' from Q to $(I \cup \{t\})$, where the subgoals in $\mathcal{G}$ are mapped to t and the other subgoals are mapped to tuples in I as in ρ.

When t is not critical to Q, for every instance I, $Q(I \cup \{t\}) = Q(I)$. Hence, for every instance I, $Q|_{(\mathcal{G},t)}(I) \subseteq Q(I)$. This gives us the condition

$$t \notin crit(Q) \Rightarrow \forall \mathcal{G} \text{ compatible with } t,\ Q|_{(\mathcal{G},t)} \subseteq Q \qquad (2)$$

Using Statements 1 and 2, we can now characterize the perfect privacy condition using query containment:

$$\begin{aligned} & Q_V \text{ is perfectly private wrt } Q_S \\ & \quad \Leftrightarrow crit(Q_V) \cap crit(Q_S) = \emptyset \\ & \quad \Leftrightarrow \forall t \in Tup(\mathcal{I}),\ t \notin crit(Q_V) \vee t \notin crit(Q_S) \\ & \quad \Leftrightarrow \forall t \in Tup(\mathcal{I}), \forall \mathcal{G}_S, \mathcal{G}_V \text{ compatible with } t, \\ & \qquad\quad Q_S|_{(\mathcal{G}_S,t)} \subseteq Q_S \vee Q_V|_{(\mathcal{G}_V,t)} \subseteq Q_V \end{aligned}$$

Our final goal is to identify cases when the above check is tractable. This will, however, not be easy as long as the condition has a universal quantifier over all the tuples in the table. We show that when attributes have sufficiently large domains, the universal quantifier over the domain of

Query Class	**Efficiency**	**Query Class Description**
Queries with no self-joins	$O(nr)$	Queries where each subgoal is associated with a unique relation symbol Example: $Q(a_1, a_2) :- R(a_1, b_1, b_2), S(a_2, b_2, b_1), T(a_1, b_3, b_1)$ Complexity of $Q \subseteq Q' : O(nr)$, where Q has no self joins
Maximal Row Tableaux	$O(n^2r)$	Query Q represented as tableau. Each variable appears only in one column. No two rows r_1, r_2 sharing the same relation symbol are s.t. $\forall i, r_1[i] = \mathsf{c} \Rightarrow r_2[i] = \mathsf{c}$, for all $\mathsf{c} \in K_Q$ Example: $Q(a_1, a_2) :- R(a_1, b_1, 0), R(a_2, b_2, 1), T(2, b_2, b_3)$ Complexity of $Q \subseteq Q'$: NP-complete
Queries with $k \leq 1$	$O(n^3r)$	Queries with at most one self-join per relation Example: $Q(a_1, a_2) :- R(a_1, b_1, b_2), R(a_2, b_1, b_3), S(a_1, b_2, b_3), S(a_2, b_1, b_3)$ Complexity of $Q \subseteq Q' : O(n^2r)$; Q has at most 1 self join [20]
Simple Tableaux	$O(m2^{2k} \cdot n^3r^2)$	Query Q represented as tableau. Each variable appears only in one column. Each column containing a repeated non-distinguished variable b_i contains no other repeated symbol. Example: $Q(a_1, a_2) :- R(a_1, b_2, b_3), R(a_2, b_2, b_4), R(a_1, b'_2, b_3), R(a_2, b''_2, b_5)$ Complexity of $Q \subseteq Q' : O(n^3r^2)$, where, Q_1, Q_2 are simple tableaux queries [2]
Acyclic Queries	$O(m2^{2k} \cdot rn^2 \log n)$	The hypergraph constructed using the variables in the query Q as nodes and subgoals in the query as (hyper-)edges is acyclic. Example: $Q(a_1, a_2) :- R(a_1, b_2, b_3), R(a_2, b_3, b_4), R(a_1, b_5, b_6)$ Complexity of $Q \subseteq Q' : O(rn^2 \log n)$, where, Q is acyclic [7].
Queries with Bounded Query-Width	$O(m2^{2k} \cdot rc^2n^{c+1} \log n)$	The hypergraph constructed using the variables in the query Q as nodes and subgoals in the query as (hyper-)edges has a query-width bounded by c. Example ($c = 2$): $Q(a_1, a_2) :- R(a_1, b_3, b_1), R(a_2, b_4, b_1), R(a_1, b_5, b_2), R(a_2, b_6, b_2)$ Complexity of $Q \subseteq Q' : O(crn^{c+1} \log n)$, where Q has bounded tree-width [7]

Figure 3: Tractable Perfect Privacy

tuples can be eliminated. Given a set of subgoals $\mathcal{G}$, let G be a subgoal such that every tuple compatible with G is also compatible with $\mathcal{G}$. Let $Q|_{(\mathcal{G},G)}$ be constructed analogous to $Q|_{(\mathcal{G},t)}$. We show, in Section 4, that if $Q|_{(\mathcal{G},G)} \not\subseteq Q$, then some tuple compatible with G is critical to Q. Using this property, we provide an alternate characterization for perfect privacy as follows: A query Q_V does not disclose any information about Q_S if and only if for every set of compatible subgoals $\mathcal{G}_S$ in Q_S and $\mathcal{G}_V$ in Q_V:

$$Q_S|_{(\mathcal{G}_S,G^\star)} \subseteq Q_S \vee Q_V|_{(\mathcal{G}_V,G^\star)} \subseteq Q_V,$$

where $G^\star$ is the most general unifier of the subgoals in $\mathcal{G}_S \cup \mathcal{G}_V$ (see Section 4).

2.2.3 *Using the Connection*

We can draw three key insights from this alternate definition. First, it is now intuitively clear why the problem of enforcing perfect privacy is in Π_2^p. For every set of compatible subgoals (universal quantification over a finite number of choices) we need to perform a query containment check (which is in NP for conjunctive queries [5] and is in Π_2^p when queries have inequalities [23]). Next, we observe that if the maximum number of subgoals in a query which share the same relational symbol is bounded by a constant (alternately, the number of self-joins per relation is bounded), then the problem of enforcing perfect privacy is only as hard as performing a constant number of containment checks. Finally, the query containments we are required to check are not between two arbitrarily complex queries, and we can use their structure to propose efficient algorithms for enforcing perfect privacy even for query subclasses where tractability of query containment is not known.

