, , 1–29 ()

Providing Security and Interoperation of Heterogeneous Systems *

STEVEN DAWSON dawson@csl.sri.com
Computer Science Laboratory, SRI International, Menlo Park, CA 94025, USA

SHELLY QIAN sqian@securesoft.com
SecureSoft, Inc., 275 Shoreline Dr., Suite 520, Redwood Shores, CA 94065 USA

PIERANGELA SAMARATI samarati@dsi.unimi.it
Università di Milano, Dip. Scienze Informazione, Polo di Crema, 26013 Crema - Italy

Editor: Vijay Atluri

Abstract. Interoperation and information sharing among databases independently developed and maintained by different organizations is today a pressing need, if not a practice. Governmental, military, financial, medical, and private institutions are more and more required to become part of a distributed infrastructure and selectively share their data with other organizations. This sharing process inevitably opens the local system to new vulnerabilities and enlarges the space of possible threats to the data and resources it maintains. As a complicating factor, in general, data sources are heterogeneous both in the data models they adopt and in the security models by which protection requirements are stated.

We present a modeling and architectural solution to the problem of providing interoperation while preserving autonomy and security of the local sources based on the use of wrappers and a mediator. A wrapper associated with each source provides a uniform data interface and a mapping between the source's security lattice and other lattices. The mediator processes global access requests by interfacing applications and data sources. The combination of wrappers and mediator thus provides a uniform data model interface and allows the mapping between restrictions stated by the different security policies. We describe the practical application of these ideas to the problem of trusted interoperation of health care databases, targeted to enforcing security in distributed applications referring to independent heterogeneous sources protected by mandatory policy restrictions. We describe the architecture and operation of the system developed, and describe the tasks of the different components.

Keywords: Secure Interoperation, Mandatory Access Control, Query Processing

1. Introduction

Interoperation of database systems independently developed and evolved is a pressing need. Organizations in both the commercial and military sectors are faced with the problem of integrating independent data sources and accessing them as if they were a single system. First initiated within the context of federated architectures [20], interoperation proposals have been more recently developed with reference to mediator-based architectures [23]. Unlike federated databases, approaches

* A preliminary version of this paper appeared under the title "Secure Interoperation of Heterogeneous Systems: A Mediator-Based Approach," in *Proc. of the IFIP 14th International Conference on Information Security (SEC'98),* Vienna-Budapest, 31 August – 2 September", 1998 [8].

based on mediation do not require the specification and management of a federated schema or multidatabase language. They therefore remove previous impediments to true interoperation and automation [19]. The interest in and success of mediated architectures is witnessed by a large number of research projects and prototypes under development (e.g., [17, 4, 14]). Most attention has been devoted, however, to data and query management issues in interoperation, while less attention has been devoted to the protection of information [25]. On the one hand this may reflect a natural relationship between the problems, since successful protection of information requires precise knowledge of how information is managed. On the other hand, security is a basic requirement for interoperation. We can easily imagine that no organization would make its data available for interoperation without a guarantee that the information will be protected as required. The potential loss of control over the data made available for interoperation, and, possibly, the fear of compromising the security of all the information managed can definitely affect the development of shared infrastructure for real-world applications. Effective information sharing and interoperation can happen only if the holders of the different databases have assurance that access constraints on information they own or manage will be respected. The major requirements of integration/mediation efforts with respect to security issues can be summarized as follows:

- **Transparent access.** Users of the application should be able to access information from any or all of the sources in a uniform and consistent way. Specialized knowledge of the individual sources should not be required.

- **Autonomy.** Application users must be permitted access to information that they can access from any individual source. Furthermore, the sources must continue to function after integration as they did prior to integration.

- **Security.** Users of the application must be denied access to any information that they cannot access from any source individually. In other words, local security policies must be respected.

The satisfaction of these requirements is often complicated by the different protection characteristics of the component systems. Different systems may enforce different access control policies and/or require different constraints to be satisfied in order to allow access to data. This heterogeneity and the potential conflicts that may result need to be resolved to allow interoperation while compromising neither the autonomy nor the security of the individual components.

Previous proposals addressing secure interoperation issues framed the problem within a federated database context, where a global schema, possibly under control of a central authority, is defined on the local data sources [10, 12, 15]. Moreover, access control is generally assumed to be regulated by discretionary policies, where access decisions are taken with respect to authorizations stated by users. Although mandatory security has been investigated, and some interoperation aspects have been addressed [11, 21], no system providing the functionalities above has been presented.

In this paper we consider the problem of securing information upon interoperation in the context of a mediator-based distributed system where access to data is enforced by mandatory policies, and the security lattices at the different data sources may differ. Our goal is to allow data sources to interoperate and to make their data available to external applications in such a way that their autonomy and security are not compromised. By the assumption of a mediator-based architecture, our approach requires neither the definition of a global schema nor the existence of a global centralized authority. Each application can define its own local (virtual) database. Schema specification and security constraints are maintained locally at each source/application. The consideration of mandatory-based policies, in contrast to discretionary ones, seems appropriate since it avoids the burden of dealing with identities. Indeed, while it appears to be appropriate and practically feasible to require applications and sources to know about security classifications with which information they store or use may be labeled, it appears rather impractical to impose similar requirements on identities. The necessity of maintaining information about users' identities managed by other sources or applications, which can also become meaningless at the local level, would compromise flexibility and introduce a considerable management burden. A mandatory policy requires instead only knowledge of the security lattices and the specification of how the levels in them relate. Data classification and clearance assignments can be performed locally, according to the requirements of autonomy and security. Moreover, once the relationships between the different policies have been specified, new databases (or tables in them) can be made available to applications without need for further specifications.

Our approach to secure interoperation is based on the use of a mediator [23] and wrappers. A wrapper for each source provides a uniform data model interface to the system, resolving the problem of data model heterogeneity. The wrapper also provides mappings between security levels of application subjects and security levels of the local lattice. The mediator interfaces the applications and the data source wrappers. It provides for the definition of mappings between the application and the data source security lattices, and for ensuring their consistency. It also provides tools for the definition of a *virtual* application schema in terms of the schemas of local data sources. The mediator provides interoperation by processing every application query for global access control, query processing, and access retrieval at the data sources. This paper describes how the schema and security specifications are stated at the data sources and, through the mediator, at the applications. We present the architecture of the mediator-based system and discuss the tasks of the different software components and their interaction during system operation. To make the discussion of the system more concrete, we present our approach in the context of a hypothetical secure information integration effort. The goal of the effort is to develop a trusted mediated application, called MedInfo, which integrates three separate and independent sources. Two of the sources, referred to as Clinic and Hospital, are multilevel secure (MLS) relational databases containing (actual) data from two units of a health care network. The third source is the well-known Medline medical research citations database. Medline was developed and is maintained by

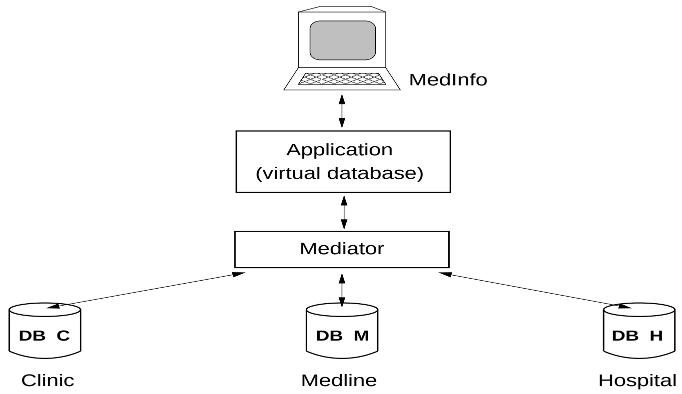


Figure 1. MedInfo application

the (United States) National Library of Medicine and is accessible via the World Wide Web. Each of the sources has its own schema, semantics, security policy, query language, and data model. While MedInfo has both a schema and a security policy, it has no data of its own, and thus can be viewed as a virtual database. The data for MedInfo are supplied by the three sources participating in the application. Figure 1 depicts the MedInfo application.

The remainder of this paper is organized as follows. Section 2 illustrates the specification of the data source and application security policies and of the mappings between them. Section 3 discusses the definition and labeling of the schema at the data sources. Section 4 illustrates how the virtual application schema is defined and how the security levels of the virtual relations are determined. It also illustrates how the relationships defining the application schema are used in query processing. Section 5 illustrates how the access control on application queries is executed by both the mediator and the data sources. Section 6 illustrates the system architecture, while Section 7 illustrates the content of the knowledge base core of the mediator process. Section 8 illustrates the different steps in the mediation process, and Section 9 gives an example of secure mediation of a query. Section 10 discusses related work. Concluding remarks appear in Section 11.

