

Unauthorized Inferences in Semistructured Databases*

Csilla Farkas¹ Alexander Brodsky² Sushil Jajodia²

¹ Information Security Laboratory, Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29208

² Center for Secure Information Systems, Department of Information and Software Engineering, George Mason University, Fairfax, VA 22030-4444

Abstract

In this paper we study the problem of providing controlled accesses to confidential data stored in semistructured databases. More specifically, we focus on privacy violations via data inferences that occur when domain knowledge is combined with non-private data. A formal model, called *Privacy Information Flow Model*, is proposed to represent the information flow and the privacy requirements. To enforce these privacy requirements, the *Privacy Mediator* architecture is suggested. Privacy Mediator guarantees that recipients of data would not be able to logically entail information that violates the privacy requirements, assuming that inference is done from the currently released and previously disclosed data, and the domain knowledge. To reason about logical entailment of disclosed data, an inference algorithm is developed that is proven to be *sound* and *complete*. The inference algorithm is developed for a tree-like, semistructured data model, selection-projection queries, and domain knowledge representation based on Horn-clause constraints.

Keywords: privacy, inference problem, semistructured data, logic entailment, soundness, completeness

Contact Author: Dr. Csilla Farkas, Dept. of Computer Science and Engineering, USC
Phone: (803)576-5762, Fax: (803)777-3767, E-mail: farkas@cse.sc.edu

*This work was partially supported by the National Science Foundation under grants IIS-0237782 (Farkas), 9734242 (Brodsky), and CCR-0113515 and IIS-0242237 (Jajodia).

Unauthorized Inferences in Semistructured Databases

1 Introduction

Privacy is defined as an individual's control over disclosure and use of personal information [68]. In the age of globally interconnected information systems, the protection of privacy has become increasingly difficult. Data protection methods, like authentication, access control, and cryptography, prevent security violations via direct attacks. However, seemingly safe actions, like release of authorized data, may also lead to indirect disclosure of private data via inferences. Medical databases are especially vulnerable to such attacks because they contain large amounts of confidential information and the corresponding medical domain knowledge, leading to the undesired inferences, is complex and unknown by non-medical workers.

For example, consider the scenario in which Jane Smith would like to undergo genetic testing to determine whether she is at high risk for developing breast cancer. The presence of mutated BRCA1 and BRCA2 genes indicates increased risk of developing breast cancer. Ms. Smith would like to keep the testing and the test results hidden from her health insurance company. She pays for the testing and instructs the physician's office and the medical laboratory not to release to the health insurance company the information that she was tested nor the results. * Since Ms. Smith's test results are positive, her physician prescribes the drugs Raloxifene and Herceptin to reduce the risk of developing breast cancer. Ms. Smith obtains the medications from her pharmacy, which then sends the prescription and the billing information to her insurance company. Unfortunately, this unique combination of drugs is used only to treat people who are at high risk of developing breast cancer. Consequently, the insurance company assumes that Ms. Smith has breast cancer and terminates her insurance.

Ms. Smith's privacy has been violated. How could it be prevented? Ms. Smith, like most people, wants to protect her private data and at the same time obtain services like insurance coverage. Clearly, there is a trade off between privacy and data sharing, needed to obtain services. However, we believe that it is possible to achieve the "right balance" between privacy and data sharing so that individuals will be able to make informed decisions. Since the insurance company used the prescription information to infer Ms. Smith's illness, a naive solution to protect her privacy could be to keep the prescription information private. However, while this solution would guarantee her privacy, it would also force her to pay for all of her medications. An improved solution is to withhold just enough information about the prescriptions so that the insurance company will not be able to infer Smith's illness, e.g., submit only one of the prescriptions. The second approach protects Ms. Smith's privacy and allows her to get reimbursed for some of her expenses.

To address privacy problems in various domains, such as the medical domain, appropriate legislative and law-enforcement mechanisms are required. It is also necessary to develop technical solutions which not only enforce the privacy law and organizational rules but also address the privacy versus information sharing problem as presented in the example above.

*Note that the scope of this paper does not include the ethical problem whether a patient should be allowed to hide relevant information from health insurance organizations.

This is exactly the focus of this paper.

Privacy protection requires methods to control accesses to confidential and personal data. Access control models, like Discretionary (DAC), Mandatory (MAC), and Role-Based Access Control (RBAC) [5, 55, 54] prevent unauthorized direct accesses to data. Recently, authorization frameworks based on logic formalism have also surfaced [23, 19, 24, 39, 9, 6, 7]. They propose policy languages that go beyond traditional access control methods to address obligation, provision, and delegation of authorization. The main focus of these models is the control of direct information accesses without considering unauthorized inferences.

The inference problem was first studied in statistical databases [28] where inferences, derived from released statistics about groups of individuals, may disclose information about a particular individual. During the 1980s, researchers investigated the inference problem in multilevel security (MLS) relational databases. Unauthorized inference channels occur if a high security data item can be disclosed using low security data items and meta-data, like database dependencies (see [38] for an overview). However, the achievements of MLS database inferences have not been applied to evaluate the privacy implications of inferences in semistructured databases. The rapid development of the Internet, wide spread use of interconnected databases, and the increase of the amount of digitalized personal data increase the risk of unauthorized disclosure of private data. Existing methods [22, 46, 49, 53, 59, 69, 8, 25, 41] focus on preventing privacy violations via direct data accesses without considering the inference problem.

Recent research on privacy preserving data mining [3, 21, 20, 50, 51] and data linkage [37, 36, 67, 61] is motivated by the need to protect sensitive data at the growing presence of data mining applications. Privacy concerns include individual user's personal information and information about the activities of a group of users. Work on data linkage addresses the problem of disclosing personal data from aggregate information or from separately released, non-confidential data of an individual. Prevention methods are based on techniques like perturbation and generalization.

This paper addresses two limitations of the current methods. First, none of the existing works considers applications of domain knowledge over released data as the source of unwanted inferences. Second, they do not address the problem of inferences over heterogeneous data. Data, corresponding to an individual, often originate from different sources where each source may maintain its own specific data model and format. The inference engine needs to be able to handle data that may not conform to specific structure.

This paper develops a framework and algorithmic solutions for preventing privacy violations via illegal data inferences that occur when domain knowledge is combined with non-private data. Our solution works on a semistructured data model [44, 2, 16, 17], without the rigid structure requirements of traditional database management systems. Semistructured data can incorporate irregular, unknown, or rapidly changing structure [62], thus it is a promising candidate to represent heterogeneous data.

We propose a formal model, called *Privacy Information Flow (PIF) Model*, to specify how information is transferred among participants (e.g., people and organizations) as well as to specify privacy requirements. The privacy information flow model consists of two main layers: a *Privacy Information Flow (PIF) Graph* that is created for every application domain (e.g., the health care domain) and participant type (e.g., patients), and a *Privacy Contract* that creates customized requirements for each individual participant (e.g., for patient Jane

Smith). The PIF graph represents the permitted data items that can be transferred between two participants. A privacy contract specifies, for an individual participant, the information that is not allowed to be transferred to or logically entailed by other participants (e.g., a health insurance company).

We also develop a privacy architecture, called *Privacy Mediator* (PriMe), that enforces the privacy requirements on every data transfer and provides maximal data availability. PriMe extends a *standard access control mechanism* with an *inference engine*. All data transfers are performed via PriMe which guarantees that no data transfer is performed if it violates the privacy requirements, even if the disallowed data item is not directly transferred but can be logically entailed from the domain knowledge and the accumulated and transferred data. On the other hand, PriMe guarantees maximal data availability, that is, all data transfers that do not cause privacy violations will be allowed. The main enabling mechanism of PriMe is an *inference engine* to reason about logical entailment of data.

We develop a sound and complete inference algorithm that is used by the inference engine. The inference engine generates all disclosed information from the current and previously disclosed data, and the domain knowledge. We consider a tree-like semistructured database model, similar to [14] which is suitable to describe information from heterogeneous data sources as well as traditional relational databases. We consider selection-projection *queries*. Traditional database dependencies, such as functional, multi-valued, and join dependencies, and the domain knowledge are represented as *Horn-clause constraints*. We say that an answer T to a query q is disclosed by the domain knowledge \mathcal{K} from the answers T_1, \dots, T_n to queries q_1, \dots, q_n , respectively, if for every database that satisfies \mathcal{K} the following holds: If T_1, \dots, T_n are in the answers to queries q_1, \dots, q_n , respectively, then T must be in the answer to q . We prove that the developed disclosure inference algorithm is *sound*, i.e., all information that is generated by the algorithm is indeed disclosed, and *complete*, i.e., all the information that can be disclosed is generated by the algorithm.

The organization of the paper is as follows. In Section 2 we define the semistructured data model, the select-project queries, and the notion of disclosure inference. In Section 3 we present the privacy information flow model and the privacy mediator architecture. Section 4 contains the developed disclosure inference algorithm and the proof of its soundness and completeness. In Section 5 we give an overview of related research. Finally, in Section 6 we conclude and suggest future research topics and improvements to our model.

2 Data Model, Queries, and Disclosure Entailment

Semistructured data models emerged during the late 1990s [44, 2, 14, 16, 17, 15] to accommodate data that do not conform to a specific structure or the structure of the data changes rapidly. Biological databases, World Wide Web, and integrated heterogeneous databases are examples of applications requiring semistructured data model. Intuitively, semistructured databases are graph-like or tree-like structures, where internal components represent complex values, and leaves represent atomic values. In this paper we consider edge-labeled, tree-like databases, conforming to the data model proposed by Buneman et al. [14, 15, 16, 17]. Data accesses are permitted through select-project queries. We suggest a Horn-clause representation of database constraints and domain knowledge.

2.1 Data Model

In our model a *database* is represented as an edge-labeled tree. We start with an example of a simple *Medical Database* (MED-DB), shown in Figure 1. The database contains data collected from heterogeneous sources, like pharmacies, health insurance companies, and a medical offices. Databases, maintained by the different organizations, may conform to different schemas. For example, different pharmacies may store information about customers differently.

