Privacy-preserving Semantic Interoperation of Heterogeneous Databases

PENG LIU, PRASENJIT MITRA, and CHI-CHUN PAN
Penn State University

Two major challenges to enabling secure interoperation among web-information sources are resolving semantic heterogeneity across websites and maintaining the privacy of the data and metadata of organizations owning the websites. In this paper, we propose SACE, a novel, implemented middleware toolkit that enables privacy-preserving secure semantic access control and allows queries to be answered across websites despite their heterogeneity. SACE implements role-based access control after accounting for differences in roles across organizations. The novelty of SACE lies in its architecture that routes data and queries through different encrypted channels and reduces hacking vulnerabilities to a minimum. Although a middleware-based solution, in order to preserve the privacy of the metadata, SACE discloses minimum information to even the mediator encrypted ontologies, encrypted ontology-mapping tables and conversion functions, encrypted role hierarchies and encrypted queries. The toolkit also requires minimal changes to existing websites or their underlying databases. We show that despite using encrypted query and encrypted data translations, the toolkit provides acceptable performance.

Categories and Subject Descriptors: ... [...]: ...

General Terms: ...

Additional Key Words and Phrases: ...

1. INTRODUCTION

The world-wide-web enables easy communications like never before. Although the potential benefits of information sharing over the web can be significant, secure information sharing among diverse organizations faces two fundamental challenges:

(1) How to *automatically* resolve the *semantic heterogeneity* among information sources in a scalable way (in terms of the number of sources) such that the need for human involvement can be minimized, and lower cost and higher efficiency can be achieved.

(2) How to handle the corresponding *privacy* concerns of the organizations involved in information sharing. (We will illustrate such privacy concerns shortly.)

School of Information Sciences and Technology, University Park, PA, 16802, {pliu,pmitra,cpan}@ist.psu.edu

To the best of our knowledge, there exists no work that handles both these challenges satisfactorily.

Based on how the semantic heterogeneity among information sources is resolved, we can classify existing information sharing schemes as either *syntactic-level* information sharing or *semantic level* information sharing. To illustrate, let us assume that two organizations A and B have two different information sources about the same subject domain (e.g., intelligence against terrorism). In most cases, if not all, the *data schema* of the two sources are (quite) different, since (a) the same attribute (of an entity) can be given different names by Organizations A and B (e.g., an attribute named 'salary' in Organization A may be named 'stipend' in B); (b) the same attribute name used by both A and B may mean different things (e.g., having different semantic scopes); (c) the same attribute used by A and B can use different measurement units (like 'dollar' and 'euro'). We denote such differences as "semantic heterogeneity" in general.

Syntactic-level information sharing resolves these semantic heterogeneity by a syntactic symbol-level mapping relation (usually implemented as a table) that explicitly tells which attribute *symbol(s)* used by A should be replaced by which attribute symbol(s) used by B when an employee of B wants to access some information from A, and vice versa. Syntactic-level information sharing can neither handle Challenge 1 well nor handle Challenge 2 well, although role mediation technology [Ahn and Mohan 2004] makes the (role-based) inter-organization access control job simpler. Since substantial human intervention is involved in mapping schemas, syntactic-level information sharing is not only expensive but also error prone, especially when an information sharing system involves a number of organizations. Furthermore, existing syntactic-level information sharing schemes may seriously jeopardize the *privacy* of the organizations involved; and such privacy concerns (and risks) may yield too many disincentives to make information sharing rewarding.

Here, the major privacy concern that an organization has is about the privacy of such sensitive metadata (or meta information) as how data are organized in the organization (e.g., data schema), how accesses are controlled in the organization (e.g., the internal access control policy and role hierarchy), and the semantics of the data used in the organization (e.g., the ontology). If a mediator trusted by both A and B generates and stores the schema mapping, this may cause substantial privacy loss especially if the mediator is compromised or is malicious.

To minimize the involvement of human beings, Dawson, Qian, and Samarati proposed the idea of mediator-based information sharing [Dawson et al. 2000]. Their work is based on manually-generated query-folding rules that are used to translate queries to resolve the semantic heterogeneity among sources. However, semantic level mediation does not inherently remove privacy vulnerabilities. In their system, although the data access is controlled using database access control policies, the information about the queries, the schemas of the databases, and the role lattices of the two organizations are all exposed to the mediator and any intruder that breaks into the mediator or a communication link of the system.

Preserving the privacy of the metadata enumerated above is important since these metadata may disclose substantial business secrets of an organization; and

may make it much easier for an attacker to infer unauthorized information (based on the information he is authorized to access and the metadata). Furthermore, preserving the privacy of the metadata enables an organization to gain more privacy. For example, this privacy enables organizations to expose aggregate information (of her data) while still preserving the confidentiality of the granularity of the stored data, i.e., whether just the average is stored or the entire data.

As indicated by the above discussion, existing information sharing schemes rely on a *fully trusted* and highly secure mediator to preserve the privacy of organizations. However, such an approach is not (very) practical, since (a) building a highly secure mediator is not only very expensive but also very difficult, if not impossible, because almost every host providing services could be hacked; (b) from the trust management point of view, such a continuous high trust requirement is very difficult to be satisfied and as a result such a mediator (third party) is very difficult to be recruited. (Fundamentally, the more trust you assume, the more vulnerable the system.) Nevertheless, the above discussion shows that preserving the privacy of the metadata while enabling semantic interoperation is a difficult problem, because, often, the technologies proposed for enabling semantic interoperation depend heavily on mediation based on the metadata.

In this work, we present SACE — Semantic Access Control Enabler — a novel semantic-level information sharing system. SACE is a secure and robust system that addresses the two challenges mentioned above. SACE, to our best knowledge, is the first framework that can address the two fundamental challenges.

SACE has several unique features, which are as follows:

⊙ *SACE performs semantic access control.* We refer to access control performed along with semantic translations of roles, queries, and data across information sources as semantic access control in the rest of the paper. Compared with [Dawson et al. 2000] in which manually generated rules are used, SACE is more automatic because it leverages the pre-defined semantics of constructs in ontologies, like class-subclass relationships, to establish semantic access control. The impact of SACE lies in enabling immediate and seamless access of data from remote sources to users of one organization, while satisfying the requirements and access policies of a second organization, without involving any human intervention at the time of processing queries.

⊙ *SACE preserves privacy of metadata.* Minimizing the trust requirements on the mediator is the key to make the information sharing system more resilient to (privacy) attacks. In SACE, even the mediator only gets encrypted queries and encrypted metadata tables and encrypted matching rules and functions to enable the semantic translation of the queries. This minimizes the trust requirements on the mediator. SACE uses a trusted expert initially to establish the semantic-mapping rules, but does not require the mediator, involved in the day-to-day mediation of queries, to be trusted (not to disclose information). Nevertheless, the unique "encrypted" query-mediation protocol of SACE ensures that each mediator can still correctly enforce the information sharing policy and resolve the semantic heterogeneity. To the best of our knowledge, SACE is the first system that can handle both of the two fundamental challenges (i.e., how to automatically resolve semantic heterogeneity and how to preserve privacy) satisfactorily.

⊙ *SACE has a scalable middleware-based architecture.* SACE has a scalable architecture. It scales seamlessly when additional sources are added. The only performance bottleneck might be that SACE deploys a single mediator, however, the mediator can easily be replicated to remove that impediment. Our experimental results show that the overhead of enabling secure interoperation is minimal and we achieved high throughput of the queries while using SACE.

The rest of the paper is organized as follows. In Section 2, we discuss a few preliminaries and assumptions that SACE is based upon. In Section 3, we show the system architecture of our system. We discuss the core techniques that SACE uses to achieve secure interoperation in Section 4. In Section 5, we analyze the security and privacy of SACE. In Section 6, we outline the related work and conclude in Section 7.

2. PRELIMINARIES

In this section, we introduce the setting where our solution can be employed and the assumptions that SACE operates under.

We assume that an end-user can use a browser client to access multiple websites maintained by different organizations. The websites sharing information with each other use the Internet (e.g., using VPN) to communicate with each other. These websites provide the information only to authorized users typically using dynamically generated webpages. The end-user has to login to the site belonging to the end-user's "home" organization — local organization where the user has direct access — by using some means of authentication like an username and a password.

Each organization enforces its own RBAC policy. A website has an associated ontology, a role-lattice showing the lattice of roles of users authorized to access the website, and a table listing authorized users and their access permissions.

2.1 Private Ontologies Expressing Metadata

We assume that associated with each information source is an ontology that specifies the relationships among the terms used in the database underlying a website. All terms used in the database schemas — table names and column names — are included in the ontology associated with the database. Organizations might want to keep their associated ontologies confidential and not publish them.

A requestor of data can interpret the data returned only by using metadata elements. For example, given a table or a column of data, the requestor needs to know what data the table or column corresponds to and what the semantics of the data is. However, in our setting, the metadata can be used only within the organization but cannot be disclosed to other organizations. This is a major challenge that SACE overcomes by translating queries and translating data so that they conform to local schemas and other metadata.

2.2 Information Sharing and Access Control

Before an organization can share information with another, it must set up a policy on what information can be shared and with whom. We refer to this policy as the *information sharing policy* of the organization.

In order to enforce access control according to an organization's information sharing policy, each underlying database has access control rules stored in an au-

thorization table. Current databases use several access control methods. One of the most popular access control methods is RBAC (Role-based Access Control) [Ferraiolo and Kuhn 1992]. A database administrator assigns each user a set of roles and grants each role permissions to a set of tables or columns of tables. Typically roles correspond to sets of users with similar characteristics or functions. We assume that each organization enforces its own RBAC policy. The organization uses an authorization table to maintain the organization's syntactic access control policy, where each syntactic authorization is a 3-tuple (subject, object, authority). Here, a subject is a role identifier; an object is a table or column identifier; and an authority can be a combination of read, write or delete. The users of organization A (B) know the schema of A (B)'s databases, but do not know the schema of B (A)'s databases.

For the current discussion, we assume that there are no negative rules and all the rules are positive. However, our framework can still operate in the presence of negative access control rules by blocking access to data at the website responding to a data request.

2.3 Our Middleware-based Solution

Objective: The objective of SACE is to enable users of organization A to access B's data using queries written against A's database schema without violating B's access control policies and without exposing the schemas and the data of the databases.

Solution: In this work, we take a middleware-based approach. For simplicity, we assume each information sharing policy only concerns two organizations, although multi-organization information sharing policies are certainly possible.

3. SYSTEM ARCHITECTURE

We present two alternative architectures for SACE: the *middleware-centric architecture* is shown in Figure 1; the *mini-middleware architecture* is shown in Figure 2. Here, (a) each architecture has two aspects: the *online* aspect, which shows how an inter-organization query is processed in runtime, and the *offline* aspect — the initial processing that takes place in preparation for the online aspect. (b) Also, for simplicity, we show the offline aspect in the mini-middleware architecture. (c) In both architectures any component not belonging to any organization belongs to the Mediator (where the middleware is running) which is located at the partially trusted third party's site. (d) For clarity the communication links between the Mediator and the two organizations are not drawn.

3.1 Middleware-Centric Architecture

First, we show how the architecture works. Second, we overview its privacy preserving features. To make our presentation clear, we separate the semantic access control aspect and the privacy preserving aspect, although the implementations of these two aspects are actually seamlessly integrated with each other. Third, we talk about the pros and cons of this architecture.

