Secure Mediated Databases

K.S. Candan
University of Maryland
candan@cs.umd.edu

Sushil Jajodia*
George Mason University
jajodia@gmu.edu

V.S. Subrahmanian†
University of Maryland
vs@cs.umd.edu

Abstract

With the evolution of the information superhighway, there is now an immense amount of information available in a wide variety of databases. Furthermore, users often have the ability to access legacy software packages developed by external sources. However, sometimes both the information provided by a data source, as well as one or more of the functions available through a software package may be sensitive – in such cases, organizations require that access by users be controlled. HERMES ("<u>HE</u>terogeneous <u>R</u>easoning and <u>ME</u>diator <u>S</u>ystem") is a platform that has been developed at the University of Maryland within which mediators may be designed and implemented. HERMES has already been used for a number of applications [14, 5]. In this paper, we will provide a formal model of security in mediated systems. We will then develop techniques that are sound and complete and respect security constraints of packages/databases participating in the mediated system. The security constraints described in this paper have been implemented – we will describe the existing implementation.

1 Introduction

Over the last few years, there has been considerable work on security in databases [10]. Most of this work has been limited to the realm of relational databases [3, 17, 23, 25], though of late, some work has been done on extending these security paradigms to object-oriented databases [11, 13, 16, 24, 31], deductive databases [7], and other paradigms. Despite the differences in the underlying data paradigm, all these frameworks share a single trait that we (cynically?) term the principle of paranoia.

> THE PRINCIPLE OF PARANOIA. The DBMS must take all steps necessary in order to insure that the user u cannot to infer any item in a pre-designated set $\mathcal{S}(u)$ of items that are to be kept secret from the user.

However, with the evolution of the information superhighway, there is now an immense amount of information available in a very wide variety of databases. Wiederhold [29, 30] has proposed the

*Partially supported by a grant from ARPA, administered by the Office of Naval Research under grant number N0014-92-J-4038, by National Science Foundation under grant IRI-9303416, and by National Security Agency under contract MDA904-94-C-6118.

†Partially supported by the Army Research Office under grant DAAL-03-92-G-0225, by the Air Force Office of Scientific Research under grant F49620-93-1-0065, by ARPA/Rome Labs contract Nr. F30602-93-C-0241 (Order Nr. A716), and by an NSF Young Investigator award IRI-93-57756.

concept of a *mediator* – intuitively, a mediator is a program that integrates multiple databases. For now, we will not worry about the structure and content of mediators – that will be described in detail in Section 2. Instead we will assume, for this section only, that we have a mediator program M that integrates some software packages $P_1, \ldots, P_k$. An important aspect of mediated systems is that each of the individual packages $P_1, \ldots, P_k$ may enforce its own unique *local* security policy. Some may represent completely "open-source" software/data, while others may place certain restrictions on the use of certain facilities and/or data residing within it. In contrast to the principle of paranoia commonly enforced in ordinary databases, mediated systems must attempt to be *maximally cooperative* to the user, yet at the same time, they must respect the security constraints of the individual databases/packages participating in the mediated system. Thus, for instance, two packages P_1 and P_2 may both be able to satisfy a user's request – however, package P_1 uses secure data, while package P_2 uses open-source data. In this case, the mediated system may reasonably use package P_2 to respond to the user's query, even though package P_1 feels this data should be kept hidden from the user. Notice that in this case, the user could directly query P_2 and get the data without using the mediator at all, so the mediator might as well do it for him, unless a *global* security condition maintained by the mediator prevents this. Figure 1 shows a schematic diagram of security schemes maintained by within a mediated system. Thus, in the case of mediated systems, we may wish to implement a policy of *cautious cooperation.*

THE PRINCIPLE OF CAUTIOUS COOPERATION. If a user's query can be answered using open-source information, then the mediator will answer the query unless doing so will *directly* violate the system's *global* security constraints. However, the system will always respect the rights of individual packages participating in the mediated system, and ensure that no single package violates its own local security policy.

The principle of cautious cooperation ensures that a given query will cause no direct violation of global integrity constraints, but may leave the path open for future violations or for inferential violations. A slightly more conservative policy, that we term the principle of *conservative cautious cooperation*, will answer a query posed by the user iff the answer to that query will not yield an "inference path" (sequence of open-source queries coupled with logical reasoning) that the user may use to violate security (unless such a path existed prior to the query being issued by the user).

THE PRINCIPLE OF CONSERVATIVE CAUTIOUS COOPERATION. If a user's query can be answered using open-source information, then the mediator will answer the query unless doing so will cause there to be a sequence of queries (that only reflect open-source accesses) such that if the user asks this sequence of queries, then he will be able to violate the system's *global* security constraints. However, the system will always respect the rights of individual packages participating in the mediated system, and ensure that no single package violates its own local security policy.

Interest in enforcing security in heterogeneous systems has been recently addressed in two interesting papers by Idris et. al. [15] and Jonscher and Dittrich [18] who have studied issues of security in federated databases. [18] describes a heterogeneous system called CHASSIS that provides mechanisms for decentralized authorization in federated systems. It deals primarily with relational and object oriented databases. Idris et. al. [15] develop techniques whereby

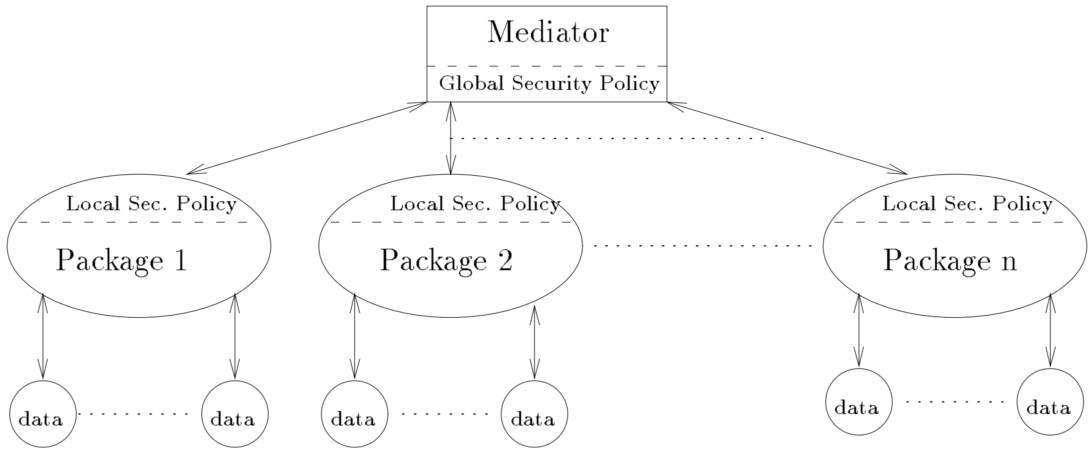

Figure 1: Schematic Diagram of Mediated Systems

dynamic changes to the security level of a user can be handled elegantly. We do not address this problem. In contrast, our work sets out a formal model of secure mediated systems, establishes formal notions of soundness and completeness w.r.t. a secure mediated system, and develops techniques that are provably sound, complete, and secure. The formal model is backed up by an implementation that has already been used for integrated terrain reasoning for military and rescue operations by the US Army [5], for robotic reasoning and planning by NIST [14], and more recently, for law enforcement operations.

The organization (and contributions) of this paper is as follows: in Section 2, we will briefly explain the HERMES system (short for "HEterogeneous Reasoning and MEdiator System") that has been developed at the University of Maryland. HERMES provides a platform within which mediators may be implemented. Full details may be found in [28]. In Section 3, we will define a formal model of security in mediated systems and formally describe the principles of cautious and conservative cautious cooperation. In section 5, we will describe the implementation of these security paradigms that has been carried out within HERMES. We will give an example of how HERMES has been used to enforce these security policies in an example that involves accessing face-recognition packages (and face databases), accessing two INGRES databases and accessing flat files.