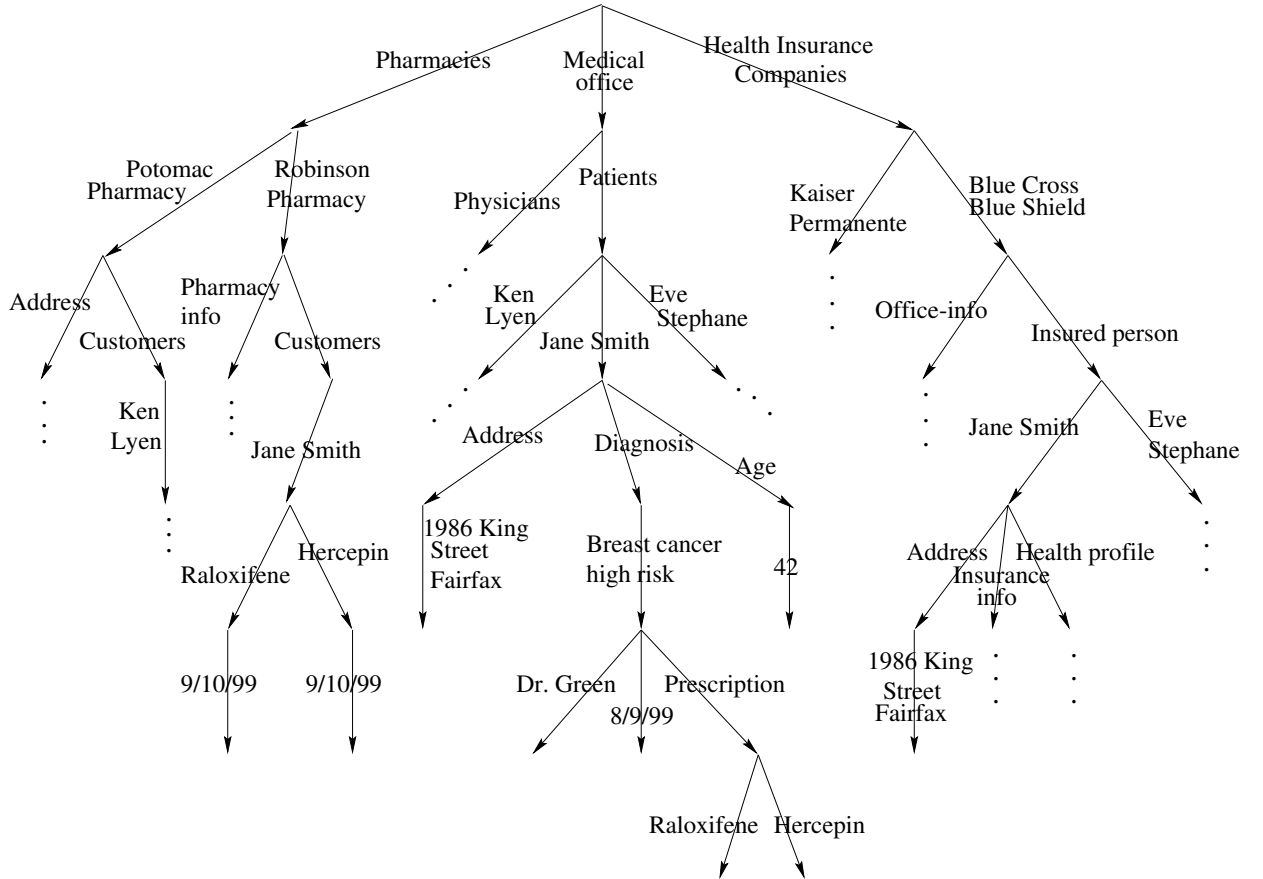


Figure 1: Medical Database (MED-DB)

We capture edge-labeled trees with the symbolic *tree-expressions* (TE).

Definition 2.1 (Tree-Expression, TE)

Let L be a set of labels containing constants, null-values, and variables.

1. The empty tree $\{\}$ is a tree-expression (TE), called the empty tree.
2. A single edge $\{l\}$, where $l \in L$, is a TE.

3. Let t_1, \dots, t_n be TEs and l_1, l_2, \dots, l_n labels in L . Then $\{l_1 \Rightarrow t_1, l_2 \Rightarrow t_2, \dots, l_n \Rightarrow t_n\}$ is a TE, where $l_i \neq l_j$ for two different i and j . In this case we say that $\{l_1 \Rightarrow t_1, l_2 \Rightarrow t_2, \dots, l_n \Rightarrow t_n\}$ represents the tree whose root has the outgoing edges labeled l_1, l_2, \dots, l_n to which the trees t_1, t_2, \dots, t_n are attached, respectively. For clarity, if the empty tree t_i is attached to the edge l_i , i.e., $\{\dots, l_i \Rightarrow \{\}, \dots\}$, we omit the empty tree, i.e., represent the previous expression as $\{\dots, l_i, \dots\}$

The TE representation of the Pharmacy subtree of MED-DB is given below.

$$\begin{aligned} &\{Pharmacies \Rightarrow \\ &\quad \{Potomac Pharmacy \Rightarrow \\ &\quad\quad \{Address \Rightarrow \{\dots\}, \\ &\quad\quad Customers \Rightarrow \\ &\quad\quad\quad \{KenLyen \Rightarrow \{\dots\}\}\}, \\ &\quad Robinson Pharmacy \Rightarrow \\ &\quad\quad \{Pharmacy info \Rightarrow \{\dots\}, \\ &\quad\quad Customers \Rightarrow \\ &\quad\quad\quad \{Jane Smith \Rightarrow \\ &\quad\quad\quad\quad \{Raloxifene \Rightarrow \{9/10/99\}, \\ &\quad\quad\quad\quad Herceptin \Rightarrow \{9/10/99\}\}\}\dots\} \end{aligned}$$

Note that in our model for every node of a TE each outgoing edge label is unique, i.e., there are no two outgoing edges with the same label. Intuitively, this limitation of our model is very similar to the restriction of the Data Type Definition (DTD) language over the eXtensible Markup Language (XML). This restriction will be necessary when generating the disclosure cover of disclosed data as described in Section 4. From now on, we use the terms *tree-expression* (TE) and *tree* interchangeably.

Labels in TEs may be constants, null-values, or variables. Constants represent edge labels of the database. Null-values correspond to label values not known by the user, i.e., values that have not been released to the user and cannot be inferred by the user. Edges labeled by null-values are needed to preserve structural information and equality between null-values. Initially, all null-values are unique. Two null-values are equated only if the database constraints or the selection condition in the where clause of the query justify this equality. This feature allows to represent cases when a user knows that two data items must have the same value without knowing the actual value. There are two kinds of variables in our model: 1) label-variables and 2) tree-variables. Label-variables correspond to a set of possible values of the edge labels that are known by the user. We use the letters x, y, z to represent label-variables. Tree-variables correspond to a set of possible subtrees that are known by the user. We use the letter t to represent tree-variables. We say that a TE is *ground* if all of its labels are constants. A database is a ground TE. We use TEs in queries, query answers, and to represent domain knowledge and disallowed information.

2.2 Queries

In this section we define the syntax and semantics of the queries that we consider. Our syntax follows the well-known “select-project” format of SQL queries. Intuitively, **select** defines the content and structure of the query answer, while **from** and **where** defines the

database being searched and the search conditions. For example, consider the query which retrieves pharmacy prescription information of customer Jane Smith from the MED-DB database (Figure 1).

Query 1:

```

select  $\{x \Rightarrow t\}$ 
from  $\{ Pharmacies \Rightarrow \{y \Rightarrow \{Customers \Rightarrow \{x \Rightarrow t\}\}\} \} \leftarrow \text{MED-DB}$ 
where  $x = \text{Jane Smith}$ 

```

where x, y are label-variables, t is a tree-variable, and *Pharmacies*, *Customers* are constants (used as labels).

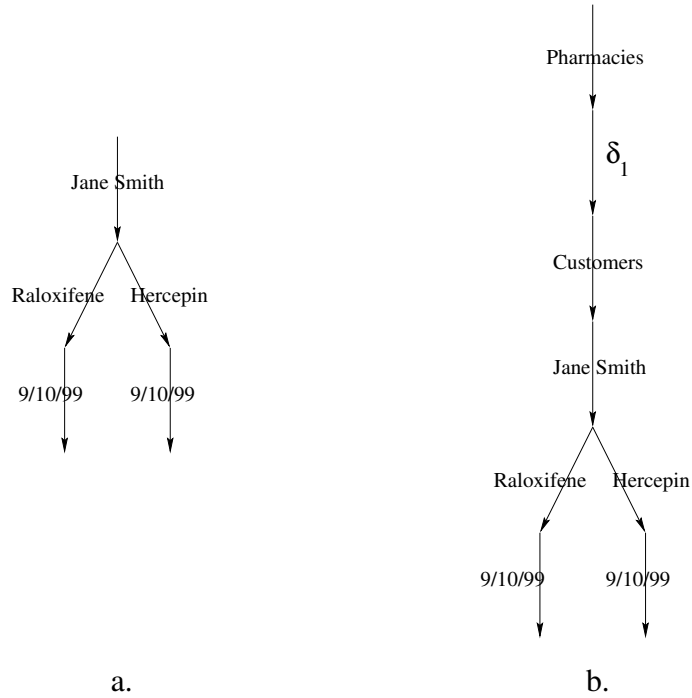


Figure 2: Answer (a) and Disclosed Query Data (b) of **Query Q**

Intuitively, the answer to Query 1 is computed by finding all instantiations to the variables x, y and t that makes the TE $\{ Pharmacies \Rightarrow \{y \Rightarrow \{Customers \Rightarrow \{x \Rightarrow t\}\}\} \}$ “match” the MED-DB tree in Figure 1. Then, only those instantiations that satisfy the **where** clause are taken (i.e., those in which x is instantiated to *Jane Smith*). For each such instantiation the structure $x \Rightarrow t$ is retrieved to construct the combined answer tree. For our example, the answer is shown in Figure 2.a.

More formally, we consider *select-project* queries of the following form:

```

select  $s$ 
from  $P \leftarrow DB$ 

```

where c_1, \dots, c_m (1)
 where

- P and s are TEs with no null-values, i.e., they only involve constants and label- and tree-variables. It is required that all variables of s also appear in P .
- c_1, \dots, c_m are equalities among constants and label-variables that appear in P , like $x = y$, $x = 3$, where x, y are variables, and 3 is constant.

To define the semantics of queries, we need the notions of query pattern, tree-union and tree-mapping. Intuitively, a query pattern encodes P and the equalities c_1, \dots, c_m in a single TE.