SACE is a middleware system that requires almost no changes to be done on the legacy systems of any organizations involved. SACE can typically be implemented as a web service (i.e., the Mediator) hosted by a partially trusted third party. The goal of the offline procedure of SACE is to translate the syntactic access

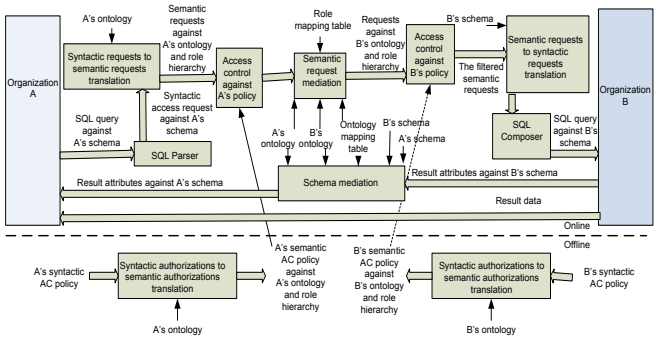


Fig. 1. The Centralized Middleware Architecture of SACE

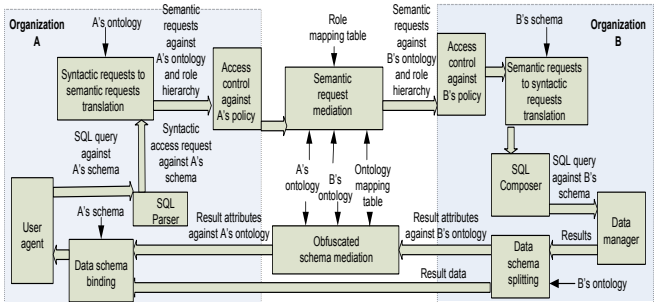


Fig. 2. The Minimum Middleware Architecture of SACE

control policy of each organization to a semantic access control policy against the organization's ontology and role lattice.

The online aspect of SACE shows how a data access request across websites is automatically mediated in a way transparent to both websites. To illustrate, let us assume an employee of Organization A needs some information from B. In this section, we outline the steps SACE takes to answer queries across websites and discuss the core techniques in detail in the next section. In Step 1, since the employee does not know B's database schema, the employee's SQL query is written against A's schema. In Step 2, a SQL parser is used to "decompose" the query into several syntactic access requests that will be automatically translated into some semantic access requests in Step 3 via the syntactic to semantic request translation procedure. In Step 4, these semantic requests are checked against A's semantic access control policy (to make sure that no employee can get some external data that violate the local access control policy). Note that this semantic policy is prepared offline.

However, at this stage these requests are expressed with A's ontology and role lattice, and they cannot be directly processed by Organization B. Hence, in Step 5 these requests are translated into several semantic accesses requests expressed with

B's ontology and B's role lattice via an algorithm called *semantic request mediation*. This algorithm uses mapping rules between terms in A's ontology and terms in B's ontology and the mapping between roles in A's role lattice and those in B's role lattice. In Step 6, these requests are checked against B's semantic access control policy. In Step 7, the filtered yet authorized semantic requests will be translated into some syntactic access requests against B's schema, so that in Step 8 the access requests can be restored to a SQL query again (by the SQL Composer). In Step 9, the SQL query is processed by B's DBMS without any security checking. However, the query results cannot be directly returned to A because they are not expressed against A's schema and the end-user in A can be confused about the meaning of the results. To make the results understandable to the user in A, and to ensure that the user in A will not know anything about B's schema (note that this is one of B's privacy concerns), and to minimize the Mediator's overhead, the query results will be "split" into two parts: the schema part and the data part. And the data part will be directly transmitted to Organization A, while the schema part will be sent to the Mediator so that the schema mediation algorithm can translate the schema of the returned results from B's schema to A's.

From the privacy preserving perspective, a key feature of SACE is that all the metadata stored and used at the Mediator, which include the database schemas and ontologies of both organizations, the ontology-mapping table and the role mapping table, are encrypted and the Mediator cannot decrypt them. In this way, SACE removes the requirement that the Mediator must be trusted not to disclose any sensitive metadata, and good privacy can be preserved even if the Mediator is hacked. Another important privacy preserving feature is that both the cross-organization queries and the results are encrypted in an end-to-end manner. Nevertheless, as we will show shortly in Section 4, the Mediator can still correctly enforce the access control policies and resolve the semantic heterogeneity.

Compared with the minimum-middleware architecture which we will present next, the pros of this approach are that this architecture is more transparent to each organization, and nothing special is needed when an organization issues an across-organization data access request. The cons are that (a) this architecture is more vulnerable to schema inference attacks, as we will show shortly in Section 5; (b) the Mediator may be so heavily loaded that it could become a performance bottleneck, especially when multiple organizations are served by one mediator.

Finally, it should be noticed that only the simplest cross-organization information sharing scenario is mentioned above. More complicated scenarios such as those involving data translation will be discussed in detail in Section 4.

3.2 Minimum Middleware Architecture

To make the presentation concise, we only address the differences from the middleware-centric architecture. First, the task of SQL parsing and the task of syntactic to semantic request translation are now done by a stub at Organization A. On the other end, the job of SQL composition and the job of semantic to syntactic request translation are now done by a stub at Organization B. Second, when the query results are returned, exact schema mediation is done in the middleware-centric architecture, but *obfuscated* schema mediation is done in the minimum middleware architecture. Accordingly, the data schema splitting module at Organization B and

the data schema binding module at Organization A need also be different.

In particular, to provide more privacy, the minimum-middleware architecture does schema obfuscation, which works as follows. When the query results are returned (from B to A) and when the schema part and the data part are split, the schema part will be "obfuscated" first in such a way that each schema term will be blended with a couple of "equivalent" terms in B's ontology. As a result, when the Mediator receives the obfuscated terms, it can only perform mediation using the obfuscated schema, and accordingly, A will receive obfuscated terms against A's ontology. Next, the Data Scheme Binding component uses A's database schema to de-obfuscate the schema part.

From the privacy preserving perspective, since database schema information is no longer stored or used at the Mediator and since schema obfuscation is performed during the process of schema mediation, the hacker breaking into the Mediator can no longer precisely correlate (encrypted) ontologies and database schemas. As a result, even if the hacker can infer some terms in the ontology, he could not know exactly which term is part of a schema.

Compared with the middleware-centric architecture, the pros are that there is more privacy and the Mediator is more efficient. The cons are that this architecture is less transparent to each organization, and each organization needs to handle some business of semantic access control.

In summary, we treat the two architectures above as two alternative architectures that tradeoff primarily between privacy and transparency (to organizations).

4. CORE TECHNIQUES

In this section, we present the set of core techniques for the SACE system and demonstrate their uniqueness and merits.

4.1 Mapping Roles

The first task before semantic access control can be enabled is to map roles from one organization to another. Initially the security officer in an organization, say Org-A, decides manually which roles in a remote organization, say OrgB, can access its data and what permissions should be granted to those roles. Using this information, the security officer maps the roles in a remote organization are mapped to roles in OrgA's role lattice. For example, a role *Investigative_Agent* in the *FBI* role lattice can be mapped to the role *Field_Agent* in the *CIA* role lattice. If a role from OrgB cannot be directly mapped to OrgA's role lattice, OrgA's role lattice is augmented by adding new roles such that OrgB's roles can be mapped to it. By connecting Org-B's roles to OrgA's roles lattice, the advantages of RBAC are afforded to permitted users from OrgB.

We assume that the role mapping is *consistent*. A consistent role mapping is such that if it maps a role A in role lattice *R1* to a role B in role lattice *R2*, then any descendant role C of role A in role lattice *R1* must not be mapped to a role D in role lattice *R2* that is an ancestor of B. For example, we get an inconsistent mapping if we map *Controller* to *Field_Agent* and *Investigative_Agent* to *Supervisor* in the *CIA* role lattice for obvious reasons. In SACE, access permissions are not transitively transferred. That is, if a role, *W1.R1*, is mapped to *W2.R2* and *W2.R2* is mapped to *W3.R3*, where *W1*, *W2*, and *W3* are three different websites with

separate access control policies, role *W1.R1* does not automatically gets mapped to *W3.R3* and would not automatically be granted access to the resources accessible by *W3.R3*.

After the security-officer has created the role-mappings, the mediator performs the following pre-processing to augment the role-mapping table. Consider the case where a role A in an organization does not have a matching role in another organization. However, recall that a role has access to all resources to which its sub-roles have access. Therefore, the mediator augments the role-mapping table by following all paths down the lattice starting at A. For each path the mediator chooses the role, X, highest on the path such that X has a mapping to a role Y in the other organization, and adds a mapping from A to Y.

4.2 Semantic Mapping Rules

Other than role mapping, the other task that must be accomplished before semantic query answering across organizations can be enabled is establishing semantic mapping rules. Each pair of organizations employ a trusted expert who establishes the mapping rules between the ontologies of the two websites. The expert provides three types of mapping rules: (1) Binary Mapping Rules: These rules are contained in mapping tables containing triples. Each triple lists two concepts from two different ontologies and specifies the relationships among them. For example, an entry in a table may be the triple
(O1.Vehicle, O2.Automobile, OWL.SubClassOf) indicating
$O2.Automobile$ is a $SubClass$ (as defined in the namespace OWL) of $O1.Vehicle$. (2) Mapping Functions: These functions are used to convert data values. For example, a conversion function between $Dollar$ and $Pound-Sterling$. Another example is when one organization may keep $first\_name$ and $last\_name$ as two attributes, but the other keeps both of them in a single attribute called $Full\ Name$. (3) Complex Mapping Rules: These rules show how a concept in one ontology representing a table can be expressed as a SQL query using concepts in another ontology. Complex Mapping rules have the following components: (i) A SQL Query using the terms in the responding organization's ontology: the requestor's query will be translated into this SQL query or its variant, (ii) A Table, T, in the requesting organization's ontology and used in the requestor's query, and, (iii) Binary Mapping Rules mapping all the results of the SQL query (in (i)) to attributes in table T. For example, a complex mapping rule is as follows:

```
Table: O1.LuxuryCar
Query: SELECT c.CarID, c.Make, c.Model, c.Year
         FROM car c, prices p
         WHERE c.ID = p.ID AND
               p.Price > 40,000
Mapping: O1.LuxuryCar.VehID = O2.Car.CarID
         O1.LuxuryCar.Mk = O2.Car.Make
         O1.LuxuryCar.Md = O2.Car.Model
         O1.LuxuryCar.Yr = O2.Car.Year
```

indicates a table *LuxuryCar* in a database associated with organization 1, can be expressed as a join between the tables *car* and *prices* in the database associated

with organization 2. Note that the attributes from the tables *O2.car* and *O2.price* may not be the same in the table *O1.LuxuryCar*. In reality, the attributes in the table *LuxuryCar* may be named *CarID, Mk, MD, Yr*. The correspondence between the attributes *CarID, Mk, MD, Yr* in the table *O1.LuxuryCar* and the attributes *CarID, Make, Model, Year* in the table *O2.Car* respectively must also be stated as entries in the mapping table for the rule to be correctly interpreted.