Figure 3 enumerates our complexity results. The first column enumerates the subclasses of conjunctive queries where perfect privacy can be enforced in time polynomial in the size of the queries. The second column gives the bound on

n	Total number of subgoals: $n = \lvert Q_S \rvert + \lvert Q_V \rvert$
m	Number of relations appearing in Q_S and Q_V
r	Max arity of relation appearing in Q_S and Q_V
k	Max number of self-joins per relation in either query

Figure 4: Size of instance (Q_S, Q_V)

the time required to enforce perfect privacy when Q_S and Q_V are from the specified query class; Figure 4 explains our notation. Notice that the case of queries without self-joins is actually a special case of our framework. For maximal row tableaux, a subclass of tableaux queries, query containment is intractable. However, since the query containments we check are of a special form, we show that perfect privacy in this scenario can be enforced with a very simple algorithm. Tractability of the rest of the query classes leverage tractability results for query containment [2, 20, 7]. When Q_S and Q_V are from different classes, the tractability result for the less efficient query class holds.

3. CRITICAL TUPLES

In this section we propose an alternate characterization of critical tuples for conjunctive queries in terms of query containment. In order to state the alternate characterization, we first need to describe how to construct $Q|_{(\mathcal{G},t)}$ from a conjunctive query Q, a set of compatible subgoals $\mathcal{G}$ in Q, and a tuple t which is compatible with all the subgoals in $\mathcal{G}$. The construct $Q|_{(\mathcal{G},t)}$ has interesting properties (Lemmas 3.1 and 3.2) which we will exploit in the rest of the paper. Finally, we recharacterize the critical tuple condition for conjunctive queries in terms of query containment (Theorem 3.1).

Denote by $Q - \mathcal{G}$ the query constucted by removing all the subgoals in $\mathcal{G}$ from the query Q. Denote by $Q|_{(\mathcal{G},t)}$ the query $\rho_t(Q - \mathcal{G})$, where ρ_t is a *partial valuation* which maps the variables appearing in subgoals in $\mathcal{G}$ to corresponding constants in tuple t, and is the identity function for all other variables in the query $Q - \mathcal{G}$. Formally ρ_t is defined as follows: Let V_Q denote the variables appearing in Q, and $V_\mathcal{G}$ denote the set of variable appearing in $\mathcal{G}$. Let K denote the set of all constants. Then, ρ_t is a function $\rho_t : K \cup V_Q \rightarrow K \cup (V_Q - V_\mathcal{G})$ such that,

$$\rho_t(x_i) = \begin{cases} t[j], & \text{if } G[j] = x_i, \text{ for some } G \in \mathcal{G}_S \cup \mathcal{G}_V \\ x_i, & \text{otherwise} \end{cases}$$

$Q|_{(\mathcal{G},t)}$ has the following two properties.

LEMMA 3.1. *Let Q be a conjunctive query and t be a tuple. For every set of subgoals $\mathcal{G}$ in Q which are compatible with t,*

$$\forall I,\ Q|_{(\mathcal{G},t)}(I) \subseteq Q(I \cup \{t\}).$$

LEMMA 3.2. *Let Q be a conjunctive query, t a tuple and I a database instance. Let ρ be a valuation mapping the subgoals in Q to tuples in $(I \cup \{t\})$. Let $\mathcal{G}$ be the set of subgoals in Q mapped to t under ρ. If ρ maps the goal of Q to t^o, then $t^o \in Q|_{(\mathcal{G},t)}(I)$.*

We can now state our alternate characterization of critical tuples in terms of query containment.

THEOREM 3.1. *A tuple t is critical to a conjunctive query Q if and only if there is some set of subgoals $\mathcal{G}$ in Q which are compatible with t, such that $Q|_{(\mathcal{G},t)} \not\subseteq Q$.*

4. PERFECT PRIVACY

In this section, we give an alternative characterization of perfect privacy in terms of query containment. In particular, we use our new characterization of critical tuples to give a novel characterization of perfect privacy in terms of query containment. Our new characterization, however, involves a universal quantifier over all the tuples in the domain, making this characterization very expensive to check. The major technical contribution of this section is to eliminate this universal quantification over all tuples (Theorem 4.2).

We first discuss the case where in every database instance in $\mathcal{I}$ the tuples are independent of each other (Section 4.1). We also discuss the scenario with key constraints (Section 4.2).

4.1 Perfect Privacy with Independent Tuples

In the case when the tuples are independent of each other, a query Q_V does not disclose any information about the query Q_S if and only if no tuple is critical to both Q_S and Q_V [18]. Hence, given our novel characterization for critical tuples (Theorem 3.1), perfect privacy can be rephrased as

THEOREM 4.1. *A conjunctive query Q_V does not disclose any information about another conjunctive query Q_S if and only if for every tuple $t \in Tup(\mathcal{I})$, and for every set of subgoals $\mathcal{G}_S$ in Q_S and $\mathcal{G}_V$ in Q_V where all the subgoals in $\mathcal{G}_S \cup \mathcal{G}_V$ are compatible with t,*

$$Q_S|_{(\mathcal{G}_S,t)} \subseteq Q_S \vee Q_V|_{(\mathcal{G}_V,t)} \subseteq Q_V.$$

Consider an algorithm which naively implements the above condition for perfect privacy (see Algorithm 1). There are three steps which would make such an algorithm intractable.

- H1: **for every** tuple $t \in Tup(\mathcal{I})$ (Line 2).
- H2: **for every** set of subgoals $\mathcal{G}_S$ in Q_S and $\mathcal{G}_V$ in Q_V which are compatible with t (Line 4).

Algorithm 1 Perfect Privacy (naive).

```
Input: Conjunctive Queries $Q_S$ and $Q_V$.
Output: Whether $Q_S$ and $Q_V$ share a critical tuple.
 1: //for every tuple in the domain
 2: for all tuples $t \in Tup(\mathcal{I})$ do
 3:    //for every set of subgoals in the query
 4:    for all $\mathcal{G}_S \subseteq Q_S$, $\mathcal{G}_V \subseteq Q_V$ having the same relational
       symbol as $t$ do
 5:       if $\mathcal{G}_S \cup \mathcal{G}_V$ is compatible with the tuple $t$ then
 6:          //consider the partial valuation $\rho_t$ mapping the
             //subgoals in $\mathcal{G}_S \cup \mathcal{G}_V$ to the tuple $t$
 7:          $\rho_t : A_Q \cup B_Q \rightarrow A_Q \cup B_Q \cup K_Q$ such that
             $\rho_t(x_i) = \begin{cases} t[j], & \text{if } G[j] = x_i, \text{ for some } G \in \mathcal{G}_S \cup \mathcal{G}_V \\ x_i, & \text{otherwise} \end{cases}$
 8:          $Q_S|_{(\mathcal{G}_S,t)} = \rho_t(Q_S - \mathcal{G}_S)$, $Q_V|_{(\mathcal{G}_V,t)} = \rho_t(Q_V - \mathcal{G}_V)$.
 9:          if $(Q_S|_{(\mathcal{G}_S,t)} \not\subseteq Q_S$ AND $Q_V|_{(\mathcal{G}_V,t)} \not\subseteq Q_V)$ then
10:             return $Q_S$ and $Q_V$ SHARE a critical tuple.
11:          end if
12:       end if
13:    end for
14: end for
15: return $Q_S$ and $Q_V$ do NOT SHARE a critical tuple.
```