Definition 2.2 (Query Pattern)

Given a query Q of the form (1), the *query pattern* q of Q is a TE defined by construction, as follows:

1. Initially, let q be P
2. If the conjunction of the equalities c_1, \dots, c_m in the **where** clause implies that
 - (a) $x = l$ where x is a variable and l is a constant, then
 - replace all occurrences of x in q with l
 - (b) $x = y$ where both x and y are variables, then
 - if only one of the variables, say x , appears in the “select” clause of the query, then replace all occurrences of y in q with x
 - otherwise (neither of them or both of them appear in the “select” clause, then replace one of the variables, say x , with y in q

For example, the query pattern for Query 1 is generated from

$$\{ \text{Pharmacies} \Rightarrow \{y \Rightarrow \{ \text{Customers} \Rightarrow \{x \Rightarrow t\}\}\} \}$$

by replacing x with the constant *Jane Smith*. That is

$$\{ \text{Pharmacies} \Rightarrow \{y \Rightarrow \{ \text{Customers} \Rightarrow \{ \text{Jane Smith} \Rightarrow t\}\}\} \}$$

A similar example can be constructed to find the names of the customers who purchased both Raloxifene and Herceptin on the same day. The select-project query is:

Query 2:

```
select {x ⇒ v}
from { Pharmacies ⇒ {y ⇒ {Customers ⇒ {x ⇒
  {Raloxifene ⇒ v, Herceptin ⇒ w}\}\}\} } ← MED-DB
```

where $v = w$

The corresponding query pattern is constructed by replacing the variable w with the variable v (condition 2.b of Definition 2.2):

$$\{ \textit{Pharmacies} \Rightarrow \{ y \Rightarrow \{ \textit{Customers} \Rightarrow \{ x \Rightarrow \{ \textit{Raloxifene} \Rightarrow v, \textit{Herceptin} \Rightarrow v \} \} \} \} \}$$

Next, we define the *tree-union* of two TEs that allows to unite single trees. This operation is needed to combine separate query answers into a single tree.

Definition 2.3 (Tree-union)

Let T, T_1, T_2 be TEs. The *tree-union*, denoted by \sqcup , is defined recursively as follows:

1. $\{ \} \sqcup T = T \sqcup \{ \} \stackrel{def}{=} T$
2. If $T_1 = \{ l_1 \Rightarrow t_1, \dots, l_k \Rightarrow t_k, l_{k+1} \Rightarrow t_{k+1}, \dots, l_m \Rightarrow t_m \}$
and $T_2 = \{ l_{k+1} \Rightarrow t'_{k+1}, \dots, l_m \Rightarrow t'_m, l_{m+1} \Rightarrow t'_{m+1}, \dots, l_n \Rightarrow t'_n \}$
(i.e., l_{k+1}, \dots, l_m are the common labels), then
 $T_1 \sqcup T_2 \stackrel{def}{=} \{ l_1 \Rightarrow t_1, \dots, l_k \Rightarrow t_k, l_{k+1} \Rightarrow t_{k+1} \sqcup t'_{k+1}, \dots, l_m \Rightarrow t_m \sqcup t'_m, l_{m+1} \Rightarrow t'_{m+1}, \dots, l_n \Rightarrow t'_n \}$

$$\{ l_1 \Rightarrow t_1, \dots, l_k \Rightarrow t_k,$$

$$l_{k+1} \Rightarrow t_{k+1} \sqcup t'_{k+1}, \dots, l_m \Rightarrow t_m \sqcup t'_m,$$

$$l_{m+1} \Rightarrow t'_{m+1}, \dots, l_n \Rightarrow t'_n \}$$

For example, consider the following two TEs:

TE 1:

$$\{ \textit{Pharmacies} \Rightarrow \{ \textit{Potomac Pharmacy} \Rightarrow \{ \textit{Customers} \Rightarrow \{ \textit{Ken Lyen} \} \} \} \}$$

TE 2:

$$\{ \textit{Pharmacies} \Rightarrow \{ \textit{Robinson Pharmacy} \Rightarrow \{ \textit{Customers} \Rightarrow \{ \textit{Jane Smith} \Rightarrow \{ \textit{Raloxifene}, \textit{Herceptin} \} \} \} \} \}$$

The only common label of the two TEs is the *Pharmacies*. The tree union on **TE 1** and **TE 2** yields the single tree:

$$\{ \textit{Pharmacies} \Rightarrow \{ \textit{Potomac Pharmacy} \Rightarrow \{ \textit{Customers} \Rightarrow \{ \textit{Ken Lyen} \} \}, \textit{Robinson Pharmacy} \Rightarrow \{ \textit{Customers} \Rightarrow \{ \textit{Jane Smith} \Rightarrow \{ \textit{Raloxifene}, \textit{Herceptin} \} \} \} \} \}$$

To define the semantics of our select-project queries, we need the notions of tree-mapping and satisfaction of a TE. We will illustrate these concepts with an example after the formal definition of query semantics. We begin with the definition of the tree-mapping.

Definition 2.4 (Tree-mapping)

First, we define symbol-mapping λ from the edge labels of a TE T_1 to the edge labels (constants) and subtrees of a ground TE T_2 as a function that

1. preserves constants
2. preserves equalities and
3. maps a tree-variable to a subtree of T_2 .

The *tree-mapping* Λ , originating from λ , is defined as a mapping such that

1. $\Lambda(\{\}) = \{\}$
2. $\Lambda(t) = \lambda(t)$, where t is a tree-variable
3. $\Lambda(\{l_1 \Rightarrow t_1, \dots, l_k \Rightarrow t_n\}) = \{\lambda(l_1) \Rightarrow \Lambda(t_1), \dots, \lambda(l_k) \Rightarrow \Lambda(t_n)\}$

Definition 2.5 (Ground TE satisfies a TE)

Let T_1 be a TE and T be a ground TE. We say that T *satisfies* T_1 if there exists a tree-mapping Λ such that $\Lambda(T_1)$ is a subtree of T . In this case, we say that T satisfies T_1 with respect to Λ .

Similarly, we say that a ground TE T *satisfies* a set $\{T_1, \dots, T_k\}$ of arbitrary TEs if there exists a tree-mapping Λ such that T satisfies all T_1, \dots, T_k with respect to Λ .

Example 2.1 Consider the TE, where x and y are label-variables:

$$T_1 = \{x \Rightarrow \{ \textit{JaneSmith} \Rightarrow y \} \}$$

and the ground TE:

$$T_2 = \{ \textit{Customer} \Rightarrow \{ \textit{JaneSmith} \Rightarrow \{ \textit{Raloxifene}, \textit{Herceptin} \} \} \}.$$

There are two tree-mappings from T_1 to T_2 using symbol-mappings λ_1 and λ_2 , where

$$\lambda_1(x) = \textit{Customer}, \lambda_1(\textit{Jane Smith}) = \textit{Jane Smith}, \lambda_1(y) = \textit{Raloxifene}$$

$$\lambda_2(x) = \textit{Customer}, \lambda_2(\textit{Jane Smith}) = \textit{Jane Smith}, \lambda_2(y) = \textit{Herceptin}.$$

Consequently,

$$\Lambda_1(T_1) = \{ \textit{Customer} \Rightarrow \{ \textit{JaneSmith} \Rightarrow \textit{Raloxifene} \} \}$$

$$\Lambda_2(T_1) = \{ \textit{Customer} \Rightarrow \{ \textit{JaneSmith} \Rightarrow \textit{Herceptin} \} \}$$

Therefore, T_2 satisfies T_1 with respect to Λ_1 and Λ_2 . □

Now, we can formally define the semantics of a select-project query.

Definition 2.6 (Query semantics)

Let Q be a select-project query of the form given in (1), and DB be a ground TE. The *answer* to Q from DB is a ground TE, defined as follows:

Let $\Lambda_1, \dots, \Lambda_k$ be all tree-mappings such that DB satisfies q with respect to Λ_i $i = 1, \dots, k$ and q is the query pattern of Q (Definition 2.2). The answer to the query is $\Lambda_1(s) \sqcup \dots \sqcup \Lambda_k(s)$, where s is the formula defined in the **select** of Q and \sqcup is the tree-union.

When a query is evaluated, all *tree-mappings*, that maps the the query pattern to the subtrees of the database, are generated, i.e., the database satisfy the query pattern with respect to these mappings. Consider Query 1 again. All *tree-mappings* from

$$q = \{ \text{Pharmacies} \Rightarrow \{ y \Rightarrow \{ \text{Customers} \Rightarrow \{ \text{JaneSmith} \Rightarrow t \} \} \} \}$$

to the subtrees of MED-DB are generated. In our example, only one such tree-mapping exists that originates from the following λ :

$$\begin{aligned} \lambda(\text{Pharmacies}) &= \text{Pharmacies} \\ \lambda(\text{Customers}) &= \text{Customers} \\ \lambda(\text{Jane Smith}) &= \text{JaneSmith} \\ \lambda(y) &= \text{Robinson Pharmacy} \\ \lambda(t) &= \{ \text{Raloxifene} \Rightarrow 9/10/99, \text{Herceptin} \Rightarrow 9/10/99 \} \end{aligned}$$

The tree-mapping, Λ , maps q to the subtree

$$\{ \text{Pharmacies} \Rightarrow \{ \text{RobinsonPharmacy} \Rightarrow \{ \text{Customers} \Rightarrow \{ \text{JaneSmith} \Rightarrow \{ \text{Raloxifene} \Rightarrow 9/10/99, \text{Herceptin} \Rightarrow 9/10/99 \} \} \} \} \}$$

Therefore, the *answer* to Query 1 is:

$$\{ \text{JaneSmith} \Rightarrow \{ \text{Raloxifene} \Rightarrow 9/10/99, \text{Herceptin} \Rightarrow 9/10/99 \} \}$$

The corresponding tree is given in Figure 2.a.

2.3 Constraints and Logical Entailment of Disclosure

From the perspective of privacy and security, we are only interested in information about the original database that is disclosed if a query is answered. Structural and presentation

requirements of the returned answer, enforced by the **select** clause of the query, are not needed. We assume, that each query is associated with a single participant. This participant may be a single user, e.g., a patient or a doctor, or an organization, e.g., a pharmacy or a health insurance company. In either case, we model the data that can be disclosed by that participant, whether it is a single user or a group users working for an organization. From now on, we use the terms “participant” and “user” interchangeably.