Expanding Mapping Tables Like in the role-mapping tables, after an automated mapping tool or an expert has expressed the mapping table, the mediator performs some pre-processing to "complete" it by adding the "inferable" relationships. We show two such inference rules below:

```
Mapping\_Table(A,B,R),Equivalent(A,C)
  => Mapping\_Table(C,A,R)
Mapping\_Table(B,A,``SubClassOf"),SubClassOf(A,C)
  => Mapping\_Table(B,C,``SubClassOf")
```

Note that these rules are inferable using the semantics of "Equivalent" and "SubClassOf'. For example, if "O1.Vehicle" maps to "O2.Conveyance", and "O1.Vehicle" is equivalent to "O1.transportVehicle", but no relationship has been established between "O1.transportVehicle" — perhaps due to errors of omission by the expert — the mediator establishes an entry in the mapping table between "O1.transportVehicle" and "O2.Conveyance".

We now proceed to explain the main tasks that must be accomplished in order to answer queries across organizations.

4.3 Schema Obfuscation and Semantic Translation

As indicated before, we assume that associated with each website, there exists an ontology. All the terms used in a database, associated with a website, to name tables and attributes must have a term that it is mapped to in the ontology. After, an end-user logs on to his local website and poses a query in SQL using the local schema, the query must be sent to the mediator for forwarding to the responding database. The requestor may not want to send the responding database the query as posed using the terms used in the requestor's database nor might the requestor know the schema terms in the responder's database. Recall that the mediator contains the ontology mapping table based on the requestor's and the responder's published ontologies and can only accept queries that use terms from the requestor's ontologies. Therefore, the requestor has to translate the term used in the local schema to a term in the requestor's ontology.

This query translation step provides for two advantages:

(1) Privacy: The terms used in an organization's database are not transmitted in any query.

(2) Extensibility and Maintenance: The requestor has the freedom to add any new terms without requiring a change in the encrypted mediated ontology table.

The first advantage preserves the privacy of the metadata even when the transmitted query may be intercepted by a hacker and decrypted. The second advantage

provides an enormous advantage related to the maintenance of the encrypted ontology tables and autonomy and extensibility of the requestor's database. As long as the newly added term can be mapped to the requestor's published ontology, there is no need for any change in the encrypted mediated ontology. Recall that the system is the most vulnerable when the encrypted ontology mapping tables are being constructed because at that point the expert has to examine the ontologies in their clear-text format. The mapping between the terms used in the requestor's database and those in the requestor's ontology and establishing the translation rules among them is handled within the organization and can be performed securely without any information leak unless the requestor's organization has been infiltrated by an insider personnel in which case a lot more than the metadata is compromised.

For example, given a query `select price from ourAutos;` where *ourAutos* is the table in organization A, the client-side SACE stub, performs a semantic translation, by looking up an internal ontology that indicates that the term *ourAutos* maps to the term *Automobiles* in the external ontology that is published to the mediator. Therefore, the query is rewritten to `select price from Automobiles;` and sent to the mediator.

4.4 Query Processing at the Mediator

After the query has been rewritten by using the terms in the requestor's ontology, the rewritten query is sent to the mediator. The mediator parses the query, identifies the tables, and attributes in the query. It tags the attributes with the tablename of the table in which the attribute appears. Attribute tagging is necessary to disambiguate between two attributes that have the same name but appear in two different tables. For example, given a query `select price from Vehicle;`, SACE parses the query and extracts the schema elements $Vehicle$ and $Vehicle.price$.

4.4.1 *Translation Using Binary Mapping Rules.* The mediator then performs *query translation*. We use a modified version of the query folding algorithm initially proposed by Qian et al [**?**]. It looks up the mapping table to find a mapping table entry for $Vehicle$ and $Vehicle.price$. Let us say the following entries are found:
$(Vehicle, Car, SubClass)$, $(Vehicle, Boat, SubClass)$,
$(Vehicle, Conveyance, Equivalent)$,
$(Vehicle.price, Conveyance.MSRP, Equivalent)$,
$(Vehicle.price, Car.MSRP, Equivalent)$, and
$(Vehicle.price, Boat.MSRP, Equivalent)$. If equivalent terms for the terms in the query are found in the mapping table, the mediator uses each equivalent term to create a rewritten query. In our example, the rewritten query is: `select MSRP from Conveyance;`.

4.4.2 *Translation Using Mapping Functions.* For attributes that must be converted before the data can be sent to the requestor, the mediator rewrites the query by using the expert-provided functions. For example, the requestor sends the query `select Price from rentalPrice;` where the rentalPrice is sought in US Dollars, and the renting agency that quotes the renting price in UK Pound-Sterling. Assume that *Price* matches to Pr in the responding database's ontology, $O2$ and *rentalPrice* matches to the table *priceOfRental* and the expert provided the function D2P() that converts prices quoted in PoundSterling to Dollars. The

query is rewritten as `select D2P(Pr) from priceOfRental;`. Similarly, if one organization has an attribute called $Name$, and another has two attributes termed $FirstName$ and $LastName$, and the expert provided a function *concatenateName* that concatenates $FirstName$ and $LastName$ to derive a $Name$, the query
`select Name from employee;`
will be rewritten to
`select concatenateName(FirstName,LastName) from emp;` assuming that the matching table for *employee* is *emp* in the responding organization's ontology.

4.4.3 *Translation Using Complex Mapping Rules.* In this case, the algorithm first maps the terms representing tables in the query to tables in the complex mapping rules. Then, the query is rewritten using the SQL query present in the complex mapping rule and the binary mapping rules between the results of the query and the attributes. For example, given the query `select Mk, Md from LuxuryCar;` sent by organization 1, and the complex mapping rule indicated above, we see that $LuxuryCar$ matches the table in the complex rule. The attributes Mk and Md have equivalent terms returned by the query. The mediator translates these two attributes to $car.Make$ and $car.Model$. Then, the query can be rewritten and is translated to:

```
Query: SELECT c.Make, c.Model
         FROM car c, prices p
         WHERE c.ID = p.ID AND
               p.Price > 40,000
```

This translated query can now be sent to organization 2.

4.4.4 *Specializing Terms in Queries.* After the mediator translates query, it also translates the roles to replace the role of the requestor with a role having equivalent permissions in the responder's role lattice by looking up the role-mapping table. The query now needs to be checked to see if the requestor has access to the tables and attributes that are needed to answer the query. This checking can be done at two places:

—Responder: The query can be sent to the responder and the responder can try to run the query and check that the role asking the query has permissions to the requested tables and attributes.

—Mediator: All organizations send their role-based access-control tables in the encrypted form to the mediator. The mediator checks to determine if the role asking query has permissions to the requested tables and attributes.

The two methods correspond to the minimum-middleware and the middleware-centric architectures. In the first method, consider the case where the query is not answered because the requestor does not have permissions to access data required to answer the query. In this case, the responder-side stub of SACE must *refine* the query by replacing a term T to which the user posing the query does not have access rights by the subclasses of T to see if the user posing the query has rights to the subclasses of T. This process of refinement is repeated until the user has access rights to the terms in the refined query or until the query cannot be no further refined because the T does not have any subclasses because it is a

leaf node in the ontology. In case the query can no longer be refined, an access error is sent to the requestor of the query. For example, if the requestor's role is not allowed access to the table *Conveyance*, the mediator would then rewrite the query `select MSRP from Conveyance;` as `select MSRP from Car; select MSRP from Boat` and resend these queries for authentication. If the access-control check is achieved at the responder's site (minimum-middleware architecture), the responder-side stub of SACE might have to and forth several times as the mediator walks down the ontologies and tries to see if the requestor has access to subclasses of the query terms.

To prevent these round-trips, in the second method, in the centralized-middleware architecture, the mediator keeps an encrypted copy of the role-based access control table for each organization. It checks this table to verify access control and keeps refining the query by rewriting it using subclasses until a query that has access permissions is generated. This final query is then send to the requestor saving several roundtrips. However, in this scenario, the drawbacks are as follows: (1) an organization may not want to share its access control tables with the mediator even if it is encrypted, and (2) everytime the organization has to make changes to its access-control tables, the mediator has to be informed about the updates.

Now, we discuss another scenario: when the equivalent term for $Vehicle$ was not present in the mapping table. In this case, too, the query would also be rewritten as: `select MSRP from Car; select MSRP from Boat`
In the example above, we showed query rewriting using only the mapping table. However, as discussed earlier, a query can also be rewritten using complex mapping rules using a variant of the query folding [Dawson et al. 2000] algorithm. Also, if data values appear in the query, e.g., the *WHERE* clause of a query contains $price > \$50,000$, the data value *50,000* may need to be translated using the expert-provided conversion functions (e.g., *Dollar* to *Euro*) if the responding organization maintains prices in *Euro*'s.

After the process of semantic translation and role translation, if either process fails to find a valid entry in the corresponding mapping tables, the query is not forwarded to the responder and sent back to the requestor with an error message indicating that access was denied.

We now show the algorithm for the minimum-middleware architecture in Algorithm 1.

We now describe the semantic expansion performed by the SACE mediator below. Given a query, for each term (table or attribute name) in the query, the mediator first expands the term by adding the synonyms of the term. Then, the the mediator looks up the ontology-mapping table for any of the terms in the expanded term-list. If multiple matching rules are found all the matches are used to generate different rewritings of the query. We use the query-folding algorithm designed by Qian to perform the query rewritings. If no matching entry is found in the ontology-mapping table, the mediator replaces each term in the synonym list with all the subclasses of the term as indicated by the ontology hierarchy and repeats the process until there are no subclasses left. We refer to this process as the *narrowing of semantic scope*. We use the term *semantic scope* to refer to the real-world meaning or the extent of the semantics of the term. The semantics of a table is defined by the semantics

Algorithm 1: The Minimum Middleware Algorithm

```
Input  : Query Q, User_Role R, Source S, Destination D
Output : Ack A
begin
    Parse query Q to identify the tables and attributes Q
    for each attribute a in Q do
        Tag a with the source-id S and the table-name T where a is the name
        of an attribute in T
    Ontology_Table OT ← lookup the ontology-table between S and D
    Obtain rewritings of Q using binary mapping rules obtained from OT:
    (i) by replacing all tables in Q with mapping table-names, and
    (ii) by replacing all attributes in Q with mapping attribute-names
    if any attribute in Q needs conversion then
        Retrieve the appropriate conversion function from OT
        Modify the rewritten query by using the conversion function
    if a complex mapping rule contains any table, T in Q, and mapping for
    all attributes of T used in Q then
        Obtain rewritings of Q using the SQL query that is part of the
        complex mapping rule
        Modify the SQL query to select (the mapping attributes of) only
        those attributes that are in Q
    Role Mapping:
    while a mapping for the user_role R has not been found do
        Lookup the role-mapping table for an entry containing R
        if a matching entry does not exist then
            R ← Subclasses of R in role lattice of S
    if any term in Q could not be rewritten because of lack of mapping rules
    then
        return "Failure"
        return error messages obtained from D
    else
        return "Success"
end
```

Algorithm 1: The Minimum Middleware Algorithm

of the data items present in the table and the semantics of an attribute name is defined by the data values in the column representing that attribute.