- H3: Checking whether $Q_S|_{(\mathcal{G}_S,t)} \subseteq Q_S$ OR $Q_V|_{(\mathcal{G}_V,t)} \subseteq Q_V$ (Line 9).

To identify query classes for which Algorithm 1 is tractable, each of the above three steps should be tractable.

Step H3: First, given sets of subgoals $\mathcal{G}_S$ and $\mathcal{G}_V$ which are compatible with a tuple t, constructing the queries $Q_S|_{(\mathcal{G}_S,t)}$ and $Q_V|_{(\mathcal{G}_V,t)}$ is at most linear in the size of the two queries. Performing the two containment checks may not be polynomial in the size of the queries [5]. However, there are subclasses of conjunctive queries for which Step H3 is tractable [7, 20]. These query classes include queries with no self joins, queries with at most one self join, simple tableaux queries, acyclic queries and queries with bounded query-width. We discuss these cases in Section 5.

Step H2: Since we are only considering conjunctive queries (without inequalities), checking whether a set of subgoals $\mathcal{G}$ is compatible with a tuple t is linear in the number of subgoals. It is sufficient to check for every $G \in \mathcal{G}$, if $G[i]$ has a constant, then $t[i]$ has the same constant. Hence, Steps H2 and H3 can be performed in time $O(m2^{2k}(nr + T(\mathsf{H3})))$, where $T(\mathsf{H3})$ is the time taken to perform Step H3, k is the maximum number of subgoals in either query which share the same relational symbol, n is the total number of subgoals which appear in both the queries, r is the maximum arity of a relation which appears in both the queries, and m relation symbols appear in both queries. If k is a constant, both the Steps H2 and H3 can be done in time polynomial in the size of the inputs if the two queries Q_S and Q_V belong to one of the query classes listed above.

Step H1: Step H1 involves a universal quantification over the set of all tuples in the domain. Clearly, assuming that the domain of tuples is very small (bounded by some polynomial in the size of the input) ensures that the algorithm is tractable. However, such an assumption would greatly limit the utility of our result. In most cases, the size of the domain of tuples is very large compared to size of the queries. Hence, in the rest of the section we propose an alternate characterization for perfect privacy where we eliminate the universal quantification over the domain of tuples. Interstingly, we are able to remove the quantification over the domain of tuples when we assume that the domain of tuples is sufficiently large.

Definition 4.1 (Sufficiently Large Domains). *Let $Q(a_1, a_2, \ldots, a_m) :- G_1 \wedge G_2 \wedge \ldots \wedge G_n \wedge \mathcal{C}_Q$ be a conjunctive query. We say that the domains of the attributes appearing in Q are* sufficiently large with respect to Q, *if in every subgoal G_ℓ every variable x_i can be assigned to a distinct constant c_i in the domain of the associated attribute A_i, and c_i does not appear in Q.*

The plan for the rest of the section is as follows. First, we generalize the construction $Q|_{(\mathcal{G},t)}$ to $Q|_{(\mathcal{G},G)}$, where G is a subgoal such that every tuple compatible with G is also compatible with all the subgoals in $\mathcal{G}$. We then show that this construction can also be used to reason about critical tuples. More specifically, we show that $Q|_{(\mathcal{G},G)} \subseteq Q$ if and only if every tuple compatible with G is not critical to Q. We then use this construction and propose an alternate characterization for perfect privacy (Theorem 4.2). This new characterization does not involve a universal quantifier over all the tuples in the domain. Finally, we prove Theorem 4.2 and bound the running time of a simple algorithm which implements Theorem 4.2.

Let us start with some definitions. Let $\mathcal{G}$ be a set of compatible subgoals in conjunctive query Q. We say that a subgoal G is *compatible* with $\mathcal{G}$ if every tuple compatible with G is also compatible with $\mathcal{G}$. We call a subgoal $G^\star$ the *most general unifier* of $\mathcal{G}$ (denoted by $mgu(\mathcal{G})$) if a tuple is compatible with $G^\star$ if and only if the tuple is compatible with $\mathcal{G}$. Since we are only dealing with conjunctive queries without inequalities, it is easy to see that the most general unifier of a set of subgoals is unique and well defined.

Given a query Q, a set of compatible subgoals $\mathcal{G}$ and a subgoal G which is compatible with $\mathcal{G}$, denote by $Q|_{(\mathcal{G},G)}$ the query $\rho_G(Q - \mathcal{G})$. Here, ρ_G is a partial valuation defined as follows:

$$\rho_G : A_Q \cup B_Q \rightarrow A_Q \cup B_Q \cup K_Q \cup V \text{ such that,}$$

$$\rho_G(x_i) = \begin{cases} G[j], & \text{if } G_\ell[j] = x_i, \text{ for some } G_\ell \in \mathcal{G}_S \cup \mathcal{G}_V \\ x_i, & \text{otherwise} \end{cases}$$

The following two lemmas show interesting properties of $Q|_{(\mathcal{G},G)}$. Lemma 4.1 shows that the query $Q|_{(\mathcal{G},t)}$ is contained in the query $Q|_{(\mathcal{G},G)}$. Lemma 4.2 connects $Q|_{(\mathcal{G},G)}$ and critical tuples.

