

# Privacy-preserving Semantic Interoperation and Access Control of Heterogeneous Databases

Prasenjit Mitra, Chi-Chun Pan, Peng Liu  
Penn State University  
University Park, PA 16802, U.S.A.  
pmitra@ist.psu.edu

Vijay Atluri  
Rutgers University  
Newark, NJ 07102, U.S.A.  
atluri@rutgers.edu

## ABSTRACT

Today, many applications require users from one organization to access data belonging to another organizations. While traditional solutions offered for the federated and mediated databases facilitate this by sharing *metadata*, this may not be acceptable for certain organizations due to privacy concerns. In this paper, we propose a novel solution – *Privacy-preserving Access Control Toolkit* (PACT) – that enables privacy-preserving secure semantic access control and allows sharing of data among heterogeneous databases without having to share metadata. PACT uses encrypted ontologies, encrypted ontology-mapping tables and conversion functions, encrypted role hierarchies and encrypted queries. The encrypted results of queries are sent directly from the responding system to the requesting system, bypassing the mediator to further improve the security of the system. PACT provides semantic access control using ontologies and semantically expanded authorization tables at the *mediator*. One of the distinguishing features of the PACT is that it requires very little changes to underlying databases. Despite using encrypted queries and encrypted mediation, we demonstrate that PACT provides acceptable performance.

## 1. INTRODUCTION

Today, more and more applications involve information access across databases (*information sources*) owned by different organizations. The most two common approaches to accomplish this are solutions offered in the area of *federated databases* and solutions employing *mediators*.<sup>1</sup> Although these can handle issues such as *data type differences*, *value differences*, *semantic differences* and *missing values*, these are extremely limited in handling the increasing need of protecting the *privacy* and *confidentiality* of the *metadata* while allowing such information accesses.<sup>2</sup>

<sup>1</sup>Here, we assume data from several sources are not stored in a single database or data warehouse.

<sup>2</sup>Here, confidentiality concerns disclosing information about the metadata to an outsider who is not involved in the in-

Federated database systems implement one-to-one connections between all pairs of databases that need to talk to each other. These connections allow one database system  $D_1$  (or employees of one organization or entity  $E_1$ ) to query another  $D_2$  (owned by another organization  $E_2$ ) in terms that  $D_2$  can understand. Federated database systems inherently require that  $D_1$  and  $D_2$  reveal their *data schema* (and the associated semantics), a main type of metadata, to each other. However, this requirement may raise serious privacy concerns when “there is an increasing need for sharing information across autonomous organizations in such a way that no information apart from the answer to the query is revealed” [4]. For example, organizations like FBI and CIA may never want to reveal their metadata and divulge crucial information about what information is stored in their sources. Moreover, storing the schema on more systems obviously increases the threat to the confidentiality of the schema.

When mediator-based systems are used to support information access across heterogeneous databases<sup>3</sup>, a *mediator* trusted by  $E_1$  and  $E_2$  generates and stores a *mapping* between the schemas of  $D_1$  and  $D_2$  to resolve the semantic heterogeneity. Essentially, mediators know data schemas. Although a privacy control policy can be enforced by the mediators, such a solution has to rely on *fully trusted* and highly secure mediators to preserve the privacy and confidentiality of metadata. And 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 unlikely to be deployed. (Fundamentally, the more trust you assume, the more vulnerable the system.) Nevertheless, the above discussion shows that preserving the privacy of metadata while enabling semantic interoperation is a difficult problem, since, often, the technologies proposed for enabling semantic interoperation depend heavily on insecure mediation based on the metadata.

In this work, we present PACT – Privacy-preserving Access Control Toolkit – a new mediation solution for information access across heterogeneous databases. PACT seeks

formation sharing setting, while privacy concerns disclosing information about the metadata to a party who is legitimately involved in the information sharing setting.