Query patterns may contain variables that are used in the evaluation of the query but the actual instantiations of these variables were never released to the user. We call these variables “unknown” variables. We replace each of these unknown variables with a unique null-value. The reason of assigning unique null-values is that we do not want to introduce any equalities among the “unknown” variables that are not the consequences of a query selection condition or a domain knowledge. Such unwarranted equalities could lead to incorrect inferences.

We define the notion of *disclosed query data* to represent the disclosed information explicitly.

Definition 2.7 (Disclosed Query Data)

Let Q be a select-project query of the form given in (1), and DB be a ground TE. The *disclosed query data* by Q from DB is a TE, defined as follows:

1. Let q be the query pattern of Q (Definition 2.2), $\Lambda_1, \dots, \Lambda_k$ be all tree-mappings such that DB satisfies q with respect to Λ_i $i = 1, \dots, k$, and $X = x_1, \dots, x_l$ be the variables of q that are not in s .
2. Modify Λ_i such that $\lambda_i(x_j) = \delta_j$, where δ_j is a *unique null-value* for all $j = 1, \dots, l, i = 1, \dots, k$.
3. The disclosed query data is defined as: $\Lambda_1(q_1) \sqcup \dots \sqcup \Lambda_k(q_k)$ where \sqcup is the tree-union.

For example, the *disclosed query data* of Query 1 is:

$$\{ \text{Pharmacies} \Rightarrow \{ \delta_1 \Rightarrow \{ \text{Customers} \Rightarrow \{ \text{JaneSmith} \Rightarrow \{ \text{Raloxifene} \Rightarrow 9/10/99, \text{Hercepin} \Rightarrow 9/10/99 \} \} \} \} \}$$

The corresponding tree is depicted in Figure 2.b.

We represent domain knowledge and database constraints as a set of Horn-clause constraint. These constraints are used in the model to derive information that is not explicitly released to the user, but can be deduced from the released data and the associated constraints.

Definition 2.8 (Horn-clause constraints)

A *Horn-clause constraint* is an expression of the form

$$\forall x_1, \dots, x_m (B_1 \wedge \dots \wedge B_n \rightarrow H)$$

where $n \geq 1$, B_1, \dots, B_n are *tree expressions* that may contain variables and constants and x_1, \dots, x_n are all the label and tree-variables in $B_1 \wedge \dots \wedge B_n \rightarrow H$. Each H has one of the following forms:

1. H is a TE with no null-values. We require that all label- and tree-variables of H must appear in B_1, \dots, B_n . In this case we say that the constraint is *tree-generating*.
2. H is an equality of the form $a = b$, where each a and b are either constants or label-variables that appears in B_1, \dots, B_n . In this case we say that the constraint is *equality-generating*.

We will refer to $B_1 \wedge \dots \wedge B_n$ as the *body* and H as the *head* of the constraint. We require that all labels of H appear in the body. We will use the shorthand $B_1 \wedge \dots \wedge B_n \rightarrow H$ for the Horn-clause constraint $\forall x_1, \dots, x_m (B_1 \wedge \dots \wedge B_n \rightarrow H)$.

Horn-clause constraints can express database dependencies and domain knowledge [66]. Consider again the medical database (MED-DB) and the domain knowledge that the combination of the drugs *Raloxifene* and *Herceptin* is used to treat patients who are likely to develop breast cancer. Individually, Raloxifene and Herceptin are also used to treat other illnesses, e.g., Raloxifene is used to treat osteoporosis. This domain knowledge of MED-DB can be represented as Horn-clause constraint of the following form, where x, y_1 , and y_2 are label-variables.

$$\begin{aligned} & \{Pharmacies \Rightarrow \{y_1 \Rightarrow \{Customers \Rightarrow \{x \Rightarrow Raloxifene\}\}\}\} \wedge \\ & \{Pharmacies \Rightarrow \{y_2 \Rightarrow \{Customers \Rightarrow \{x \Rightarrow Herceptin\}\}\}\} \rightarrow \\ & \{Medical\ Office \Rightarrow \{Patients \Rightarrow \{x \Rightarrow \{Diagnosis \Rightarrow Breast\ cancer\ high\ risk\}\}\}\} \end{aligned}$$

To demonstrate the equality-generating dependencies, assume that the database satisfies the requirement that a patient has only one address. That is, the address of a patient must be the same in the database of the Medical Office and in the database of the Health Insurance Companies. This can be represented by the rule:

$$\begin{aligned} & \{Medical\ Office \Rightarrow \{Patient \Rightarrow \{x \Rightarrow \{Address \Rightarrow y_1\}\}\}\} \wedge \\ & \{Health\ Insurance\ Companies \Rightarrow \{Blue\ Cross \Rightarrow \{Insured\ Person \Rightarrow \\ & \{x \Rightarrow \{Address \Rightarrow y_2\}\}\}\}\} \rightarrow y_1 = y_2 \end{aligned}$$

Definition 2.9 (Constraint satisfaction)

Let d be a Horn-clause constraint of the form $B_1 \wedge \dots \wedge B_n \rightarrow H$. We say that a ground tree T satisfies d if for every tree-mapping Λ where T that satisfies $\{B_1, \dots, B_n\}$ with respect to Λ the following holds:

1. If d is a tree-generating constraint, then T also satisfies H with respect to Λ , i.e., $\Lambda(H)$ is a subtree in T .
2. If d is a equality-generating constraint where H is of the form $a = b$, then $\lambda(a) = \lambda(b)$ in T , where λ is the symbol-mapping for Λ .

Consider again the previous Horn-clause constraint. This constraint means that if there exists a tree-mapping Λ from the TEs

$$T_1 = \{Pharmacies \Rightarrow \{y_1 \Rightarrow \{Customers \Rightarrow \{x \Rightarrow Raloxifene\}\}\}\}$$

and

$$T_2 = \{Pharmacies \Rightarrow \{y_2 \Rightarrow \{Customers \Rightarrow \{x \Rightarrow Hercepin\}\}\}\}$$

to a database such that $\Lambda(T_1)$ and $\Lambda(T_2)$ are subtrees of the database, then

$$\Lambda(\{Medical\ Office \Rightarrow \{Patients \Rightarrow \{x \Rightarrow \{Diagnosis \Rightarrow Breast\ cancer\ high\ risk\}\}\}\})$$

must also be a subtree of the database.

Let Λ be a tree-mapping that maps

$$\{Pharmacies \Rightarrow \{y_1 \Rightarrow \{Customers \Rightarrow \{x \Rightarrow Raloxifene\}\}\}\}$$

to

$$\{Pharmacies \Rightarrow \{Robinson\ Pharmacy \Rightarrow \{Customers \Rightarrow \{Jane\ Smith \Rightarrow Raloxifene\}\}\}\}$$

and

$$\{Pharmacies \Rightarrow \{y_2 \Rightarrow \{Customers \Rightarrow \{x \Rightarrow Hercepin\}\}\}\}$$

to

$$\{Pharmacies \Rightarrow \{Robinson\ Pharmacy \Rightarrow \{Customers \Rightarrow \{Jane\ Smith \Rightarrow Hercepin\}\}\}\}$$

The corresponding symbol-mapping λ is:

$$\lambda(y_1) = \lambda(y_2) = Robinson\ Pharmacy$$

$$\lambda(x) = Jane\ Smith$$

Λ maps the head of the dependency

$$\{Medical\ Office \Rightarrow \{Patients \Rightarrow \{x \Rightarrow \{Diagnosis \Rightarrow Breast\ cancer\ high\ risk\}\}\}\}$$

to the subtree

$$\{Medical\ Office \Rightarrow \{Patients \Rightarrow \{Jane\ Smith \Rightarrow \{Diagnosis \Rightarrow Breast\ cancer\ high\ risk\}\}\}\}$$

Therefore, MED-DB satisfies the above Horn-clause constraint.

An example of equality generating dependency equates the address of the same patient in the Medical Office database and the Health Insurance Companies database. The dependency is represented as:

$$\{MedicalOffice \Rightarrow \{Patients \Rightarrow \{x \Rightarrow \{Address \Rightarrow y\}\}\}\} \wedge \{HealthInsuranceCompanies \Rightarrow \{z \Rightarrow \{Insuredperson \Rightarrow \{x \Rightarrow \{Address \Rightarrow w\}\}\}\}\} \rightarrow y = w$$

The mappings

$$\lambda(x) = Jane\ Smith$$

$$\lambda(y) = 1986\ King\ Street\ Fairfax$$

$$\lambda(z) = Blue\ Cross\ Blue\ Shield$$

Then $\lambda(w)$ must be be 1986 King Street Fairfax

Definition 2.10 (Logical Entailment of Disclosure)

Let \mathcal{K} be a set of Horn-clause constraints, T_1, \dots, T_n and T are TEs. We say that T_1, \dots, T_n *disclose* T under \mathcal{K} , denoted as $T_1, \dots, T_n \models_{\mathcal{K}} T$, if for every database DB that satisfies \mathcal{K} the following holds: If DB satisfies T_1, \dots, T_n with respect to a tree-mapping Λ then DB also satisfies T with respect to Λ .

Definition 2.10 guarantees the existence of a subtree $\Lambda(T)$ in any database that satisfies \mathcal{K} and contains $\Lambda(T_1), \dots, \Lambda(T_n)$. In our example, assume that a database satisfies the domain knowledge that Ralixofene and Hercepin are used to treat high-risk breast cancer patients, and that a patient, say Ms. Smith, takes these drugs. Then it is logically entailed that the database contains the information that Ms. Smith is a high-risk breast cancer patient.