2. Security policy specifications

We assume that access control at each source and application is regulated by a mandatory policy. Mandatory policies govern the access by subjects to the information on the basis of classifications, or security levels, assigned to subjects (clearance) and to data objects. Levels are partially ordered according to a dominance ($\geq$) relationship, forming a lattice. With respect to information secrecy, access is regulated by the *no-read-down* and *no-write-up* principles, according to which a subject can read only information classified at a level dominated by the

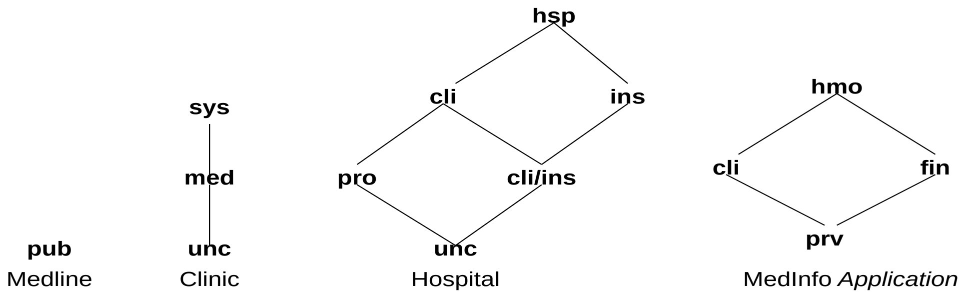


Figure 2. Security lattices

subject's level and write only information classified at a level dominated by the subject's level [1].

2.1. Application and source security lattices

At each source and application x, a security lattice $\mathcal{L}_x = \langle L_x, \geq_x \rangle$ is defined stating the security levels used at the source/application and the dominance relationships between them. The security lattice at a source specifies the classifications that can be assigned to the information stored at the source (and made available for interoperation) and the clearances that subjects accessing the source can be assigned. The security lattice defined at each application defines the possible clearances that subjects of the applications can assume and the possible classifications that can be assigned to the (virtual) relations of the application schema.

Figure 2 illustrates possible security lattices for the application and sources of our running example. The lattices are interpreted as follows. Medline is a publicly accessible source and thus has a trivial security lattice consisting of the single level **pub**, applied to all objects. The clinic uses a simple security lattice with three levels: **sys**, which is used to protect the most sensitive data in the database; **med**, which labels clinical data; and **unc**, representing unclassified data. The hospital employs a more elaborate lattice. The highest level is **hsp**. The next two lower levels, **cli** and **ins**, are used to label clinical- and insurance-related data, respectively. Level **cli/ins** represents the intersection of **cli** and **ins**, and reflects the need to provide access to certain data to both **cli**- and **ins**-classified subjects. Level **pro** corresponds to provider-sensitive data, and **unc** to unclassified data. The MedInfo application uses a lattice of intermediate complexity. The top level is **hmo**, with lower levels **cli** and **fin** representing the partition of data into clinical and financial categories. The lowest level is **prv** (Private) — the application does not support unclassified users.

2.2. Cross-lattice relationships

The security lattice defined at each application defines the possible clearances that subjects of the application can assume and the possible classifications that can be assigned to the (virtual) relations of the application schema. Users connecting to the application will have knowledge of these classifications only. They do not need any knowledge of the security lattices of the sources. Since the application provides access to the different data sources, to determine visibility of the application subjects on the data at each source, subjects' clearances must be mapped into corresponding clearances at the local sources. For this purpose the mediator requires, for each application, the specification of how the security levels of the application security lattice "map to" the security levels of each source that the application may need to access. Such mappings are specified by means of dominance relationships *from* the security levels of the application to the security levels of the *source*. More precisely, for each application A, the mapping is specified as a set of dominance relationships of the form $(l_i \geq l_j)$, where $l_i \in L_A$ is a security level in the application security lattice and $l_j \in L_{S_j}$ is a security level in the lattice of some source S_j. In the following, we refer to such relationships as *cross-lattice relationships*, in contrast to the lattice relationships (or edges) locally specified at the application or at the sources. Figure 3 illustrates an example of cross-lattice relationships between the lattice of the MedInfo application and those of the three data sources introduced in Figure 2. Cross-lattice relationships are represented by dashed arrows. In the following, we denote by Map_A the set of cross-lattice relationships specified for the security levels of application A. We use the notation $l_i \geq_x l_j$ to denote relationships specified within an individual lattice $\mathcal{L}_x$. We write $l_i >_x l_j$ if $l_i \geq_x l_j$ and $l_i \neq l_j$. We use the notation $l_i \geq l_j$ to denote a dominance relationship holding in an individual lattice (application or source) or between levels of the application and levels in a source either explicitly (i.e., belonging to Map) or because of the combination of the cross-lattice and lattice relationships. For instance, with reference to the lattices and cross-lattice relationships of Figure 3, **med**$\geq$**unc**, **hmo**$\geq$**sys**, and **hmo**$\geq$**unc**; also, **hmo**$\geq$**sys** $\in \mathsf{Map}_{\mathsf{MedInfo}}$.

Notice that the cross-lattice relationships specify which levels of the applications dominate which levels of the data sources and are unidirectional. No dominance relationship is specified between the levels of the sources and those of the applications. This is a characteristic of the problem under consideration where, given a subject connected at a security level in the application lattice, we need to retrieve all information visible to him, that is whose (source) level is *dominated* by the level of the subject. In the processing of queries from a specific application, only the cross-lattice relationships between the application lattice and the source lattices and the relationships in the individual lattices involved are considered. Indeed, these relationships are the only ones needed to determine the visibility of subjects submitting queries.

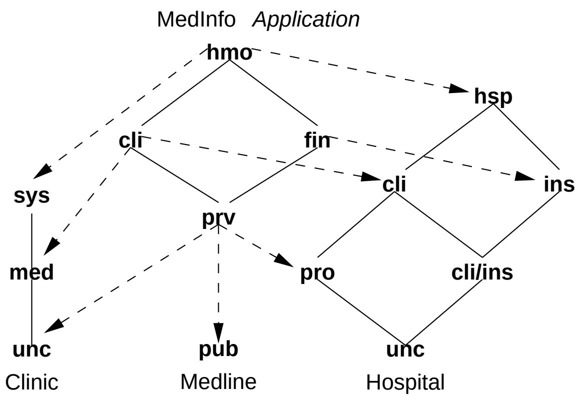


Figure 3. Security lattices and mappings

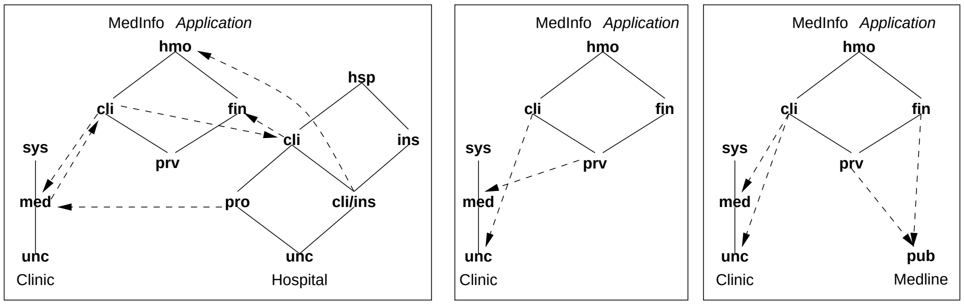
2.3. Correctness of the specifications

The mappings between levels of the application and levels of the sources are used to determine the (application) security level to be assigned to the virtual application relations and the (source) security levels to be assigned to a requesting application subject for local access control by the source. For these translations between levels to be correct, we require the cross-dominance relationships specified for each application to be consistent and absent from ambiguities and redundancies.

As already stated, cross-lattice relationships are unidirectional: from applications to sources. However, since each source could also serve as an application, dominance relationships between two lattices can be specified in both directions, leading to the possibility of cycles. In such cases, the mediator must guarantee consistency in the combination of different relationships. Within an individual security lattice, the consistency property requires that no two levels in the lattice dominate each other, that is, forming a cycle. Intuitively, a cycle would permit an illegal (from lower to higher levels) flow of information. Our consistency property naturally extends this requirement to the consideration of multiple lattices and cross-lattice relationships between them. Given a set of individually consistent lattices, consistency requires the combination of interlattice and cross-lattice relationships not to introduce any dominance relationships between levels in a lattice $\mathcal{L}_x$ that is not already present in the lattice $\mathcal{L}_x$ itself.

Intuitively, an inconsistency corresponds to a path of dominance relationships from a level l_i to a level l_j in a lattice $\mathcal{L}_x$, which is not dominated by l_i according to the order $\geq_x$ specified in the lattice. This concept is made clear by the following property.

PROPERTY 1 (CONSISTENCY) *For all lattices* $\mathcal{L}_x = \langle L_x, \geq_x \rangle$, *and levels* $l_i, l_j \in L_x : l_i \geq l_j \Longrightarrow l_i \geq_x l_j$.



