Secure Interoperation of Heterogeneous Systems: A Mediator-Based Approach

S. Dawson[1], *S. Qian*[2], *P. Samarati*[1]
[1]*Computer Science Laboratory, SRI International*
333 Ravenswood Ave., Menlo Park, CA 94025 USA
{dawson, samarati}@csl.sri.com

[2]*SecureSoft, Inc.*
275 Shoreline Dr., Suite 520, Redwood Shores, CA 94065 USA
sqian@acm.org

Abstract

This paper addresses the problem of ensuring protection in the interoperation of heterogeneous data sources. We present a system that allows data sources enforcing mandatory multilevel policies to interoperate and make their data available to external applications still maintaining autonomy and security. Sources can be heterogenous with respect to both the data model and the security lattices governing access control. Our approach is based on the use of wrappers and a mediator. A wrapper associated with each source provides a uniform data interface and mapping between the source's lattice and other lattices. The mediator processes global access requests by interfacing applications and data sources. We show how the relationships between the security lattices are stated and controlled for inconsistencies, and how global queries are processed by the mediator and by the local wrappers for both data retrieval and enforcement of security constraints. We also describe the architecture and operation of the system, and describe the tasks of the different components.

Keywords

Database security, interoperability, multilevel security, distributed systems.

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. This interoperation is often complicated by the different protection requirements 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. In this paper we consider this problem in the context of a 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. The major requirements of this integration/mediation effort 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.

Previous proposals address 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 [2, 4, 6]. Moreover, access control is generally assumed to be regulated by discretionary policies, where access decisions are taken with respect to authorizations stated by users. Mandatory security policies in distributed systems have been investigated, and some interoperation aspects have been addressed [3, 8]. However, no working system providing the functionalities above has been presented. In this paper we illustrate a system that provides interoperation between data sources enforcing mandatory policies. 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.

Our approach to interoperation, satisfying the above requirements, is based on the use of a mediator [9] 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 more concrete we present our approach in the context of a hypothetical information integration effort. The goal of the effort is to develop a trusted mediated application, called MedInfo, which integrates two separately maintained health-care databases, Clinic and Hospital, and a Web-based medical citations system, Medline. 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, each of which has its own schema, semantics, security policy, query language, and data model. 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 lattices and of the mappings between them. Section 3 illustrates how the virtual application schema is defined and how the security levels of the virtual relations are determined. Section 4 illustrates

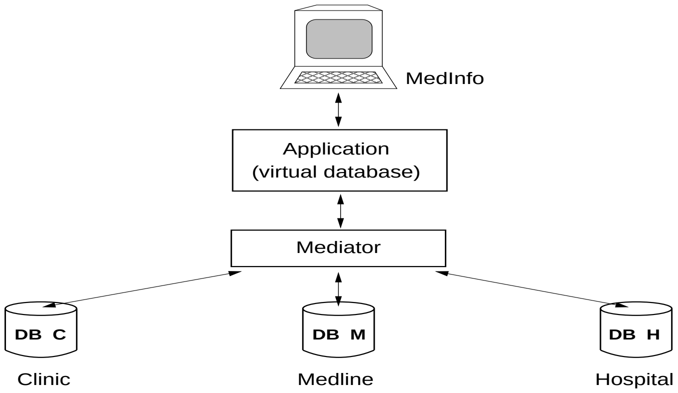

Figure 1 MedInfo application

how the access control on application queries is executed by both the mediator and the data sources. Section 5 illustrates the system architecture and the processing and execution of application queries. Concluding remarks appear in Section 6.

2 SECURITY SPECIFICATIONS

2.1 Local Source and Application Lattices

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 classification levels assigned to subjects and objects containing information. Levels are partially ordered according to a dominance ($\geq$) relationship, forming a lattice. With respect to information secrecy, access is controlled according to the *no-read-down* and *no-write-up* principles, according to which a subject can read only information with level dominated by the subject's level and write only objects with level dominating the subject's level [1].