3 Privacy Information Flow Model and Privacy Mediator

3.1 The Model

In this section we present a *privacy information flow (PIF) model* to represent allowed information flow within a domain and define the privacy requirements. The PIF model has two main components: 1) a *privacy information flow (PIF) graph* and 2) *individual privacy requirements (IPR)*. Before giving formal definitions, we first illustrate the purpose and

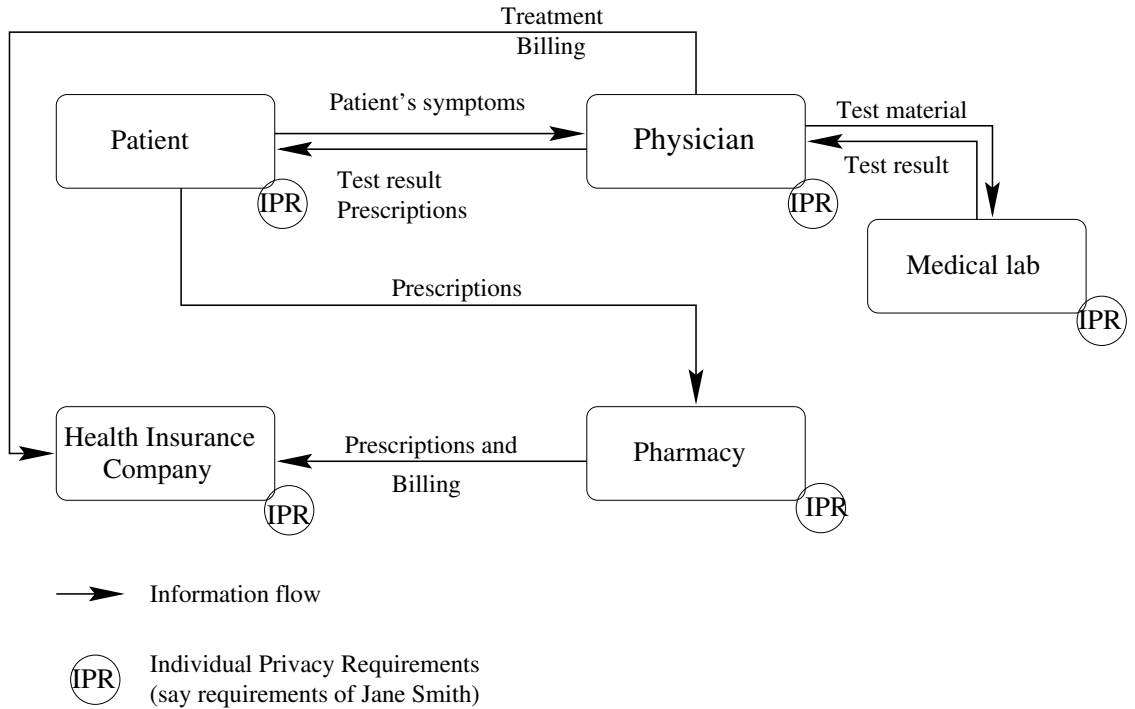


Figure 3: Medical Information Flow: Patient's perspective

functionality of these components using the medical example of the Introduction. Figure 3 shows the PIF graph built for our example. Each node in the graph represents the type of a participant. In our example the participants are *Patient*, *Physician*, *Pharmacy*, *Medical lab* and *Health Insurance Company*. In general, participant-types may correspond to individuals, organizations, or their groups that share the same properties and roles. Each arrow in the PIF graph indicates the direction of the information flow and its label *TI* the type of the transferred information. For example, a physician may transfer *treatment* and *billing* information to the health insurance company, *test material* to the medical lab, and *test results* and *prescriptions* to the patient. Intuitively, the label *TI* indicates that nothing but *TI* is allowed to be transferred between the two participants. However, *TI* may be further limited by an individual participant, e.g., by a patient, based on her/his individual privacy requirements (IPR's). For example, Jane Smith may disallow her physician to transfer any treatment information about her genetic and chronic diseases to the health insurance company.

We begin by defining the transferred information specification, then follow with the formal definition of the Privacy Information Flow Graph.

Definition 3.1 (Transferred Information Specification)

Let v_1 and v_2 denote two participant types, and DB_1, \dots, DB_n databases of participants type v_1 . The *transferred information specification* (TI) from v_1 to v_2 is defined as a set of queries Q_1, \dots, Q_m over the databases DB_1, \dots, DB_n .

From the perspective of privacy, the TI specifications represent all data that is permitted to be transferred from a user of type v_1 to an other user of type v_2 . However, not all data are required to be transferred for each transaction.

Definition 3.2 (Privacy Information Flow (PIF) Graph)

A *PIF graph* is a directed, labeled graph $G = (V, E, \epsilon)$, where

1. V is a nonempty set of *participant-types* (nodes of the graph),
2. E is a set of directed edges, also called information flow channels, which are ordered pairs of nodes, i.e., $E \subseteq V \times V$.
3. ϵ is a mapping that associates with every edge (v_1, v_2) in E a *transferred information specification* TI . Intuitively, TI indicates that nothing but this information is allowed to be transferred from participant-type v_1 to participant-type v_2 .

Consider the PIF graph given in Figure 3. The set of participant-types V is { Patient, Physician, Medical lab, Pharmacy, Health Insurance Company}. Some of the information flow channels are $(Patient, Physician)$, $(Physician, Patient)$, $(Patient, Pharmacy)$, $(Pharmacy, Health Insurance Company)$, etc. The mapping ϵ associates each directed edge with a TI, e.g., TI *Prescriptions* is associated with the directed edge $(Pharmacy, Health Insurance Company)$, i.e., $\epsilon (Pharmacy, Health Insurance Company) = Prescriptions$, etc. Each TI is represented as a set of queries. For example, the set of queries for the TI *Prescriptions* is {Query3}, where Query 3 is:

Query 3:

```

select {Pharmacy  $\Rightarrow$  y, Customers  $\Rightarrow$  x, Drugs  $\Rightarrow$  w, Date  $\Rightarrow$  z}
from {Pharmacies  $\Rightarrow$  {y  $\Rightarrow$  {Customers  $\Rightarrow$  {x  $\Rightarrow$  {w  $\Rightarrow$  z  $\Rightarrow$  {}}}}}}  $\leftarrow$  MED-DB

```

Variable y corresponds to the name of the pharmacy, x to the name of the customer, w to the prescription information, and z to the date of the purchase. This means that only information disclosed by $\{Pharmacy \Rightarrow y, Customers \Rightarrow x, Drugs \Rightarrow w, Date \Rightarrow z\}$ from database *MED-DB* is allowed to be transferred from a participant of type pharmacy to a participant of type health insurance company with the appropriate instantiations to x, y, z, w . Note that a TI may have a set of queries, depending on the actual data structure of the original databases.

Information flow among participants may happen either when a participant requests some information from other participants (e.g., health insurance company submits a query to a physician to obtain information about a patient's treatments) or when a participant supplies some information to other participants without any request (e.g., a pharmacy sending billing information to a health insurance company).

Typically, a PIF graph is designed for each domain with respect to information regarding a participant-type, e.g. patient in our example. In addition to the PIF graph, individual participants may have their special requirements.

Definition 3.3 (Individual Privacy Requirements (IPR))

Given a PIF graph $G = (V, E, \epsilon)$, the *individual privacy requirements* for an individual participant are described as a mapping m that associates with every participant-type in V a *Disallowed Inferred Information* DI^2 . Similarly to the Transferred Information, DI^2 is expressed as a set of queries Q_1, \dots, Q_k and their disclosed data. Personalization of the disallowed information is done by conditioning all queries Q_1, \dots, Q_k on the individual.

The main difference between TI and DI^2 is that TIs are associated with edges of the PIF Graph, while DI^2 s are associated with participant types. This difference allows to detect undesired inferences even if transfers from different data sources contribute to them.

For example, assume that Ms. Smith disallows that her high risk breast cancer diagnosis be released to the health insurance company. This DI^2 is represented as

```

select {Patient  $\Rightarrow$  Jane Smith, Diagnosis  $\Rightarrow$  High risk breast cancer}
from {Medical Office  $\Rightarrow$  {Patients  $\Rightarrow$  {Jane Smith  $\Rightarrow$  {Diagnosis  $\Rightarrow$ 
    Breast cancer high risk}}}}  $\leftarrow$  MED-DB

```

The direct transfer of these data is prevented by the PIF graph. However, indirect information disclosure may still occur from the prescription information sent from the pharmacies to the insurance company. To evaluate whether a DI^2 is violated each data transfer at the recipient's site is evaluated to disclose any of the specified DI^2 .

3.2 Privacy Contract, Policy and Mediator

A *privacy contract* is composed of

1. A PIF graph $G = (V, E, \epsilon)$,
2. Domain knowledge set \mathcal{K} of Horn-clause constraints, and

3. Individual Privacy Requirements, m

The *privacy policy* for the above privacy contract is defined as follows:

1. Every individual *participant* must be of a single participant-type $v \in V$. For example, Jane Smith is of the type *Patient*, Blue Cross Blues Shield is of the type *Health Insurance Company*.
2. The Transfer Control unit (see Figure 4) collects for every participant v the data that have been received from any of the other users or can be entailed from the received data and the domain knowledge.
3. A participant-type v_1 is allowed to send a data item *data* to a participant v_2 if and only if the following conditions hold:
 - (a) *data* permitted by the PIF graph, that is data disclosed from the answers of the permitted queries of the TI $\epsilon(v_1, v_2)$, and
 - (b) No disallowed data item in $m(v_2)$ is logically entailed from \mathcal{K} , the prior knowledge of v_2 (i.e., data previously received by v_2) and data item *data*.