Lemma 4.1. *Let Q be a conjunctive query, $\mathcal{G}$ be a set of compatible subgoals, G be a subgoal (not necessarily in Q) which is compatible with $\mathcal{G}$, and t be a tuple compatible with G. Then*

$$Q|_{(\mathcal{G},t)} \subseteq Q|_{(\mathcal{G},G)}$$

Lemma 4.2. *Let Q be a conjunctive query, $\mathcal{G}$ be a set of compatible subgoals in Q, and G be a subgoal (not necessarily in Q) which is compatible with $\mathcal{G}$. Every tuple compatible with G is not critical to Q if and only if*

$$Q|_{(\mathcal{G},G)} \subseteq Q$$

We are now ready to state our alternate characterization for perfect privacy.

Theorem 4.2. *Let Q_S and Q_V are conjunctive queries and the domains of attributes are sufficiently large with respect to each of the queries. Q_V does not disclose any information about Q_S if and only if for every set of compatible subgoals $\mathcal{G}_S$ in Q_S and $\mathcal{G}_V$ in Q_V,*

$$Q_S|_{(\mathcal{G}_S,G^\star)} \subseteq Q_S \textit{ OR } Q_V|_{(\mathcal{G}_V,G^\star)} \subseteq Q_V$$

where $G^\star = mgu(\mathcal{G}_S \cup \mathcal{G}_V)$.

In the remainder of this section, we prove the above theorem. Proving the "if" direction of the above condition is quite easy. Suppose tuple t is critical to both Q_S and Q_V. Then from Theorem 3.1, there are sets of subgoals $\mathcal{G}_S$ in Q_S and $\mathcal{G}_V$ in Q_V such that

$$Q_S|_{(\mathcal{G}_S,t)} \not\subseteq Q_S \text{ AND } Q_V|_{(\mathcal{G}_V,t)} \not\subseteq Q_V$$

Since t is compatible with $\mathcal{G}_S$ and $\mathcal{G}_V$, t is also compatible with $G^\star = mgu(\mathcal{G}_S \cup \mathcal{G}_V)$. Hence, using Lemma 4.1, we get

$$Q_S|_{(\mathcal{G}_S,G^\star)} \not\subseteq Q_S \text{ AND } Q_V|_{(\mathcal{G}_V,G^\star)} \not\subseteq Q_V$$

Proving the other direction, however, is trickier. We need to show that if $Q_S|_{(\mathcal{G}_S,G^\star)} \not\subseteq Q_S$ and $Q_V|_{(\mathcal{G}_V,G^\star)} \not\subseteq Q_V$, then Q_S and Q_V share a critical tuple. From Lemma 4.2, all we can say is that there are tuples t_S and t_V such that t_S and t_V are compatible with $G^\star$ (and hence compatible with $\mathcal{G}_S$ and $\mathcal{G}_V$), such that t_S is critical to Q_S and t_V is critical to Q_V. In fact, unless we assume the domains are sufficiently large, this is all the above condition guarantees.

The sufficiently large domain assumption ensures that given a query Q there is a valuation which maps every symbol in Q to a distinct constant in the respective domain. Using this property, we can prove a stronger version of Lemma 4.2 which in turn will help us to prove the theorem.

Definition 4.2 (Fine $Q|_{(\mathcal{G},G^\star)}$-Critical Tuple). *Let Q be a query, $\mathcal{G}$ a set of subgoals in Q and $G^\star$ be a subgoal compatible with $\mathcal{G}$. A tuple t_f which is critical to a query Q is called a* fine $Q|_{(\mathcal{G},G^\star)}$-critical tuple[1] *if there is a database instance I_f and a valuation ρ_f such that*

- *ρ_f is a valuation mapping $Q|_{(\mathcal{G},G^\star)}$ to tuples in I_f such that every variable appearing in $Q|_{(\mathcal{G},G^\star)}$ is mapped to a distinct constant not appearing in the query.*
- *$\rho_f(G^\star) = t_f$.*
- *ρ_f maps the goal of $Q|_{(\mathcal{G},G^\star)}$ to a tuple t_f^o such that $t_f^o \notin Q(I)$.*

Lemma 4.3. *Let Q be a conjunctive query, $\mathcal{G}$ be a set of compatible subgoals in Q, and $G^\star$ be a subgoal (not necessarily in Q) which is compatible with $\mathcal{G}$. Under the sufficiently large domain assumption, $Q|_{(\mathcal{G},G^\star)} \not\subseteq Q$ if and only if one of the tuples compatible with $G^\star$ is a fine $Q|_{(\mathcal{G},G^\star)}$-critical tuple.*

Lemma 4.3 guarantees that if $Q_S|_{(\mathcal{G}_S,G^\star)} \not\subseteq Q_S$ and $Q_V|_{(\mathcal{G}_V,G^\star)} \not\subseteq Q_V$, then there exist t_S and t_V compatible with $G^\star$ such that t_S is a fine $Q_S|_{(\mathcal{G}_S,G^\star)}$-critical tuple and t_V is a fine $Q_V|_{(\mathcal{G}_V,G^\star)}$-critical tuple. We use this property to show that t_S is critical to Q_V.

Consider a mapping $\gamma : K \rightarrow K$ (from the domain of all constants to the domain of all constants) such that $\gamma(t_V[i]) = t_S[i]$ and $\gamma(t_S[i]) = t_V[i]$, and γ is an identity function for constants not appearing in t_V or t_S. It is easy to see that γ has the following properties.

[1]A similar definition is used in the proof of hardness of checking perfect privacy in [18]