(a) (b) (c)

Figure 4. An example of inconsistent(a), ambiguous(b), and redundant(c) relationships

Figure 4(a) illustrates an example of inconsistent specifications. The figure contains three sets of inconsistent specifications. The first inconsistency is due to the presence of the mappings **cli**$\geq$**cli** from MedInfo to Hospital and **cli/ins**$\geq$**hmo** from Hospital to MedInfo. Such mappings, together with the dominance relationshps within the lattice, would form a cycle of the dominance relationship involving two levels in a individual lattice. Such a cycle would allow **cli** subjects to indirectly acquire **hmo**-classified information. The second inconsistency is similar and is due to the presence of the mappings **cli**$\geq$**cli** from MedInfo to Hospital and **cli**$\geq$**fin** from Hospital to MedInfo. The combination of such mappings would imply a dominance relationship between **cli** and **fin** within the MedInfo lattice, thus indirectly allowing, through the mediation of Hospital, the flow of information from **fin** to **cli**, violating the policy of the application. Finally, the third inconsistency is due to the presence of the dominance relationship between **med** in Clinic and **cli** in Medline, between **cli** in Medline and **cli** in Hospital, and between **pro** in Hospital and **med** in Clinic. The combination of such relationships together with the interlattice relationship between **cli** and **pro** would allow a hospital subject classified **pro** access to **cli**-labeled data, clearly violating the policy of the hospital. Note that inconsistencies correspond to paths from a level to a level that it does not dominate (either greater or incomparable). For instance, in Figure 4(a) the existence of the two opposite dominance relationships between **cli** in Medline and **med** in Clinic is not to be considered an inconsistency. Their semantics is merely that the two security levels are equivalent.

The consistency property is the only constraint we need to ensure that no unauthorized information flow will occur. For the correctness of the security-level translations we also require cross-lattice relationships not to be ambiguous or redundant, as stated by the following properties.

PROPERTY 2 (NONAMBIGUITY) *For all applications A, sources S, and levels $l_i, l_j \in L_A$, $l_u, l_v \in L_S : l_i >_A l_j, l_u >_S l_v, (l_i \geq l_v) \in \mathsf{Map}_A \Longrightarrow (l_j \geq l_u) \notin \mathsf{Map}_A$.*

The nonambiguity property states that a level l_j that is dominated by a level l_i in the application cannot map, in a source, to a level that dominates a level to which l_i maps. Intuitively, the fact that l_i dominates l_j in the application means that l_i has visibility on more data than l_j, and such a situation should be respected in the mapping. The rejection of ambiguous specifications ensures that the schema classification and access control obey this principle. Figure 4(b) illustrates an example of ambiguous specifications where the dominance relationship between **cli** and **prv** in the application lattice appears reversed on the levels to which they map. According to such (ambiguous) mapping, lower-level **prv** subjects would have a larger visibility on the source data than higher-level **cli** subjects. Also, **cli** cleared users could augment their visibility by connecting at a lower level. Such a situation is obviously ambiguous. It is worth noticing that the mappings are ambiguous but in themselves not inconsistent since all the constraints stated by the interlattice and cross-lattice relationships can simultaneously hold. Note instead that it is completely legitimate for two levels in a dominance relationship in the application lattice to map to incomparable levels in a source lattice. For instance, in Figure 3 both **fin** $\geq$ **ins** and **pro** $\geq$ **prv** belong to Map_A. The meaning of such relationships is that subjects at level **fin** can view, at Hospital, information classified **pro** (through the mapping specified for **prv**), as well as information classified **ins**. It is also completely legitimate for two incomparable levels in the application lattice to map to two levels in a dominance relationship in a source lattice. For instance, with reference to Figure 2, **cli** of the application could be defined to dominate **cli** in Hospital, while **fin** could be defined to dominate **pro**.

Property 3 (Nonredundancy) *For all applications A and sources S the following two conditions must be satisfied:*
(i) For all $l_i, l_j \in L_A,\ l_u \in L_S : l_i >_A l_j, (l_i \geq l_u) \in \mathsf{Map}_A \Longrightarrow (l_j \geq l_u) \notin \mathsf{Map}_A$.
(ii) For all $l_i \in L_A,\ l_u, lv \in L_S : l_u >_S l_v, (l_i \geq l_u) \in \mathsf{Map}_A \Longrightarrow (l_i \geq l_v) \notin \mathsf{Map}_A$.

The nonredundancy property states that *(i)* two levels in a dominance relationship in the application cannot be defined to map to the same level in a source lattice, and *(ii)* each level in the application lattice cannot map to two levels in a source lattice such that one of the source levels dominates the other. In other words, redundancy happens when there is both a direct and an indirect dominance relationship between a pair of security levels. Figure 4(a) illustrates an example of ambiguous specifications. There is a cross-lattice edge from both **fin** and **prv** in the application to **pub** in Medline. Although one of the relationships is implied by the other, and therefore holds, the direct edge from **fin** to **pub** is redundant. If it is intended that both **fin** and **prv** should map to **pub**, the cross-lattice edge from **fin** to **pub** is redundant and should be removed. Analogously, the mapping from **cli** to **unc** is redundant given the mapping **cli** to **med**. Like ambiguity, redundancy does not cause any improper information flow but it would introduce unnecessary complications and ambiguity in the translation process.

Note that the nonredundancy constraint does not forbid two levels of the application to map *indirectly* to the same level, which is completely legitimate. For instance, in Figure 3, both **fin** and **prv** in the application would translate to **pub** in

Medline. It is also possible for an application level to directly or indirectly dominate more than one level in a source lattice if the levels are incomparable; that is, none of them dominates the others. This may happen, for instance, when a security level at the application dominates two levels at a source but it does not dominate their least upper bound. For instance, with reference to the lattices of Figure 2, level **fin** could be defined as dominating (mapping) to both **pro** and **ins** in Hospital.

It is worth noticing that the cross-lattice relationships are not required to be complete. That is, it is not necessary that an explicit dominance relationship be specified for each level in the classification lattice of the application and some level of each data source. For instance, in Figure 3 no direct relationship is specified for the **fin** level of the MedInfo application with respect to levels of the classification lattice of Clinic. The relationships between **fin** and the levels of Clinic are derived from those specified for **prv**. This situation reflects the fact that application subjects cleared at **fin** and those cleared at **prv** have the same access privileges to the information in source Clinic (both map to **unc**). In this example, a dominance for **fin** wrt the levels of Clinic exists because of the cross-lattice relationship specified for **prv**. Note, however, that since no completeness property is required, it can happen that a security level in an application does not dominate any level in a data source. This is completely legitimate and reflects a situation where a security level of the application does not have any access to the information maintained at the data source. For instance, consider a situation like that in Figure 3 but where there is no edge between **prv** of MedInfo and **unc** of Clinic. Such a framework reflects a situation in which subjects cleared **fin** and **prv** are not allowed to see any information and should then be given no access to data of Clinic (queries by them will be blocked by the mediator and will not even be passed to Clinic). Note also that some levels of a source may have no levels in the application that dominate them. This reflects a situation where the source does not make available for interoperation information classified at certain levels. For instance, consider again the situation of Figure 3 and suppose that there is no edge between **hmo** of the application and **sys** of Clinic. In this case, no level at the application has visibility on information classified at **sys** at the Clinic, which can then be accessed only through direct connection by cleared users.

Before closing this section, we note that in a framework where all sources and applications have a bottom element intended to denote "public" access (i.e., all subjects are cleared for access to some data), completeness of the specifications can be enforced by the automatic specification of a dominance relationship between the minimal element of the application lattice (when no relationship is explicitly stated for it) and the minimal elements of the lattices of the sources. For the sake of generality, we do not make this assumption and require the specifications to adhere only to the properties stated earlier.

3. Source schema definition and labeling

Each source provides and maintains some data. We make no assumptions on the underlying data models used at the sources. For instance, with respect to our sam-

Labels	Rules	Application_query		Resource_query
{**prv**}	R_1:	patient, patient_private	$\hookleftarrow$	H.patients[**unc**]
{**cli**}	R_2:	patient	$\hookleftarrow$	C.patients[**med**]
{**prv**}	R_3:	physician	$\hookleftarrow$	C.physicians[**unc**]
{**prv**}	R_4:	physician,ph_specialty	$\hookleftarrow$	H.providers[**pro**]
{**cli**}	R_5:	event	$\hookleftarrow$	C.events[**med**], C.physicians[**unc**]
{**cli**}	R_6:	event	$\hookleftarrow$	H.providers[**pro**], H.events[**cli**]
{**cli**}	R_7:	visit	$\hookleftarrow$	C.location[**med**], C.visit[**med**]
{**cli**, **fin**}	R_8:	visit	$\hookleftarrow$	H.visit[**cli/ins**]
{**prv**}	R_9:	research	$\hookleftarrow$	M.publication[**pub**], M.author[**pub**]
{**cli**}	R_{10}:	research	$\hookleftarrow$	C.events[**med**], C.physicians[**unc**]
{**cli**}	R_{11}:	research	$\hookleftarrow$	H.providers[**pro**], H.events[**cli**]
{**fin**}	R_{12}:	ins_physician	$\hookleftarrow$	H.providers[**pro**], H.insurance[**ins**]

Figure 5. Sample MedInfo application and resource relationships

ple integration effort, Clinic and Hospital are MLS relational databases containing (actual) data from two units of a health care network. The medical research citations database is instead accessible through the World Wide Web via an HTML interface. A wrapper at each source provides a uniform relational interface between the source and the mediator.