At each source and application, a security lattice is defined stating the security levels with which information stored at the source is classified and the dominance relationships between them. Figure 2 illustrates possible security lattices for the application and sources of our running example. The semantics of the dashed arrows connecting them is explained in the next subsection. The illustrated lattices are interpreted as follows. Medline is a publicly accessible source and thus has a trivial security lattice consisting of the single label **pub**, which is 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 subjects using both **cli** and **ins** classifications. 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.

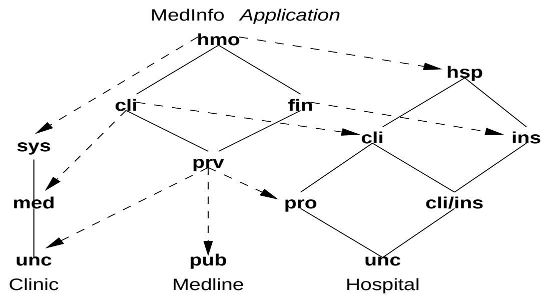

Figure 2 Security lattices and mappings

2.2 Application Security Specifications

Like sources, each application has associated a security lattice defining the security levels with which (virtual) information at the application can be classified and with which users can be cleared. A user connecting to an application can enter the system as a subject with any security level dominated by his/her clearance, among those appearing in the application lattice. Since the application provides global access to the different data sources, it is necessary to establish a mapping between the security levels used by the application and the security levels at the different data sources. Each application defines such a mapping by specifying *dominance* relationships between security levels in the application lattice and security levels in the lattices at the sources. Figure 2 illustrates possible dominance relationship between the security levels of our application and the three data sources considered. Cross-lattice dominance relationships are indicated by dashed arrows. Intuitively, the existence of an arrow from a level L_1 in the application to a level L_2 in the source lattice implies that L_1 *dominates* L_2 ($L_1 \geq L_2$) and, therefore, subjects classified L_1 by the application should be given read access to information classified L_2 at the source. For instance, the fact that the application level **cli** dominates the clinic level **med** indicates that subjects connected at level **cli** at the application should have visibility of information labeled **med** at the clinic.

2.3 Cross-lattice Consistency

For the correctness of the access control and labeling policy, it is important that the dominance relationships between the different lattices be consistent. By consistent we mean that both the following conditions are satisfied:

1. the relationships must not imply any information flow from higher to lower levels in any individual lattice,
2. the relationships must be nonambiguous.

Intuitively, a violation of type 1 corresponds to a cycle involving at least one lattice edge in the dominance relationships. In the mapping of Figure 2 such violations do not occur, because mappings are specified in just one direction: from the application to the sources. Recall, however, that each data source can potentially serve as an application, and vice versa. Hence, lattices can be connected by dominance relationships in opposite directions, leading to the possibility of cycles. For instance, consider the lattices and relationships illustrated in Figure 2, and suppose that a dominance relationship is specified between **med** in Clinic and **hmo** in MedInfo. Such a relationship introduces an inconsistency, because **med** both dominates and

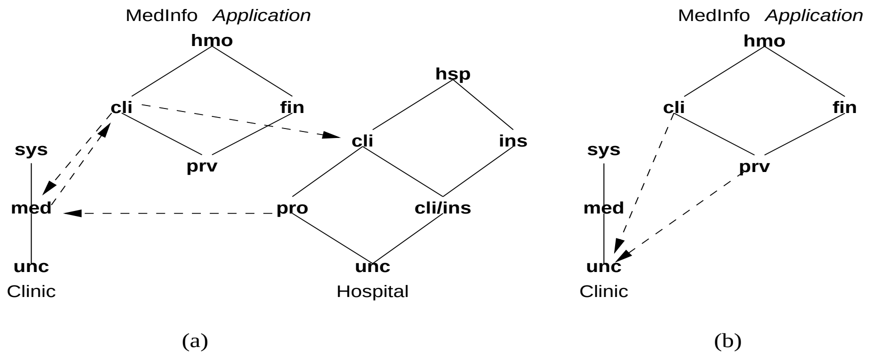

Figure 3 An example of inconsistent (a) and ambiguous (b) relationships

is dominated by **hmo**, but the two levels cannot be considered equivalent. Cross-lattice inconsistencies can also be caused by relationships involving more than two lattices. Such inconsistencies are less obvious, and their detection requires the evaluation of all the cross-lattice relationships in the system. Figure 3(a) illustrates an example of inconsistency involving three lattices. There is a 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. It is easy to see that the combination of these cross-lattice relationships, together with the dominance relationship between **cli** and **pro** in Hospital, introduces an inconsistency. The existence of such an inconsistency, under interoperation, would permit a hospital subject classified **pro** access to **cli**-labeled data, clearly violating the policy of the hospital. Note that inconsistencies correspond only to cycles that involve at least one lattice edge. For instance, in Figure 3 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.