2 Background: The HERMES System

In this section, we will briefly describe the basic theory behind mediated systems proposed in [21, 27, 1]. Illustration is provided via the HERMES implementation.

A *domain*, $\mathcal{D}$, is an abstraction of databases and software packages and consists of three components: (1) a set Σ whose elements may be thought of as the data-objects that are being manipulated by the package in question, (2) a set $\mathcal{F}$ of functions on Σ – these functions take objects in Σ as input, and return, as output, objects from their range (which needs to be specified). The functions in $\mathcal{F}$ may be thought of as the predefined functions that have been implemented in the software package being considered, (3) a set of relations on the data-objects in Σ – intuitively, these relations may be thought of as the predefined relations in the domain, $\mathcal{D}$.

A constraint Ξ over D is a first order formula where the symbols are interpreted over D. Ξ is either true or false in D, in which case we say that Ξ is solvable, or respectively unsolvable in D, where the reference to D will be eliminated if it is clear from context. The key idea behind a mediated system is that constraints provide the link to external sources, whether they be databases, object bases, or other knowledge sources. This idea is developed extensively in [21, 27, 1] and we do not elaborate on them here.

For example in HERMES, a *domain call* is a syntactic expression of the form

$$\texttt{domainname} : \langle\texttt{domainfunction}\rangle(\langle\texttt{arg1}, \ldots, \texttt{argn}\rangle)$$

where `domainfunction` is the name of the function, and $\langle$`arg1`, …, `argn`$\rangle$ are the arguments it takes. Intuitively, a domain call may be read as: in the domain called `domainname`, execute the function `domainfunction` defined therein on the arguments (`arg1`, …, `argn`). The *result* of executing this domain call is coerced into a set of entities that have the same type as the output type of the function `domainfunction` on the arguments (`arg1`, …, `argn`)

A *domain-call atom* (DCA-atom) is of the form `in(X, domainfunction`$\rangle$($\langle$`arg1`, …, `argn`$\rangle$) where `in` is a constraint that is satisfied just in case the entity `X` is in the set returned by the domain call in the second argument of `in(-,-)`. In other words, `in` is the polymorphic set membership predicate.

More concretely, `in(A,ingres:select_eq('criminals',"name","john smith"))` is a DCA-atom that is true just in case `A` is a tuple in the result of executing a selection operation (finding tuples where the `name` field is `john smith`) on a relation called `criminals` maintained in a `INGRES` database system.

A *mediator/constrained database* is a set of rules of the form

$$A \quad \leftarrow \quad D_1 \,\&\, \ldots \,\&\, D_m \,\&\, A_1, \ldots, A_n.$$

where $A, A_1, \ldots, A_n$ are atoms, and $D_1, \ldots, D_m$ are DCA-atoms. Note that for simplicity, we restrict constraints to DCA-atoms of the form described above. This does not however, detract from the generality of the techniques described in the paper – It has been shown in [22] that all the types of constraints considered by Kanellakis et. al. can be captured within this framework (Lu, Nerode, and Subrahmanian present further details [21]).

Given any mediator M that integrates information from domains $\mathtt{d_1}, \ldots, \mathtt{d_n}$, we use the notation $(M, \{\mathtt{d_1}, \ldots, \mathtt{d_n}\}) \models A$ to indicate that the atom A can be inferred by using the mediator M in conjunction with the domains $\mathtt{d_1}, \ldots, \mathtt{d_n}$. A formal model-theoretic and fixpoint semantics justifying this construction has been developed in [21] – those details are not necessary for the purpose of this paper.

2.1 Motivating Example for Security in Mediated Databases

We introduce a running example which also motivates our approach. This example has been addressed in the existing HERMES implementation [28].

Suppose a police-officer is conducting a drug investigation. For this purpose, he asks the following query:

> POLICE QUERY. Find all people who have been seen with Cesar Cali and who have a criminal record and whose bank accounts have had at least two deposits of over $ 8,000 each during the period August–Sep. 1994.

When processing this query, the following types of databases/software packages may need to be accessed, and some of these packages may impose various types of security restrictions.

1. A **face-recognition** package may need to be accessed – this package contains two functions `segmentface` and `matchface`. The `segmentface` function takes as input, a photograph, and returns as output, a list of file names containing just the faces occurring in those photographs. Thus, for instance, if `segmentface` is called with a file called `surv.1` which contains three people, then this function will create three files, one containing just the face of each of the three people in the photograph (with background clutter eliminated). The `matchface` takes two arguments – each a localized face (such as those generated by the `segmentface` function, or such as those in a mugshot database) and returns the name of the person involved if the two faces "match" – otherwise it returns "false." In this example, the functions in the face-recognition package are accessible to all police officers, but the data (see below) may not be.
2. Two face databases – one called **survdata** contains photographic data collected by the local police department in the course of this investigation – it is accessible to all police officers with clearance levels of 1 or higher, The second database, called **mugshotdb** is a mugshot database maintained at a centralized police site that is only accessible to police officers with clearance levels of 2 or more.
3. There is a third database called `bank_activity` that is an INGRES database having the schema `(Name,Acctnum,Amount,Activity,Day,Mth,Yr)`. In the USA, all banks are required by law to report "suspicious" transactions – such an INGRES database may be created using such transaction reports. We will assume that the `bank_activity` database is accessible to all police officials with clearance levels of 2 or more.
4. Finally, there is a fourth database, called `criminal` that is also an INGRES database having the schema `(Name,Conviction,Day,Month,Year)`. This database has an interesting *content-dependent clearance policy.* Officers with clearance level 2 or higher can access all the data in this – however officers with clearance level 1 can only access data on crimes committed since Jan. 1, 1990.

Figure 2 shows the scenario outlined in this example. Similarly, Table 1 gives the name and contents of the `survdata`, while Table 2 gives the form and contents of the mugshot database. The `bank_activity` and `criminals` relational databases are shown in Figures 3 and 4, respectively.

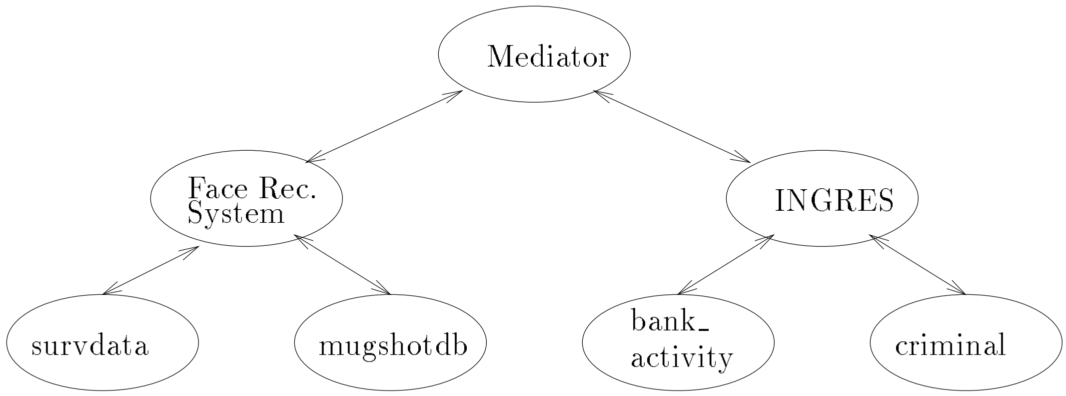

Figure 2: Mediated System used in the Police Example

Name	Acctnum	Amount	Activity	Day	Mth	Yr
John Smith	77732	9126.00	deposit	15	Aug.	1994
John Smith	77732	6232.00	deposit	10	Sep.	1994
John Smith	77732	8654.00	deposit	18	Nov.	1994
Jane Doe	36542	1932.00	deposit	11	Nov.	1994
Jane Doe	36542	9500.00	deposit	17	Nov.	1994
Jane Doe	36542	9500.00	deposit	1	Jan.	1995
David Jones	94531	8700.00	deposit	24	Nov.	1994
David Jones	94531	8900.00	deposit	21	Dec.	1994
Elena Sontag	45329	8195.00	deposit	1	Dec.	1994
Elena Sontag	45329	9876.00	deposit	18	Dec.	1994