We also do not make any assumptions on the labeling policy applied at each source. This allows the data sources to maintain complete autonomy. For example, some sources may classify information at the table level, while other sources may provide classification at the attribute or element level. Regardless of the specific classification policy applied at a source, each object (relation produced by the wrapper) at the source has associated a security level. In case of finer-grained classification, the level assigned to each object will be the greatest lower bound of the security levels of the information in the object, as is common practice in mandatory security policies [9]. As described in Section 5, the security level associated with the object is the level that will be considered by the mediator to filter access requests to the object. Finer-grained access control (within the object) will be enforced at the source itself.

4. Application schema definition and labeling

4.1. Application/resource relationships

At each application, a virtual database is defined using information collected from the different sources. The relationships between application and resource schemas are described in a general form that relates application queries to resource queries. This means that a relationship description may establish a correspondence between an arbitrary join of relations in the application schema and an arbitrary join of relations in a resource schema. In other words, relationships between application relations and resource relations are many to many. The relationship descriptions are created by a person with knowledge of both the application and the database resource with which it is related. Once created, the descriptions are stored in the knowledge base for use in the transformation process, as described in Section 8.

The relationships are expressed by rules of the form

$$application\_query \hookleftarrow resource\_query$$

where *application_query* and *resource_query* are essentially bodies of Datalog queries formulated in terms of the application and resource schemas, respectively, and all the predicates in *resource_query* refer to relations of a single source. These rules can be understood as "the query *application_query* in the application schema *corresponds to* the query *resource_query* in the resource schema (for some particular resource)", or, in other words, "*resource_query* provides *some* answers for *application_query*". Figure 5 illustrates an example of relationships between the MedInfo application schema and the three database sources. The relationships are shown in an abstract form in which the attributes of the relations have been omitted. The meaning of the security levels appearing between square brackets and reported in column Labels is described in Section 4.3. The first rule in Figure 5 states that a query on the relation H.patients in the hospital database provides answers for a query on relations Patient and Patient_private in the application. Answers for a query on application relation Patient are also provided by a query on clinic relation C.Patients, as stated by the second rule. The detailed form of the first two relationship rules would appear as follows:

R_1: $\mathsf{patient}(Id,Ln,Fn,Adr,Dob,Ms,Sex)$, $\mathsf{patient\_private}(Id,Ssn,Race) \hookleftarrow$
$\mathsf{H.patients}(Id,Ln,Fn,Adr,Ssn,Dob,Ms,Sex,Race)$

R_2: $\mathsf{patient}(Id,Ln,Fn,Adr,Dob,Ms,Sex) \hookleftarrow \mathsf{C.patients}(Id,Ln,Fn,Adr,Dob,Ms,Sex)$

Note that more than one relation can appear in the right-hand side of a rule — rule R_5 is one example. The intuitive meaning of this situation is that *application_query* contains information produced by the *join* of the relations appearing in *resource_query*. Also, different rules may be defined with the same relation on the left-hand side (*application_query*). For instance, rules R_5 and R_6 both produce information for queries on virtual relation event. In this case the semantics is that the information in such relation is the *union* (in relational database terms) of the information collected through the different rules. Note the distinction between joining and unioning of the source relations.

The application/resource relationships are used by the mediator to translate, by means of a process known as *query folding*, queries submitted to the application into queries on the sources. Intuitively, each relationship defines a possible way in which a query can be folded by the mediator and therefore a possible way in which some results can be obtained from the data sources, as illustrated in the following section.

4.2. Query folding

Before proceeding with the application schema labeling and system operation, it is useful to briefly illustrate how the resource relationships are used in the mediation process.

The mediator processes queries with a query folding approach. Using the resource descriptions stored in the knowledge base, query folding attempts to rewrite a given application query (expressed in the mediation language) into queries on available database resources. The folding process involves finding a suitable resource replacement for each literal in the body of the application query. If such replacements can be found, and if the conjunction of the replacement literals is consistent with the semantics of the original query, the rewritten query is called a *complete folding*, or, simply, a *folding* of the original query. The folding algorithm used in the prototype mediation system is guaranteed to find all complete foldings of a given application query. This means that the mediator will extract from the participating databases the maximum number of answers for the query. We illustrate the basic concept of the query folding approach through some examples. We refer the reader to [7, 18, 5] for details.

Consider the first two relationship rules of Figure 5, whose detailed form has been given in Section 4.1, and consider the following simple query on the (virtual) relation Patient in the MedInfo application:

```
SELECT last_name, first_name FROM Patient
```

After translation into the mediation language, the query becomes

$$\mathtt{q1}(Ln, Fn) :- \mathsf{patient}(Id, Ln, Fn, Adr, Dob, Ms, Sex)$$

which is then transformed into the two different queries

$$\mathtt{q1a}(Ln, Fn) :- \mathsf{H.patients}(Id, Ln, Fn, Adr, Ssn, Dob, Ms, Sex, Race)$$
$$\mathtt{q1b}(Ln, Fn) :- \mathsf{C.patients}(Id, Ln, Fn, Adr, Dob, Ms, Sex)$$

to be forwarded to Hospital (query q1a) and Clinic (query q1b).

Note that foldings need not have such a simple correspondence between query and individual resources. To illustrate consider the following extension of the query above:

```
SELECT last_name, first_name, race
FROM Patient, Patient_private
WHERE Patient.id = Patient_private.id
```

with translation into the mediation language

$$\mathtt{q2}(Ln, Fn, Race) :- \mathsf{patient}(Id, Ln, Fn, Adr, Dob, Ms, Sex),\ \mathsf{patient\_private}(Id, Ssn, Race)$$

This query requests the attribute *Race* from the logical relation Patient_private. Examining the rules for Patient, we see that this additional information is maintained in the hospital database in relation H.patients, but the clinic database has no corresponding information. Nevertheless, if the two databases contain some common patient information, the information missing from C.patients might be filled in by H.patients. This possibility is reflected in the queries produced by the folding algorithm:

$$\begin{aligned}&\mathsf{q2a}(Ln, Fn, Race) :-\\&\quad \mathsf{H.patients}(Id, Ln, Fn, Adr, Ssn, Dob, Ms, Sex, Race)\\&\mathsf{q2b}(Ln, Fn, Race) :-\\&\quad \mathsf{C.patients}(Id, Ln, Fn, Adr, Dob, Ms, Sex),\\&\quad \mathsf{H.patients}(Id0, Ln0, Fn0, Adr0, Ssn0, Dob0, Ms0, Sex0, Race),\\&\quad Id = Id0\end{aligned}$$

In the first folding (query q2a) the relation H.patients alone is used to provide answers for q2. The second folding (query q2b) will attempt to supply answers through a join of H.patients with C.patients. Based on the resource descriptions, queries q2a and q2b are the only foldings that may provide answers to query q2.

Having clarified how rules are used in the query processing to produce the view of the virtual relation for the application subject, we are now ready to discuss the security labeling of the application schema.

4.3. Application schema labeling

The security levels of the application schema are derived from the security levels assigned to the objects at the sources. Classification of the application schema is obtained by classifying relationship rules as follows. For each rule, the levels of the relations appearing in the right-hand side of the rule are considered. According to the semantics of the cross-lattice mapping, a (source) relation classified at level l_i should be accessible to all subjects classified at a level l_j in the application that dominates l_i (according to the lattice and cross-lattice relationships). Level l_i of the source relation must then be translated into the set Al_i of minimal levels in the application lattice that dominate l_i. Note that Al_i is a set, since multiple minimal levels may exist that dominate l_i. For instance, in Figure 3, the set of minimal levels in the application that dominate **cli/ins** in Hospital is {**cli**,**fin**} (meaning the relation can be accessed by subjects classified **cli** or above as well as by subjects classified **fin** or above). If the rule contains only one relation, this is the set of levels to be assigned to the rule. If the rule contains more than one relation, we need instead to determine the application levels that have visibility on *all* the relations in the right-hand side. These are the levels that dominate a level in *each* Al_i, for all l_i occurring in the right-hand side. Such levels are computed by calculating the cartesian product of all Al_i and substituting each set in the cartesian product with the least upper bound of the levels appearing in the set. Nonminimal elements are then eliminated from the set of levels so produced. To illustrate, consider rule R_5 in Figure 5, where the levels of the source relations are reported within square brackets, and the lattices and cross-lattice relationships of Figure 3. Relations C.events and C.physicians are labeled **med** and **unc**, which map to sets {**cli**} and {**prv**}, respectively. Their cartesian product contains only one element, which is {**cli**,**prv**}, whose least upper bound is {**cli**}. Consider instead rule R_8. Relation

H.visit is labeled **cli/ins**, which, as already discussed, translates in the application to the set of levels {**cli**,**fin**}. Finally, consider rule R_{12}. Level **pro** translates to {**prv**}, while **ins** translates to {**fin**}. The cartesian product contains only the set {**prv**,**fin**} whose least upper bound is **fin**. Note that, since an application level can dominate some levels in the source but not their least upper bound (see Section 2.3), it is important to first translate the source levels into application levels and then compute the least upper bound. Computing the least upper bound prior to translation could overclassify information, forbidding subjects to access information for which they have sufficient clearance. For instance, consider again rule R_{12}. The least upper bound between the levels appearing in the right-hand side is **hsp** in the source, which would translate to **hmo** in the application, thus not allowing **fin** visibility to the information produced by the rule. The security levels associated with the rules are used at access control time to determine the dependencies that can be evaluated in the folding process and the queries to be forwarded to the sources for producing the information to be returned to a subject at a given level. In particular, each (virtual) relation may be expressed through different relationship rules; that is, it may appear on the left-hand side of different relationship rules. All such rules may have different classifications associated with them. This reflects the fact that not all subjects are cleared for all the data that can be retrieved through a query. The higher the subject's clearance the greater the number of rules applicable to the subject. For instance, with reference to query `q2` illustrated in Section 4.2 and the classifications of Figure 3, the folding algorithm will produce both queries `q2a` and `q2b` for subjects at level **cli** and **hmo**. It will produce only query `q2a` for subjects at level **prv** and **fin**.