For each table or attribute in the query, even after semantic expansion, if no entry in the ontology-mapping table matches any term in the semantically expanded list of synonyms, then the list is modifying by replacing each term in the list with all its subclasses in the otnology. The mediator then repeats this process until all

table-names or attribute-names have been matched, or the list cannot be expanded or replaced anymore. of searching for an entry in the ontology-mapping table searches for the terms in the responder's process of finding matching terms in the responder's ontology is For example, say, a query `select price from Vehicle` has been expanded by using the synonyms of $Vehicle$ in the source ontology to {*Vehicle, transportVehicle*}. If neither *Vehicle*, nor *transportVehicle* appears in ontology-mapping table, the list is expanded to include the subclasses of *Vehicle* and *transportVehicle*, say, to the set {*landVehicle, waterVehicle*}. If *landVehicle* or *waterVehicle* have entries in the ontology-mapping table, (e.g., mapping *landVehicle* to *Automobile* and mapping *waterVehicle* to *Boat*), they are used to rewrite the query as `select price from Automobile; select price from Boat;`

We now give an example to illustrate our query rewriting algorithm:

EXAMPLE 1. *Consider a query posed by the user "Bob" in "FBI". The requestor-side component of SACE looks up the roles of "Bob" in the "FBI"'s user-role-assignment table. The clien-side component then sends the query and the role information to the mediator. For example, if the query is:*

$$select * frompersonnel$$

and the role of "Bob" is "Field_Agent". The mediator then semantically expands "personnel". Say the ontology says that "personnel" has a synonym "employee". The mediator the information in the query as:[Data=Personnel,Employee Role=Fieldusing the terminology used in the remote website.

4.5 Role Translation

Now that the query has been rewritten, the mediator proceeds to translate the role of the requestor and map it to a role in the responding organization. The mediator looks up the role of the requestor of the query in the role-mapping table. If an entry is found, the mediator replaces the requestor's role with the equivalent role obtained from the role-mapping table. If an entry is not found in the role mapping table corresponding to the requestor's role, the mediator replaces the requestor's role with all roles that are descendants of the role in the requestor's role lattice and repeats the process of looking up the role-mapping table to find an equivalent role. If an entry is found, the mediator replaces the requestor's role with this newly found role.

For example, if the query is sent by a *Controller* in the FBI, the mediator looks up the mapping-table and replaces it with the role *Supervisor* in the CIA. space It may be recalled that by agreeing to allow the *Controller* and *Supervisor* to be matched, the CIA had indicated that the two roles should have the same permissions with respect to the CIA's data. However, if the mapping between *Controller* and any role in the CIA's role lattice does not exist, the mediator replaces *Controller* by its sub-roles *Investigative_Agent* and *Criminal_Agent*. The guiding principle is that a role higher up in the lattice has all the permissions granted to a role that is its descendant. Now, if the role-mapping table contains an entry equating FBI's *Investigative_Agent* to CIA's *Field_Agent*, the query posed by the an user with the role *FBI.Controller* can be sent to the CIA with the role *CIA.Field_Agent*.

Similar to semantic expansion and narrowing of semantic scope for terms in

queries, the translated role may need to be modified. In case the translated role does not have access to all elements sought by the queries, the mediator traverses the role lattice of the organization serving the data.

4.6 Answering Queries at the Responder Site

Note that the responder now gets a query that conforms to its own ontology. Furthermore, the role that poses the query is also from the responder's role hierarchy. Recall, at the requestor site, the query underwent semantic expansion to arrive at the query `select price from Automobile;`. Assume, that $O1.Automobile$ was translated to $O2.Conveyance$. Similarly, at the responder site the query is rewritten by replacing $Conveyance$ with $Transporter$, where $Conveyance$ appears in the ontology of the responding organization, but the actual database is named *Transporter* and the internal ontology of the responding organization indicates that *Conveyance* and *Transporter* are synonymous. The query now becomes `select price from Transporter;`. The reason for such an expansion is that the ontology-mapping table might contain a rule for $Conveyance$, however, the schema of the database contains a table by the name $Transporter$ and the responder's ontology indicates that $Conveyance$ and $Transporter$ are synonymous. In the centralized-middleware architecture, the access control has already been done. In the medium-middleware architecture, the the responder-side stub checks to see if the role, say *Supervisor*, is permitted to access the table $Transporter$ and access the attributes in the table it seeks. *Transporter.price*. If the access control check succeeds the result is processed (described in the next subsection) and sent back to the user. If access is denied, the responder-side stub narrows $Transporter$ by replacing it with its descendants in the ontology and checks again to see if the role *Supervisor* has access to its descendants.

4.6.1 *Data Translation.* In the general case, the attributes in matching tables might be different. To handle this case, we need data translations. Data translations may be performed at the mediator or at the responder using hints sent by the mediator. Data translations may be of two types:(1) the mediator gets the responder-side attributes that match the query attributes from the mapping table narrowing the query attributes if a matching attribute is not found in the mapping table. (2) if complex-mappings or conversion functions exists, use the expert-provided function to perform the data conversion.

For example, the CIA might have an entry called $MSRP$ and the FBI has an entry called $Price$ and the mapping table has an entry that says that the two are equivalent. If instead of the CIA and the FBI, it was the CIA and Scotland Yard, SACE would use an expert-provided conversion function to convert the currency before sending the data back.

We omit the details of the query translation and the data translation algorithm because of space limitations. The full version of these algorithms can be found in [Liu et al. 2004a].

4.7 Multi-site Join Queries

In information integration and interoperation scenarios, one needs to compose information from multiple sites. Therefore, these systems must support the execution of queries that require information to be retrieved from different sites and

composed. We refer queries that require information from multiple information sources as *multi-site join* queries. The problem of optimizing and executing queries that require information from multiple sources has been studied extensively in the context of distributed databases. We consider some of the known techniques and evaluate them not only based on their response times but also based on the privacy-preserving qualities of the algorithms. The latter analysis is an original contribution of our work.

4.8 Where to Perform the Join?

In existing works on distributed databases, there are predominantly three paradigms:

(1) *Query Shipping*: In this case, the query is shipped from the requestor to the responder of the query and the results of the query are shipped back from the responder to the requestor.

(2) *Data Shipping*: In this case, the requestor asks the responder to send the data from multiple tables or parts of tables to the receiver. The requestor then caches the data at the requestor's site and performs the query execution there.

(3) *Hybrid*: In this case, some subqueries of the requestor's query can be answered using query shipping, that is, they are sent to the responder and only the results of these subqueries sent back to the requestor. On the other hand, other subqueries are answered using data shipping. That is, the data needed to answer these subqueries are brought into the requestor's cache if the data is not already present, and then the subquery is evaluated on the shipped data at the requestor's site. The two sets of results are then put together at the requestor's site to answer the query.

In SACE, we can opt to implement multi-site joins in any of these three ways. Typically, in distributed databases, a hybrid strategy has been found to work best.

The following are the advantages and disadvantages of query and data shipping:

(1) If we seek to optimize the time taken to execute a single query once, query shipping gives the best response time.

(2) Data shipping is advantageous when the data can be cached and reused for future queries and when communication costs are significant. In this case, we save the communication costs from the requestor or the mediator to the responder.

(3) Even when future queries can use data shipping, the response time may be less while using query shipping in some scenarios. For example, if the requestor has a slow machine while the responder has a high-capacity server, the query processing at the responder combined with the communication time may be less than the query processing time at the requestor.

In our scenario, apart from the requestor and the responders of a query, we have the mediator in the middle. Therefore, we can have the following choices:

(1) *Middleware-based Join*: In this method, the requestor uses query shipping and wants the results of the query returned to it. In this situation, we have the following sub-cases:

(a) The mediator uses query shipping to ship the subqueries to the responders and computes the results to the original query from the responses to the sub-queries.
(b) The mediator uses data shipping and asks the responders to send it the data required to answer the query. The mediator then caches this data, computes the answer to the query and returns the final answer to the requestor. The cached data can be used to answer future queries without requiring to go the sources provided the data is still valid.

(2) *Requestor-based Join*: This case also has the following sub-cases:
(a) In this case the requestor essentially wants to use data shipping. However, since the requestor does not know the corresponding table and attribute names at the responder, the requersot cannot simply request the data to be shipped to it. Instead, it forwards the query to the mediator indicating that it wants the data to be shipped directly back to the requestor. The mediator rewrites the requestor's query and forwards the query to the responders asking them to send the data back directly to the requestor. All the responders send data back directly to the requestor of the query.
(b) In this case, the requestor wants to use query shipping, the mediator performs the query rewriting and breaks up the query into appropriate subqueries, but the results of the subqueries are sent back directly to the requestor. In both these sub-cases, the requestor joins the data to arrive at the final result. The middleware may optionally send the requestor a rule stating how that data should be joined at the requestor or the requestor may figure that out itself.

(3) *Responder-based Join*: In this case, the mediator generates a plan where the data is not returned to the mediator but forwarded from one responder to another with partially computed join results. A responder receives a partially computed join, joins its own data to it, and forwards it to another responder. The last responder forwards the result to the requestor.

The disadvantages of the join methods with respect to query response time is listed below. A disadvantage of one method that is not a disadvantage for a second method can be considered an advantage fo r the second method.

(1) The mediator-based join algorithms are all routed through the mediator. Therefore, the mediator can be a performance bottleneck. A solution is to replicate and have multiple mediators serving the clients.
(2) If the data is cached at the mediator instead of the requestor's, less data may be cached. Also, by caching data at the requestor, the requestor has a choice to cache data that is of higher priority to it, whereas even if the mediator is augmented to consider priority of the cached data, it will consider the overall priority of the data with regards to multiple requestors instead of the priority of the data for one requestor.
(3) In the case where the requestor joins the data, and data shipping is used, a large amount of data might need to be communicated through the network bringing down the network performance.

The advantages and disadvantages of the different join methods with respect to privacy and security of the metadata and the data are listed below:

(1) Requestor-based Join:
   (a) An apparent disadvantage of the requestor-based join method is that the requestor may see the data obtained from all the responders. However, if the requestor has the rights to run a query on the data, the requestor must have select rights on the data anyway.
   (b) The advantage of the requestor-based join method is that the data is sent directly from the responder to the requestor using a secure channel and does not pass through any intermediate nodes.

(2) Middleware-based Join:
   (a) The middleware-based join methods are susceptible to denial of service attacks and become a single-point of failure. Replication can alleviate the problem to some extent.
   (b) The major drawback of a middleware-based join is that if an intruder hacks into the mediator the intruder has access to all the data. Even if the data is encrypted, by collecting and analyzing the data passing through the middleware, the intruder may be able to infer valuable information about the data stored in the different organizations.

(3) Responder-based Join: The disadvantage of performing a responder-based join is that the data is forwarded from one responder to another. In this scenario, an unauthorized user in one responder can easily examine the forwarded data that it might not have the rights to access. Therefore, we do not implement this method in SACE.

We studied the three join strategies and choose the requestor-based join method since it conforms to our principle of separate query and data channels. Recall that separating query and data channels provides the maximum privacy and security by reducing the bottleneck and the vulnerability on the mediator. The choice of which sub-strategy to employ — query shipping or data shipping — is an optimization question depending upon whether the same query will be rerun or not. For example, conside the following case. While running a query using query shipping, it involves 30 units of data being shipped to the requestor, and running data shipping results in 50 units of data being brought to the requestor. If the query is run two times or more and the data does not change rapidly, data shipping will be preferred, because if SACE uses query shipping, it will involve $2x30 = 60$ units of data being brought to the requestor, whereas if the data is cached at the requestor and can be reused, data shipping results in 50 units of data being communicated. Lastly, the third strategy is difficult to implement and involves partial disclosures of schemas.

4.9 Additional Optimizations

Additionally, we use blocking and multi-threading to enable pipelining of the query processing. Pipelining the query processing enhances the performance of multi-source joins.