Figure 3: The relation `bank_activity`

picturefile	contents
file1	Cesar Cali and John Smith
file2	John Smith and Jane Doe
file3	Cesar Cali and David Jones
file4	Cesar Cali and Elena Sontag
file5	Cesar Cali and Jim Perry
file6	Cesar Cali and Anne Dow

Table 1 : Contents of survdata

photo	person
pic1	Cesar Cali
pic2	John Smith
pic3	David Jones
pic4	Adam Payton
pic5	Jim Perry
pic6	Vic Simpson
pic7	Anne Dow
pic7	Elena Sontag
pic8	Kevin Wylie.

Table 2 : Contents of mugshotdb

2.2 Expressing the Police Query

Using the above data, it is possible to express the police query as a predicate `pq`. The query $\leftarrow$ `pq`(`Person`) is satisfied just in case `Person` satisfies the conditions in the police query. Three

Name	Conviction	Day	Mth	Yr
John Smith	drug	1	Feb.	1992.
Vic Simpson	assault	13	Mar.	1992
Elena Sontag	drug	8	Jun.	1987
Kevin Wylie	theft	17	May.	1993
David Jones	drug	19	Apr.	1992.
Jane Doe	espionage	14	Sep.	1990.

Figure 4: The Relation `criminals`

"sub" predicates are defined – `pq1(Person)` is satisfied if `Person` was seen with Cesar Cali by accessing the face recognition system. `pq2(Person)` is satisfied if `Person` had made two or more deposits of 8000 dollars or more during Aug.-Dec. 1994 and `pq3(Person)` is satisfied if `Person` has a criminal history. Note that no information about clearances is encoded in the rules given below. How these clearances are handled in the presence of such a mediator will become clear in the next section. A somewhat enhanced version of this example has been completely implemented in HERMES – the clauses below is a simplified version of the implemented version.

$$
\begin{array}{rcl}
\mathtt{pq1(Person)} & \leftarrow & \mathtt{in(Face1, face : segmentface(survdata))}\ \& \\
 & & \mathtt{in(Shot1, face : matchface(Face1, mugshotdb))}\ \& \\
 & & \mathtt{in(``Cesar\ Cali'', face : givename(Shot1))}\ \& \\
 & & \mathtt{in(Face2, face : segmentface(survdata))}\ \& \\
 & & \neq \mathtt{(Face1, Face2)}\ \& \\
 & & = \mathtt{(Face1.origin, Face2.origin)}\ \& \\
 & & \mathtt{in(Shot2, face : matchface(Face2, mugshotdb))}\ \& \\
 & & \mathtt{in(Person, face : givename(Shot2))}. \\
\mathtt{pq2(Person)} & \leftarrow & \mathtt{in(Tuple1, ingres : select\_geq(bank\_activity, amount, 8000))}\ \& \\
 & & = \mathtt{(Tuple1.activity, ``deposit'')}\ \& \\
 & & \mathtt{in(Tuple2, ingres : select\_geq(bank\_activity, amount, 8000))}\ \& \\
 & & = \mathtt{(Tuple2.activity, ``deposit'')}\ \& \\
 & & \neq \mathtt{(Tuple1, Tuple2)}\ \& \\
 & & = \mathtt{(Tuple1.name, Tuple2.name)}\ \& \\
 & & = \mathtt{(Tuple1.name, Person)}\ \& \\
 & & = \mathtt{(Tuple1.yr, 1994)}\ \& \\
 & & = \mathtt{(Tuple2.yr, 1994)}\ \& \\
 & & \mathtt{between(Tuple1.mth, aug, dec)}\ \& \\
 & & \mathtt{between(Tuple2.mth, aug, dec)}. \\
\mathtt{pq3(Person)} & \leftarrow & \mathtt{in(Tuple, , ingres : select\_eq(criminals, name, Person))}. \\
\mathtt{pq(Person)} & \leftarrow & \mathtt{pq1(Person)}\ \&\ \mathtt{pq2(Person)}\ \&\ \mathtt{pq3(Person)}.
\end{array}
$$

The `between` predicate is implemented in the obvious way.

If one considers this example, we notice that the following properties hold:

$$
\begin{array}{rcl}
(M, \{\mathtt{d_1}, \ldots, \mathtt{d_n}\}) & \models & \mathtt{pq(``JohnSmith'')} \\
(M, \{\mathtt{d_1}, \ldots, \mathtt{d_n}\}) & \not\models & \mathtt{pq(``JaneDoe'')} \\
(M, \{\mathtt{d_1}, \ldots, \mathtt{d_n}\}) & \models & \mathtt{pq(``DavidJones'')} \\
(M, \{\mathtt{d_1}, \ldots, \mathtt{d_n}\}) & \models & \mathtt{pq(``ElenaSontag'')}.
\end{array}
$$

Note that the $\models$ relation expresses what can be concluded from a mediated system. However, it does not say anything about *who* can draw such conclusions. In particular, a police official with clearance level 1 cannot conclude that Elena Sontag satisfies the police query because such officers cannot access information on crimes committed before 1990 (and she committed her crime in 1987).

3 Security in Mediated Systems

In a mediated system involving a mediator M that integrates packages/domains $\mathtt{d_1}, \ldots, \mathtt{d_k}$, each package may have its own unique security authorization policy, which it would like to enforce. There are two possible ways of handling security.

- **Mediator-Enforced Security:** If an end-user is accessing that package through the mediator M, then the package may choose to "trust" the mediator and allow the mediator to enforce the security policy of the package. This necessitates that the appropriate security criteria be encoded as rules within the mediator, and furthermore, that these rules be enforced. This is relatively easy to do within HERMES and we will only briefly mention, in Section 3.1.1, how this may be accomplished.
- **Package-Enforced Security:** It is far more likely that packages will wish to retain control of their security procedures, i.e. the package may choose not to trust the mediator (or mediator-author) to properly enforce its security policy. In this case, the mediator serves two purposes:
  - First, it may try to process the user's query in a way that "disturbs" the user as little as possible, using packages that either allow mediator-enforced security, or impose relatively few restrictions on security. Packages that enforce stringent security will only be used when other means of processing the query fail.
  - When a package that uses package-enforced security is queried, then the mediator serves merely as a communication medium, conveying the security screening requests of the package to the user, and returning the user's responses to the package. This happens only when the preceding item doesn't lead to a path that bypasses package-enforced security.

In the rest of this paper, we will concentrate on how the mediator handles package-enforced security. We will prove that these methods are sound and complete.

3.1 A Formal Model

3.1.1 Global Security

We will assume that the mediator has a special meta-predicate, called SECRET that takes two arguments. The first argument is either a domain-call or an atom, while the second argument is an integer, denoting a clearance level. $\mathtt{SECRET}(A, i)$ means that any user with a security level strictly less than i has no access to information about the atom A. We assume that the function SECRET satisfies the axiom scheme:

$$i \leq j \,\&\, \mathtt{SECRET}(A, j) \quad \rightarrow \quad \mathtt{SECRET}(A, i) \tag{1}$$

The axiom enforces the fact that if A is secret to somebody having a high clearance level (i.e. j) then A is secret w.r.t. people with lower clearance levels. Note that the above axiom can be expressed as a rule within the mediator.