The Privacy Mediator (PriMe) architecture, given in Figure 4, can be used to enforce the privacy policy on all data transfers. The *transfer control module* of PriMe is responsible for enforcing the third requirement of the privacy policy. This requires the use of an inference

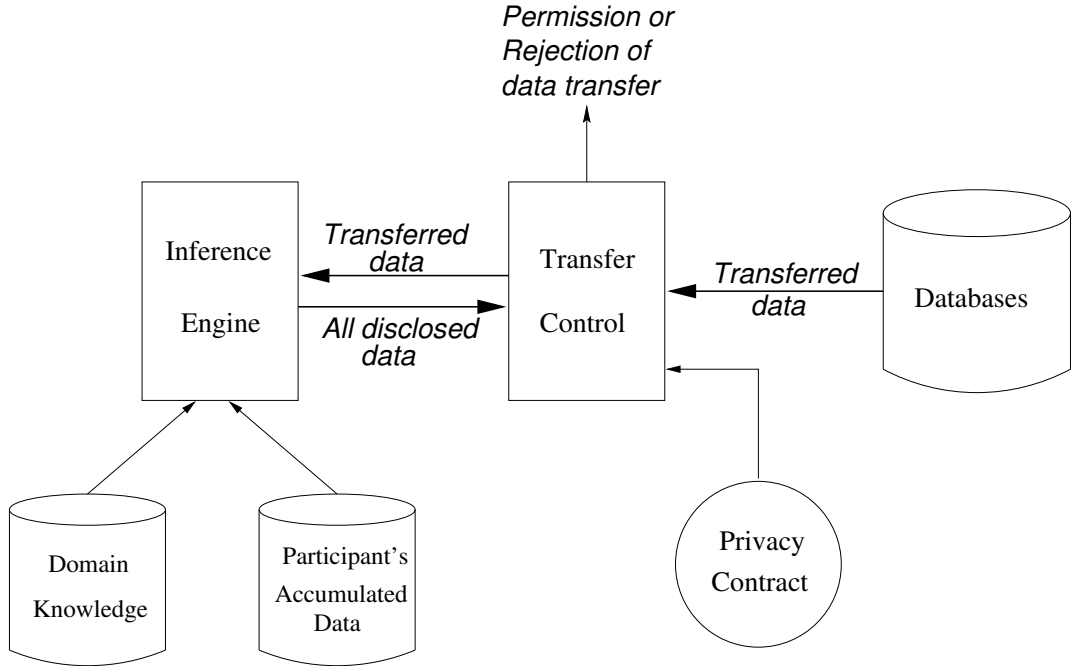


Figure 4: Privacy Mediator (PriMe) Architecture

engine, which uses the domain knowledge and the accumulated data to generate all logically entailed information.

The current system architecture is based on a centralized, trusted module (Transfer Control) to evaluate information transfers. Therefore, all traffic must pass through this module before reaching the destination user. Transfer control keeps the history file for each user (note that a user may be a single person, or a group of people working together for an organization). The history file contains all directly transmitted data that were received by the user and all logically entailed data. The history file can be periodically purged of outdated data, depending on the domain requirements.

To represent the functionality of PriMe, consider the scenario when Jane Smith's health insurance company requests information about her prescriptions from a pharmacy. The disclosed query data for this request is permitted by the PIF graph.

$$\{Pharmacies \Rightarrow \{ \delta_1 \Rightarrow \{ Customers \Rightarrow \{ Jane\ Smith \Rightarrow \{ Raloxifene \Rightarrow 9/10/99, Hercepin \Rightarrow 9/10/99 \} \} \} \} \}$$

Next, the inference engine applies the Horn-clause constraint

$$\{Pharmacies \Rightarrow \{y_1 \Rightarrow \{Customers \Rightarrow \{x \Rightarrow Raloxifene\}\}\}\} \wedge$$

$$\{Pharmacies \Rightarrow \{y_2 \Rightarrow \{Customers \Rightarrow \{x \Rightarrow Hercepin\}\}\}\} \rightarrow$$

$$\{Medical\ Office \Rightarrow \{Patients \Rightarrow \{x \Rightarrow \{Diagnosis \Rightarrow Breast\ cancer\ high\ risk\}\}\}\}$$

on the disclosed data, and derives the TE

$$\{Medical\ Office \Rightarrow \{Patients \Rightarrow \{Jane\ Smith \Rightarrow \{Diagnosis \Rightarrow Breast\ cancer\ high\ risk\}\}\}\}$$

which is indeed in the disallowed information $m(HealthInsuranceCompany)$; therefore the transfer control mechanism of PriMe refuses the requested data transfer.

Note, that refusal of a data request if it satisfies the TI specifications, can be suspicious. In the above example, the insurance company may infer, that Ms. Smith has some serious medical condition that she is now permitting to be released. Solutions, different from rejecting a data transfer, may be better for some applications. In particular, releasing partial answer in a manner that do not disclose the private data, seems promising.

4 Disclosure Inference

In this section we develop an algorithm to generate a logically entailed tree. First, we define the concept of minimality and complete, logically entailed tree of a ground-TE and a set of Horn-clause constraints. Intuitively, the minimality requirement (soundness) ensures that only disclosed data is present in the TE. The completeness requirement ensures that all disclosed data is given in the TE.

Definition 4.1 (Disclosure Cover of T under \mathcal{K} ($DC_{\mathcal{K}}(T)$))

Let \mathcal{K} be a set of Horn-clause constraints and T be a TE with no variables (but possibly with null-values). A TE \mathcal{S} (with no variables) is called a *disclosure cover* of T under \mathcal{K} , denoted as $DC_{\mathcal{K}}(T)$, if \mathcal{S} satisfies the following properties:

1. \mathcal{S} is *sound*; i.e., $T \models_{\mathcal{K}} \mathcal{S}$
2. \mathcal{S} is *complete*; i.e., for every TE T' with no variables, if $T \models_{\mathcal{K}} T'$ then \mathcal{S} satisfies T' , i.e., there exists a symbol-mapping λ that maps the labels of T' to the labels of T such that $\Lambda(T')$ is a subtree of \mathcal{S} , where Λ is the tree-mapping originating from λ .

Note that when generating disclosure cover the TEs contain constants and null-values but not variables. Null-values represent that the data values are unknown to the user but some equalities among the unknown values are entailed either by the domain knowledge or by the query conditions. To preserve this information we require that the tree-mappings do not map constants to null-values. Null-values can be mapped to a constant or a different null-value.

The algorithm (Algorithm: Disclosure Cover) to generate the disclosure cover of a TE T under Horn-clause constraints \mathcal{K} is given in Figure 5. As the input, the Disclosure Cover Algorithm receives a TE T (tree-union of the result of the current query and previously disclosed data) and a set of Horn-clause constraints \mathcal{K} (domain knowledge). Similarly to the Chase process [66], the Disclosure Cover Algorithm applies \mathcal{K} on T until no more changes occur. At that point, the resulting TE, denoted as \mathcal{S} , satisfies \mathcal{K} and no more information can be derived from T and \mathcal{K} .

Theorem 4.1 (Disclosure Cover)

Algorithm Disclosure Cover effectively computes a disclosure cover of T under \mathcal{K} , denoted by $DC_{\mathcal{K}}(T)$; i.e., the algorithm terminates and the computed \mathcal{S} is (1) *sound* and (2) *complete*.

Proof: Applications of the constraints on \mathcal{S} do not create new symbols; and the height of \mathcal{S} is bounded. Therefore, there are only a finite number of paths that can be created, thus the algorithm must *terminate*.

For *soundness*, let DB be a ground tree that satisfies \mathcal{K} and also satisfies T . Then there must exist a tree-mapping Λ from T to DB . The proof of soundness is based on the inductive statement that after each application of a constraint on \mathcal{S} in the algorithm, the resulting tree \mathcal{S} is still satisfied by DB . This clearly holds for the initial state, when $\mathcal{S} = T$. Assume that the application of a dependency d equates two values $\lambda(a)$ and $\lambda(b)$ in \mathcal{S} . But then, there must exist a tree-mapping Λ , originating from λ such that $\Lambda(B_1), \dots, \Lambda(B_n)$ are subtrees of DB . Since DB satisfies \mathcal{K} , $\lambda(a)$ must be equal to $\lambda(b)$ in DB , therefore Λ is still a tree-mapping from \mathcal{S} to DB . If inconsistency occurs in step 2.a of the algorithm, then \mathcal{S} is vacuously disclosed. Now, assume that d is a tree-generating dependency that generates a subtree $\Lambda(H)$ in \mathcal{S} . There must exist a tree-mapping Λ , originating from λ such that $\Lambda(B_1), \dots, \Lambda(B_n)$ are subtrees of DB and $\Lambda(H)$ must be a subtree of DB since DB satisfies d . But then, Λ is still a tree-mapping from \mathcal{S} to DB . Similar argument can be constructed for each step of the dependency application. This concludes the proof of soundness.

Algorithm:	Disclosure Cover
INPUT	1. Ground TE T 2. Set \mathcal{K} of Horn-clause constraints
OUTPUT	Disclosure cover $DC_{\mathcal{K}}(T)$ that is the set \mathcal{S} generated by the algorithm
METHOD	<p>1. Initially, set \mathcal{S} to T.</p> <p>2. Apply constraints in \mathcal{K} on \mathcal{S}, until no more changes occur or inconsistency occurred, as follows: If there exists a constraint</p> $d = B_1 \wedge \dots \wedge B_n \rightarrow H$ <p>and a tree-mapping Λ from $\{B_1, \dots, B_n\}$ to \mathcal{S} (when null-values of \mathcal{S} are treated as constants), apply (d, Λ) on \mathcal{S} as follows:</p> <p>(a) If d is an equality-generating constraint, i.e., of the form $B_1, \dots, B_n \rightarrow a = b$ then</p> <ul style="list-style-type: none"> • If both $\lambda(a)$ and $\lambda(b)$ are null-values in \mathcal{S} then replace all occurrences of $\lambda(a)$ with $\lambda(b)$. • Otherwise, if one of $\lambda(a), \lambda(b)$, say $\lambda(a)$, is not a null-value then replace all occurrences of $\lambda(b)$ in \mathcal{S} with $\lambda(a)$. • Otherwise, i.e., if both $\lambda(a), \lambda(b)$ are constants, <i>do nothing</i>. In this case, if $\lambda(a) \neq \lambda(b)$, we say that <i>inconsistency</i> occurred. <p>(b) If d is an tree-generating constraint, i.e., of the form $B_1, \dots, B_n \rightarrow H$, where H is a TE, then</p> <ul style="list-style-type: none"> • Set \mathcal{S} to $\mathcal{S} \sqcup \Lambda(H)$ <p>3. If inconsistency occurred during Step 2 (i.e., the λ maps a and b to two different constants. Then, the original database didn't satisfy the functional dependency, then set \mathcal{S} to a special value ALL. ALL indicates that any TE (with no variables) is disclosed.</p> <p>4. Return \mathcal{S} as output.</p>

Figure 5: Disclosure Cover

For *completeness*, if inconsistency occurred during step 2 of the algorithm, then completeness trivially follows.

If no inconsistency occurred then assume, by contradiction, that for some $T', T \models_{\mathcal{K}} T'$ but \mathcal{S} does not satisfy T' (i.e., there does not exist a tree-mapping Λ such that $\Lambda(T')$ is a subtree of \mathcal{S}). To show that such mapping must exist, we construct a ground tree DB from \mathcal{S} at the end of the dependency application (step 2) of the algorithm by replacing each null-value with a unique constant. Clearly, DB satisfies \mathcal{K} and also satisfies T (initial condition) but, based on our initial hypothesis, DB does not satisfies T' . This is a contradiction to

the fact that $T \models_{\mathcal{K}} T'$. This completes the proof of completeness. □

An important corollary of Theorem 4.1 is that the inference disclosure is decidable. The proof of this corollary is based on the correctness of the developed inference algorithm.