Consider the case where two responders A and B run two sub-queries and on the results returned by them C, the requestor, performs a join. If either A or B

has a sub-query that takes a long time to run and waits for the entire sub-query to run before returning the data all at once to C, then C has to wait for those results before it can start computing the join. An alternative is for A and B to return rows belonging to the results-set of the subqueries they are executing to C as soon as the rows become available. However, sending a large number of messages with a few number of rows is also not very cost effective, due to the latency and costs involved in processing the numerous small messages. Therefore, SACE requires responders to allow result-sets of subqueries to be retrieved in blocks, the size of which can be set using a configuration parameter.

On the requestor-side, the inherent parallelism scheme of executing subqueries at multiple sites and joining the data at the requestor can only be fully utilized if the requestor can receive the data (and then process them) from several sites simultaneously. In order to enable parallel receiving of results and data from responding sites, the SACE stub at the requestor uses multiple threads each of which receives data from one site.

4.10 Privacy Preserving Features

In this sub-section, we outline the privacy-preserving features of SACE.

4.10.1 *Preserving the Confidentiality of Metadata.* First, all the metadata stored at the mediator are encrypted, but the mediator does not know the decryption key. All the database schema elements and the data elements in the query are encrypted. A's ontology, role hierarchy, and database schema are encrypted by a key only known to A, called the *master* key of A. Accordingly, the ontology-mapping rules encrypted by two keys: all the terms belonging to A's ontology are encrypted by A's master key; all the terms belonging to B's ontology are encrypted by B's master key. For example, consider the query sent by requestor, A, to the mediator cited above: `select Price from rentalPrice` Here $Price$ and $rentalPrice$ will be encrypted using the master key of A. The rewritten query, R, is `select D2P(Pr) from priceOfRental`. In R, Pr and $priceOfRental$ will be encrypted using the master key of the responding organization B. Similarly, for the complex mapping rule shown above, the terms in the SQL query is encrypted using the key of the responding organization B, and the table $O1.LuxuryCar$ is enrypted using the key of the requesting organization A. The mapping rules relating the attributes of $LuxuryCar$ with the results returned by the SQL query are encrypted using both the master keys of A and B, i.e., the left-hand-side containing terms from $O1$ are encrypted using A's master key and the right-hand-side containing terms from the SQL query are encrypted using B's master key. The role-mapping table is encrypted in a similar manner.

In our scenario, we have minimum trust in the mediator. We envision that the two organizations will employ a mutually trusted *expert* to establish an encrypted set of ontology-mapping rules that the mediator can use. If the expert is an automated tool, the tool is evoked for a short period, establishes the ontology-mapping rules, and is then killed. An expert is considered "killed" if it maintains no history of the ontologies passed to it or of the intermediate or final results of the ontology mapping process. (The expert is trusted not to disclose any information.) After establishing the rules, the expert encrypts them using the master keys of the two

organizations and sends the ontology-mapping rules to the mediator.

4.10.2 *Preserving the Confidentiality of Queries and Data.* Second, the confidentiality of the queries and the data is maintained by encrypting the queries and the data using several encryption keys.

The requestor encrypts the queries and sends them to the mediator. The mediator does not accept any query with a wild-card character, (`select * from`) since the mediator does not know the attributes of the tables and cannot rewrite the query. The requestor-side application must expand the query with the wild-card character by replacing the wild-card character with all the attributes in the source table before sending the query to the mediator.

Using the encrypted ontology-mapping rules, the mediator can translate the queries without decrypting the schema elements in the requestor's query. The encrypted data values in the queries are translated and transformed to equivalent data values encrypted using the data encryption key of the responder. This encryption is done with a transformation function provided by the expert and does not require the mediator to decrypt the data values.

4.10.3 *Privacy Preserving Data Translation.* Third, the query results are encrypted first before being transmitted to the requestor via the data channel. Similarly, note that when the results are returned, the schema part (i.e., the attribute names) will be encrypted by a key only

Before the data can be sent, as discussed above, data translation may be required. Data translation may cause additional privacy-related vulnerabilities. Data translation can be handled in various ways in SACE, however, they have different resilience to privacy attacks.

A naive approach is to let the mediator translate the data. However, it is quite difficult for the mediator to do data translation based on encrypted values. Finally, when the query results are returned, since some data translation needs to be done by the mediator, these results cannot bypass the mediator; and as a result, more confidentiality breaches can occur. An added disadvantage is that the mediator becomes a bottleneck since all traffic is routed through the mediator.

The privacy problems of the naive approach clearly indicate that a better approach may be to let organizations translate the data. However, if we let organization B know exactly how A's database schema is different from B's, B can infer substantial information about A's schema and this can be a big privacy concern to many organizations. To remove this privacy concern, we propose two novel schemes to handle data translation.

Scheme 1: ontology-based data translation. During initialization of SACE, A and B send the trusted expert their schemas S_A and S_B. Next, the expert proposes a third ontology (denoted O_3) that contains a mediating term only those terms values need translation between S_A and S_B. For privacy and efficiency, O_3 should be minimum. Next, the expert returns O_3, a mapping table $MappingTable(S_A, O_3)$ that captures the data translation requirements, and the corresponding data transaction functions to A; O_3,
$MappingTable(S_B, O_3)$ and the corresponding data translation functions to B.

During run time, when a query is issued by A, e.g.,

`select * from personnel where salary > $50K`, if term 'salary' is not in $MappingTable(S_A, O_3)$, the query is encrypted and sent to the mediator as usual. Otherwise, 'salary' will be replaced by the equivalent term in O_3 (e.g., 'stipend') and the corresponding data translation function will be applied to change $50K to 45K Euro. Next, the query will be encrypted and sent to the mediator. The mediator needs to do nothing different (except that no term in O_3 should be changed), since 'stipend' should be in O_A. So the access control part will not be affected.

Then, the filtered query will be sent to B. B will first decrypt the query, then find that 'stipend' is in
$MappingTable(S_B, O_3)$, then 'stipend' will be will be replace by the equivalent term in S_B (e.g., 'wage'), then the corresponding data translation function will be applied to translate 45K Euro to 32K Pounds. Now, B's database server can correctly execute this query.

After obtaining the results, B will use $MappingTable(S_B, O_3)$ to translate the format of the results from Pounds to Euro. Then, the data part of the results will be transmitted back through the data channel, while the schema part will be mediated by the mediator. Once A receives the results, A will use
$MappingTable(S_A, O_3)$ to translate the format of the results from Euro to dollars.

In summary, scheme 1 has very good privacy. O_3 provides a middle layer that blocks the schema inference attacks; no decryption is needed at the mediator; and no clear text is available during mediation.

Scheme 2: privacy preserving data translation without O_3. Scheme 1 has very good privacy, but it needs O_3, two translations on the way from A to B, and two more translations on the way from B to A. The goal of Scheme 2 is to remove the needs of O_3 and maximize the efficiency (the stubs on A and B are thinner). Scheme 2 works as follows.

First, during initialization the trusted expert identifies all the data translation needs. For each such need from A to B, a specific conversion function is produced and specified as: { E(table_name, A's master key), E(attribute_name, A's master key), E(Function, B's master key) }. The expert will do the same thing for the needs from B to A. Next, the expert will forward all the conversion function specifications to the mediator.

During run time, when the mediator receives an encrypted query, if an encrypted attribute matches any of the conversion function, the mediator will use the ontology mapping table to rewrite the attribute as { E(O_B_attribute_name, B's master key), E(Function, B's master key)}.

When B receives the query, B will first decrypt the query and the function, then apply the function to the value associated with the attribute before the query is submitted to B's database server. Once the results are generated by the database, the value associated with the specific attribute will again, be processed by the function in the reverse way (note that the function is a two-way function).

A problem with this method is that when the tables have attributes that need to be converted, say split, the mediator has to keep a list of all problematic attributes that need to be converted (in either direction) and all the corresponding conversion functions. Finally, compared with Scheme 1, scheme 2 has less privacy since B knows the exact data value (and measurement units) that A uses, although the

attacker does not know anything about this information.

5. SECURITY AND PRIVACY ANALYSIS

SACE tackles both security and privacy issues of inter-organization information sharing. SACE's primary security concern is whether unauthorized accesses can be caused and whether data confidentiality can be compromised. SACE's primary privacy concern is whether the sensitive metadata stored on a mediator can be *inferred* by the attacker.

In general, since the queries and the data are encrypted using different keys and pass through different channels and since each data channel is encrypted using different keys and only the answer to one query passes through one data channel, the chances of the entire system being compromised is reduced. However, there are questions about whether SACE's semantic access control scheme can allow unauthorized accesses, and whether the privacy preserving features can prevent SACE from achieving "correct" information sharing. We have investigated these questions carefully, and our study shows that SACE's semantic access control scheme is "correct" (or secure) and SACE's privacy preserving features will not cause incorrect information sharing. Nevertheless, due to space limitations, we could not provide the correctness proof of SACE here, and we would refer the interested readers to [**?**].

In this section, first, we analyze the amount of privacy that can be achieved when SACE is under various inference attacks. We show why SACE can achieve much more privacy than existing information sharing solutions. In addition, we compare the middleware-centric architecture and the minimum-middleware architecture from the perspective of privacy preserving. Second, we analyze how SACE is resilient to the attacks that may cause unauthorized accesses. Note that it is easy to see that our key management scheme is secure and data confidentiality can be ensured.

5.1 Correctness of the Semantic Access Control Scheme

In this section, we only address the security of SACE in terms of the semantics of our semantic access control scheme. We assume there are no (active) attacks against SACE.

To see whether our semantic access control scheme is secure, we must first define "security" for semantic access control. Our definition has two aspects: (a) within a single organization and (b) across organizations. First, within a single organization, a *syntactic access control* scheme is secure if a data object x can only be accessed by those who have authorizations to access x. In contrast, the corresponding *semantic access control* scheme (built on top of the syntactic authorizations) is *secure* if for any user (or role), whenever a semantic access request is authorized, there exists a syntactic access request that can allow the user to access the same data and will be authorized by the syntactic access controller. Moreover, if every request authorized by the syntactic access controller will be authorized by the semantic access controller, we say the semantic access control is *tight*. Note that when the semantic access control is not tight, some availability may be lost.

Now we can define the security of semantic access control across organizations in a similar way. We assume a cross-organization semantic access control scheme

is always developed to enforce a specific syntactic cross-organization access control policy. Given a sound (i.e., the policy is consistent) and complete (i.e., every request is either authorized or denied) syntactic cross-organization access control policy, the corresponding cross-organization semantic access control scheme is *secure* if whenever the semantic scheme allows a user to access a data object, there exists a syntactic cross-organization access request that will allow the user to access the data object and will be authorized by the syntactic cross-organization access control policy.

In the following, we show that SACE's semantic access control scheme is secure. For this purpose, we need to introduce a notation called *semantic scope*. We assume each term e in an ontology has a specific semantic scope, denoted $SC(e)$, which indicates how broad the terms semantics can cover. For example, the semantic scope of "horse" is *broader* than "race-horse" but *narrower* than "animal", denoted $SC(\text{“}race-horse\text{”}) \subset SC(\text{“}horse\text{”}) \subset SC(\text{“}animal\text{”})$. Moreover, we assume each column or attribute (name) of a table has a specific semantic scope, and a tables semantic scope is the *union* of the semantic scopes of all of its columns. Finally, we assume semantic scopes can be compared with each other universally across organizations. The relation between two semantic scopes is of four types: *equivalent*, broader, narrower, or *intersecting*.