In general, the SECRET meta-predicate may be defined within the mediator either by specifying a set of facts. For instance, in the Police example, we may have a security constraint that people strictly below the clearance level of 3 cannot access any information in the INGRES database about Edgar Brown (who may be a special undercover operative). This can be encoded as the fact:

```
SECRET(ingres:select_eq('criminals',name,''Edgar Brown''),3).
```

A *global security definition* (GSD, for short) is a finite set of facts of the form

$$\texttt{SECRET(A,i)} \quad \leftarrow$$

In addition, all GSDs are assumed to include axiom scheme 1. We will see later, in Section 4.3, that the above axiom is not sufficient – one more axiom will be needed to satisfy certain intuitions about inferential security – security when the user may use logical reasoning to infer a supposedly restricted piece of information.

3.1.2 Local Security

Consider a domain, $\texttt{d}$, participating in a mediated system. When a call is made to this package by the mediator, the call is of the form $\texttt{d}:\texttt{f}(<\texttt{arguments}>)$. A *local security check* policy for domain $\texttt{d}$ is a function $\texttt{VIOL}_\texttt{d}$ that takes as input, a domain call $\texttt{d}:\texttt{f}(<\texttt{arguments}>)$ and a clearance level i and returns a boolean value as output. Intuitively, if $\texttt{VIOL}_\texttt{d}(\texttt{d}:\texttt{f}(<\texttt{arguments}>), i) =$ true (resp. false) then this means that the query violates security (resp. does not violate security) at level i. All local security check functions are assumed to satisfy the condition

$$i \leq j \,\&\, \texttt{VIOL}_\texttt{d}(\texttt{d}:\texttt{f}(<\texttt{arguments}>), j) \;=\; \text{true} \;\rightarrow\; \texttt{VIOL}_\texttt{d}(\texttt{d}:\texttt{f}(<\texttt{arguments}>), i) \;=\; \text{true}.$$

Recall that external domains can be called by the mediator only through invocations of the form $\texttt{in}(\texttt{X}, \texttt{d}:\texttt{f}(<\texttt{args}>))$. We say that domain $\texttt{d}$ *securely satisfies this call w.r.t. a user having clearance level* i just in case there exists a substitution θ such that:

1. $\texttt{X}\theta$ is one of the objects returned by evaluating the function $\texttt{f}$ in domain $\texttt{d}$ on the specified arguments $<\texttt{args}>$ and
2. $\texttt{VIOL}_\texttt{d}(\texttt{d}:\texttt{f}(<\texttt{arguments}>), i) = \mathit{false}$.

Notice that any statement of the form $\texttt{SECRET}(\texttt{d}:\texttt{f}(<\texttt{args}>), \texttt{i})$ can be considered to be a part of the definition of $\texttt{VIOL}_\texttt{d}$ by setting $\texttt{VIOL}_\texttt{d}(\texttt{d}:\texttt{f}(<\texttt{arguments}>), i) =$ true if $\texttt{SECRET}(\texttt{d}:\texttt{f}(<\texttt{args}>), \texttt{i})$ is true. *Hence, without loss of generality, in the rest of this paper, we will assume that global security definitions only contain statements of the form* $\texttt{SECRET}(A, i)$ *where* A *is a non-DCA atom.*

POLICE EXAMPLE REVISITED. As an example, let us return to the case of the police query described earlier on in this paper. Consider a police officer with clearance level 1, and consider the INGRES $\texttt{criminals}$ relation. Suppose this officer issues the call

$$\texttt{in}(\texttt{X}, \texttt{ingres}:\texttt{select\_eq}('\texttt{criminals}', \texttt{conviction}, \texttt{drugs}))$$

asking for the tuples involving people convicted on drug charges. Notice, in particular, that

$$\texttt{VIOL}_\texttt{d}(\texttt{ingres}:\texttt{select\_eq}('\texttt{criminals}', \texttt{conviction}, \texttt{drugs}), 1) = \texttt{false}.$$

The *actual answer* to this query should return three tuples, viz.

Name	*Conviction*	*Day*	*Mth*	*Yr*
John Smith	drug	1	Feb.	1992.
Elena Sontag	drug	8	Jun.	1987
David Jones	drug	19	Apr.	1992.

Actual Solution to Query

Actually, due to the security criterion which says that police officers who have clearance 1 can only access crimes committed after Jan. 1, 1990, the answer returned to the police officer issuing the above query will *not* include Elena Sontag. Thus, he will see the answer:

Tuple-id	*Name*	*Conviction*	*Day*	*Mth*	*Yr*
t_1	John Smith	drug	1	Feb.	1992.
t_2	David Jones	drug	19	Apr.	1992.

Solution Returned to Police Officer with Clearance Level 1

Given the tuple-ids $\mathtt{t}_1, \mathtt{t}_2$ shown in the above table, INGRES securely satisfies the call $\mathtt{in}(\mathtt{X}, \mathtt{ingres} : \mathtt{select\_eq}(\mathtt{'criminals'}, \mathtt{conviction}, \mathtt{drugs}))$ w.r.t. the user having clearance level $\mathtt{i} = \mathtt{1}$ via two substitutions – viz. $\{\mathtt{X} = \mathtt{t}_1\}$ and $\{\mathtt{X} = \mathtt{t}_2\}$. □

Security Violations. In the event a given query violates security, then the answer to be returned is a special symbol, $\perp_{\mathtt{d},\mathtt{f}}$. If the given query does not violate security, then the relevant function is invoked within the package and the computed answer is returned. Thus, for example, suppose the police officer with clearance level 1 asked the query:

$$\mathtt{in}(\text{“}\mathtt{Elena\ Sontag}\text{''}, \mathtt{ingres} : \mathtt{project}(\mathtt{select\_eq}(\mathtt{'criminals'}, \mathtt{conviction}, \mathtt{drugs}), \mathtt{name})).$$

This query asks specifically whether Elena Sontag has been convicted on drug charges. This information is supposed to be inaccessible to the police officer. We may model the value returned by the local INGRES database, in this case, as being the value $\perp$. At this point, the reader may suspect that the police officer can violate security as follows:

> Simulated Police Officer's Reasoning at Clearance Level 1. He reasons that if Elena Sontag had no conviction, then the above query would have failed. Hence, as it did not fail, but instead returned $\perp$, the system has something to hide about Elena Sontag. If the officer knows of the implemented security policy, he may then conclude that Elena Sontag had a conviction on drug charges prior to 1990. This is a security violation. □

The above problem is easily avoided by applying a technique similar to the well-known query modification[24]. Whenever a query of the form

$$\leftarrow \mathtt{in}(\mathtt{X}, \mathtt{d} : \mathtt{f}(< \mathtt{args} >))$$

is asked by a user at clearance level i, our system will in fact treat that query as (the slightly more complex query)

$$\leftarrow \mathtt{in}(\mathtt{X}, \mathtt{d} : \mathtt{f}(< \mathtt{args} >))\ \&\ \mathtt{X} \neq \perp_{\mathtt{d:f}}.$$

In the case of the query considered earlier, this means that the query asked by the police officer with clearance level 1 will in fact fail, and s/he will not know whether it failed because Elena Sontag had no convictions, or whether it failed because Elena Sontag had a conviction prior to 1990.