5. Access control

Access control is enforced at both the application (mediator) and source levels. This "hybrid" approach, in contrast to mediator-only or source-only controls, ensures autonomy and security of the sources while avoiding unnecessary processing and forwarding of queries. In fact, with local control the sources will always control every request and possibly enforce finer grained restrictions, while with mediator control queries are filtered, and only the (local) queries that can potentially retrieve information for which the requesting subject has clearance are generated and forwarded to the local sources.

At the application level, the mediator controls access to the virtual relations by limiting the use of the rules that determine how an application query can be answered. (Note that since we address the problem of providing global visibility of information, we consider that only read accesses are requested at the application level, while write accesses are carried out locally at each source. This assumption reflects the real-world application of the kind of system under consideration.) Suppose that an application user connects to the system as a subject at level L and issues a query to the mediator. The query is formulated entirely in terms of the application schema, and hence references only the virtual relations of the application. For each relation referenced in the query, the mediator determines the set of rules on which the subject has visibility, that is, with at least one level dominated by the level of the subject. If the set of rules accessible by a subject for a given relation is empty, access to that relation is definitely prohibited, and the query is rejected. Otherwise, the mediator uses the accessible rules to attempt to answer the subject's query, according to the query folding process illustrated in Section 4.2. As an example, consider a subject with level **prv** who issues a query on relation research. Figure 5 contains three rules for research, the first labeled {**prv**}, and the second and third labeled {**cli**}. The mediator will use only the first rule in attempting to answer the query, since the last two rules should not be visible to the subject (**prv** does not dominate **cli**). Note, however, that access to rules does not, by itself, ensure that a query will be answered. Even if rules are accessible for each relation referenced in the subject's query, the mediator may still fail to find a way to answer the query [18]. In this case, the query is not rejected, but instead an empty set of answers is returned.

At each source, access control may be enforced at a finer level than is currently supported by the application, depending on the capabilities of the particular source. Any source query generated by the mediator from an application query is issued to the (wrapper of the) source and labeled with the security level of the subject who issued the application query. The wrapper translates the subject level into the corresponding levels for the source and forwards the query to the source. This translation consists of determining the set of maximal levels in the source lattice dominated by the subject's level. For example, suppose an application subject operating at given level **fin** issues a query, and that this query results in a query to the hospital database. There are two maximal labels in the hospital lattice dominated by **fin**: **ins** and **pro**. Hence, two access requests are issued to the source: one at **ins** and one at **pro**. Note that in case of rules joining relations at incomparable levels, single-level requests will be submitted on each of the relation and not on their join, which will be computed afterwards. This to guarantee a proper view on the data to application subjects whose level dominates incomparable levels involved in a join but not their least upper bound. For instance, with reference to a request by a **fin** subject on relation ins_physician (rule R_{12}), the request at level **ins** will return the data from H.insurance and the request at level **pro** will return the data from H_providers. Once data have been gathered, the join will be computed. Note instead that, according to the fact that joins are labeled with the least upper bound of the relations involved, none of the two levels would have been returned any information on a request on the join. This approach, which works well when the subject is restricted to the use of one level at the time, would not be appropriate in our context where an application level can map to multiple incomparable levels and should be therefore guaranteed the visibility on the union of the information visible to each of them. The computation of the join afterwards allow us to make use of the access control system at the source, working under the stated assumption, without compromising visibility of the information. Although application-level access control is sufficient to ensure that source-level queries resulting from an application query will not be rejected by the source, it is not strong enough to allow source-level access control to be bypassed. For example, a source may permit tuple-level labeling of data. In such cases, either the source must enforce this finer level of access control, or the mediator (or wrapper) must filter out answers based on the requesting subject's level and the labeling of the data. Autonomy, efficiency, and assurance considerations argue in favor of maintaining source-level access control.

6. The system architecture

The mediator system has been implemented and demonstrated using the MedInfo application [6]. The core mediator components, as well as the translators and wrappers, are implemented in ANSI C. The translators are modules that can either be operated as stand-alone programs or be linked directly with the core mediator components. Each wrapper is a stand-alone program that is called as needed by the mediator. The mediator itself runs as a single Unix process, and each database query issued by the mediator is managed by a subprocess of the mediator that calls the appropriate wrapper program. Communication between any wrapper process and its parent process is carried out using POSIX standard pipes to maximize portability of the mediator system. A limited amount of scripting code (e.g., Perl) is used to interface certain stand-alone components and for CGI processing. The two relational sources Clinic and Hospital are Trusted Oracle databases maintained on a Trusted Solaris platform. The Medline database is maintained separately (by the National Library of Medicine) and made available via the Internet.

The overall architecture is illustrated in Figure 6. The dotted areas delimit the the application, the mediator, and the data sources. The cylinders represent information maintained, while rectangles represent software components.

7. Knowledge base

The knowledge base component maintains information regarding schemas and security policies of participating databases and the relationships between the application and each of the participating databases. The query transformer, the query language/data model translators, and the security policy translators all rely on information maintained in the knowledge base. Thus, the knowledge base is central to the mediator. Population of the knowledge base with meaningful and consistent knowledge of the sources and their relationships to the application is critical to the construction of a successful mediated application. In a sense, an instance of the mediation architecture with an empty knowledge base can be viewed as a mediator "template" or "framework", which upon construction of the knowledge base becomes a mediator.

The knowledge base is populated and maintained by an administrator interacting with the knowledge base editor. The main activities supported by a knowledge base editor are

- Description of source schemas, language constraints, and representation structures

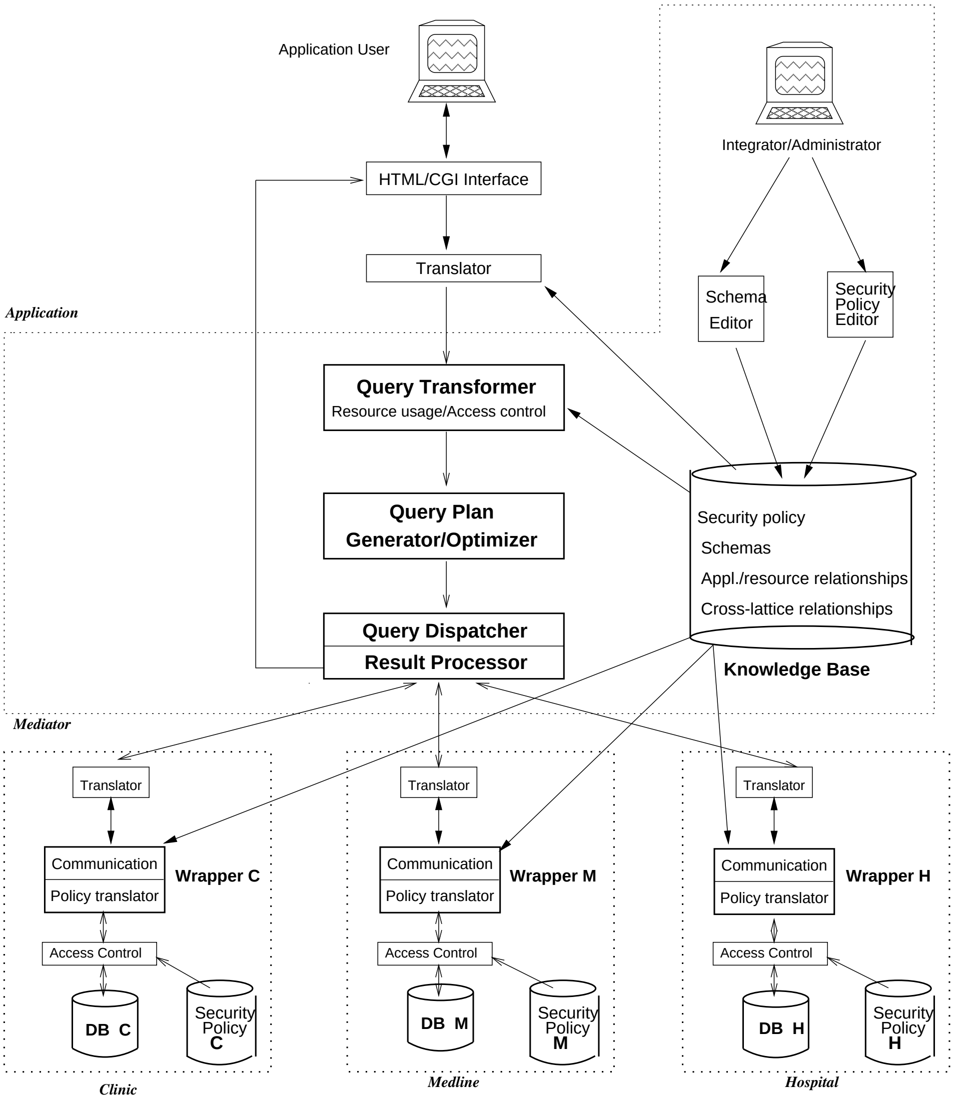


Figure 6. Mediation architecture

- Specification of relationships between source schemas, in the form of mappings that relate queries answerable by one source to queries answerable by another source or application