A violation of type 2 happens when there is both a direct and an indirect dominance relationship between a pair of security levels. Figure 3(a) shows an ambiguous set of mappings. There is a cross-lattice edge from both **cli** and **prv** in the application to **unc** in the clinic. Although one of the relationship is implied from the other one, and therefore holds, the direct edge from **cli** to **unc** presents an apparent ambiguity in the mappings. If it is intended that both **cli** and **prv** should map to **unc**, the cross-lattice edge from **cli** to **unc** is redundant and should be removed.

In our system, insertion of cross-lattice relationships is controlled and allowed only if no inconsistencies would be introduced. In the remainder of this paper we therefore assume that the considered cross-lattice relationships are consistent.

3 SCHEMA DEFINITION AND LABELING

3.1 Source Schema

Each source provides and maintains some data. We make no assumptions on the underlying data models used at the sources, which can be completely heterogeneous. As discussed in Section 5, a wrapper associated with each source provides a uniform relational interface between the source and the mediator system. In addition, we make no assumptions regarding 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 stored at the source will have associated a security level. In case of fine 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 [5]. As it will be described in Section 4, the security level associated with the object is the level that will be considered by the mediator to filter access request to the object. Finer-grained access control (within the object) will be enforced at the source itself.

3.2 Application Schema Definition

At each application a virtual database is defined using information collected from the different sources. At a high level, the relationships between application and resource schemas are described in a form that relates application relations to resource queries. The relationships are expressed by rules of the form *relation* $\hookleftarrow$ *resource_query*, where *relation* is a (virtual) relation at the application level and *resource_query* is a conjunctive query expressed in terms of the schema of exactly one data source. The intuitive meaning of the rule is that *relation* contains information produced by the *join* of the relations appearing in *resource_query*. These rules can be understood as "the query *resource_query* provides *some* answers for a query on *relation*. Different rules may be defined for the same *relation*. In this case the semantics is that the information in *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. If two relations appear together in the right hand side for the same rule for a given application relation, they are joined to obtain the data in the relation. If they appear, instead, in different rules they are unioned. This distinction is crucial for understanding the labeling of application relations. Figure 4 illustrates an example of application rules defining the virtual relations of the MedInfo application. For simplicity, relationships are shown in an abstract form in which the attributes of the relations are omitted. For example, the first rule states that a query on the relation C.patients in the clinic database provides answers for a query on relation patient in the application. The meaning of the security labels is described next.

3.3 Application Schema Labeling

We have already discussed how, at the interface level with the system, each object in a data source is visible as a relation with an associated security level. The security classification of each virtual relation R defined at the application level is derived from the security levels of the relations referenced by R. More precisely, each virtual relation R has associated a non-empty set, possibly singleton, of security levels in the application lattice. This set of security levels is obtained through labeling of the rules as follows. For each rule, the least upper bound (lub) L of the security levels of the relations appearing in the right hand side of the rule is computed. Since all the relations in the right hand side of a rule belong to one source, and therefore use levels in the same security lattice, this level is well defined and can be easily determined. The set of security levels associated with a rule is the set containing the minimal elements of the security levels of the application lattice that