<sup>3</sup>Mediators are also used for information integration [10], but the integration aspect is out of the scope of this paper.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

to solve the aforementioned problem. To the best of our knowledge, PACT is the first framework that can preserve the privacy of metadata while enabling semantic interoperation. Besides preserving the privacy of metadata, another key contribution of PACT is *semantic access control* which we will explain shortly.

It is important to note that PACT is very different from secure multi-party computation originated by Yao [22], and (relational) database specific secure multi-party computation solutions proposed in [4, 14]. While the focus of this research is on how to allow  $E_1$  and  $E_2$  to integrate their data in such a way that a function  $f(x, y)$ , where  $x \in D_1$  and  $y \in D_2$ , can be computed by both  $E_1$  and  $E_2$  without anything else about  $x$  or  $y$  being revealed. In contrast, PACT focuses on metadata instead of data; and PACT focuses on information access instead of information integration.

PACT has several unique properties:

⊙ *PACT achieves ‘blind’ mediation using encrypted metadata.* All the metadata used by a mediator are encrypted and the mediation algorithm does not require any decryption, yet semantic mediation can be correctly done without any human intervention (in runtime). In this way, PACT preserves privacy of metadata.

⊙ *PACT greatly reduces the trust requirements on the mediator.*  $E_1$  and  $E_2$  no longer need to trust that the mediator will not disclose their metadata (without authorization). The mediator is not required to be highly secure either. This property makes PACT a *semi-trusted* third party approach to information sharing instead of a trusted third party approach.<sup>4</sup> This makes PACT a very practical solution.

⊙ *PACT exploits semantic mediation to enhance privacy.* As we mentioned above, mediation typically compromises privacy, however, in PACT, mediation is exploited to yield more privacy. This is achieved via a novel *schema obfuscation* technique we will present shortly in Section 4, where *synonyms* in *ontologies* are used to ‘obfuscate’ attributes in schemas and other metadata. This property means that the terms in a database are never shared outside the database but secure interoperation can be achieved without having to share any metadata with other organizations.

⊙ *PACT ‘outsources’ access control to the mediator.* As a result, if a query from  $E_1$  will be denied by  $E_2$ , the query can be denied much earlier at the mediator. In this way, PACT enhances the performance of the system by avoiding a roundtrip to the responding database for queries that will be eventually denied or that need to be rewritten because the responding database allows only partial access to the data being requested. This is achieved via a novel *semantic access control* scheme that allows PACT to seamlessly integrate semantic mediation and access control. We will present this scheme in Section 4.

⊙ Despite the costs of key management and encryption, our experiments show that the overhead of enabling secure interoperation is quite small and we achieved high throughput of the queries while using PACT. The overhead of symmetric encryption is negligible; and many if not most PACT mediation operations can be directly performed using the mapping tables stored at the Mediator. The only performance bottleneck might be that PACT deploys a single mediator, however, the mediator can easily be replicated to remove

<sup>4</sup>Note that PACT assumes that the software at the mediator has not been tampered and it runs the mediation algorithm faithfully.

that impediment.

## 2. PRELIMINARIES

In this section, we introduce some preliminary concepts used in the rest of the paper.

### 2.1 Data Schema and Ontologies

We denote the data schema for an organization as  $\{T_1, T_2, \dots, T_m\}$ , where  $T_i$  is a table (or relation) denoted as  $T_i(a_{i1}, a_{i2}, \dots, a_{ik})$ . Each  $a_{ij}$  is an *attribute* of  $T_i$ .

We assume that associated with each information source is an *ontology* that specifies the relationships among the terms used in the information source. We use the Web Ontology Language (OWL) [6] to express our ontologies. In our model, an *ontology* is a set of *concepts* or *terms* (denoted  $C_1, C_2, \dots, C_L$ ) that have three types of relation among them:  $\{equivalentClass, subClassOf, differentFrom\}$ . Two classes related by *equivalentClass* “have the same instances” [6]. The relation *subClassOf*( $C_i, C_j$ ) means that the semantic scope of  $C_i$  is narrower than that of  $C_j$ , or  $C_j$  is *broader* than  $C_i$ . The relationship *differentFrom* is used to indicate that two classes or individuals are different. Among the 3 relations, *equivalentClass* and *subClassOf* are transitive, but *differentFrom* is not. We use the terms “concepts” and “terms” interchangeably in the rest of the treatise.