Now we make two more assumptions about SACE. First, we assume that role mapping in SACE will never map a role in Organization A to a more powerful role in Organization B. Second, we assume the (underlying) syntactic cross-organization access control policy will never allow an employee of A to get a piece of information from B's source which he is not authorized to access (that type of information) in A. This assumption implies that a semantic access control scheme across organizations is secure only if for each user or role, the union of the semantic scopes of the accessible data objects (a table or a column) from any foreign organization is a subset of the union of the semantic scopes of the data objects accessible within the home organization.

To show that SACE's semantic access control scheme is secure, we need only to show that no access request denied by the syntactic cross-organization access control policy will be authorized by the semantic scheme. For this purpose, consider the scenario when an employee of Organization A wants to get some data from B, we first show that when the request is checked against B's semantic access control policy (after the semantic mediation step), it will be denied if it is not authorized by the syntactic access control policy. This is true since when we translate B's syntactic access control policy to a semantic policy offline, we make sure that the 'object' term will never have a broader semantic scope than the corresponding syntactic object (e.g., a column) in the syntactic policy. Hence, when the employee's request is checked against B's semantic policy, no access to an term that has a broader semantic scope than the columns in the syntactic policy will be authorized. Moreover, when the filtered semantic request is translated to a syntactic request against B's schema, we ensure that the semantic scope of the resulted syntactic request will never be broader than the semantic request. This means that the syntactic request will never violate the syntactic policy.

Second, we show that the employee will never get data from B that violate A's

syntactic access control policy. To show this is true, we just need to show that after the employee's request is checked against A's semantic policy, the filtered request when being processed by SACE will never violate A's syntactic policy, since it is easy to see that any request violating A's syntactic policy will be filtered out during the check. This statement is true since (1) when this request later on goes through the semantic request mediation process, SACE ensures that the semantic scope of the resulted request against B's ontology will never be broader than that of this request; (2) according to the way the resulted request is checked aginst B's (semantic and syntactic) policy, the request will never be able to get data that have a broader semantic scope than the request.

5.2 Privacy Analysis of SACE

For two organizations using SACE to share information with each other, the primary privacy concerns of the organizations are as follows. (Note that due to the middleware-based architecture, SACE makes it very difficult, if not impossible, for one organization to infer the other organization's database schema, ontology and role hierarchy.)

⊙ Whether the Mediator can infer the sensitive metadata stored on the mediator, though all the metadata are encrypted.

⊙ Whether an outside attacker can infer the sensitive metadata after he successfully breaks into some components of the SACE system.

Accordingly, our threat model focuses on four *attack modes*:

⊙ (A) An outside attacker breaks into the mediator and infers metadata, but the attacker does not maintain history of queries (and responses);

⊙ (B) The same as Mode A except that the attacker maintains a complete history of queries (and responses).

⊙ (C) The mediator infers the metadata;

⊙ (D) The attacker breaks into the mediator as well as an organization.

In the following, we show that SACE can provide very good privacy to the organizations involved. Note that since the privacy issues associated with data translation are already addressed in Section 4, they will not be mentioned here.

In Mode A, some representative privacy attacks are:

⊙ Based on some specific statistics of the metadata associated with an organization, such as the size (or "shape") of her ontology and database schema, the attacker guesses who the organization it. SACE makes this attack difficult to succeed since all the ontologies are encrypted and two ontologies with similar statistics may belong to two very different organizations.

⊙ The attacker tries to figure out the meaning of each encrypted term in an ontology. Moreover, the attacker exploits the (equivalent, broader, narrower, irrelevant) relation among the encrypted terms to help him. However, precisely guessing the meaning of encrypted terms based on only some use statistics is difficult, especially when the organization identities are also kept private by SACE. Moreover, even after some terms are disclosed, it is still very difficult for the attacker to guess the database schema.

⊙ The attacker tries to figure out the meaning of each encrypted attribute in a database schema. This attack is similar to the above one, but breaking a database schema can cause much more damage than breaking an ontology. For this reason,

in the minimum-middleware architecture, database schemas are no longer stored at the Mediator. Moreover, all the schema information that the Mediator can see is obfuscated. In this way, SACE may provide maximum privacy.

⊙ Based on the encrypted role mapping table and access control policies and the relationships among the encrypted terms, the attacker may be able to infer something about the role hierarchy of an organization, such as knowing that one role is more "powerful" than another role, etc.. However, SACE makes this attack very difficult to succeed since (a) no role hierarchy is stored on the mediator; (b) knowing which roles are more powerful does not disclose much information of the role hierarchy (when the attacker does not know the clear text version); (c) note that the minimum-middleware architecture no longer keeps access control policies on the mediator, which makes this attack even harder.

In Mode B, some representative privacy attacks are as follows.

⊙ Based on the history observed by the attacker, the ontology mapping table and the role-mapping table, the attacker could reconstruct the (encrypted) ontologies and access control policies. Hence, obfuscating the ontologies and policies stored on the Mediator do not help much. Nevertheless, the ability to reconstruct encrypted ontologies and policies does not significantly improve the attacker's ability in inferring them.

⊙ By monitoring the frequencies of requests across organizations, the attacker may infer which organization is popular, and which pair of organizations is a gang, etc.. This attack can allow the attacker to get more information about the organization identifies than Mode A attacks, but this attack is still difficult to succeed if the attacker does not know who the set of organizations are. Although we may use dummy requests to obfuscate the attacker, this could cause denial of service and may bring down the middleware.

⊙ Since a lot history data are available, histogram-based privacy analysis and inference are possible. To counter this attack, a unique design feature of SACE is that we make no data available to the Mediator and all inference can only be based on encrypted schema and terms. Although the value part of each query is encrypted, such cipher text is useless to the attacker. Therefore, SACE is very resilient to such inference attacks.

In particular, we decouple semantic level mediation from (query result) data transmission. The middleware will only be involved in the mediation tasks, and the result data will be encrypted and transmitted to the requestor through a separate channel that will never be touched by the middleware, although to make the result data meaningful to the requestor, the schema part of the results will still need to be blindly mediated by the middleware in cipher text.

In terms of Mode C, this mode is actually a special case of Mode A and Mode B, so no more discussion is needed.

In Mode D, when the attacker breaks into an organization (e.g. FBI) and assumes an authorized account (not necessarily the root), he will get (clear-text) access to all the data, database schema, role hierarchy and ontology accessible to the account as well as the other organizations' data that the account is authorized to access. Now the question is whether he can figure out the encrypted ontology mapping table, and the metadata of the other organizations (e.g., CIA). Although now the

attacker can compose special probing queries to do this job, it should be noticed that since the corresponding security loss (data confidentiality and integrity) is already much more significant than the inference problem, the organization should instead work hard to block this attacker rather than making the middleware more private preserving.

Finally, of course we can assume such attack modes as the attacker breaking into both of the two organizations (and the Mediator) and the attacker stealing the authorization of the DBA of an organization. Of course, such attack modes will allow the attacker to gain more information about the metadata, but as we argued above, in such modes the attacker can cause much more damage by directly attacking the system instead of just inferring the metadata.

5.3 On Attacks that May Cause Unauthorized Accesses

Now we address how active attacks can cause unauthorized accesses in SACE. We found that in general such attacks could be handled by some existing security mechanisms. For example, the attacker may change some records of the ontology mapping table in such a way that he can allow an organization to access some data that the organization is not authorized to access.

To tackle this problem, first, we can deploy some integrity protection techniques. For example, we may assume the Mediator will sign each query authorized; assume the Mediator's signing key is in tamper-resistant storage; assume the Mediator's program will be certified and can be verified whenever being loaded for execution. Second, a time-stamped checksum of the metadata may be firstly signed with the private key of the security expert before being stored at the Mediator. During runtime, the Mediator may periodically verify this signature before checking any access control.

For another example, denial-of-service may be caused, but DOS attacks do not cause unauthorized accesses. To tackle this problem, the Mediator can be replicated across the network, or several different mediators can serve the same pair of organizations.

6. EXPERIMENTS AND RESULTS

We have implemented a prototype of SACE based primarily on minimum-middleware architecture. In this section, we present the evaluation results and show that SACE's performance is in general good. The evaluation focuses on comparing the performance of SACE with the non-encrypted version of SACE and a direct query system in terms of a set of system parameters such as the number of data requesters and the result data size.

To assess the performance of SACE, we programmed SACE prototype in Java; and Jena 2 Semantic Web Framework is applied to handle the ontology files. In particular, the prototype is implemented with Sun Java Web Service Developer Pack 1.5 with Apache Tomcat 5.0.19 Web Container. The Java Virtual Machine used is Sun JDK version 1.5.0-b64. We performed several experiments on the FBI-CIA information sharing scenario, where both FBI and CIA have an individual organizational database managed by MySQL DBMS version 4.1.8. Moreover, three Web services are running in three computers (connected by a 100 Mbps LAN switch) on behalf of FBI, CIA, and the mediator, respectively. The detailed specifications

Tier	FBI	Middleware	CIA
CPU	Pentium 4 2.53GHz	Dual Xeon 1.8GHz	Pentium 4 2.4GHz
RAM	768MB	512MB	512MB
OS	Linux 2.6.9	Linux 2.6.9	Linux 2.6.8
Database tables	15	-	18
Role hierarchy nodes	5	-	5
Ontology	289 triples	-	325 triples

Table I. System specifications

of the experimental testbed are in Table I. Each computer is equipped with an IDE hard drive with about 45 MB/second sequential read throughput.

The communications among software components are implemented by JAX-RPC. There are three types of implementation to create JAX-RPC clients: static stub, dynamic proxy, and dynamic invocation interface (DII). Clients use either static stub or dynamic relied on pre-generated implementation-specific classes. Therefore, in our implementation, we use dynamic invocation interface for flexible system design and easy deployment.

6.1 Metadata and Data Sets

We generated the database schema and data sets for both FBI and CIA databases based on the information sharing scenario. In general, each table contains 50 to 10000 records depend on its functionality, and the corresponding access control rules are stored in a separate authorization table. The database schema is shown in Table II.

We designed the organizational ontology for each organization. Each ontology has about 20 classes and about 300 triples if they are represented in N-TRIPLE format [rdf 1999]. The ontology covered all concepts used in the database. The syntactic database schema can be mapped to semantic concepts defined in ontology using a syntactic-semantic mapping table. We show the complete ontology used in the FBI-CIA scenario in [Liu et al. 2004b]. Regarding role-based access control, each organization has 5 roles as shown in Fig. 3. Each role has diffrent privileges to access the tables in database. Parent roles have all the privileges their children roles have. Finally, the mapping table in the mediator was designed for information mapping between the FBI and CIA databases.