In addition to the knowledge required for semantic mediation, the knowledge base contains information on security policies and their relationships for trusted interoperation. A security policy editor is provided that enables the administrator to perform the following security-related activities:

- Description of security policies of the mediated application and data sources
- Specification of mappings between the security policy of one source and that of another source or application
- Identification and resolution of potential security violations that may result from interoperation

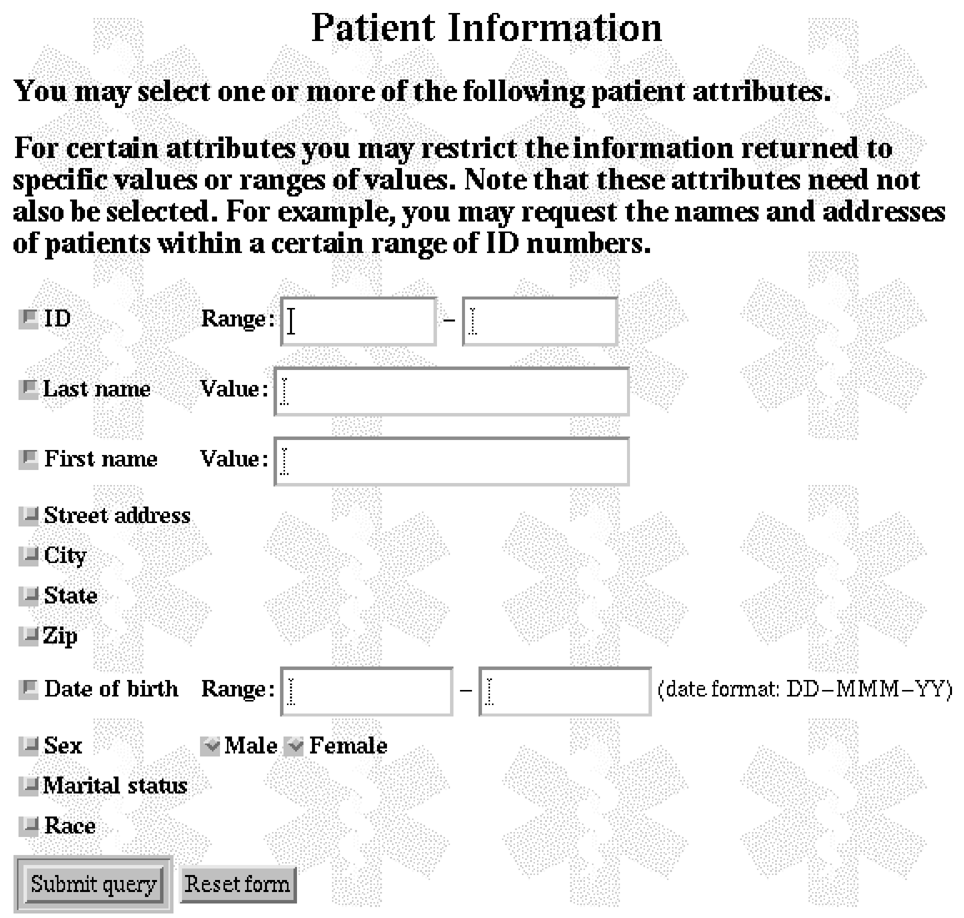
Perhaps the most important function of the security policy editor is the identification of potential security violations. The editor allows the integrator/administrator to specify security relationships one by one. After the addition of each relationship, the editor determines whether the added relationship would introduce a security violation, identifying also all relationships involved in the potential violation. The administrator can then decide to withdraw the relationship that causes the violation or remove one or more relationships until the violation is corrected. Only then is the addition of a new relationship permitted.

8. The secure mediation process

At a high level, the mediation process consists of the following steps:

1. Translate a subject's application query from the query language of the application (e.g., HTML) into the mediation language.
2. Transform the application query (now expressed in the mediation language) into (possibly multiple) target database queries (still in mediator language) including only relations for which the subject has sufficient clearance (mediator access control).
3. Generate and optimize a global execution plan for target queries.
4. Generate local query plans for target queries.
5. Execute query plans.
   (A) Translate target queries from mediator language to database language.
   (B) Issue target queries on databases. Wrappers at databases determine appropriate clearance levels for queries by invoking the security policy translator and pass them on for local access control.
6. Process results and return answers to the subject query.

The user interface simply provides a means for a user to issue queries in terms of the application schema and to view the answers to that query provided by the mediator. The HTML/CGI interface provides a graphical interface (via a Web browser) to the mediated application. The user logs in at a certain clearance level


Patient Information

You may select one or more of the following patient attributes.

For certain attributes you may restrict the information returned to specific values or ranges of values. Note that these attributes need not also be selected. For example, you may request the names and addresses of patients within a certain range of ID numbers.

ID Range: –
Last name Value:
First name Value:
Street address
City
State
Zip
Date of birth Range: – (date format: DD–MMM–YY)
Sex Male Female
Marital status
Race
Submit query Reset form

Figure 7. Patient query form

and then may query the system by entering SQL queries into a simple HTML fill-out-form interface like that illustrated in Figure 7. A CGI (Common Gateway Interface) script recasts the HTML query in SQL and forwards it to a translator. The user views results formatted as HTML tables.

The remainder of this section describes the main software modules participating in the secure query processing and their operation.

8.1. Query translator

The first stage in the mediation process is the translation of an application query into the mediation language. In the prototype system, an application query is expressed (after being recast from HTML) in a subset of SQL (restricted to conjunctive, or select-project-join queries). After translation into the mediation language, the user's query is forwarded to the query transformer, which is the primary module of the mediator core.

Translation is also used in a later stage of mediation, during the execution of queries generated by the mediator from an application query. Here the queries must be translated from the mediation language into the query language of the target database. Of the three databases mediated by the demonstration system, Clinic and Hospital use SQL as the query language, while Medline uses HTML as its query language. Thus, the prototype system includes modules for translating between the mediation language and SQL and for translating between the mediation language and Medline's HTML encoding. Normally, Medline is accessed via a Web browser by filling out an HTML form. Access to Medline from the mediator is achieved by generating an HTML query that is equivalent to the query generated by the form interface. Thus, Medline remains truly autonomous — its interface need not be modified to enable access from the mediator. Indeed, Medline would be unable to distinguish mediator accesses from Web browser accesses.

8.2. Query transformer

The query transformer attempts to rewrite a given query, formulated in terms of the application schema, into one or more queries that can be answered by one or more participating databases. To accomplish this rewriting, the mediator performs the query folding process described in Section 4.2 subject to restrictions enforced by the access control as discussed in Section 5. Using descriptions of the relationships between application and target database schemas and the labeling specified, query folding rewrites a query on the application schema into a set of queries on one or more database schemas, if possible. Each query in the resulting set of transformed queries is guaranteed to yield a subset of the answers to the original query. Furthermore, the set of transformed queries produced by query folding is complete, in the sense that the set will yield the maximum amount of information (from the participating databases) relevant to the original query.

8.3. Query plan generator/optimizer

If query transformation is successful, we have a set of queries to be issued to one or more databases. Some of the queries may be answerable completely by one database. For example, queries `q1a`, `q1b`, and `q2a` (from Section 4.2) are all single-database queries. Others may involve multiple databases, for example, query `q2b`, which involves both the hospital database and the clinic database. Individual queries involving multiple databases must be broken down, or decomposed, into subqueries that can be answered by individual databases. Some portions of a query may not be answerable by any database source. Such portions include joins between subqueries on different sources and the evaluation of selection conditions not supported by a source. These parts of the query must then be evaluated by the mediator.

8.3.1. Global query plan The details of how the set of transformed queries will be evaluated are represented in a *global query plan*. The global plan specifies the order of evaluation of individual queries in the set, how the individual queries are broken down, if necessary, into subqueries, and what processing remains to be done by the mediator. A query plan consists of a sequence of subplans. Each subplan is made up of a group of query plans. In principle, all the queries produced in the transformation stage can be evaluated in parallel, since each query is independent of the others. In practice, the ability to evaluate the queries in parallel depends on the capabilities of the database resources (since several queries may need to be issued to the same database source) and the communication, storage, and processing resources available to the mediator. In the prototype system, the degree of actual parallel evaluation is determined at the plan execution stage (query dispatch and result processing). Hence, the global plan is generated to represent the maximum amount of parallelism possible in principle. This implies that the global plan will consist of a single group of query plans, one for each query in the transformed set. Generation of an evaluation plan for each query in the transformed set involves the following steps: *(1)* Identification of the database source for each literal in the body of the query, *(2)* decomposition of the query into single-source queries, and *(3)* formulation of a query for mediator processing, if necessary.