To enable information access across heterogeneous information sources, within each organization, an in-house process generates (semantic) mappings between its database (i.e., table names and attribute names in the database schema) and its ontology – using a toolkit deploying existing schema and ontology mapping techniques (see [19] for a survey) – and stores this *database-ontology mapping* for future use. Due to this mapping, attributes in data tables are not necessarily always part of an ontology.

We assume each organization  $E$  has a set of *employees* or *users*. Each user is authorized to access part of  $E$ ’s database, so the user usually knows only part of  $E$ ’s schema. Similarly, we assume each user  $U$  uses a specific *user ontology* which corresponds to the part of  $E$ ’s ontology that  $U$  can access. Due to privacy protection, we assume the employees of one organization never know the schema of another organization.

### 2.2 Role-based Access Control

PACT uses role-based access control (RBAC). We assume that each database enforces its own RBAC policy. An user must be associated with at least a *home organization* which identifies the user and assigns the user one or more roles. We denote an access control policy as  $\{R, R \rightarrow P\}$ , where  $R$  is a *role hierarchy*,  $P$  is the set of *privileges*, and  $R \rightarrow P$  is a mapping from  $R$  to  $P$ . A privilege  $p_i$  is denoted as  $\{object, action, sign\}$ . We assume the *object* be an attribute of the table  $T_i.a_{ij}$  or a table  $T_i$ ; the *action* is **select**, **update**, **insert** and **delete**; and *sign*  $\in \{+, -\}$ . ‘+’ means a positive privilege (allow access) and ‘-’ means negative (deny access). Each role can be mapped to multiple privileges. Each organization maintains the following tables in its database: (a) a subject-role assignment table, (b) a privileges table, and (c) a role-privileges assignment table. The subject-role assignment table lists users and their roles, e.g., (Bob, manager) indicates that the user Bob has the role of “manager”. The role-privileges assignment table lists tuples of the form (role,  $p_i$ ), e.g., (manager,  $p_i(\text{employeeTab}, \text{select}, +)$ ) indicates that the role “manager” have **select**

access to the table “employeeTab”.

For clarity, in the next two sections we assume that there are no negative privileges. Nevertheless, note that our framework can be easily extended to support negative privileges.

### 3. SYSTEM ARCHITECTURE

The architecture of PACT is shown in Figure 1. PACT has two phases: the *offline* phase – the initial processing that takes place before any query is processed; and the *online* phase, which shows how an inter-organization query is processed in runtime.

PACT is a middleware system that requires very few changes to be done on the legacy systems of any organizations involved. The offline procedure of PACT is to (1) translate the (syntactic) access control policy of each organization to a semantic access control policy against the organization’s ontology, and (2) prepare the other metadata used by the mediator.

To illustrate the online aspect of PACT, suppose an employee of Organization A needs some information from organization B. In Step 1, since the user does not know B’s data schema, the user’s SQL query is written against the user’s user ontology. In this way, the actual column and table names used in the query will be ‘obfuscated’ by A’s ontology. Then the obfuscated query will be encrypted.

In Step 2, a SQL parser is used to “decompose” the query into several column-level or table-level access requests. 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 3, the mediator translates these requests into several semantic accesses requests expressed with B’s ontology and role lattice via an algorithm called *semantic request mediation*. This algorithm uses encrypted mappings between terms in A’s ontology and B’s ontology and the mapping between roles in A’s role lattice and B’s role lattice. In Step 4, these requests are checked against B’s semantic access control policy. In Step 5, the filtered yet authorized semantic requests will be decrypted and translated into some syntactic access requests against B’s schema. In Step 6, the SQL query is processed by B’s DBMS. The DBMS may forgo the security checking since it has already been done. However, the query results cannot be directly returned to A because they are not expressed against A’s ontology and the user can be confused about the meaning of the results. In Step 7, the responder translates the data and sends it back to the user.