Algorithm 2 Perfect Privacy (Sufficiently Large Domains)

```
Input: Conjunctive Queries $Q_S$ and $Q_V$
Output: Whether $Q_S$ and $Q_V$ share a critical tuple.
 1: //for every set of subgoals in the two queries
 2: for all $\mathcal{G}_S \subseteq \mathcal{G}_{Q_S}$, $\mathcal{G}_V \subseteq \mathcal{G}_{Q_V}$ do
 3:    //if all the subgoals in $\mathcal{G}_S \cup \mathcal{G}_V$ are compatible.
 4:    if $\mathcal{G}_S \cup \mathcal{G}_V$ are compatible subgoals of relation $R$ of arity
       $r$ then
 5:       //consider the partial valuation $\rho'$ mapping the
          //subgoals in $\mathcal{G}_S \cup \mathcal{G}_V$ to $G^\star = mgu(\mathcal{G}_S \cup \mathcal{G}_V)$
 6:       Let $G^\star$ be a subgoal such that
             $1 \le i \le r, \exists G \in \mathcal{G}_S \cup \mathcal{G}_V : G[i] = \mathsf{c}_i \Rightarrow G^\star[i] = \mathsf{c}_i$
             $1 \le i, j \le r, \exists G, G' \in \mathcal{G}_S \cup \mathcal{G}_V,$
                $G[i] = G'[j] \Rightarrow G^\star[i] = G^\star[j]$
          Let $\{g_1, g_2, \ldots\}$ be the set of distinct variables appearing
          in $G^\star$.
 7:       $\rho_{G^\star} : A_Q \cup B_Q \to A_Q \cup B_Q \cup K_Q \cup V$ such that
          $\rho_{G^\star}(x_i) = \begin{cases} G^\star_j, & \text{if } G_\ell[j] = x_i, \text{ for some } G_\ell \in \mathcal{G}_S \cup \mathcal{G}_V \\ x_i, & \text{otherwise} \end{cases}$
 8:       $Q_S|_{(\mathcal{G}_S, G^\star)} = \rho_{G^\star}(Q_S - \mathcal{G}_S)$,
          $Q_V|_{(\mathcal{G}_V, G^\star)} = \rho_{G^\star}(Q_V - \mathcal{G}_V)$.
 9:       //check if $Q_S|_{(\mathcal{G}_S, G^\star)} \subseteq Q_S$ and $Q_V|_{(\mathcal{G}_V, G^\star)} \subseteq Q_V$
10:       if $\left(Q_S|_{(\mathcal{G}_S, G^\star)} \not\subseteq Q_S \text{ AND } Q_V|_{(\mathcal{G}_S, G^\star)} \not\subseteq Q_V\right)$ then
11:          return $Q_S$ and $Q_V$ SHARE a critical tuple.
12:       end if
13:    end if
14: end for
15: return $Q_S$ and $Q_V$ do NOT SHARE a critical tuple.
```

Lemma 4.4. *Let Q_S and Q_V be conjunctive queries, $\mathcal{G}_S$ be a set of subgoals in Q_S, $\mathcal{G}_V$ be a set of subgoals in Q_V and $G^\star$ be a subgoal compatible with $\mathcal{G}_S \cup \mathcal{G}_V$. Let t_S and t_V be tuples compatible with $G^\star$ such that t_S is a fine $Q_S|_{(\mathcal{G}_S, G^\star)}$-critical tuple, and t_V is a fine $Q_S|_{(\mathcal{G}_V, G^\star)}$-critical tuple. Let $\gamma : K \to K$ be a mapping such that $\gamma(t_V[i]) = t_S[i]$ and $\gamma(t_S[i]) = t_V[i]$. The mapping γ has the following properties under the sufficiently large domain assumption*

- *γ is a bijection.*
- *for every constant c appearing in Q_V, $\gamma(\mathsf{c}) = \mathsf{c}$.*

With Lemma 4.4, the rest of the proof of Theorem 4.2 is straightforward. Let $I'_V = \gamma(I_V)$. Then from Lemma 4.4, $I'_V \cup \{t_S\} = \gamma(I_V \cup \{t_V\})$. Lemma 4.4 also shows that γ changes the value of a constant only if the constant does not appear in Q_V. Hence, ρ is a valuation mapping Q_V to some I if and only if $\gamma \circ \rho$ is a valuation mapping Q_V to $\gamma(I)$. Therefore,

$$\begin{aligned} &Q_V(I_V \cup \{t_V\}) \not\subseteq Q_V(I_V) \\ &\quad\Rightarrow Q_V(\gamma(I_V \cup \{t_V\})) \not\subseteq Q_V(\gamma(I_V)) \\ &\quad\Rightarrow Q_V(I'_V \cup \{t_S\}) \not\subseteq Q_V(I'_V) \end{aligned}$$

Hence, $t_S \in crit(Q_S) \cap crit(Q_V)$, completing the proof. □

Algorithm 2 sketches a simple algorithm which implements Theorem 4.2. Since we have eliminated the universal quantifier over tuples, we are left with only two steps which contribute to the complexity of Algorithm 2.

- H2': **for every** set of subgoals $\mathcal{G}_S$ in Q_S and $\mathcal{G}_V$ in Q_V which are compatible (Line 2).
- H3': Checking whether $Q_S|_{(\mathcal{G}_S, G^\star)} \subseteq Q_S$ OR $Q_V|_{(\mathcal{G}_V, G^\star)} \subseteq Q_V$ (Line 10).

We bound the running time of Algorithm 2 in terms of the notation in Figure 4 and use it in Section 5 to motivate our search for tractable query classes. Given a set of compatible subgoals $\mathcal{G}_S$ in Q_S, $\mathcal{G}_V$ in Q_V (say, of relation R), the most general unifier $G^\star$ can be constructed in time $O(r_R(|\mathcal{G}_S| + |\mathcal{G}_V|))$, where r_R is the arity of relation R. The queries $Q_S|_{(\mathcal{G}_S, G^\star)}$ and $Q_V|_{(\mathcal{G}_V, G^\star)}$ can be constructed in time $O(r_R(|Q_S| - |\mathcal{G}_S| + |Q_V| - |\mathcal{G}_V|))$. Hence in total this takes $O(rn)$. Let $T(\mathsf{H3'})$ be the time taken to perform the two containment checks in (H3'). The number of sets of subgoals $\mathcal{G}_S \cup \mathcal{G}_V$ which share the same relation symbol is at most $m2^{2k}$. Hence, the running time can be bounded by

$$O(m2^{2k}(nr + T(\mathsf{H3'}))) \tag{3}$$

4.2 Perfect Privacy with Key Constraints

We can extend our results to the case when the relations have key constraints as shown in the following theorem.

Theorem 4.3. *Let Q_S and Q_V be conjunctive queries and assume that the domains of attributes are sufficiently large with respect to the queries. Q_V does not disclose any information about conjunctive query Q_S if and only if for every relation R appearing in both Q_S and Q_V,*

$$\forall \mathcal{A}_i \in \mathcal{K}_R, \quad \forall \mathcal{G}_S \subseteq Q_S, \mathcal{G}_V \subseteq Q_V, \ \textit{compatible on } \mathcal{A}_i \textit{ on } R$$
$$\left(Q_S|_{(\mathcal{G}_S, \hat{G_S}_i)} \subseteq Q_S \ \textit{OR} \ Q_S|_{(\mathcal{G}_V, \hat{G_V}_i)} \subseteq Q_V\right)$$

where $\hat{G_S}_i$ and $\hat{G_V}_i$ are the most general unifiers of $\mathcal{G}_S$ and $\mathcal{G}_V$, respectively, such that they are both compatible on the attributes in $\mathcal{A}_i$.