Our evaluation benchmark consists of four types of queries. We generated 25 queries for each type. Each query is executed 10 times and we calculate the mean execution time for each. Finally, the query result data size ranges from 0.31KB to 1,359KB. The four types of queries are:

(1) Basic type query: These queries only involve semantic translations. For example, when an FBI agent wants to get some data from the CIA database, his query, that is originally written against FBI's schema, can be:
SELECT username, passwd **FROM** Person **WHERE** username ='john031'.
After the mediator performs a semantic translation, it becomes:
SELECT userid, pwd **FROM** Personnel **WHERE** userid ='john031'.
Notice that although the query in the given example is valid in both organizations, the result retrieved from CIA may not be the desired information for the

Schema of FBI database:

Number of records	Schema
50	Person (ID, username, passwd, orgID)
100	Organization(ID, parentOrganization, locationID, name)
500	LocationArea(ID, building, city, state, country)
50	PersonDescription(ID, fullname, agerange, complexion, height, build, hair, sex, driving, phonenumber, clothing)
1000	Event(ID, analysis, reporterID, activityID, note, predecessor, type)
10000	ActivityTable(ID, person, organization, vehicle, location, duration, TimeFrame, resultedIn, willresultIn, frequency, rationale)
10000	TimeFrame(ID, minute, hour, day, month, year)
1000	Vehicle(ID, make, model, color, status, plate)

Schema of CIA database:

Number of records	Schema
50	Personnel(ID, userid, pwd, description, location, organization)
100	Suspect(ID, firstname, lastname, location, note)
500	Cell(ID, number, establishedIn)
1000	Note(ID, comment, Time)
50	TerroristOrganization(ID, parentOrganization, locationID, established)
10000	Location(ID, building, city, province, country)
50	PersonalProfile(ID, name, cell, age, height, ethnicity, address)
10000	Events(ID, owner, activity, noteID, predecessor, eventType)
10000	ActivityInformation(ID, suspect, automobileID, location, startTime, endTime, result)
10000	Time(ID, h, d, m, y)
2000	Automobile(ID, model, manufactory, color, size, condition, registration)

Table II. Database schema and the number of records for each table in the databases

user in FBI. For example, the users' password should not be released to any other organizations. We can prevent this problem by a proper design of the mapping table.

(2) Query with semantic expansion: These queries involve query folding. For example, an FBI agent's across-organization query can be originally written as:
SELECT fullname, phonenumber **FROM** PersonDescription **WHERE** fullname ='John Peterson'.
After semantic translation and expansion by the mediator, the query becomes:
SELECT firstname, lastname, number **FROM** PersonProfile, Cell **WHERE** PersonProfile.ID = Cell.ID **AND** PersonProfile.firstname = 'John' **AND** PersonProfile.lastname = 'Peterson'.

(3) Query with data translation: These queries contain values or columns that SACE needs to translate. For example, when a FBI agent issues the following query:
SELECT ID **FROM** PersonDescription **WHERE** height > *5.9*.
Since in FBI the metric for height is in feet, but the height in CIA is in centimeter, the value (i.e., 5.9) will be converted to centimeter, and the translated query will be:
SELECT ID **FROM** PersonProfile **WHERE** height > *179.83*.

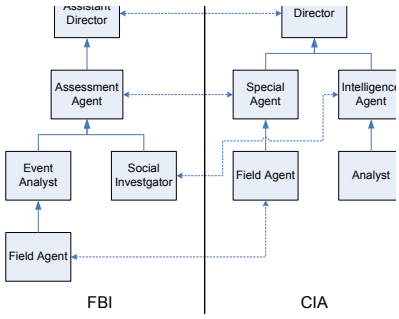


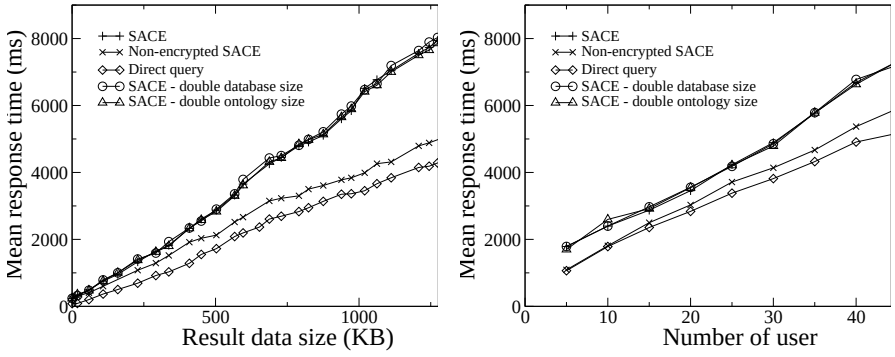
Fig. 3. Role hierarchy

(4) Hybrid type: Translation Using Complex Mapping Rules and Mapping Functions. For example, the query:
SELECT fullname, phonenumber **FROM** PersonDescription **WHERE** height $>$ *5.9*
will be translated to:
SELECT firstname, lastname, number **FROM** PersonProfile, Cell **WHERE** PersonProfile.ID = Cell.ID **AND** height $>$ *179.83*.

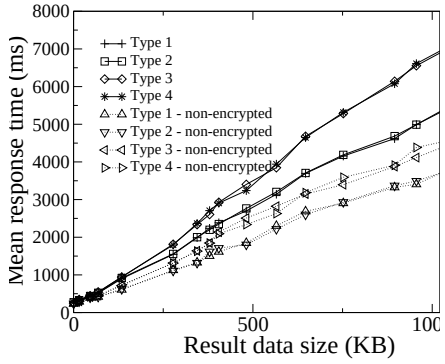
6.2 End-to-End Response Time

The mean end-to-end response time is determined by the average time used to process a cross-organization query when multiple such queries are processed. We measure the response time as the time elapsed since a user's request a query until she got the response to the query from SACE. The response time includes both computation and communication costs. In particular, to process a query request, the system performs query parsing, semantics translation, data tanslation, and access control check. In the encrypted version of SACE, data and metadata encryption and decryption are also required. We used both symmetric and asymmetric encryption algorithms in our system. We assume each organization has one master key for symmetric encryption and a RSA keypair for asymmetric encryption. They length of the master key is 112 bits, while the length of RSA keypair is 1024 bits. The subject and object in each 3-tuple was encrypted by Triple DES [des] with a master key for each organization. The ontology files (.owl) were translated to N-TRIPLE format. Subject, predicate and object terms are encrypted separately using the master key. For each query request, the requesting organization has to exchange a session key with the responding organization, the requesting stub will parse the SQL statement and encrypt every non-keyword term using the session key. They keep the SQL keyword non-encrypted, so the mediator is able to parse the request and decompose the request into encrypted syntactic access requests. By encrypting Data and meta data, SACE can provide uncompromising privacy preservation. However, we have to pay extra computation costs to accomplish the security requirements.

To compare the overhead of SACE processing, we have implemented a simple direct query system using the same architecture as SACE but without semantic access



(a) (b)



(c)

Fig. 4. End-to-end response time of SACE, non-encrypted SACE, and the direct query system. (a) The effects of result data size on response time. (b) The effects of number of users on response time at peak traffic. (c) The effects of query types on response time.

control. The requesting organization sends a query request to the mediator, then the mediator bypass this request to the responding organization. Access control is done by the responding organization by checking the role's privilege. The query result will be sent to the requesting stub directly. Also, the users in the requesting organization need to know the database schema in the responding organization.

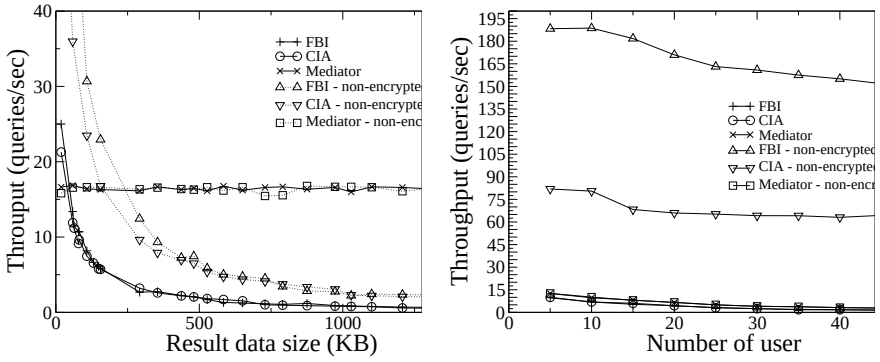
6.2.1 *The Result Data Size.* To evaluate the impact of the result data size, we generate one request to FBI to query the CIA's database using the query sets described in the previous section. The results in Fig. 4(a) show that, in general, when the result data size increased, the response time of SACE increases. This is easy to understand considering the amount of data to be transfered and processed. Larger result sets require not only longer transfer time, but also need more computation time to encrypt or decrypt. To compare the encrypted SACE and the non-encrypted SACE, the encrypted SACE requires more time than the non-encrypted SACE to complete a query request when the result data size increased. In particular, the encrypted SACE requires 15 percent to 20 percent more in mean response time when the number of user is larger than 10. Closer inspection reveals that the response time of SACE is roughly double as the direct query system in our experiment. Also, the mean response time of the non-encryption SACE increases roughly linear, while the mean response time of SACE increases slightly worse than linear. This can be explained considering that the encryption algorithms are non-linear algorithms. Second, we compare SACE and the direct query system. The results in Fig. 4(a) show that SACE requires two to two and half time more than the direct query system. More specifically, when the result data size is less than 500KB, data encryption causes about 35 percent more in mean response time. As the result data size increases, the overhead of encryption keep increasing. When the result data size is 1359.73KB, the overhead of mean response time is about 60 percent for the encrypted SACE. Recall that the simple direct query system involves only minimal computation and has roughly the same communication costs if the result data size are equal. Its results show the lower bound of the response time of the web services architecture in the information sharing scenario. Also, the mean response times of both the non-encrypted SACE and the direct query system increase roughly linear. The difference of the mean response times between the non-encrypted SACE and the direct query system is around 0.5 seconds for most tested queries. The results show that without the encryption, SACE only add a small overhead during the query processing and the impact on the mean response time is nearly a linear function of result data size. Furthermore, the encryption computation dominated the overall process time, and the overhead caused by semantic-level access control check is relatively small.

6.2.2 *The Number of Users.* To evaluate the impact of the number of users, we create multiple query traffic loads to the system, and we fix the size of each query result at about 100KB. That is about 50 to 100 records in our database depending on which tables are queried. For most frequent cross-organization queries, for example, to query the suspects list from the CIA's database, the rows of result set is usually is usually less than 100KB. Fig. 4(b) shows the average response time when the number of user in the system increases response time in all systems increases. In

general, users in encrypted SACE will expect longer response time, because of the encryption and decryption computation costs. But compared to the results in Fig. 4(a), the difference between the encrypted system and the non-encrypted system (or direct query system) in multi-user environment is smaller than singal user testing. This can be explained as follows: When a query from a user sent to the middleware, the request will be blocked and waited the result until it has received a response from the middleware. During the waiting time, the request will not use CPU resource, therefore, the CPU can process other queries. However, in the non-encrypted system, although the queries do not need as much time as the encrypted system to be processed, the requests still need to be blocked due to the network delay and JAX-RPC's overhead. This result shows that SACE has good scalability in a multi-user environment.

6.2.3 *The Size of Database and Ontology.* We also evaluated the impact of the database size and the size of the ontology files. Results in both Fig. 4(a) and Fig. 4(b) show that by doubling the database size, the affect of response time is very small can be neglected. It is because SACE is a information sharing layer above the physical data storage (database systems). Therefore, SACE will not handle how data is stored and indexed. This design philosophy keeps SACE not only platform independent, but also database independent. Ideally, SACE should be able to use any database management system. However, different database management system may have some special features and extensions that may cause difficulty processing the queries. In our implementation, the SQL parser we used followed the standard ANSI SQL-92 specification. Hence, any non-standard SQL statements will be rejected. Also, changing the size of ontology files for both organizations will not change the response time significantly. The reason is if the additional relationship in the ontology will not affect the concepts used to process queries, it is obvious the response time will not change. Even when we changed the relationship between the concepts used to process queries, the computation time required for the inference processes is relatively small. Furthermore, since the filtered stub generated by the mediator will be stored as a hash table, the changes of response time due to changing the access control rules during the on-line process is negligible. The results show that SACE can easily be extended to large database systems in practical applications.