Labels	Application		Resource
{**prv**}	patient	$\hookleftarrow$	C.patients[**unc**]
{**prv**}	patient	$\hookleftarrow$	H.patients[**unc**]
{**prv**}	physician	$\hookleftarrow$	C.physicians[**unc**]
{**prv**}	physician	$\hookleftarrow$	H.providers[**pro**]
{**cli**}	event	$\hookleftarrow$	C.events[**med**], C.physicians[**unc**]
{**cli**}	event	$\hookleftarrow$	H.providers[**pro**], H.events[**cli**]
{**cli**}	visit	$\hookleftarrow$	C.location[**med**], C.visit[**med**]
{**cli, fin**}	visit	$\hookleftarrow$	H.visit[**cli/ins**]
{**prv**}	research	$\hookleftarrow$	M.publication[**pub**], M.author[**pub**], M.keyword[**pub**]
{**cli**}	research	$\hookleftarrow$	C.events[**med**], C.physicians[**unc**]
{**cli**}	research	$\hookleftarrow$	H.providers[**pro**], H.events[**cli**]

Figure 4 Sample MedInfo application and resource relationships

dominate L. To illustrate, consider the fifth rule in Figure 4, where the levels of the source relations are reported within square brackets, and the lattices and cross-lattice relationships of Figure 2. Relations C.events and C.physicians are labeled **med** and **unc**, respectively, and the lub of these labels is **med**. The set of minimal labels in the application lattice that dominate **med** is {**cli**}. Consider instead the eighth rule, relation H.visit is labeled **cli/ins**, which is also trivially the lub. The set of minimal labels in the application lattice that dominate **cli/ins** is {**cli, fin**}.

The security levels associated with a rule determine the visibility of the rule by application subjects. In particular, according to the no-read-down principle, a subject has visibility on a rule only if its level dominates at least one of the levels associated with the rule. This is equivalent to requiring the subject to dominate the levels of all the relations appearing in the right hand side of the rule. The security levels associated with the rules are used at access control time to determine the dependencies that can be evaluated, and therefore the source to be contacted, in producing the information in the relation to be returned to a subject a given level.

4 ACCESS CONTROL

Our system enforces access control at both the application and source levels. 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, i.e., 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. As an example, consider a subject with level **prv** who issues a query on relation research. Figure 4 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 [7]. 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 queries are issued to the source: one at **ins** and one at **pro**. 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.

5 THE SYSTEM ARCHITECTURE

The mediator system has been implemented and demonstrated using the MedInfo application. The core mediator components, as well as the translators and wrappers, are implemented in C and currently run on Unix workstations. The mediator itself runs as a single Unix process, and each database query issued by the mediator is managed by a subprocess that invokes the appropriate wrapper. In the MedInfo application, the Clinic and Hospital databases are both Trusted Oracle databases maintained on a Trusted Solaris platform. The Medline database is maintained separately (by the National Library of Medicine) and is accessed in real time on the World Wide Web.

The overall system architecture is illustrated in Figure 5. The dotted areas delimit the components at the application, at the mediator, and at the data source. The cylinders at each source/application represent the information maintained at the source/application. In particular, each data source maintains information about the local security lattice (Section 2.1), and database schema and its labeling (Section 3.1). Each application (which, recall, can itself be a source) maintains a local security lattice and its relationships to lattices at the different sources (Section 2.1), a virtual database schema, and the rules for query transformation (Section 3.1). The knowledge base component storing schema and lattice mappings plays a central role in the mediator, as all components enforcing query transformation and access control on information are maintained in the knowledge base. 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. The rectangles appearing in Figure 5 correspond to the different software components necessary for system operation. In the following we briefly describe each of them. We then describe how queries submitted by applications are mediated and processed at the data sources.