From the privacy preserving perspective, a key feature of PACT is that all the metadata stored and used at the mediator, which include the ontologies of both organizations, the ontology-mapping table and the role-mapping table, are encrypted and the mediator cannot decrypt them. In this way, PACT removes the requirement that the mediator must be trusted not to disclose any sensitive metadata, and good privacy and confidentiality can be preserved even if the mediator is hacked.

*Remark.* Although the Access Controller may be moved into organization B to further enhance the privacy of access control policies, outsourcing access control to the mediator may substantially improve the system’s performance, as we will explain shortly in Section 4.

## 4. CORE TECHNIQUES

In this section, we present the set of core techniques used by the PACT system and demonstrate their uniqueness and merits. First, we discuss the offline operations of PACT. Second, we discuss the runtime operations of PACT. Although for clarity we only address the scenario with two organizations, PACT can easily handle multiple organizations with one or more mediators.

### 4.1 Offline Processing

In this section, we show how each piece of the metadata used by the mediator (shown in Figure 1) is prepared. First, organization A’s ontology (B’s ontology) is prepared by an in-house process to include the terms used in  $D_A$  ( $D_B$ ), in the way we mentioned in Section 2. Then A’s ontology (B’s ontology) will be encrypted by organization A (B) using a specific *master key* denoted  $K_A$  ( $K_B$ ) as follows<sup>5</sup>: all and only the terms are encrypted, none relation among them is. Note that in a PACT system a master key (e.g.,  $K_A$ ) is only known to its owner (e.g., organization A), no one else. The encrypted version of A’s ontology is the version stored at the mediator.

Second, the *ontology-mapping table* may be generated in either a semi-automated way or a totally automated way. In the first case, an ontology *matcher* (i.e., a human expert trusted by both organization A and B) will be able to access both A’s ontology and B’s ontology in cleartext. He understands the semantics of both ontologies; he will use specific ontology mapping techniques [19] to map; and he will generate certain entries or rules in the mapping table which we will explain shortly. Then the resulted mapping table will be encrypted as follows: for each entry, all and only the terms from A’s ontology are encrypted using  $K_A$ ; all and only the terms from B’s ontology are encrypted using  $K_B$ . The encrypted version of this mapping table is the version stored at the mediator. It is important to note that (1) the human expert has nothing to do with the mediator (software); they are totally different entities; (2) the expert can instantly “forget” the mapping table once it is generated because runtime mediation does not need the expert at all.

Compared with the semi-automatic method, totally automatic ontology-mapping table generation does not need a trusted human expert, so it achieves more privacy, although sometimes it may not achieve perfect accuracy (of mapping). Automatic ontology-mapping techniques are well studied [19]. In particular, in [16] we developed a privacy-preserving ontology-mapping scheme where two encrypted ontologies can be automatically matched through a peer-to-peer protocol without (a) revealing the terms of any ontology or (b) involvement of any third parties. (The main idea is to leverage commutative encryption techniques.)

It should be noticed that ontology-mapping is **not** schema-mapping. To enhance privacy, we do not map A’s schema to B’s schema directly. Hence the terms used in the ontology-mapping table are typically just a synonym of a schema term. This obfuscation technique will be further addressed in Section 4.2.

Finally, the *role-mapping table*, which maps organization A’s role hierarchy to B’s role hierarchy (and vice versa), can be generated in a similar way.

**Encrypted ontology mapping rules.** The ontology-map-

<sup>5</sup>Note that we use symmetric key cryptography.