Corollary 4.1 *The following problem is decidable:*

Given a set \mathcal{K} of Horn-clause constraints and TEs T and T' , determine whether $T \models_{\mathcal{K}} T'$.

4.1 Implementation Considerations

In our model, every data transfer must be evaluated by PriMe that checks for privacy violations. A straightforward solution is to use a centralized model, where PriMe acts like a security proxy server between the participants. In this case, PriMe knows all the security requirements (i.e., privacy contract) and the domain knowledge. In addition, PriMe hosts the participants' history files and runs the inference engine. The advantage of the centralized location is that a single, high-performance hardware device can be used to perform the transfer analysis. Since all historical data and the domain knowledge is stored locally, fast data access can be achieved. The speed of data processing is determined by the complexity of the inference algorithm. The algorithm manipulates on a generic tree, and its complexity is bounded by the *chase* process of applying the domain constraints. Each iteration (application of a constraint) checks for possible tree-mappings from the body of the constraint to the history-database tree, and thus is bounded by the number and complexity of the constraint's elements as well as the labels of the generic tree. The complexity of the algorithm is polynomial in the size of the input. Clearly, the more data and constraints PriMe handles the slower its performance. This may create a bottleneck in centralized processing. In addition, a central processing unit represent a single point of failure.

A more scalable solution could be to implement a distributed PriMe architecture, where trusted processing units guard the incoming traffic to each participant. However, this model requires that the participants must support the usage of the PriMe model, but, they are not in control of PriMe. Since we consider a controlled environment, where the participants may try to acquire information only in legally accepted ways (i.e., an insurance company will not try to hack the database of a health care provider) this is not an unrealistic assumption. Each local PriMe unit keeps track of the incoming traffic and analyzes whether a given transfer discloses any forbidden information to the participant. If a privacy violation is detected, the originator of the transfer is contacted and the transfer is rejected. The complexity of each local PriMe is the same as for centralized, but the input data and the domain knowledge are usually much smaller than for the centralized model. Also, there is no single point of failure is present.

5 Related Work

One of the earliest considerations of privacy violations in databases focused on statistical inferences, where statistics about groups of individuals are provided but no information about any particular individual is released. However, it is still possible that confidential information is disclosed by correlating different statistics. Inference control in statistical

databases have been considered extensively (e.g., [18, 28, 30, 31, 32, 40]). A number of inference control mechanisms, such as query size and query overlap control, data swapping, and multidimensional transformation, were developed and their limitations established. The main problem is that simple inference control mechanisms are easily subverted and; therefore, insufficient. On the other hand, mechanisms that provide security with high assurance, are not-practical in general purpose databases and thus are limited to certain applications.

Since the 1980s research on data inference and violations of data confidentiality shifted to multilevel secure (MLS) relational databases that contain data classified at different security levels (see [38] for a survey). Most of the inference channels in MLS relational databases are raised by combining *meta-data* (e.g., database constraints) with data to obtain information that has higher security classification than the original data. Techniques to detect privacy violations via inference channels in MLS relational databases fall into two general categories. The first category includes techniques that detect inference channels during database design time [13, 26, 27, 34, 35, 42, 45, 47, 57, 60, 63]. Inference channels when detected, are typically removed by modifying the database design and/or by increasing the classification levels of some of the data. These techniques often result in over-classification of data and; therefore, reduce the availability of data. Techniques in the second category seek to eliminate inference channel violations during query time [12, 29, 43, 56, 58, 64, 69]. If an inference channel is detected, the query is either refused or modified to avoid security violations. This technique allows higher data availability while protecting data confidentiality.

During the 1990s, with the further development of electronic databases and computer networks new privacy problems surfaced [1, 4, 10, 11, 22, 33, 46, 48, 49, 52, 53, 59, 65, 69, 70, 71]. Simultaneously, works to provide control accesses to eXtensible Markup Language (XML) have surfaced [8, 25, 41]. These models focus on defining access control models on XML documents, thus preventing privacy violations via direct data accesses. While these mechanisms are necessary, none of the above works provide technical solutions to enforce privacy requirements in the presence of possible inferences, or give assurance on the level of protection.

The works closest to ours are [12, 33, 69]. Our previous work [12] proposes an integrated security mechanism that guarantees data confidentiality in MLS relational databases against illegal inferences. We considered inferences that occur when database constraints are combined with non-sensitive data to obtain sensitive information. We developed the concepts of data-dependent and data-independent disclosure inferences. In data-dependent mode, inference was based on actual data items, while in data-independent mode, inference was based on queries only. A sound and complete inference algorithm was developed for data-dependent mode, and a complete inference algorithm for data-independent mode. The main limitation of this model is that it focuses only on multilevel secure relational databases and thus it is unable to accommodate integrated heterogeneous databases and requires database management support for multilevel security.

Dreyer and Olivier [33] propose an information can-flow graph to represent positive (permitted) and negative (not permitted) information flow among the entities. They assume that users are not trustworthy and information flow between users should be taken into account, too. They distinguish between the static (potential) and dynamic (actual) aspects of the information flow. They present algorithms to extend the static can-flow graph to incorporate the dynamic aspect without introducing unauthorized information flow. How-

ever, their work lacks the formal definition of the suggested model, as well as proofs of the correctness of the presented algorithms are missing. Finally, they do not consider inferences as a source indirect data flows.

A security mediator to ensure data confidentiality in clinical databases is presented by Wiederhold et. al [69]. The purpose of the proposed mechanism is to support information sharing while preserving data confidentiality. The security mediator, developed under the TIHI project, screens the queries and answers to the queries to detect privacy violations. The security requirements are encoded as rules and enforced by the mediator. The paper provides a general framework and the description of a prototype for enforcing domain specific privacy requirements. However, the authors do not formally define the type of rules used by the security mechanism and the application of these rules, thus no guarantee of the privacy protection can be established. Further, they do not consider possible inferences as sources for privacy violations and their model is suitable only for relational databases.

6 Conclusions

In this paper we presented a formal model, called *Privacy Information Flow (PIF) Model* to represent allowable information transfer among participants (e.g., people and organizations) of a domain, and to specify privacy requirements at both organizational and individual level. We used a *Privacy Information Flow (PIF) Graph* to express general privacy requirements for a specific application domain (e.g., the medical domain) and participant type (e.g., patients). *Individual Privacy Requirements* are used to extend the PIF graph to accommodate individual privacy preferences.

We developed a sound and complete inference algorithm, used by a *Privacy Mediator (PriMe)*, to guarantee that the privacy policy is correctly enforced even in the presence of data inferences that occur when non-private data is combined with domain knowledge to disclose private information. We adopted a tree-like, semistructured data model and used Horn-clause constraints to represent domain knowledge. Although, we demonstrated the functionality of our model through medical examples, it is not limited to any particular domain.

We conclude by listing some suggestions for future research.

Privacy Mediator Architecture: In our current model all data transfer is performed through a single transfer control module that enforces the privacy requirements. This might create a bottleneck. A future improvement of our model could be to assign individual transfer control mechanisms to each participant and define methods to correctly distribute the domain knowledge and privacy restrictions among the participants.

Data model: In this paper, we adopted a tree-like data structure. This can be extended to accommodate graph structures that possibly have cycles. Another limitation of the current work is that the structure of the tree is preserved through explicit label values (constants); thus in the presence of label-variables and null-values, we cannot tell whether two outgoing edges from the same node are different or correspond to the same edge.

Queries: Variables used in the TE in the **from** clause of a select-project query may function as wild card symbols, i.e., they correspond to any edge label of the database. However, they are limited to match a single edge only and not an arbitrary path. While it is possible, with the knowledge of the data structure, to use a set of select-project queries to mimic a query

with variables corresponding to an arbitrary path, it is cumbersome. A future extension of our model would be to incorporate path-variables that match any path of arbitrary length. **Inference:** Currently the disclosure inference algorithm generates all disclosed data using actual data items (trees). Since it might be computationally expensive, in some applications it might be desirable to establish possible inferences using only the past and present queries and the database constraints.

Finally, in real life applications it might happen that private information cannot be fully disclosed from the query answers and the domain knowledge but can be guessed with high probability. Although our logic based framework allows the incorporation of probabilities in the knowledge rules, additional mechanisms are needed to protect private information from attacks based on probabilistic methods.

References

- [1] Genetic information and the workplace. Technical report, Department of Labor, Department of Health and Human Services, Equal Employment Opportunity Commission Department of Justice, 1998.
- [2] Serge Abiteboul. Querying semi-structured data. In *ICDT*, pages 1–18, 1997.
- [3] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proc. of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, TX*, pages 439–450, 2000.
- [4] J. G. Anderson. Clearing the way for physicians’ use of clinical information systems. *Communications of ACM*, 40(8):83–90, August 1997.
- [5] D.E. Bell and L.J. LaPadula. Secure computer systems: Mathematical foundation and model. Technical report, Mitre Corp. Report No. M74-244, Bedford, Mass., 1975.
- [6] S. Benferhat, F. Autrel, and F. Cuppens. A possibilistic logic encoding of access control. In *FLAIRS Conf.*, pages 481–485, 2003.
- [7] S. Benferhat, R. E. Baida, and F. Cuppens. A stratification-based approach for handling conflicts in access control. In *SACMAT*, pages 189–195, 2003.
- [8] Elisa Bertino, Silvana Castano, Elena Ferrari, and Marco Mesiti. Controlled access and dissemination of XML documents. In *Workshop on Web Information and Data Management*, pages 22–27, 1999.
- [9] C. Bettini, S. Jajodia, S. Wang, and D. Wijesekera. Flexible support for multiple access control policies. *ACM Trans. Database Syst.*, 26(2):214–260, 2001.
- [10] J. Biskup and H. H. Bruggemann. The personal model of data - towards a privacy oriented information system (extended abstract). In *Proc. of the Fifth International Conference of Data Engineering, February 6–10, 1989, Loas Angeles, California, USA*, pages 348–355. IEEE Computer Society, 1989.