5. FINDING TRACTABLE CASES

In this section we show how to utilize the alternate characterizations for perfect privacy to identify query classes where checking perfect privacy is tractable. We restrict our discussion to the scenario with independent tuples, since the reason for the intractability of checking perfect privacy is the same in the other case. We will be using the notation from Figure 4 in this section.

We utilize two observations to identify tractable query classes. Our analysis of the running time of Algorithm 2 (Equation 3) shows that apart from looping over all sets of compatible subgoals $\mathcal{G}_S$ in Q_S and $\mathcal{G}_V$ in Q_V (H2') and checking two containments (H3'), all the other steps involved can be implemented in time polynomial in the size of the queries. So for queries with a bounded number of self-joins per relation (k), any query class for which query containment is tractable also permits tractable checking of perfect privacy (Section 5.1). More interestingly, in our definitions of privacy, the query containment checks we are interested in are of a special form. We illustrate this for a subclass of tableau queries, where the query containments we are interested in are so simple that we can check perfect privacy in time $O(n^2r)$, even though query containment in that class is not tractable in general (Section 5.2). In fact, perfect privacy is tractable even if the number of self-joins per relation is not bounded by a constant. We also briefly discuss the complexity of checking perfect privacy when the two queries are from two different query classes (Section 5.3).

5.1 Using Tractability of $Q_1 \subseteq Q_2$

Since the seminal paper by Chandra and Merlin [5], which showed the NP-hardness of query containment for conjunctive queries, there has been a lot of work in finding subclasses of conjunctive queries for which the query containment problem is tractable. We utilize this work to identify four major classes of queries where checking perfect privacy is tractable – queries with at most one self join per relation (using results in [20]), simple tableau queries with bounded k (using results in [2]), acyclic queries with bounded k, and queries with bounded query-width with bounded k (using [7]).

5.1.1 Queries with at most one self join per relation

We start with a very simple case. Saraiya [20] shows that checking $Q \subseteq Q'$ is tractable whenever Q and Q' are such that for every subgoal G' in Q' there are at most two subgoals in Q which are compatible with G'.

Theorem 5.1 (Tractability of Containment [20]). *Let Q_1 and Q_2 be two conjunctive queries such that every subgoal in Q_2 is compatible with at most two subgoals in Q_1. Then $Q_1 \subseteq Q_2$ can be checked in time $O(n^2 r)$*

The algorithm presented in [20] runs in time $O(n^2 r)$.

For the class of queries with at most one self-join, there are at most two subgoals of a relation in any query. Hence, any $Q|_{(\mathcal{G},\hat{G})}$ will also be a query with at most one self join. Therefore, $Q|_{(\mathcal{G},\hat{G})} \subseteq Q$ can be checked correctly using the algorithm in [20]. Moreover, since $k(Q) \leq 2$ for any query, we only need to loop over $O(n)$ sets of the form $\mathcal{G}_S \cup \mathcal{G}_V$.

Theorem 5.2 (Tractability of Perfect Privacy). *Let Q_S and Q_V be two queries having at most one self-join. Then perfect privacy can be checked in time $O(n^3 r)$.*

5.1.2 Acyclic & Bounded Query-width Queries

There has been extensive work on relating the structure of conjunctive queries and the hardness of query containment. In particular, Chekuri and Rajaram [7] relate the hardness of query containment to the cyclicity of the query graph. The *query graph* is a hyper graph whose nodes are the symbols appearing in the query and hyper-edges are the subgoals in the query. A query is *acyclic* if the query graph is acyclic.

The "degree of cyclicity" in a query is usually measured using *query-width.* Acyclic queries have a query-width of 1. The basic idea behind query-width is the following. An acyclic query can be decomposed into a tree, with each vertex in the tree corresponding to one subgoal in the query.[2] This tree preserves properties like which subgoals share variables. For a cyclic query one may be able to construct a tree with each vertex in the tree corresponding to more than one subgoal. The query-width is the largest number of subgoals which are mapped to a single vertex in the "best" tree decomposition. We refer the reader to [7] for formal definitions.

Theorem 5.3 (Tractability of Containment [7]). *Let Q_2 be a query with a query-width bounded by c. Then $Q_1 \subseteq Q_2$ can be checked in time $O(c^2 r n^{c+1} \log n)$.*

This gives us the following result on the complexity of checking perfect privacy.

[2]This can be done by using the GYO-reduction on the query-graph ([25]).

$$Q_S(name) \ :- \ P(pid, name), D(pid, \mathsf{Hepatitis}, b_1), D(pid, \mathsf{Cancer}, b_2)$$

	PID	PName	Disease	Medication
Q_S		name		
P	pid	name		
R	pid		Hepatitis	b_1
R	pid		Cancer	b_2

Figure 5: Query Q_S and its tableau representation

Theorem 5.4 (Tractability of Perfect Privacy). *Let Q_S and Q_V be queries having query-width bounded by c and at most k self-joins per relation. Then perfect privacy can be checked in time $O(m 2^{2k} c^2 r n^{c+1} \log n)$.*

5.1.3 Simple Tableau Queries

In many database schemas, different attributes in the schema are incomparable. The hospital database schema in Figure 1 is one such example. Hence, in any conjunctive query on an instance of the hospital schema, a variable cannot be bound to two different attributes. Such queries are called *tableau queries*, or short *tableaux.*

Definition 5.1 (Tableau Query [2]). *A tableau query Q is a conjunctive query where every variable (distinguished or non-distinguished) cannot be associated with two different attributes in the relational schema.*

As the name suggests, a tableau query is usually represented in a tabular form. The columns of this table are the attributes that appear in the query. The rows of the table are the subgoals of the query, and all definitions from Section 2.1 that involve subgoals thus extend to rows of a tableau query. The goal of the query is represented by the first row in the table called the summary row. Each row is labeled with the relation associated with the corresponding subgoal. Figure 5 illustrates a tableau representation of a tableau query. Note that no variable appears in more than one column. Even for this restricted class of queries, Aho et al [2] show that the query containment problem is NP-hard. They identify a further subclass of queries, called *simple tableau queries*, wherein query containment is tractable. The query illustrated in Figure 5 is an example of a simple tableau query.