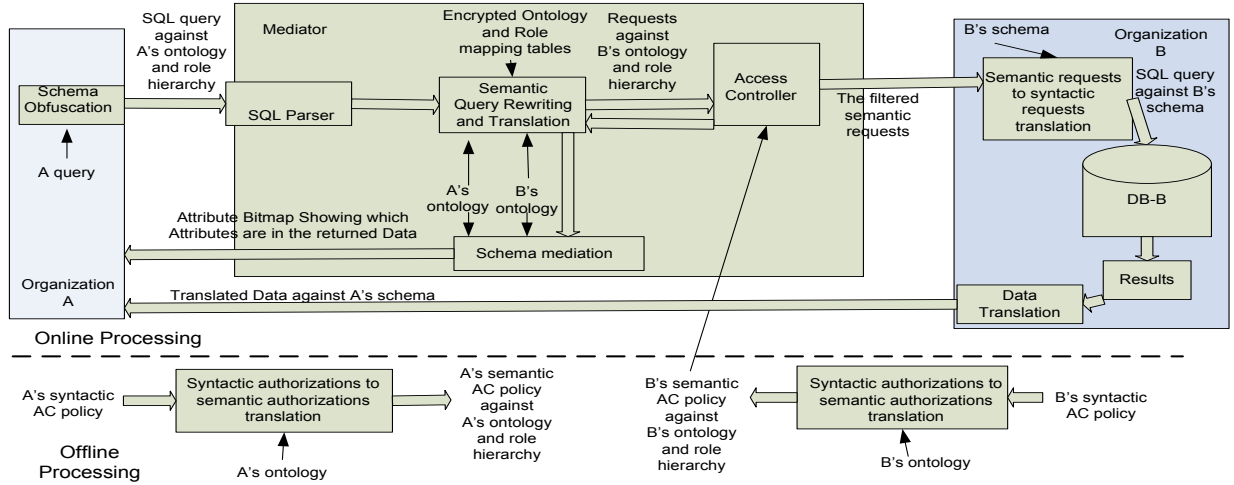


Figure 1: Architecture of PACT

ping rules used in our system are of three forms:

(1) *Binary Mapping Rules*: A binary mapping rule is a triple listing two concepts from two different ontologies and the relationship among them. For example, an entry in the mapping table may be the triple:

$(O1.E_{K_1}(\text{Vehicle}), O2.E_{K_2}(\text{Automobile}), \text{OWL.subClassOf})$

Here,  $O1$  ( $O2$ ) denotes the ontology of organization 1 (2), and  $K_1$  ( $K_2$ ) is the master key of organization 1 (2). This entry indicates that  $O2.Automobile$  is a *subClassOf* (as defined in the namespace OWL [6]) of  $O1.Vehicle$ .

(2) *Split/Merge Attribute Rules*: Consider the example, where Organization 1 uses an attribute *Name* and that is split into two attributes—names *FirstName* and *LastName* in the information system of Organization 2. To capture such rules, we use the “split/merge attribute” rules, e.g.,

$(O1.E_{K_1}(\text{Name}),$

$\text{Merge}(O2.E_{k_2}(\text{FirstName}), O2.E_{k_2}(\text{LastName}))).$

(3) *Mapping Functions*: These functions are used to convert data values. For example, a conversion function between *Dollar* and *PoundSterling*, *Metre* and *Feet*, etc.

(4) *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 requesting organization’s query will be translated into this SQL query or its variant; (ii) A table,  $T$ , in the requestor’s ontology and used in the requestor’s query, and, (iii) Binary Mapping rules, Split/Merge Attribute rules or Mapping Functions mapping all the results of the SQL query (in (i)) to attributes in table  $T$ .