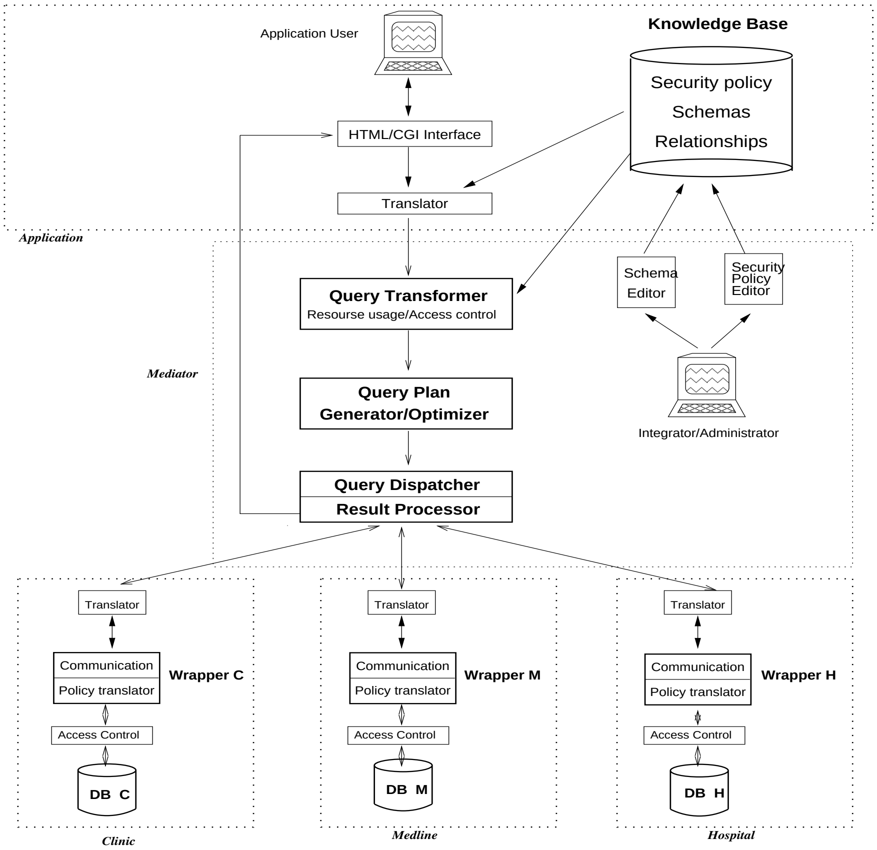

Figure 5 Mediation architecture

5.1 Components of the system

This section outlines system components, organized according to functionality.

1. **Knowledge Base Population and Data Model Translation**
   (a) **Security Policy Editor.** It assists in the following activities:
   - Specification of security policies of applications and data sources.
   - Specification of cross-lattice relationships.
   - Identification and resolution of potential security violations that may result from interoperation. Security relationships are specified one by one, and each is accepted only if it does not cause any inconsistencies.

   (b) **Schema Editor.** The main activities supported by the schema editor are:
   - Description of schemas, language constraints, and representations.
   - Specification of schemas relationships as mappings that relate queries answerable by a source to queries answerable by another source or application.

   (c) **Translators.** In the mediator all queries are expressed in a mediation language, which may not be the language used by the application or the sources. Translators at each application and source provide a mechanism for translating queries between these languages.

2. **Query Processing and Access Control at the Mediator**
   (a) **Query Transformer**
      - *Resource Usage.* Using the relationship rules stored in the knowledge base, the query transformer attempts to rewrite a given application query into queries on the sources. This rewriting process, known as *query folding* [7], involves finding a suitable resource to provide answers for each relation referenced in the application query. In general, there may be many resources that can provide partial answers to a query. Furthermore, many resources may combine to produce answers. The result of query transformation, then, is a *set* of queries representing all distinct ways in which available resources provide answers to the application query. The answer to the application query is the union of the answers of all the queries in the set.
      - *Access Control.* Access control on the virtual relations of the application is enforced by the mediator by restricting access to the relationship rules. The query transformer uses only the rules for which the application subject requesting access has visibility. Thus, the amount of information returned by the mediator in response to a subject's query depends not only on the schema relationships, but also the levels of the subjects and of the rules. For example, a subject at level **cli** will typically see more answers to a query on research than a subject at level **prv** will see for the same query.

   (b) **Query Plan Generator/Optimizer.** The result of query transformation is a set of queries to be issued to one or more data sources. Some of the queries in this set may be answerable by one data source alone. Others may involve multiple databases and must be decomposed into subqueries that can be answered by individual sources. Furthermore, some portions of a query, for example, joins involving different sources, may not be answerable by any source. Such portions must be evaluated by the mediator. The details of how the transformed queries will be evaluated are represented in a query plan. The plan specifies the order of evaluation of individual queries in the set, their decomposition, if necessary, and what processing is done by the mediator.