Definition 5.2 (Simple Tableau Query [2]). *A tableau query is* simple *if in any column with a repeated non-distinguished variable there is no other symbol that appears in more than one row.*

Theorem 5.5 (Tractability of Containment [2]). *Let Q_1 and Q_2 be two simple tableaux. Then $Q_1 \subseteq Q_2$ can be checked in time $O(n^3 r^2)$.*

For us to be able to use the above tractability result, we need to show that given a simple tableau query Q, a set of compatible subgoals $\mathcal{G}$ and a subgoal $\hat{G}$ compatible with $\mathcal{G}$, the query $Q|_{(\mathcal{G},\hat{G})}$ is also a simple tableau query.

Lemma 5.1. *Let Q be a simple tableau query, $\mathcal{G}$ a set of compatible subgoals in Q and $\hat{G}$ a row (not necessarily in Q) compatible with $\mathcal{G}$. Then $Q|_{(\mathcal{G},\hat{G})}$ is a simple tableau query.*

Using Lemma 5.1 and Theorem 5.5 we obtain the following theorem.

THEOREM 5.6 (TRACTABILITY OF PERFECT PRIVACY). *Let Q_S and Q_V be two simple tableaux with at most k rows tagged with the same relation. Perfect privacy can be checked in time $O(m2^{2k}n^3r^2)$.*

5.2 Using Tractability of $Q|_{(\mathcal{G},\hat{G})} \subseteq Q$

Our strategy to identify tractable query classes until now as limited to identifying query classes where query containment is tractable, without taking into account that the containments we need to check have a specific form. In this section we exploit this fact to identify another query class where checking perfect privacy is tractable.

We continue our discussion on tableau queries. Define a partial order $\prec$ on the rows of the tableau as follows: if two rows r_1 and r_2 are tagged with the same relation, $r_1 \prec r_2$ if

$$\forall i \forall \mathsf{c} : \mathsf{r_1[i] = c \Rightarrow r_2[i] = c}, \text{ for some constant } \mathsf{c}$$

The above condition says that $r_1 \prec r_2$ if in every column that r_1 has a constant, r_2 has the same constant. We can now define a *maximal row*.

DEFINITION 5.3 (MAXIMAL ROW). *We call a row r in a tableau a* maximal row *if there is no other row r' tagged with the same relation as r such that $r \prec r'$.*

The query class we are considering in this section are those tableau queries which only contain maximal rows. We call such tableaux *maximal row tableaux*. The query Q_S from Figure 5 is an example of a maximal row tableaux.

DEFINITION 5.4 (MAXIMAL ROW TABLEAU). *A tableau is a maximal row tableau if for every two rows r_1 and r_2 tagged with the same the relation, $r_1 \not\prec r_2$ and $r_2 \not\prec r_1$.*

Unlike in the previous cases, where we considered query classes which permitted efficient query containment, we show that query containment is NP-hard (similar to [2]) even when the two queries are maximal row tableaux.

THEOREM 5.7 (INTRACTABILITY OF CONTAINMENT). *Let Q_1 and Q_2 be two maximal row tableaux. Checking $Q_1 \subseteq Q_2$ is NP-complete.*

Nevertheless, we are only interested in specific containments. We illustrate this using the critical tuple check. To show that a tuple t is critical to Q, it is enough to show a set of subgoals $\mathcal{G}$ in Q which are compatible with t such that $Q|_{(\mathcal{G},t)} \not\subseteq Q$. If t is compatible with $\mathcal{G}$, then t is compatible with every $G \in \mathcal{G}$. Consider $Q|_{(G,t)}$, when Q is a maximal row tableau.

$Q|_{(G,t)} \subseteq Q$ iff there is a homomorphism h from the symbols in Q to the symbols in $Q|_{(G,t)}$ such that h is an identity function on constants and for any subgoal G' in Q, $h(G')$ is a subgoal in $Q|_{(G,t)}$ [5]. However, the following argument shows that there is no h such that $h(G)$ is a subgoal in $Q|_{(G,t)}$. First, G does not appear in $Q|_{(G,t)}$. Second, since Q is a maximal row tableau, for any other subgoal G' in Q having the same relation as G, there is some column i where $G[i] = \mathsf{c}$, a constant, and $G'[i]$ is either a variable or a different constant. G' appears as $\rho_t(G')$ in $Q|_{(G,t)}$. Since no variable repeats across columns, $\rho_t(G')[i] = G'[i]$. Hence, no homomorphism will map G in Q to $\rho_t(G')$ in $Q|_{(G,t)}$, for all $G' \neq G$, implying that $Q|_{(G,t)} \not\subseteq Q$.

LEMMA 5.2. *Let Q be a maximal row tableau, t a tuple and G a subgoal in Q which is compatible with t. Then*

$$Q|_{(G,t)} \not\subseteq Q$$

As a consequence of Lemma 5.2, a tuple is critical to a maximal row tableau if and only if it is compatible with some row in the maximal row tableau. Thus we can now prove the following Theorem:

THEOREM 5.8 (TRACTABILITY OF PERFECT PRIVACY). *Let Q_S and Q_V be two maximal row tableaux. Then Q_S and Q_V share a critical tuple if and only if there are subgoals G_S in Q_S and G_V in Q_V such that G_S is compatible with G_V. Hence, perfect privacy can be checked in time $O(n^2r)$.*

5.3 Queries from Different Classes

Recall that the running time of Algorithm 2 is bounded by the product of $(nr + T(\mathsf{H3}'))$ and the number of query containments to be checked. When Q_S and Q_V are from the query classes specified in Section 5.1, the number of query containments to be checked is bounded by $O(m2^k)$. $T(\mathsf{H3}')$ is bounded by the running time of query containment for the class where checking query containment is less efficient.

As we showed in Theorem 5.8, for maximal row tableaux, we need not loop over all subsets of compatible subgoals. Hence, if Q_S is a maximal row tableau, and Q_V is from one of the subclasses listed in Section 5.1 (or vice versa), the number of containments to be checked is bounded by $O(n2^k)$. Hence, we get the following theorem,

THEOREM 5.9. *Let Q_S be a maximal row tableau. The complexity of checking whether a conjunctive query Q_V is perfectly private with respect to Q_S is*

- *$O(n \cdot r)$, if Q_V has no self-joins.*
- *$O(n \cdot n^2r)$, if Q_V has at most one self-join.*
- *$O(n2^k \cdot c^2rn^{c+1}\log n)$, if Q_V has query-width bounded by c and at most k self-joins per relation.*
- *$O(n2^k \cdot n^3r^2)$, if Q_V is a simple tableau query with at most k self-joins per relation.*

6. RELATED WORK

We start with related work on privacy policy design. In mandatory access control each data item (row, column or cell in a table) is given a security level and users with a higher clearance can access the data item [4]. Discretionary access control involves granting and revoking access authorizations to users [12]. Early authorization models did not allow negative authorization (what information a user should not access). Subsequent work propose authorization mechanisms which allow positive authorizations on views and negative authorizations only on base tables (like in [3]).