In section 2, we have described the semantics of such calls – the output returned is whatever function f would return if it were invoked with the specified arguments. However, when local security policies are being implemented, and if the domain/package d feels that a certain call violates its security policy, then the behavior of the domain/package may be unpredictable. In order to model this "unpredictable behavior", we will augment the range of each domain function by the addition of an artificial value, denoted $\perp$[1]. Whenever a package/domain's local security system responds, then the response can be viewed as the returning of a special answer, denoted $\perp_{\mathtt{d,f}}$ – when d and f are clear from context, we will omit these subscripts.

3.1.3 Local + Global Security

A *secure mediated system*, $\mathcal{M}$ consists of four components:

1. A mediator M
2. a set $\mathtt{d_1}, \ldots, \mathtt{d_k}$ of packages/domains integrated by M,
3. a global security policy $\mathcal{G}$ and
4. a set $\{\mathtt{VIOL}_1, \ldots, \mathtt{VIOL}_k\}$ of local security policies where $\mathtt{VIOL}_i$ is the security check policy associated with the local domain $\mathtt{d_i}$, $1 \leq \mathtt{i} \leq k$.

Consider a ground instance

$$A \quad \leftarrow \quad D_1 \,\&\, \ldots \,\&\, D_m \,\&\, A_1 \,\&\, \ldots \,\&\, A_n$$

of a HERMES rule where $D_j = \mathtt{in}(\mathtt{X}, \mathtt{d_j} : \mathtt{f_j}(< \mathtt{args} >))$. Then this rule can be fired, at clearance level i, just in case each of D_j's is securely satisfied by domain d at clearance level i and each of the A_r's is true. Note that here, we are not considering global security (i.e. the A_i's may or may not be globally secure). Given this definition of firability of rules, we may associate a notion of logical consequence – $\mathcal{M}$ has A as a logical consequence at level i, denoted $\mathcal{M} \mapsto^i A$ iff A can be obtained from $M \cup \mathcal{G}$ by repeatedly firing the rules using the notion of level-i firability described above.

Police Example Revisited. Let us return to the example of the Police Query and consider the predicate pq3 defined in the mediator. Recall, also, that $(\mathtt{M}, \{\mathtt{d_1}, \ldots, \mathtt{d_n}\}) \models \mathtt{pq3}(\text{``}\mathtt{Elena\,Sontag}\text{''})$. Consider now the mediated system $\mathcal{M}$ constructed using the above mediator and domains/packages, and that uses the security policy described so far. In particular, recall that police officers with clearance level 2 can access data on Elena Sontag's criminal records, but police officers with clearance level 1 cannot do so. It is now the case that:

$$\begin{array}{lll} \mathcal{M} & \mapsto^2 & \mathtt{pq3}(\text{``}\mathtt{Elena\,Sontag}\text{''}). \\ \mathcal{M} & \not\mapsto^1 & \mathtt{pq3}(\text{``}\mathtt{Elena\,Sontag}\text{''}). \\ \mathcal{M} & \mapsto^1 & \mathtt{pq3}(\text{``}\mathtt{David\,Jones}\text{''}). \\ \mathcal{M} & \mapsto^2 & \mathtt{pq3}(\text{``}\mathtt{David\,Jones}\text{''}). \end{array}$$

[1] Strictly speaking, we should write $\perp_{\mathtt{d,f}}$.

$\square$

We say that the mediated system $\mathcal{M}$ *securely implies A for a user having clearance level i*, denoted $\mathcal{M} \trianglerighteq^i A$, just in case:

- $\mathcal{M} \mapsto^i A$, and
- $(M \cup \mathcal{G}, \{\mathtt{d}_1, \ldots, \mathtt{d_k}\}) \not\models \mathtt{SECRET}(A, i)$, i.e. $\mathtt{SECRET}(A, i)$ is *not* a logical consequence consequence of $M \cup \mathcal{G}$ w.r.t. domains $\mathtt{d}_1, \ldots, \mathtt{d_k}$.

Intuitively, this definition says that a ground atom A is securely implied by a mediated system just in case A is true and does not violate the security policies of packages, and A is not supposed to be kept secret (using the global security constraints) from users having clearance level i.

POLICE EXAMPLE REVISITED. Let us suppose, for instance, that we have one global security constraint in our police mediator specifying that the predicate pq3("DavidJones") is available only to police officers with clearance level 2 or more. Then this may be stated as the global security definition:

$$\mathtt{SECRET}(\mathtt{pq3}(\text{``}\mathtt{DavidJones}\text{''})) \quad \leftarrow \quad .$$

There may be several reasons the police want to limit access to information about David Jones (e.g. he may be an undercover operative, etc.). In this case, recall that

$$\mathcal{M} \quad \mapsto^1 \quad \mathtt{pq3}(\text{``}\mathtt{David\ Jones}\text{''}).$$

However, $(M \cup \mathcal{G}, \{\mathtt{d}_1, \ldots, \mathtt{d_k}\}) \models \mathtt{SECRET}(\mathtt{pq3}(\text{``}\mathtt{DavidJones}\text{''}))$; hence, it is the case that

$$\mathcal{M} \quad \not\trianglerighteq^1 \quad \mathtt{pq3}(\text{``}\mathtt{David\ Jones}\text{''}).$$

However, it is still the case that

$$\mathcal{M} \quad \trianglerighteq^2 \quad \mathtt{pq3}(\text{``}\mathtt{David\ Jones}\text{''}).$$

We now formally define the concept of an *answer* that may be returned to a user having a specified clearance level in a secure mediated system.

Definition 3.1 (Secure Answers) Suppose $\mathcal{M} = (M, \{\mathtt{d}_1, \ldots, \mathtt{d_k}\}, \mathcal{G}, \{\mathtt{VIOL}_1, \ldots, \mathtt{VIOL}_k\})$ is a secure mediated system. Suppose A is an *atomic query* posed by a user having clearance level i. Then, a substitution θ is a *secure answer* to this query iff $\mathcal{M} \trianglerighteq^i (\forall) A\theta$..
Suppose Q is a *conjunctive query*. Then, a substitution θ is a *secure answer* to this query iff $\mathcal{M} \trianglerighteq^i (\forall) A_j\theta$ for every atom A_j occurring in Q.

4 Implementing Cautious Cooperation

The key function provided by any database is the ability to answer queries correctly. In this paper, we have defined the notion of a secure answer to a query. The key function of a secure mediated system is to provide secure answers to the user. We use the technique of query

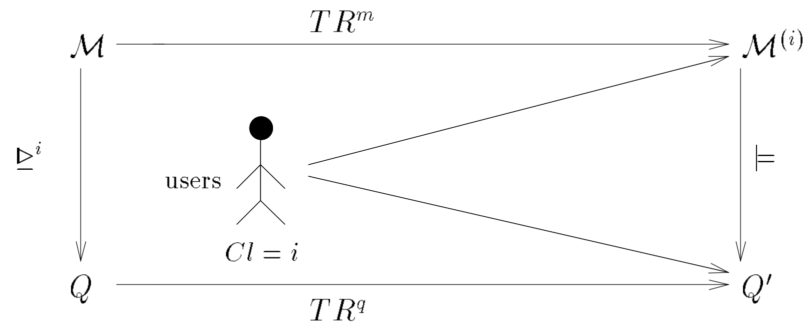

Figure 5: Implementation Approach

modification [24], augmented with other techniques, to implement the principle of cautious cooperation within HERMES.