6.2.4 *Types of Queries.* Finlay, we compare different query types and evaluate the mediating process. The impact of different query types on end-to-end response time is evaluated by sending 25 queries from each type of query and calculating the mean response time for each query. As shown in Fig. 4(c), the mean response time of type 1 and type 2 (or type 3 and type 4) query are very close. The closeness of the response times indicates that the impact of semantic expansion on the system is very small. However, the overhead of data translation increases with the size of the result data. Type 3 and type 4 queries involve data translation in one column of integer type. Data in this column needs translation four times during the transaction. This explained why the response time for type 3 and type 4 queries are higher than type 1 and type 2. According to the results, the data translation is linearly dependent on the rows of the result data table. It is easy to understand



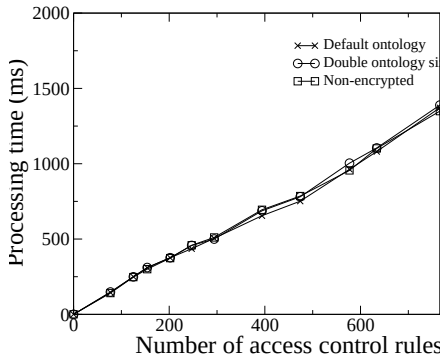
(a) (b)

Fig. 5. Component throughput of SACE and non-encrypted SACE. (a) The effects of result data size on the throughputs of FBI, CIA and mediator. (b) The effects of number of users on the throughputs of FBI, CIA and mediator at peak traffic.

considering the operation is applied on each row of data. If the result table is small, the effect of data translation is not significant. When the size of result data is larger than about 200 kB, the effect gets stronger. The difference of response time to retrieve a 1,145 KB data set for a type 1 and a type 3 query is about 1,800 ms. That is about 30 percent overhead caused by data translation. For the non-encrypted system, the difference between type 1 and type 3 (or type 2 and type 4) is smaller than in the encrypted system. Since no encryption and decryption are needed, the response times for all the types are less than that in the encrypted system.

6.3 Component Throughput

In this section, we focus on the performance of the three key components of SACE, namely the information-sharing stub at the requesting organization, the middleware at the mediator, and the stub at the responding organization. We measure the throughputs (in terms of queries per second) of the three components as follows. First, for each component, we take into account the time consumed by every operation performed by the component during the life-time of a query. In particular, the main operations are as follows. At the requesting organization, each SQL statement is parsed and translated into a semantic query. Data conversions are involved if necessary. The translated query is then encrypted and sent to the mediator. If the query is authorized, it is translated to a semantic query against the responder's



(a)

Fig. 6. Mediator performance evaluated by access control rules

ontology. Subsequently, the translated query is sent to CIA. Once CIA gets the query, it first decrypts the query and converts some values if necessary. Then, the translated semantic query is translated into a syntactic query against CIA's schema. Finally, the query results are encrypted and returned, and the requesting organization will decrypt the results. Second, we get the throughput measurements, we inject 40 queries on behalf of FBI agents (where the four query types are mixed, each type has 10 queries), then we measure the total processing time (i.e., both the time consumed in processing the request and the time consumed in processing the results are counted) consumed by each component. For example, if in total it takes t seconds for the mediator to process n queries, the mediator's throughput is n/t. The comparison between the encrypted SACE and the non-encrypted SACE systems will be shown in our results. Note that here SACE implementation is slightly different from the minimum-middleware architecture.

6.3.1 *The Result Data Size and The Number of Users.* In Fig. 5(a), the non-encrypted system outperform then encrypted system. When the query result size is less than 100KB, the non-encrypted system could be 5 times faster in both requesting and responding stubs. Even when the result size is larger than 1MB, the non-encrypted system is still 3 to 4 times faster. Note that there is neither encryption nor decryption computation needed in mediator. The mediator can process both encrypted and non-encrypted query. Therefore, the performance of mediator in both systems are close. As the result data size increased, the throughputs for both organizations drop. The mediator, however, is unaffected by the result data

size because it will not process the result data. Fig. 5(b) shows the impact of the number of user on the throughput of each component. When the users in the system increases from 5 to 50, the throughput of FBI dropped from 9.71 queries per second to 1.34 queries per second; the throughput of CIA dropped from 9.96 queries per second to 1.23 queries per second in the encrypted SACE system. Comparing to the throughput of FBI dropped from 188.32 queries per second to 149.54 queries per second and the throughput of CIA dropped from 81.9 queries per second to 63.53 queries per second, the difference between the performance of encrypted SACE and non-encrypted SACE of the requesting stub and responding stub is huge. Notice that, in both FBI and CIA, the throughputs are roughly the same when the number of user is less than ten. That is because the system can handle small number of users very quickly, there is almost no congestion due to those users simultaneously submit requests. However, the throughput of mediators in both systems are very close. Therefore, the mediator becomes the bottleneck in the non-encrypted SACE system. Since SACE is a very flexible system, we can add more mediators to the system to improve the overall performance. Here we only test one mediator and two organizations. But the architecture can be easily extended to multi-organization and multi-mediator architecture. For mediators, more organizations means more users, more ontologies and more entries in mapping tables. Load balancing can be applied when there are multiple mediators are available.

6.3.2 *The Mediator.* Fig. 6(a) shows the impact of the number of access control rules on the mediator's performance. The processing time increases linearly as the number of the access control rules increases. It is because the mediation process is a linear algorithm. We also performed the same test on the mediator to process non-encrypted data. As expected, the mediator is not significantly affected where processing encrypted data. Finally, we use RDQL in the Jena toolkit to query the ontology files, and our experiments show that the size of ontology files has almost no impact on the processing time of the mediator as we doubled the ontology size.

7. RELATED WORK

To the best of our knowledge, there exists no prior research that enables semantic access control across heterogeneous information sources based on ontologies while preserving the privacy of the metadata of the information sources. Qin and Atluri introduced concept-level semantic access control for the semantic web [Qin and Atluri 2003]. Their work deals with how terms naming resources (whose access is being controlled) can be rewritten using other terms subject to logical rules expressed using OWL (Web Ontology Language) [Bechhofer et al.]. Qu, et al. [Qu et al. 2004], have presented an ontology-based rights expression language built on top of OWL to express access rights of resources. Damiani, et al. [Damiani et al. 2004] have discussed how policy languages can be extended for the semantic web. Agarwal and Sprick [Agarwal and Sprick 2004; Agarwal et al. 2004], and Yague and Troya [Yague and Troya 2002; Yague et al. 2003] have presented frameworks for access control policies for semantic web services.

There is a rich literature on access-control in information interoperation systems. We mention a selected few that are the most closely related to our work. Gong and Qian [Gong and Qian 1996; 1994] have discussed the complexity and composabil-

ity issues in secure interoperation. Ahn and Mohan [Ahn and Mohan 2004] have implemented RBAC-based information sharing on a syntactic level. De Capitani di Vimercati, and Samarati have shown how authorization specification and enforcement can be implemented in federated database systems [De Capitani di Vimercati and Samarati 1997].

Dawson, Qian, and Samarati, discuss how security can be provided while enabling interoperation of heterogeneous systems [Dawson et al. 2000]. They use query folding to resolve the semantic heterogeneity of the information sources. The rules that the query folding is based on is manually expressed. Furthermore, their system does not have provisions to preserve the confidentiality of the metadata of the information sources. SACE uses ontologies that are used to expand queries semantically by considering the synonyms of terms. First, when an immediate match does not exist between terms in the two ontologies, we allow the narrowing of the semantic scope of a term. Second, SACE pursues similar narrowing when a role in one organization does not have a direct counterpart in another organization from which data is being sought. Third, SACE has robust privacy-preserving features and allows for the complete confidentiality of the metadata of the information sources.

8. CONCLUSION

In this work, we have demonstrated SACE — a semantic access control enabling system. SACE provides maximum privacy and security for data, queries and metadata and enables interoperation among heterogeneous information sources. SACE incurs only a minor performance degradation in comparison to non-secure interoperation systems. We show that SACE is easy to setup without requiring any major modification of existing websites and their underlying database systems and that it scales seamlessly.

REFERENCES

Data encryption standard (DES). *FIPS PUB 46-3*.

1999. Resource description framework(rdf) model and syntax specification, w3c recommendation http://www.w3.org/tr/rec-rdf-syntax.

Agarwal, S. and Sprick, B. 2004. Access control for semantic web services. In *International Conference on Web Services (ICWS '04)*. IEEE Computer Society Press.

Agarwal, S., Sprick, B., and Wortmann, S. 2004. Credential based access control for semantic web services. In *2004 American Association for Artificial Intelligence Spring Symposium Series*.

Ahn, G.-J. and Mohan, B. 2004. Secure sharing role-based delegation. *Journal of Network and Computer Applications*.

Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., and Stein, L. Owl web ontology language reference. Tech. rep., W3C.

Damiani, E., De Capitani di Vimercati, S., Fugazza, C., and Samarati, P. 2004. Extending policy languages to the semantic web. In *ICWE*. 330—343.

Dawson, S., Qian, S., and Samarati, P. 2000. Providing security and interoperation of heterogeneous systems. *Distribute Parallel Databases 8,* 1 (January), 119—145.

De Capitani di Vimercati, S. and Samarati, P. 1997. Authorization specification and enforcement in federated database systems. *Journal of Computer Security 5,* 2, 155—188.

Ferraiolo, D. and Kuhn, D. 1992. Role based access control. In *15th National Computer Security Conference. Available from http://csrc.nist.gov/rbac/ on Aug. 30th, 2004.*

Gong, L. and Qian, X. 1994. The complexity and composability of secure interoperation. In *IEEE Symp. Security and Privacy*.

Gong, L. and Qian, X. 1996. Computational issues in secure interoperation. *IEEE Transactions Software Engineering 22,* 1, 43—52.

Liu, P., Mitra, P., and Pan, C.-C. 2004a. Privacy-preserving semantic access control across heterogeneous information sources. Tech. rep., School of Information Sciences and Technology, The Pennsylvania State University. November.

Liu, P., Mitra, P., and Pan, C.-C. November 2004b. Privacy-preserving semantic access control across heterogeneous information sources. Tech. rep., School of Information Sciences and Technology.

Qin, L. and Atluri, V. 2003. Concept-level access control for the semantic web. In *Workshop on XML Security, held in conjunction with the 10th ACM Conference on Computer and Communications Security*.

Qu, Y., Zhang, X., and Li, H. 2004. An ontology-based rights expression language. In *13th international World Wide Web conference on Alternate track papers & posters Poster, (WWW, Alt. 04)*. ACM Press, New York, NY, USA, 324—325.

Yague, M., Mana, A., J., L., and Troya, J. M. 2003. Applying the semantic web layers to access control. In *Web Semantic Workshop, DEXA 2003 Conference*. IEEE Computer Society Press.

Yague, M. and Troya, J. 2002. A semantic approach for access control in web services. In *Euroweb 2002 Conference. The Web and the GRID: from e-science to e-business, British Computer Society, World Wide Web Consortium*. 483—494.

...