Miklau and Suciu propose a model where negative authorizations are specified using a query Q_S [18]. The user is allowed to see all the information that does not disclose any information about Q_S, hence the name perfect privacy. Enforcing perfect privacy for conjunctive queries is shown to be very intractable. This model has been extended to allow for arbitrary dependencies in the database and views which have are already been published [9]. Checking perfect privacy in this setting is even harder.

Allowing only queries which disclose no information about the Q_S might be too restrictive. A less restrictive privacy guarantee based on asymptotic conditional probabilities is presented in [8]. Certian answer privacy is yet another privacy guarantee which disallows only those Q_V which disclose that some tuple is for sure in the output of Q_S [21]. Statistical databases allow answering aggregate information about a sensitive column (like salary) while guaranteeing that the exact value in any row is not disclosed [1]. Work on de-identification attempt to ensure that an individual cannot be associated with a unique tuple in an anonymized table. k-anonymity [22] ensures that every indicvidual cannot be identified within a group of tuples of size k. Anonymity metrics based on “blending in a crowd” have been proposed assuming strict restrictions on the data distribution [6]. These, however, do not guarantee privacy of sensitive information, since all the individuals in a group could have the same sensitive information. ℓ-diversity, fixes this problem and guards against limited amounts of background knowledge by ensuring that given a sensitive attribute, every individual is associated with at least ℓ distinct values (which are roughly equi-probable) for the sensitive attribute [17].

There has been a lot of work on the problem of query containment; i.e., *“Is $Q_1 \subseteq Q_2$?”*. Query containment for conjunctive queries was shown to be NP-hard [5]. Subsequently, there has been a lot of work on identifying restricted classes of queries where the problem is tractable [2, 7, 20] (we have discussed these in detail in Section 5). The results in [7] extend those in [19, 24] for acyclic queries, since acyclic queries have query width 1. More recently, query containment has been identified to be the same problem as the constraint satisfaction problem (CSP) [16]. Gottlob et al compare different restrictions on the structure of a CSP which make the problem tractable [11]. They show that the restriction on hypertree width is the most general of these restrictions.

It has been shown that query containment is even harder for conjunctive queries with inequalities [14, 23]. Though the problem is easier if each relation symbol appears at most twice in Q_1, the problem is still shown to be co-NP-complete [15]. Restriction on query-width does not make the problem easier in the presence of inequalities.

7. CONCLUSIONS

The problem of enforcing perfect privacy was known to be highly intractable even for conjunctive queries without inequalities. In this paper we identify many subclasses of conjunctive queries for which enforcing perfect privacy is tractable. This helps policy designers to decide what policies can be efficiently enforced. As future work, we would like to identify sufficiently expressible subclasses of conjunctive queries *with* inequalities.

Acknowledgements: We thank A. Evfimievski, P. Kolaitis, R. Srikant, and H. Corrada-Bravo for insightful discussions. This work is supported by NSF grant IIS-0121175, the KDD Initiative, and by a gift from Microsoft through Jim Gray. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsors.

8. REFERENCES

[1] N. R. Adam and J. C. Wortmann. Security-control methods for statistical databases: A comparative study. *ACM Comput. Surv.*, 21(4):515–556, 1989.

[2] A. V. Aho, Y. Sagiv, and J. D. Ullman. Equivalences among relational expressions. *Siam Journal of Computing*, 1979.

[3] E. Bertino, S. Jajodia, and P. Samarati. A flexible authorization mechanism for relational data management systems. *ACM Trans. Inf. Syst.*, 1999.

[4] S. Castano, M. Fugini, G. Martella, and P. Samarati. *Database Security.* Addison Wesley, 1995.

[5] A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational databases. In *STOC*, 1977.

[6] S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee. Toward privacy in public databases. In *TCC*, 2005.

[7] C. Chekuri and A. Rajaraman. Conjunctive query containment revisited. In *ICDT*, 1997.

[8] N. Dalvi, G. Miklau, and D. Suciu. Asymptotic conditional probabilities for conjunctive queries. In *ICDT*, 2005.

[9] A. Deutsch and Y. Papakonstantinou. Privacy in database publishing. In *ICDT*, 2005.

[10] Simson Garfinkel. *Database nation: the death of privacy in the 21st century.* O’Reilly & Associates, Inc., Sebastopol, CA, USA, 2001.

[11] G. Gottlob, N. Leone, and F. Scarcello. A comparison of structural csp decomposition methods. *Artif. Intell.*, 2000.

[12] P. P. Griffiths and B. W. Wade. An authorization mechanism for a relational database system. *ACM Trans. Database Syst.*, 1(3), 1976.

[13] K. Kenthapadi, N. Mishra, and K. Nissim. Simulatable auditing. In *PODS*, 2005.

[14] A. Klug. On conjunctive queries containing inequalities. *Journal of the ACM*, 35(1):146–160, 1988.

[15] P. Kolaitis, D. L. Martin, and M. N. Thakur. On the complexity of the containment problem for conjunctive queries with built-in predicates. In *PODS*, 1998.

[16] P. Kolaitis and M. Vardi. Conjunctive-query containment and constraint satisfaction. In *PODS*, 1998.

[17] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. In *ICDE*, 2006.

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

[19] X. Qian. Query folding. In *ICDE*, 1996.

[20] Y. P. Saraiya. Polynomial-time program transformations in deductive databases. In *PODS*, 1990.

[21] K. Stoffel and M. Studer. Provable data privacy. In *DEXA*, 2005.

[22] L. Sweeney. k-anonymity: a model for protecting privacy. *IJUFKS*, 10(5):557–570, 2002.

[23] R. van der Meyden. The complexity of querying indefinite data about linearly ordered domains. *JCSS*, 54(1), 1997.

[24] M. Yannakakis. Algorithms for acyclic database schemes. In *VLDB*, 1981.

[25] C. T. Yu and M. Z. Ozsoyoglu. An algorithm for tree-query membership of a distributed query. In *IEEE COMPSAC*, 1979.