The HERMES system already has a methodology for processing queries that is sound and complete w.r.t. the standard semantics of such systems defined in terms of the $\models$ relation introduced earlier that specifies what is true. However, consider (for simplicity) an atomic query A posed by the user and suppose θ is a substitution such that $(M, \{\mathtt{d_1}, \ldots, \mathtt{d_k}\}) \models (\forall) A\theta$. However, this does not mean that θ is a *secure answer* to this query – this is because the $\models$ relation does not take security into account, instead it just specifies what follows logically from the mediator M and the underlying data sources/packages $\{\mathtt{d_1}, \ldots, \mathtt{d_k}\}$.

In this section, we will show how, given any secure mediated system $\mathcal{M} = (M, \{\mathtt{d_1}, \ldots, \mathtt{d_k}\}, \mathcal{G}, \{\mathtt{VIOL}_1, \ldots, \mathtt{VIOL}_k\})$, and a query Q posed by a user having clearance level i, we can create a new secure mediated system $\mathcal{M}^{(i)}$ and a new query Q' such that

> *θ is a secure answer to query Q w.r.t. secure mediated system $\mathcal{M}$ w.r.t. clearance level i* **IFF** *θ is an answer to query Q' w.r.t. secure mediated system $\mathcal{M}^{(i)}$.*

Mathematically, this can be represented as: $\mathcal{M} \trianglerighteq^i (\forall) Q\theta$ IFF $\mathcal{M}^{(i)} \models (\forall) Q'\theta$. Figure 5 shows a diagrammatic representation of this relationship. The figure shows two transformations that are needed – the transformation TR^m describes how given $\mathcal{M}$ and a user clearance level i, we may construct $\mathcal{M}^{(i)}$. The transformation TR^q shows how, given a query Q and the user clearance level i, we may construct Q^i. In the rest of this section, we will describe these two transformations and show how they may be implemented.

4.1 The TR^m Transformation

Suppose

$$\mathcal{M} \quad = \quad (M, \{\mathtt{d_1}, \ldots, \mathtt{d_k}\}, \mathcal{G}, \{\mathtt{VIOL}_1, \ldots, \mathtt{VIOL}_k\}).$$

is a secure mediated system and i is the clearance level of a user. Consider a mediator clause of the form

$$A \quad \leftarrow \quad D_1 \,\&\, \ldots \,\&\, D_m \,\&\, A_1, \ldots, A_n.$$

where $A, A_1, \ldots, A_n$ are atoms, and $D_1, \ldots, D_m$ are DCA-atoms. We replace this clause by a clause of the form:

$$A^{(i)} \leftarrow D_1^{(i)} \,\&\ldots\&\, D_m^{(i)} \,\&\, N_1 \,\&\cdots\&\, N_m \,\& A_1^{(i)}, \&\ldots\& A_n^{(i)} \,\&\mathbf{not}(\mathtt{SECRET}(A^{(i)})).$$

where the atoms occurring in the above clause are obtained in the following way:

- If $\mathtt{D_j} = \mathtt{in}(\mathtt{X_j}, \mathtt{d_j} : \mathtt{f_j}(<\mathtt{args}>))$, then $\mathtt{D_j^{(i)}} = \mathtt{in}(\mathtt{X}, \mathtt{d_j} : \mathtt{f'_j}(<\mathtt{args}, \mathtt{i}>))$ where:
  1. $\mathtt{f'_j}(<\mathtt{args}, \mathtt{i}>)) = \mathtt{f_j}(<\mathtt{args}>)$ iff $\mathtt{VIOL_j}(\mathtt{d} : \mathtt{f}(<\mathtt{args}>, \mathtt{i})) = \mathtt{false}$. This says that the function $\mathtt{f'_j}$ behaves exactly like $\mathtt{f}$ if no violation occurs when a user with clearance $\mathtt{i}$ poses this query.
  2. $\mathtt{f'_j}(<\mathtt{args}, \mathtt{i}>)) = \perp_{\mathtt{d_j}:\mathtt{f_j}}$ otherwise.
- $\mathtt{N_j}$ is the disequality constraint $\mathtt{X_j} \neq \perp_{\mathtt{d_j}:\mathtt{f_j}}$.
- If $\mathtt{A_j} = \mathtt{p}(\mathtt{t_1}, \ldots, \mathtt{t_n})$, then $\mathtt{A_j^{(i)}} = \mathtt{p}(\mathtt{t_1}, \ldots, \mathtt{t_n}, \mathtt{i})$.

Thus, given a secure mediated system $\mathcal{M}$, we replace the mediator M in this system by the mediator $M' = \{TR^m(C) \mid C$ is a rule in $M\}$ where M is the mediator in $\mathcal{M}$.

The mediator M' obtained by this transformation contains negation – It is easy to see that this negation leads to a *stratified* rule set, and that the notion of $\models$ applied to such mediators is inference from a stratified program.

4.2 The TR^q Transformation

Any user query Q is of the form $\leftarrow \mathtt{D_1} \,\&\ldots\&\, \mathtt{D_m} \,\&\, \mathtt{A_1}, \ldots, \mathtt{A_n}$ where $A_1, \ldots, A_n$ are atoms, and $D_1, \ldots, D_m$ are DCA-atoms. Given that this query is posed by a user having clearance level i, this query can be transformed in exactly the same way as above to give the query $Q' = TR^q(Q)$ below:

$$\leftarrow D_1^{(i)} \,\&\ldots\&\, D_m^{(i)} \,\&\, N_1 \,\&\cdots\&\, N_m$$

Theorem 1 Suppose $\mathcal{M} = (M, \{\mathtt{d_1}, \ldots, \mathtt{d_k}\}, \mathcal{G}, \{\mathtt{VIOL_1}, \ldots, \mathtt{VIOL}_k\}$ is a secure mediated system and Q is a query posed by a user having clearance level i. Then:

1. **(Soundness)** If θ is a substitution such that $(TR^m(M), \{\mathtt{d_1}, \ldots, \mathtt{d_k}\}) \models (\forall)TR^q(Q)\theta$, then $\mathcal{M} \trianglerighteq^i (\forall)Q\theta$, i.e. θ is a secure answer to query Q w.r.t. $\mathcal{M}$.
2. **(Completeness)** If θ is a substitution such that $\mathcal{M} \trianglerighteq^i (\forall)Q\theta$, then $(TR^m(M), \{\mathtt{d_1}, \ldots, \mathtt{d_k}\}) \models (\forall)TR^q(Q)\theta$.
3. **(Weak Preservation of Security)** There is no atom A such that $\mathtt{SECRET}(A, , i)$ is true and such that $(TR^m(M), \{\mathtt{d_1}, \ldots, \mathtt{d_k}\}) \models (\forall)A^{(i)}$.

The above theorem has the following important impact – HERMES currently computes the $\models$ relation in a sound and complete manner, using a compiler that pre-computes a materialized mediated view (cache) as described in [22], and then using this cache for query processing instead of the rules, unless there is no option but to use the mediator rules. Various intelligent caching techniques to support this kind of query processing [2].

By having the ability to translate the secure query processing problem into one that only involves the $\models$ relation, we may use HERMES to efficiently perform secure query processing. As the reader will observe, the secure version of HERMES may be (and has been) implemented by the following simple steps:

- When the user invokes HERMES, s/he is asked to "sign-on." Based on this, the system looks up the user's clearance level in a database.
- Once the user has specified the mediator s/he wishes to work with, the TR^m transform is computed – this can be done in linear time.
- When a query Q is posed, it is translated into $TR^q(Q)$ – this too can be done in linear time.
- The query $TR^q(Q)$ is then processed w.r.t. $TR^m(M)$ as a standard HERMES query. The above theorem guarantees that the answers computed will be secure.