Consider query `q2b` given in Section 4.2. Using resource information maintained in the mediator's knowledge base, the plan generator identifies the first literal in the query as a reference to the medical center database, while the second literal refers to the hospital database. The final literal, $Id = Id0$, expresses a join condition between relations from different databases and must be evaluated by the mediator. Thus, the query is decomposed into the following two single-source queries:

$$\begin{aligned}
&\texttt{q2b1}(\mathit{Id}, \mathit{Ln}, \mathit{Fn}, \mathit{Adr}, \mathit{Dob}, \mathit{Ms}, \mathit{Sex}) :- \\
&\quad \texttt{C.patients}(\mathit{Id}, \mathit{Ln}, \mathit{Fn}, \mathit{Adr}, \mathit{Dob}, \mathit{Ms}, \mathit{Sex}) \\
&\texttt{q2b2}(\mathit{Id0}, \mathit{Ln0}, \mathit{Fn0}, \mathit{Adr0}, \mathit{Ssn0}, \mathit{Dob0}, \mathit{Ms0}, \mathit{Sex0}, \mathit{Race}) :- \\
&\quad \texttt{H.patients}(\mathit{Id0}, \mathit{Ln0}, \mathit{Fn0}, \mathit{Adr0}, \mathit{Ssn0}, \mathit{Dob0}, \mathit{Ms0}, \mathit{Sex0}, \mathit{Race})
\end{aligned}$$

and a mediator-evaluated query to perform the join and return answers to the original query:

$$\begin{aligned}
&\texttt{q2b3}(\mathit{Ln}, \mathit{Fn}, \mathit{Race}) :- \\
&\quad \texttt{q2b1}(\mathit{Id}, \mathit{Ln}, \mathit{Fn}, \mathit{Adr}, \mathit{Dob}, \mathit{Ms}, \mathit{Sex}), \\
&\quad \texttt{q2b2}(\mathit{Id0}, \mathit{Ln0}, \mathit{Fn0}, \mathit{Adr0}, \mathit{Ssn0}, \mathit{Dob0}, \mathit{Ms0}, \mathit{Sex0}, \mathit{Race}), \\
&\quad Id = Id0
\end{aligned}$$

Queries `q2b1` and `q2b2` may be evaluated in parallel, while the evaluation of `q2b3` must wait for the evaluation of `q2b1` and `q2b2` to complete.

8.3.2. Optimization When a mediator system incorporates databases distributed over a wide area (e.g., the Internet), a major factor in the cost of mediation is the amount of network traffic between the mediator and databases. If the global query plan is generated naively, its execution may result in queries on individual databases that yield much more data, and hence result in much more network traffic, than necessary. In the prototype system, optimizations are performed that eliminate two principal sources of unnecessary network traffic, both of which result from the decomposition of multiple-source queries into single-source queries. Optimization is performed through projection and cross-product elimination. Cross-product elimination reconfigures the queries so to eliminate all unnecessary (i.e., not required for the computation of the result) cross products. Projection eliminates from the query all those attributes not necessary for the computation of the result so that only the data for necessary attributes are retrieved. A necessary attribute is one that is part of the projection (occurs in the head of the query) or participates in a join or selection condition. For instance, the optimized form of queries `q2b1` and `q2b2` of Section 8.3.1 is

$$\texttt{q2b1}(Id,Ln,Fn) \;:-\; \texttt{C.patients}(Id,Ln,Fn,Adr,Dob,Ms,Sex)$$

$$\texttt{q2b2}(Id0,Race) \;:-\; \texttt{H.patients}(Id0,Ln0,Fn0,Adr0,Ssn0,Dob0,Ms0,Sex0,Race)$$

since *Ln*, *Fn*, and *Race* are the projected attributes, while *Id* and *Id0* are involved in a join.

8.3.3. Local query plan In a heterogeneous environment, not all databases will necessarily have the same query evaluation capabilities. For example, some databases may not be able to evaluate certain built-in predicates (e.g., arithmetic comparisons) supported by other databases or by the application. In addition, some databases may have certain constraints on how queries may be formulated; for example, an attribute may be required to have a value supplied in the query (input only), or may not permit a value to be supplied in the query (output only).

To ensure that individual source queries can be evaluated, a local query plan is generated for each such query in the global plan. Each local query plan can be viewed as a refinement of a source query node in the global plan. In the simplest case, the capabilities of the source are sufficient, the query meets the requirements of the source, and hence no refinement is needed. When the query contains (built-in) predicates not evaluable by the source, the query must be decomposed into a query evaluable by the source (if possible) and the remainder that must be processed by the mediator. If the local plan generator determines that the (possibly decomposed) query does not meet the constraints of the source, the mediator can avoid sending the query to the source and instead supply a null answer to the query. Note that, after local query plan generation, no local query optimization is attempted. Local query plans represent queries entirely answerable by a single database source, and thus optimization of the queries is left to the respective databases.

8.4. Query dispatcher and result processor

The plan execution component of the mediator consists of two subcomponents: a query dispatcher and a result processor.

The query dispatcher is responsible for issuing the individual queries contained in the global query plan to the appropriate database sources or to the mediator's internal query processor. The order of evaluation is specified in the global plan, and the query dispatcher may issue the queries in any manner that satisfies that order. In particular, queries that are grouped together in the plan for parallel evaluation may be evaluated in any arbitrary order, subject to the constraints of the database sources (for multiple simultaneous connections) and the mediator's own internal resources. In the demonstration prototype, the dispatcher issues grouped queries in an arbitrary sequential order. Before any database query can be issued, it must be translated from the mediation language into the database's query language. Using information supplied in each local query plan, the dispatcher identifies the appropriate translator and calls it. The dispatcher then sends the translated query to the source specified in the plan. In the case of a mediator query, no translation is required, and the query is sent directly to the mediator's internal query processor.

The result processor is responsible for combining the answers returned from database sources and formatting the processed answers for return to the user (the issuer of the original application query). The processing necessary for combining the answers to individual database queries is specified by the query plan in the form of mediator queries. Recall that this processing includes computing joins of relations from different databases as well as the evaluation of built-ins that the query processors of some databases may not handle. Thus, the mediator contains a query processor capable of evaluating select-project-join queries. For return of the results to the application user, the mediator can be configured in either of two ways. In one configuration, the result processor computes the union of all answers to the application query and returns them as a single set. In the alternate configuration, the mediator returns the answers to each query in the set produced by query transformation (folding) separately, along with an indication of the source(s) of the information.

8.5. Wrappers and local access controllers

Each database incorporated into a mediated system must have a wrapper module with which it communicates with the mediator. The purpose of a wrapper is to accept queries from the mediator, forward these queries to the query processor of the database, accept answers back from the database, and return these results back to the mediator. In a trusted environment, a wrapper must also provide a security policy translator to ensure consistent and meaningful use of security information in the mediation process.

Local access control is enforced independently by the source access control system. As already discussed, we do not make any assumption on the local access control system, apart from the fact that a mandatory (lattice-based) policy is considered and that a security level is assigned to each object made available for interoperation. For the correctness of possible joins between data returned from multiple sources (see Section 8.3.1) we require no polyinstantiation to appear in the result returned from a local query. This is consistent with the fact that the application subject is returned the data he can access according to his clearance but with no label attached. Consideration of polyinstantiation and its management at the global level would require a more sophisticated and complex mapping management across lattices, which, however, does not seem needed for the kind of applications under consideration.

Note that the assumption of the use of a lattice base policy allows also for sources that do not enforce any access control, since these can be represented as a lattice with a single node that is dominated by the bottom element of the application lattice. Source Medline of our application is an example of this.

9. An example of query securely mediated

We now illustrate an example of a secure query mediation process. Suppose that a user connects to the application as a subject at level L. He then issues query Q to the mediated application via the "HTML Interface" (Figure 6). For concreteness, suppose that L is **cli**, the lattices and the relationships between them are as illustrated in Figure 3, and that Q is a query requesting research information on hypertension. Q would proceed as follows:

1. Query Q is passed from the HTML Interface into a translator that converts Q, expressed in HTML, into Q', expressed in the mediation language. For our example query, Q' is expressed as a query on relation research.

2. From query Q' (with level **cli**), the query transformer uses the knowledge base to determine a mediated query Q'_m. In general, Q'_m may be a *set* of queries to sources C, M, and H. Each query in Q'_m remains at security level L. For the example query Q', there are three relevant relationship rules (Figure 5) for research. The transformer uses all three rules, since the query's level, **cli**, dominates the only level of all of them. Query Q'_m is a union of three queries: one, Q'_M, is a join on M.publication, and M.author; another, Q'_C, is a join on C.events and C.physicians; and the third, Q'_H, is a join on H.providers and H.events.