   (c) **Query Dispatcher/Result Processor** The final stage in the mediation process is the execution of the query plan. The dispatcher is responsible for issuing individual queries to the appropriate sources or to the mediator's internal query processor. The result processor receives and combines answers from the sources, and returns the collected answers to the requesting subject (the issuer of the original application query).

3. **Query Processing and Access Control at the Data Sources**
   (a) **Wrappers.** Each data source incorporated into the mediated system has a wrapper to interface with the mediator. The purpose of a wrapper is to accept queries from the mediator, perform security level translation, forward the query to the source, and return answers back to the mediator. For nonrelational sources, the wrapper also provides a relational interface to the mediator.

   (b) **Access Control.** The mediator performs access control at the query level by accepting or rejecting queries and by transformation using only rules for which the requesting subject has adequate clearance. In addition, access control is enforced at the data sources. This is necessary when, for example, the source supports finer grained classification (e.g., at the element level). Any filtering of results is performed by source-level access control. Thus, maintaining access control at the sources increases efficiency of the mediator, helps preserve autonomy of the sources, and helps in achieving high assurance, since the mediator is not responsible for fine-grained access control.

5.2 How Queries are (Securely) Mediated

We now illustrate the operation of the system upon submission of a query by the application. 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 5). For concreteness, suppose that L is **cli**, the lattices and the relationships between them are as illustrated in Figure 2, 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 4) 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, M.author, and M.keyword; another, Q'_C, is 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.

6 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 sources are respected. This work can be extended in several directions, some of which we are currently investigating. A first direction consists in the consideration of different security policies at the data sources; in particular different forms of discretionary policies can be considered. A second direction concerns the investigation of administrative policies for regulating accesses whereby data sources could, for instance, delegate access administration to the applications themselves. A third direction concerns the investigation and application of our approach in a federated context.

ACKNOWLEDGMENTS

This work was supported in part by DARPA/Rome Laboratory under contracts F30602-94-C-0198 and F30602-96-C-0337 and by the National Science Foundation under grant ECS-94-22688.

REFERENCES

[1] D.E. Bell and L.J. LaPadula Secure Computer Systems: Unified Exposition and Multics Interpretation Tech. Rep., The Mitre Corp., 1974.

[2] 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.

[3] L. Gong and X. Qian. Computational issues in secure interoperation. *IEEE Trans. on Software Engineering*, 22(1):43–52, January 1996.

[4] D. Jonscher and K.R. Dittrich. An Approach for Building Secure Database Federations. In *Proc. of the 20th VLDB Conf.*, 1994.

[5] T.F. Lunt, D.E. Denning, R.R. Schell, M. Heckman, and W.R. Shockly Secure Distributed Data Views, Vol.1–4 Tech. Report SRI-CSL, Computer Science Lab., SRI International, 1989.

[6] M. Morgenstern, T.F. Lunt, B. Thuraisingham, and D.L. Spooner. Security Issues in Federated Database Systems. In *Database Security, V: Status and Prospects*, pp. 131–148, 1992.

[7] X. Qian. Query folding. In *Proc. of the 12th International Conference on Data Engineering*, pp. 48–55, 1996.

[8] B. Thuraisingham and H. H. Rubinovitz. Multilevel Security Issues in Distributed Database Management Systems III. *Computers & Security*, 11:661–674, 1992.

[9] G. Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, 25(3):38–49, 1992.

BIOGRAPHY

Steven Dawson is a Computer Scientist in the Computer Science Laboratory at SRI International. His research interests include database interoperability and security, query optimization, deductive databases, and analysis and optimization of high-level programming languages.

Shelly Qian is a Principal Architect at SecureSoft, Inc. Before joining SecureSoft, she was a Senior Computer Scientist at SRI International. Her research interests include database security, semantic interoperation and integration of heterogeneous databases, and software architectures.

Pierangela Samarati is a Computer Scientist at SRI International. She is on leave from University of Milan where she is an Assistant Professor at the Computer Science Department. Her main research interests are database security, information system security, and access control models and systems.