4.3 Implementing Conservative Cautious Cooperation

The principle of cautious cooperation is rather weak – for instance, a "smart" user may be able to "crack" security in the following way: suppose $\mathtt{SECRET}(A, i)$ is in the global security definition of the mediator, and suppose

$$\mathtt{A} \quad \leftarrow \quad \mathtt{B_1} \,\&\, \ldots \,\&\, \mathtt{B_n} \,\&\, \mathtt{D_1} \,\&\, \ldots \,\&\, \mathtt{D_n}$$

is a clause in the mediator. Suppose the user knows that this clause is in the mediator – then by asking the query

$$\leftarrow \quad B_1 \,\&\, \ldots \,\&\, B_n \,\&\, D_1 \,\&\, \ldots \,\&\, D_n$$

he may be able to *infer* that A is true, independently of his security clearance level. The principle of cautious cooperation above says nothing, whatsoever, about this; yet a system manager may well be justified in using this yardstick as a measure of the security provided by the system.

Consider a user u having clearance level i. Let $\mathtt{BEL}(u,t)$ be a formulas (including ones constructed using DCA atoms) such that $\mathtt{BEL}(u,t)$ is the set of formulas believed by user u at a point t in time. Without loss of generality, we will assume that each formula in $\mathtt{BEL}(u)$ is a universally quantified disjunction of atoms or negated atoms (ordinary or DCA). When the user starts interacting with the secure mediated system, his initial set of beliefs is given by $\mathtt{BEL}(u,0)$, i.e. $t=0$ at this initial time he starts his query interaction.

Let us say that a secure mediated system $\mathcal{M} == (M, \{\mathtt{d_1}, \ldots, \mathtt{d_k}\}, \mathcal{G}, \{\mathtt{VIOL_1}, \ldots, \mathtt{VIOL}_k\})$ is *strongly secure w.r.t. level i at time t* just in case for any sequence of queries $Q_1, \ldots, Q_r$ that the user can ask, there is a set of answers $\sigma_1, \ldots, \sigma_r$ such that:

- these answers are securely implied by $\mathcal{M}$, i.e. $\mathcal{M} \trianglerighteq^i \sigma_j$ for $j = 1, \ldots, r$, or σ_j is the "no" answer denoting that the query is false, and

- the answers, together with the users original beliefs do not allow him to conclude (using classical logic) any secrets, i.e. there is no secret A in $\mathcal{G}$ such that $\mathtt{BEL}(u,0) \cup \{Q_1\sigma_1,\ldots,Q_r\sigma_r\} \models A$.

One Way to Violate Security: Let us examine one possible way to violate security. Initially, let us suppose the user's belief set is $\mathtt{BEL}(u,0)$. Suppose there exists an atom $\mathtt{SECRET}(A,i)$ in $\mathcal{G}$ where i is greater than or equal to the user's clearance. If the user's belief set contains a disjunction $(\mathtt{A}_1 \vee \cdots \vee \mathtt{A}_\mathtt{r})$ of atoms where $\mathtt{A} = \mathtt{A}_\mathtt{j}$ for some $\mathtt{1} \leq \mathtt{j} \leq \mathtt{r}$, then the user may violate security by simply asking the sequence of queries $\mathtt{A}_1, \mathtt{A}_2, \ldots, \mathtt{A}_\mathtt{r}$.

In order to handle this way of violating security, we observe that the following theorem holds.

Theorem 2 Suppose u is a user having clearance level i. suppose $\mathcal{M} = (M, \{\mathtt{d}_1,\ldots,\mathtt{d}_\mathtt{k}\}, \mathcal{G}, \{\mathtt{VIOL}_1,\ldots,\mathtt{VIOL}_k\})$ is a secure mediated system. Then: $\mathcal{M}$ is strongly secure w.r.t. user u at all times t iff there is no disjunction $(\mathtt{A}_1 \vee \cdots \vee \mathtt{A}_\mathtt{m})$ in $\mathtt{BEL}(u,0)$ such that $\mathtt{SECRET}(A_j,i)$ is implied by $\mathcal{G}$ for some $1 \leq j \leq m$. $\Box$

The above theorem is very interesting. It says, in effect, that the user has a technique to violate security iff he knew some disjunction when he started interaction with the system such that one of the disjuncts in this (known) disjunction was supposed to be kept secret from him. Note that this is an *iff* statement. As a consequence, it is enough, if the system has an accurate guess, at the beginning, of what the user actually knows,[2] to maintain security. If the user did not initially know $(A \vee Disj)$ for some secret A, then it is enough to ensure, in response to each query Q asked by the user, that any returned answer substitution θ satisfy the condition:

$$\mathtt{BEL}(u,t) \cup \{Q\theta\} \not\models \bigvee_{\mathtt{SECRET}(A,i)\in\mathcal{G}} (A \vee D)$$

where D is any disjunction and i is the security level of the user. If

Another Way to Violate Security: A more insidious way of violating security is via the "in" predicate. Recall, from Section 3.1.1, that in order to keep information about Edgar Brown secret, the user may have to place the following declaration in the mediator:

```
SECRET(ingres:select_eq('criminals',name,''Edgar Brown''),3).
```

However, the "smart" user may find out information just the same by asking the slightly modified query:

$$\leftarrow (\mathtt{X}, \mathtt{ingres}:\mathtt{project}(\mathtt{select\_eq}('\mathtt{criminals}', \mathtt{name}, \text{“}\mathtt{EdgarBrown}''), \mathtt{conviction}).$$

This query would *not* be treated as a secret by the global security definition. Hence, for each field (such as `conviction`) in the `criminals` relation, the user can project out the value of that field, after first selecting the tuple with `name` = "`Edgar Brown`". He can thus reconstruct the entire record pertaining to Edgar Brown, thus violating security (in fact, all he did was to split

[2]This may be very difficult, and almost impossible to accomplish in practice. In the absence of any other alternative, it seems reasonable to make some educated guess about what the user knows and hope for the best.

the original select query into a bunch of permissible queries whose joint answers implied the value of the select).

In order to solve the latter problem, we will add to the global security definition, the extra axiom scheme shown below:

$$\mathtt{SECRET(d : g(f(< args >), < args >), I)} \quad \leftarrow \quad \mathtt{SECRET(d : f(< args >), I)}.$$

This says that if a particular function application in a domain package is not permitted at a clearance level I, then applying that function first, followed by application of other functions is not permissible either.

Theorem 3 The above techniques implement the principle of conservative, cautious, cooperation in a sound, complete and secure manner.

5 System Implementation

The security techniques described within this paper have been implemented as a component of the HERMES Heterogeneous Reasoning and Mediator System. The entire system currently includes approximately 60,000 lines of C code running under both Unix and Windows environments.

When the secure version of HERMES is invoked from Unix, the system asks the user to identify himself/herself and then determines the person's security clearance level from a database. When the mediator selected by the user is being compiled, if the user's clearance level is not high enough, then some of the domain calls are replaced by a special domain call $call_\perp$ which returns false when called. This prevents the user from using such clauses. This is a partial implementation of the TR^m described in this paper – the rest is enforced at run-time as described below.

At run-time, when the user attempts to execute a specific domain function, the system checks to see if the user is cleared to access the corresponding domain function. Notice that at compile-time, a call of the form $\mathtt{d : f(a, X)}$ may not have been suppressed because some instances of this call (e.g. with $\{\mathtt{X = b}\}$ may be permissible – however, at run-time, there may be an attempt to execute the sub-query $\mathtt{d : f(a, a)}$ – if this subquery is secret, then the run-time implementation of TR^q will suppress this call from succeeding.

For instance when HERMES is invoked by a user with clearance level 1 the query pq("JohnSmith") succeeds because John Smith is a suspect, and no secret information is needed for this user to conclude this. In contrast, the query pq("ElenaSontag" fails because it involves security violations. In this case, the system merely says "no", and hence, the user cannot determine whether the system is ignorant of Elena Sontag, or whether the system is keeping information about her secret.