**Example 1.** A complex mapping rule is of the following form:

Concept in  $O1$ :  $E_{K_1}(\text{LuxuryCar})$   
 Mapped to query: **select**  $c.E_{k_2}(\text{CarID}), c.E_{k_2}(\text{Make}), c.E_{k_2}(\text{Model}), c.E_{k_2}(\text{Year}), p.E_{k_2}(\text{Price})$  **from**  $E_{k_2}(\text{Car})$   $c, E_{k_2}(\text{Prices})$   $p$  **where**  $c.E_{k_2}(\text{ID}) = p.E_{k_2}(\text{ID})$  **and**  $p.E_{k_2}(\text{Price}) > 40,000$

Mapping details:

$O1.E_{k_1}(\text{LuxuryCar}).E_{k_1}(\text{VehID}) = O2.E_{k_2}(\text{Car}).E_{k_2}(\text{CarID})$   
 $O1.E_{k_1}(\text{LuxuryCar}).E_{k_1}(\text{Mk}) = O2.E_{k_2}(\text{Car}).E_{k_2}(\text{Make})$   
 $O1.E_{k_1}(\text{LuxuryCar}).E_{k_1}(\text{Md}) = O2.E_{k_2}(\text{Car}).E_{k_2}(\text{Model})$

$O1.E_{k_1}(\text{LuxuryCar}).E_{k_1}(\text{Yr}) = O2.E_{k_2}(\text{Car}).E_{k_2}(\text{Year})$   
 $O1.E_{k_1}(\text{LuxuryCar}).E_{k_1}(\text{Pr}) = \text{ConvertEuro2Dollar}(O2.E_{k_2}(\text{Prices}).E_{k_2}(\text{Price}))$

This rule indicates that 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. We also show that the attribute  $O2.Prices.Price$  needs currency conversion using a provided function.

Finally, we postpone the discussion on how syntactic to semantic access control policy translation is done to the next section.

## 4.2 Realtime Query Processing

We now show how a query from the *requestor* organization is mediated and then answered by the *responder* organization in PACT.

### Step 1: Schema Obfuscation and Query Encryption

When a query is issued by an employee of the requestor organization, the query is written against the employee’s user ontology instead of the requestor’s schema. In this way, the requestor’s schema is kept private. We refer to this process as *schema obfuscation*. Conceptually, schema obfuscation replaces a schema term in the query with a *randomly* chosen synonym in the user ontology.

Besides enhancing privacy, schema obfuscation has some side benefits. (1) It increases the extensibility and maintainability of the system, because the database and its schema may be changed and new terms can be added without requiring a change in the ontology-mapping table used by the mediator, as long as the newly added database term can be mapped to a term in the existing ontology. (2) Avoiding the tasks to update the ontology-mapping table yields more security because the system is the most vulnerable when the human expert is involved.

Similar to schema obfuscation, PACT also performs *role obfuscation*.

After the query is “obfuscated”, it will be encrypted using two keys: (a) the requestor’s master key; and (b) a specific

session key (denoted  $K_s$ ) negotiated by the requestor and the responder sometime before <sup>6</sup>.

**Example 2.** Say an user at FBI poses a query:

**select** name, salary, address **from** employee **where** salary > 70000;

She wants information from the CIA database. This query is posed using the terms *salary*, *name*, *address* and *employee* from FBI's ontology. Note that the ontologies of both FBI and CIA are shown in Appendix A.

**Query encryption:** The FBI-side PACT stub encrypts *all* the obfuscated schema terms (e.g., *employee* and *salary*) in the query using FBI's master key, denoted  $K_{FBI}$ , and *all* the values (i.e., 70000) in the query using  $K_s$ , the session key, to derive the query below. Note that  $K_s$  cannot be used to encrypt the terms in the query because otherwise the responder can know the requestor's terms after he eavesdrops the query.

**select**  $E_{K_{FBI}}$ (name),  $E_{K_{FBI}}$ (salary),  $E_{K_{FBI}}$ (address) **from**  $E_{K_{FBI}}$ (employee) **where**  $E_{K_{FBI}}$ (salary) >  $E_{K_s}$ (70000).

**Step 2: SQL Query Parsing.** When an encrypted query arrives at the mediator, it is parsed and all the table and column names are extracted. The mediator expands wildcards (like **select \***) by replacing the wildcard character using the encrypted attributes associated with the table in the query.