- [11] B. Braithwaite. National health information privacy bill generates heat at SCAMC. *Journal of American Informatic Association*, 3(1):95–96, 1996.
- [12] A. Brodsky, C. Farkas, and S. Jajodia. Secure databases: Constraints, inference channels, and monitoring disclosure. *IEEE Trans. Knowledge and Data Eng.*, 12(6):900–919, 2000.
- [13] L.J. Buczkowski. Database inference controller. In D.L. Spooner and C. Landwehr, editors, *Database Security III: Status and Prospects*, pages 311–322. North-Holland, Amsterdam, 1990.
- [14] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 505–516, 1996.
- [15] Peter Buneman, Wenfei Fan, Jme Simeon, and Scott Weinstein. Constraints for semi-structured data and XML. *SIGMOD Record*, 30(1):47–54, 2001.
- [16] Peter Buneman, Wenfei Fan, and Scott Weinstein. Path constraints on semistructured and structured data. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 129–138. ACM Press, 1998.
- [17] Peter Buneman, Wenfei Fan, and Scott Weinstein. Path constraints in semistructured databases. *Journal of Computer and System Sciences*, 61(2):146–193, 2000.
- [18] F.Y. Chin. Security in statistical databases for queries with small counts. *ACM Trans. on Database Systems*, 3(1):92–104, 1978.
- [19] L. Cholvy, F. Cuppens, and C. Saurel. Towards a logical formalization of responsibility. In *ICAIL*, pages 233–242, 1997.
- [20] C. Clifton, M. Kantarcioglu, and J. Vaidya. Defining privacy for data mining. In *Proc. of the National Science Foundation Workshop on Next Generation Data Mining, Baltimore, MD*, pages 126–133, 2002.
- [21] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving distributed data mining. *SIGKDD Explorations*, 4(2):28–34, 2002.
- [22] National Research Council. For the record: Protecting electronic health information. Technical report, National Academy of Sciences, 1997.
- [23] F. Cuppens and R. Demolomba. A deontic logic for reasoning about confidentiality. In *Proc. 3rd International Workshop on Deontic Logic in Computer Science*, 1996.
- [24] F. Cuppens and R. Demolomba. A modal logical framework for security policies. In *ISMIS*, pages 579–589, 1997.
- [25] Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, and Pierangela Samarati. Design and implementation of an access control processor for XML documents. *WWW9 / Computer Networks*, 33(1-6):59–75, 2000.

- [26] S. Dawson, S. De Capitani di Vimercati, and P. Samarati. Minimal data upgrating to prevent inference and association attacks. In *Proc. of the 18th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 114–125, 1999.
- [27] S. Dawson, S. De Capitani di Vimercati, and P. Samarati. Specification and enforcement of classification and inference constraints. In *Proc. IEEE Symp. on Security and Privacy*, 1999.
- [28] D.E. Denning. *Cryptography and Data Security*. Addison-Wesley, Mass., 1982.
- [29] D.E. Denning. Commutative filters for reducing inference threats in multilevel database systems. In *Proc. IEEE Symp. on Security and Privacy*, pages 134–146, 1985.
- [30] D.E. Denning, P.J. Denning, and M.D. Schwartz. The tracker. *ACM Trans. on Database Systems*, 4(1):75–95, 1979.
- [31] D.E. Denning and J. Schlorer. A fast procedure for finding a tracker in a statistical database. *ACM Trans. on Database Systems*, 5(1):88–102, 1980.
- [32] D. Dobkin, A.K. Jones, and R.J. Lipton. Secure databases: Protection against user influence. *ACM Trans. on Database Systems*, 4(1):97–106, March 1979.
- [33] L. C. J. Dreyer and M. S. Olivier. Dynamic aspect of the infopriv model. In *Proc. 9th Database and Expert Systems Applications DEXA 98*, pages 340–345. IEEE Computer Society, Los Alamitos, 1998.
- [34] J.A. Goguen and J. Meseguer. Unwinding and inference control. In *Proc. IEEE Symp. on Security and Privacy*, pages 75–86, 1984.
- [35] T.H. Hinke. Inference aggregation detection in database management systems. In *Proc. IEEE Symp. on Security and Privacy*, pages 96–106, 1988.
- [36] T.S. Hsu, C. J. Liau, D. W. Wang, and J. K. Chen. Quantifying privacy leakage through answering database queries. In *Proc. 5th Information Security Conference (ISC)*, pages 162–175. Springer-Verlag LNCS 2433, 2002.
- [37] T.S. Hsu, C.J. Liau, and D.W. Wang. A logical model for privacy protection. In *Proc. 4th Information Security Conference (ISC)*, pages 110–124. Springer-Verlag LNCS 2200, 2001.
- [38] S. Jajodia and C. Meadows. Inference problems in multilevel secure database management systems. In M.D. Abrams, S. Jajodia, and H. Podell, editors, *Information Security: An integrated collection of essays*, pages 570–584. IEEE Computer Society Press, Los Alamitos, Calif., 1995.
- [39] Sushil Jajodia, Pierangela Samarati, Maria Luisa Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *ACM Trans. Database Syst.*, 26(2):214–260, 2001.

- [40] J.B. Kam and J. Ullman. A model of statistical databases and their security. *ACM Trans. on Database Systems*, 2(1):1–10, March 1977.
- [41] M. Kudo and S. Hada. XML document security based on provisional authorization. In *Proc. of the 7th ACM Conference on Computer and Communication Security*, November 2000.
- [42] D.G. Marks. Inference in MLS database systems. *IEEE Trans. Knowledge and Data Eng.*, 8(1):46–55, February 1996.
- [43] S. Mazumdar, D. Stemple, and T. Sheard. Resolving the tension between integrity and security using a theorem prover. In *Proc. ACM Int'l Conf. Management of Data*, pages 233–242, 1988.
- [44] Jason McHugh, Serge Abiteboul, Roy Goldman, Dallan Quass, and Jennifer Widom. Lore: A database management system for semistructured data. *SIGMOD Record*, 26(3):54–66, 1997.
- [45] C. Meadows. Extending the Brewer-Nash model to a multilevel context. In *Proc. IEEE Symp. on Security and Privacy*, pages 95–102, 1990.
- [46] B. N. Meeks. Privacy lost, anytime, anywhere. In *Communications of ACM*, volume 40/8, pages 11–13, 1997.
- [47] M. Morgenstern. Controlling logical inference in multilevel database systems. In *Proc. IEEE Symp. on Security and Privacy*, pages 245–255, 1988.
- [48] United States General Accounting Office. Medical records privacy, access needed for health research, but oversight of privacy protections is limited. Technical report, United States General Accounting Office, Report to Congressional Requesters GAO/HEHS-99-55, 1999.
- [49] D. E. O’Leary. Some privacy issues in knowledge discovery: Oecd personal privacy guidelines. *IEEE Expert/Intelligent Systems and Their Applications*, 10(2), April 1995.
- [50] S.R.M. Oliveira and O.R. Zaiane. Privacy preserving clustering by data transformation. In *Proc. of the 18th Brazilian Symposium on Databases, Brazil*, pages 304–318, 2003.
- [51] S.R.M. Oliveira, O.R. Zaiane, and Y. Saygin. Secure association rule mining. In *Proc. of the 8th pacific-Asian Conference on Knowledge Discovery and Data Mining, Sydney, Australia*, pages 74–85, 2004.
- [52] T. C. Rindfleisch. Privacy, information technology, and health care. *Communications of ACM*, 40(8):93–100, August 1997.
- [53] A. D. Rubin, D. Geer, and M. J. Ranum. *WEB Security Sourcebook*. John Wiley and Sons, Inc., 1997.
- [54] Ravi Sandhu, David Ferraiolo, and Richard Kuhn. The NISI model for role-based access control: Towards a unified standard. In *ACM RBAC 2000*, pages 47–64, 2000.

- [55] Ravi Sandhu and Pierangela Samarati. Access control: Principles and practices. *IEEE Communications*, 29(2):38–47, 1996.
- [56] G.L. Sicherman, W. de Jonge, and R.P. van de Riet. Answering queries without revealing secret. *ACM Trans. on Database Systems*, 8(1):41–59, 1983.
- [57] G.W. Smith. Modeling security-relevant data semantics. In *Proc. IEEE Symp. Research in Security and Privacy*, pages 384–391, 1990.
- [58] P.D. Stachour and B. Thuraisingham. Design of LDV: A multilevel secure relational database management system. *IEEE Trans. Knowledge and Data Eng.*, 2(2):190–209, June 1990.
- [59] L. D. Stein. *Web Security - A Step-by-Step Reference Guide*. Addison-Wesley Longman, inc., 1998.
- [60] T. Su and G. Ozsoyoglu. Inference in MLS database systems. *IEEE Trans. Knowledge and Data Eng.*, 3(4):474–485, December 1991.
- [61] H. Subramaniam and Z. Yang. Report on DIMACS working group on privacy/confidentiality of health data. Technical report, DIMACS Center, CoRE Building, Rutgers University, Piscataway, NJ, 2003.
- [62] Dan Suci. Semistructured data and XML. In *Handbook of massive data sets*, pages 743–788, 2002.
- [63] T.H.Hinke, Harry S. Delugach, and Asha Chandrasekhar. A fast algorithm for detecting second paths in database inference analysis. *Jour. of Computer Security*, 3(2,3):147–168, 1995.
- [64] B.M. Thuraisingham. Security checking in relational database management systems augmented with inference engines. *Computers and Security*, 6:479–492, 1987.
- [65] T. C. Ting. Privacy and confidentiality in healthcare delivery information system. In *Proceedings of the 12th IEEE Symposium on Computer-Based Medical Systems*, page 2. IEEE Computer Society, 1999.
- [66] J.D. Ullman. *Principles of Database and Knowledge-base Systems, Volumes 1,2*. Computer Science Press, Rockville, MD, 1988.
- [67] D. W. Wang, C. J. Liao T.S. Hsu, , and J. K. Chen. On the damage and compensation of privacy leakage. In *18th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*. Kluwer Inc., 2004.
- [68] A. F. Westin. *Privacy and Freedom*. Atheneum, New York NY, 1967.
- [69] G. Wiederhold, M. Bilello, and C. Donahue. Web implementation of a security mediator for medical databases. In T. Y. Lin and S. Qian, editors, *Database Security XI Status and Prospects*, pages 60–67. Chapman and Hall, 1998.