3. Mediated query Q'_m is passed to the query plan generator/optimizer, which computes an evaluation plan P for the component queries of Q'_m. Each of the three queries resulting from transformation in the example is a single-source query with no dependencies on the others. A simple plan allows Q'_C, Q'_M, and Q'_H to be evaluated simultaneously at sources C, M, and H, respectively.

4. Plan P is sent to the query dispatcher, which sends Q'_C, Q'_M, and Q'_H to the translators for sources C, M, and H, respectively.

5. Queries Q'_C, Q'_M, and Q'_H are translated, respectively, into Q_C, Q_M, and Q_H, which are then forwarded, respectively, to wrappers C, M, and H.

6. Wrapper C translates security level L into security level L_C for source C forwards Q_C to source C at security level L_C, resulting in data D_C, which is passed back to the translator. Wrappers M and H do likewise for Q_M and Q_H, respectively. In particular, L_C is **med**, L_M is **pub**, and L_H is **cli**.
7. Answers D_C, D_M, and D_H are translated back into the mediation language, resulting in answers D'_C, D'_M, and D'_H, which are sent to the result processor.
8. The result processor computes the union of answers D'_C, D'_M, and D'_H, and forwards it back to the subject.

10. Related work

Previous work on providing secure interoperation has mainly been performed within the context of federated database systems, characterized by the fact that a global schema, possibly under the control of a certain authority, can be assumed, and on which access restrictions can be specified or derived. Also, access control has generally been based on discretionary policies. The different proposals have addressed the various problems in this context, such as management of global vs. local identities and their authentication [12], enforcement of different policies by the sources [13], definition of a global database and top-down derivation (from the global to the local level) of authorizations [22], and specification and bottom-up derivation (from the local to the global level) of authorizations and consideration of administrative privileges [10]. All these approaches are clearly inapplicable in our context. Multilevel security issues, always within the federated database context, have been addressed in [16, 21]. In [16] all sources are assumed to use the object-oriented data model and to apply the security orderings (i.e., the lattice at all sources is the same). Security is carried out via association of security levels to messages exchanged between the sources upon interoperation and to objects upon migration (relocation) from one source to another. In [21] sources represent a horizontal partition of a global DBMS, and their security lattices are all subsets of a predefined totally ordered set. Their approach to query processing is therefore not applicable in our context.

The problem of combining security specifications of different interoperating systems while ensuring that no security breaches has been addressed in [11, 3]. In particular, [11] considers generic security specifications expressed in terms of permissions and restrictions. They characterize properties that must be satisfied for the composition not to compromise security and study the complexity of ensuring their satisfaction. The work in [3] specifically considers combination of mandatory policies, where the goal is to define an ordered set combining different security orderings in such a way that a set of (cross-set) constraints among them is satisfied. Cross-set constraints are specified in both positive ($l_i \geq l_j$) and negative ($l_i \not\geq l_j$) form. The paper characterizes the problem of combining specifications and illustrates algorithms for the computation of the global order with a logic programming approach and a graph-based approach. The advantages of considering a mediator-based architecture, where no global ordering needs to be computed, and a simpler form of constraints (positive form) suffices, making the problem simpler in our context.

Specifically targeted to providing security in a mediator-based interoperation architecture is the work in [2], where access control is based on credentials or roles. However, the paper addresses mainly aspects of authentication and secure communication with emphasis on protocols and encryption and anonymity enforcement. Credential- and role-based controls are also addressed in [26] in the context of a generic distributed infrastructure. Such controls are complementary to mandatory lattice-based controls considered by us. Another approach to providing security under interoperation [25, 24] involves dynamic checking and possible sanitization of answers (based on content) before they are returned to the subject. At present, this approach relies on manual intervention for sanitization. Nevertheless, this approach can provide complementary security services within our architecture (6), and can be viewed as an additional component positioned between the result processor and the user interface.

11. Conclusions

We have presented a system that allows multilevel secure data sources to share their data and make them available to external applications in such a way that autonomy and security of each data source are respected. We have illustrated how the different schema and security constraints are specified at the data sources and at the applications, and how application queries are processed for access control and security constraint enforcement. We have also illustrated and discussed the architecture of the system we have built to provide secure interoperation.

The architecture proposed in this paper provides the basis for secure interoperation and could be extended in several directions. The mediator and the wrappers can be enhanced to consider non-mandatory policies. For instance, discretionary policies, possibly group-based rather than identity-based, could be considered. The access control modules could be enhanced to support management of credentials. In such contexts administrative issues regulating the specification and enforcement of access restrictions at the global level will need to be investigated. The problem of managing conflicting and redundant information, or access restrictions, from different sources should also be investigated.

Acknowledgments

This work was performed while all the authors were with SRI International and was supported in part by National Science Foundation under grant ECS-94-22688 and by DARPA/Rome Laboratory under contracts F30602-94-C-0198 and F30602-96-C-0337.

References

1. D.E. Bell and L.J. LaPadula. Secure computer systems: Unified exposition and Multics interpretation. Technical report, The Mitre Corp., 1974.
2. J. Biskup, U. Flegel, and Y. Karabulut. Secure mediation: Requirements and design. In Sushil Jajodia, editor, *Database Security XII: Status and Prospects.* Kluwer, 1999.
3. P. Bonatti, M.L. Sapino, and V.S. Subrahmanian. Merging heterogeneous security orderings. In *Proc. 4th European Symp. on Research in Computer Security (ESORICS 96)*, Rome, Italy, September 1996.
4. K.S. Candan, S. Jajodia, and V.S. Subrahmanian. Secure mediated databases. In *Proc. 12th International Conference on Data Engineering (ICDE '96)*, New Orleans, Lousiana, February 1996.
5. S. Dawson, J. Gryz, and X. Qian. Query folding with functional dependencies. Technical report, SRI International, 1996.
6. S. Dawson. Optimization techniques for trusted semantic interoperation. Technical report, SRI International, November 1997.
7. S. Dawson and X. Qian. Query mediation for trusted database interoperation. In *Proc. 1997 DoD Database Colloquium*, San Diego, CA, September 1997.
8. S. Dawson, S. Qian, and P. Samarati. Secure Interoperation of Heterogeneous Systems: A Mediator-Based Approach. In *Proc. of the IFIP 14th International Conference on Information Security (SEC'98),* Vienna-Budapest, 31 August – 2 September", 1998.
9. D.E. Denning, T.F. Lunt, R. Schell, M. Heckman, and S. Shockley. Secure distributed data view (SeaView) – the SeaView formal security policy model. Technical report, SRI International, July 1987.
10. S. De Capitani di Vimercati and P. Samarati. Authorization specification and enforcement in federated database systems. *Journal of Computer Security*, 5(2):155–188, 1997.
11. L. Gong and X. Qian. Computational issues in secure interoperation. *IEEE Transactions on Software Engineering*, 22(1):43–52, January 1996.
12. D. Jonscher and K.R. Dittrich. An approach for building secure database federations. In *Proc. 20th VLDB Conference, Santiago, Chile*, 1994.
13. D. Jonscher and K.R. Dittrich. Argos — A Configurable Access Control Subsystem Which Can Propagate Access Rights. In *Proc. 9th IFIP Working Conference on Database Security*, Rensselaerville, New York, August 1995.
14. A.Y. Levy, A. Rajaraman, and J.J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of the 22nd International Conference on Very Large Databases (VLDB'96)*, pages 251–262, Mumbay, India, September 1996.
15. M. Morgenstern, T.F. Lunt, B. Thuraisingham, and D.L. Spooner. Security issues in federated database systems: Panel contributions. In C. E. Landwehr and S. Jajodia, editors, *Database Security, V:Status and Prospects, IFIP*, pages 131–148, 1992.
16. M.S. Olivier. A multilevel secure federated database. In *Proc. 9th IFIP Working Conference on Database Security*, pages 23–38, Rensselaerville, New York, August 1995.
17. Y. Papakostantantinou, S. Abiteboul, and H. Garcia-Molina. Object fusion in mediator systems. In *Proc. 22nd International Conference on Very Large Databases (VLDB'96)*, Mumbay, India, September 1996.
18. X. Qian. Query folding. In *Proc. Twelfth International Conference on Data Engineering*, pages 48–55, 1996.
19. X. Qian and T. Lunt. Semantic interoperation: A query mediation approach. Technical Report TR 94-02, SRI International, 1994.
20. A.P. Sheth and J.A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.
21. B. Thuraisingham and H.H. Rubinovitz. Multilevel security issues in distributed database management systems III. *Computers & Security*, 11:661–674, 1992.
22. C.Y. Wang and D.L. Spooner. Access control in a heterogeneous distributed database management system. In *IEEE 6th Symp. on Reliability in Distributed Software and Database Systems, Williamsburg*, pages 84–92, 1987.
23. G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, March 1992.

24. 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 60–72. Chapman & Hall, 1998.
25. G. Wiederhold, M. Bilello, V. Sarathy, and X. Qian. A security mediator for health care information. In *Proc. 1996 AMIA Conference*, pages 120–124. Journal of the AMIA, Washington, DC, October 1998.
26. M. Winslett, N. Ching, V. Jones, and Slepchin. Using digital credentials on the World Wide Web. *Journal of Computer Security*, 1997.