### Step 3: Encrypted Query Rewriting

This "decomposed" query is then sent to the mediator along with the role of the user, say  $E_{K_{FBI}}$ (Agent), posing the query. Now, the mediation algorithm (shown in Algorithm 1) is used to rewrite the query so that the semantic heterogeneity between organization A's schema and B's schema can be resolved. Note that Algorithm 1 calls the function *RewriteTerms* in Algorithm 2 to rewrite terms in a query. Note also that the mediation algorithm does not need any decryption.

To illustrate, let's revisit Example 2. First, the mediator searches for an equivalent table of the table  $E_{K_{FBI}}$ (employee). Let us assume that this search fails. Next, the algorithm tries to find mapping rules with tables in the responder's ontology that have been established as *subClassOf* of the table  $E_{K_{FBI}}$ (employee). Say, the tables  $E_{K_{CIA}}$ (manager) and  $E_{K_{CIA}}$ (staff) have been matched to be *subClassOf*  $E_{K_{FBI}}$ (employee). Furthermore, an attribute merge/split rule specifies that  $E_{K_{FBI}}$ (name) is the merge of the attributes  $E_{K_{CIA}}$ (firstName) and  $E_{K_{CIA}}$ (lastName) found in both tables  $E_{K_{CIA}}$ (manager) and  $E_{K_{CIA}}$ (staff). Another binary mapping rule indicates the equivalence of  $E_{K_{FBI}}$ (salary) and the attribute  $E_{K_{CIA}}$ (comp) in table  $E_{K_{CIA}}$ (manager) and the attribute  $E_{K_{CIA}}$ (pay) in table  $E_{K_{CIA}}$ (staff). Furthermore, assume FBI and CIA were agencies from different continents and  $E_{K_{CIA}}$ (pay) has units in *Dollars* and  $E_{K_{FBI}}$ (salary) is expressed in *Euros* and a conversion function *EuroToDollar* and its inverse *DollarToEuro* is provided by the expert. The attribute  $E_{K_{FBI}}$ (address) does not appear in any of the CIA tables and is thus dropped from the query and the mediator informs the requestor of this elimination. The mediator rewrites the input query as the union of the two queries:

(1) **select** Merge( $E_{K_{CIA}}$ (firstName),  $E_{K_{CIA}}$ (lastName)), DollarToEuro( $E_{K_{CIA}}$ (comp)) **from**  $E_{K_{CIA}}$ (manager) **where**

<sup>6</sup> A variety of key distribution protocols can be used here.

---

## Algorithm 1: The Mediator Algorithm

---

**Func Mediate**

**Input** : Query Q, User\_Role R, Source S, Destination D

**Output**: Ack A

**begin**

{Rewrite the Queries }

Ontology\_Table  $\mathcal{OT} \leftarrow$  lookup the ontology-mapping table between  $S$  and  $D$

{Rewrite the Roles }

MappingRoles  $\mathcal{R}' \leftarrow$  retrieve an Equivalent mapping role for  $R$  from the role-mapping table

**if** retrieval failed **then**

    Retrieve all roles  $\mathcal{R}'$  such that any role  $r'$  in  $\mathcal{R}'$  is *subClassOf*  $R$

{Rewrite the Query }

Parse query  $Q$  to identify the tables and attributes in  $Q$

Using mapping rules from  $\mathcal{OT}$  construct all possible rewritings of  $Q$  by replacing each table and attribute in  $Q$  by calling *ReplaceTerms*

{Access Control Check }

**for** each rewritten query  $Q'$  **do**

**for** each role  $R'$  in  $\mathcal{R}'$  **do**

        Check access( $Q'$ ,  $R'$ )