In general, when user executes a domain function call, but the domain may refuse to give an answer by returning a special symbol or string (e.g. the string Permission Denied.). Our algorithms would treat such strings as the special symbol ⊥ as described in the paper and treat such as a value as a failure for the given domain call to succeed.

6 Conclusions

Although there has been a great deal of research interest in various technical issues (such as semantic heterogeneity, schema integration, transaction processing, and query optimization) related to federated data management, research on security issues has been very limited. In [18], Jonscher and Dittrich study the access control problems that arise in tightly controlled federations. The model for sharing in [6] is based on the establishments of bilateral agreements that explicitly specify the terms of the agreements and each member's duty for protecting other member's data. In contrast to these efforts, we allow external packages to assert and enforce their own security policies, and provide a formal mathematical model for reasoning about query processing in such environments.

The ideas described within this paper have been integrated into the HERMES ("Heterogeneous Reasoning and Mediator System") platform for mediator development. The current platform allows us to integrate

- database systems (INGRES, DBASE, PARADOX are completed, while ObjectStore is under completion),
- reasoning systems (face recognition software and a US Dept. of Defense path planning system have been integrated)
- non-traditional DBMSs that have been integrated include a quadtree-based GIS, a text database management system, and multimedia data including images (GIF and TIFF files), and video data.

The security mechanisms described in this paper have been implemented on the current HERMES platform, and to our knowledge, is the first implementation of secure mediated systems that apply not just to standard DBMSs, but also apply to integration of heterogeneous software and non-structured data.

References

[1] S. Adalı, V.S. Subrahmanian, Amalgamating Knowledge Bases, III: Distributed Mediators, *International Journal of Intelligent Cooperative Information Systems*, 1994.

[2] S. Adalı, V.S. Subrahmanian, *Intelligent Caching in HERMES*, to appear: Proc. 1995 Intl. Conf. on Very Large Knowledge Bases, Twente, Holland, April 1995.

[3] Ahad, R., et al., "Supporting Access Control in an Object-Oriented Database Language," in *Proc. International Conference on Extending Database Technology (EDBT)*, Vienna, Austria, Springer-Verlag LNCS 580, 1992.

[4] Y. Arens, C.Y. Chee, C.-N. Hsu and C. Knoblock. (1993) *Retrieving and Integrating Data From Multiple Information Sources*, International Journal of Intelligent Cooperative Information Systems, 2, 2, pps 127–158.

[5] J. Benton, V.S. Subrahmanian, Using Hybrid Knowledge Bases for Missile Siting Problems, *Proceedings of the Conference on Artificial Intelligence Applications*, IEEE Computer Society, 1994.

[6] B. T. Blaustein et al., A model of secure federated data management, Technical Report, MITRE Corp., 1994.

[7] P. Bonatti, S. Kraus and V.S. Subrahmanian. (1992) *Declarative Foundations of Secure Deductive Databases,* Proc. 1992 Intl. Conf. on Database Theory, Lecture Notes in Computer Science, Vol. 646, pp. 391–406, Springer. Full version to appear in IEEE Trans. on Knowledge and Data Engineering, 1995.

[8] Y. Breitbart, H. Garcia-Molina, A. Silberschatz, Overview of Multidatabase Transaction Management, *Journal of Very Large Databases*, 2, pp. 181–239, 1992.

[9] M.W. Bright, A.R. Hurson, S. Pakzad, Automated Resolution of Semantic Heterogeneity in Multidatabases, *ACM Transactions on Database Systems*, 19, 2, pp. 212–253, 1994.

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

[11] Fagin, R., "On an Authorization Mechanism," *ACM TODS,* Vol. 3, No. 3, September 1978.

[12] J. Grant, W. Litwin, N. Roussopoulos, T. Sellis, An Algebra and Calculus for Relational Multidatabase Systems, *Proceedings of the First International Workshop on Interoperability in Multidatabase Systems*, IEEE Computer Society Press, pp. 118 – 124, 1991.

[13] Griffiths, P.G., and Wade, B., "An Authorization Mechanism for a Relational Database System," *ACM TODS,* Vol. 1, No. 3, September 1976.

[14] J. Horst, E. Kent, H. Rifky, V.S. Subrahmanian, Hybrid Knowledge Bases for Real-Time Robotic Reasoning, *Proceedings of the International Workshop on Pattern Recognition in Practice*, E. Gelsema, L. Kanal eds., North Holland/Elsevier, 1994.

[15] N. B. Idris, W. A. gray and R. F. Churchhouse. (1994) *Providing Dynamic Security Control in a Federated Database*, Proc. 1994 Intl. Conf. on Very Large Databases, pps 13–23.

[16] Jajodia, S., and Sandhu, R., "Toward a Multilevel Relational Data Model," in *Proc. ACM-SIGMOD Conf.,* Denver, May 1991.

[17] Jajodia, S., and Kogan, B., "Integrating an Object-Oriented Data Model with Multilevel Security," in *Proc. IEEE Symposium on Security and Privacy*, Oakland, California, May 1990.

[18] D. Jonscher and K. R. Diittrich, An approach for building secure database federations, *Proc. 20th VLDB Conf.*, 1994.

[19] R. Krishnamurthy, W. Litwin, W. Kent, Language Features for Interoperability of Databases with Schematic Discrepancies, *Proceedings of ACM Conference on Management of Data*, 1991.

[20] Y. Labrou and T. Finin. (1994) *A Semantics Approach for KQML – A General Purpose Communications Language for Software Agents*, Proc,. 1994 Intl. Conf. on Information and Knowledge Management, pps 447–455.

[21] J. Lu, A. Nerode, V.S. Subrahmanian, Hybrid Knowledge Bases, *IEEE Transactions on Knowledge and Data Engineering*, 1994. To appear. Available as technical report CS-TR-3037, University of Maryland.

[22] J. Lu, G. Moerkotte, J. Schue and V.S. Subrahmanian. (1995) *Efficient Maintenance of Materialized Mediated Views*, Proc. 1995 ACM SIGMOD Conf. on Management of Data, to appear, May 1995.

[23] Millen, J., and Lunt, T., "Security for Object-Oriented Database Systems" in *Proc. of the IEEE Symposium on Research in Security and Privacy*, Oakland, California, May 1992.

[24] M. Stonebraker, E. Wong, Access Control in a Relational Data Base Management System by Query Modification, *Proceedings of the ACM*, pp. 180–186, 1974.

[25] Rabitti, F., Bertino, E., Kim, W., and Woelk, D., "A Model of Authorization for Next-Generation Database Systems," *ACM TODS*, Vol. 16, No. 1, March 1991.

[26] A. Sheth, J. Larson, Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases, *ACM Computing Surveys*, 22, 3, pp. 183–236, 1990.

[27] V.S. Subrahmanian, Amalgamating Knowledge Bases, *ACM Transactions on Database Systems*, 19, 2, pp. 291–331, 1994.

[28] V.S. Subrahmanian, et. al. (1995) *HERMES: A Heterogeneous Reasoning and Mediator System*, submitted for publication.

[29] G. Wiederhold, Mediators in the Architecture of Future Information Systems, *IEEE Computer*, pp. 38–49, March 1992.

[30] G. Wiederhold, Intelligent Integration of Information, *Proceedings of the ACM Conference on Management of Data*, pp. 434–437, 1993.

[31] Winslett, M., Smith, K., and Qian, X., "Formal Query Languages for Secure Relational Databases," *ACM-TODS*, 19, 4, pp. 626–662, December 1994.