**if** access failed **then**

            Refine  $Q'$  by replacing each term in  $Q'$

            by its subclasses and check for access

**if** access succeeded **then**

                Insert refined query into

                AcceptedQueryList

{Handle Splitting/Merging of Attributes }

**for** each attribute  $a$  and data value  $d$  in a rewritten query  $Q'$  that needs conversion or split/merge **do**

    annotate the attribute in  $Q'$  with the

    conversion function or the split/merge function

{Error Handling and Return }

Set all bits in MissingAttributesBitMap  $\mathcal{B}$  to 1

**if** any table or selected-attribute in  $Q$  could not be rewritten because of lack of mapping rules or access control **then**

    return "Failure"

**else**

**if** any where-clause-attribute in  $Q$  could not be rewritten because of lack of mapping rules or access control **then**

        Set the corresponding bit in

        MissingAttributesBitMap  $\mathcal{B}$  to 0

    send AcceptedQueryList to  $D$

    send  $\mathcal{B}$  to  $S$

    return "Success"

**end**

---

$E_{K_{CIA}}$ (comp) > EuroToDollar( $E_{K_s}$ (70000));

(2) **select** Merge( $E_{K_{CIA}}$ (firstName),  $E_{K_{CIA}}$ (lastName)), DollarToEuro( $E_{K_{CIA}}$ (pay)) **from**  $E_{K_{CIA}}$ (staff) **where**  $E_{K_{CIA}}$ (pay) > EuroToDollar( $E_{K_s}$ (70000));

Note that the functions *Merge*, *EuroToDollar* and *DollarToEuro* are not evaluated at the mediator but just inserted into the query-text as shown above and the mediator sends the responder the executables of those functions. As indicated later, these functions are used by the responder to

---

**Algorithm 2:** Term Rewriting and Attribute Rewriting Function Handling a Single Table in the Query and its Attributes

---

**Func ReplaceTerms**

**Input** : Table  $T$ , OntologyMappingTable  $OT$ , Query  $Q$

**Output** : Ack  $A$

```

begin
  {Using Binary Mapping Rules }
  tableCounter  $\leftarrow$  0
  tableNames[tableCounter]  $\leftarrow$  Lookup  $OT$  for
  table-names  $T'$  equivalent to  $T$ 
  if lookup fails then
    for each table  $T'$  that are SubClassOf  $T$  do
      tableNames[tableCounter]  $\leftarrow T'$ 
      tableCounter  $\leftarrow$  tableCounter + 1

  for each table  $T'$  in tableNames do
    Reset attributeNames to null
    for each attribute  $a$  in  $T$  that appears in  $Q$  do
      Insert into attributeNames the attributes of
       $T'$  that are equivalent to  $a$ 
      if equivalent attributes to  $a$  are not
      available then
        Insert into attributeNames the
        attributes of  $T'$  that are subclassOf  $a$ 
        Lookup split/merge attribute rules  $R$ 
        Insert into attributeNames the attributes of
         $T'$  in  $R$  mapping to  $a$ 
      tableNames[tableCounter].attributeNames  $\leftarrow$ 
      attributeNames
      tableCounter  $\leftarrow$  tableCounter + 1

  {Using Complex Mapping Rules }
  for each complex mapping rule  $r$  that contains  $T$ 
  in  $Q$  do
    tableNames[tableCounter]  $\leftarrow$  the SQL query in
     $r$ 
    tableNames[tableCounter].attributeNames  $\leftarrow$ 
    mapping attributes in  $r$  for those attributes
    that are in  $T$  and  $Q$ 
    tableCounter  $\leftarrow$  tableCounter + 1

  return (tableNames, attributeNames)
end

```

---

perform data translation before sending the results back to the requestor.

**Role Refinement:** For our example, let us also assume that while performing role-translation, role  $E_{K_{FBI}}(Agent)$  gets translated to an equivalent CIA role  $E_{K_{CIA}}(field\_agent)$ . If  $Agent$  did not have an equivalent CIA role, the mediator would then look for CIA roles that are subclass of  $Agent$ . If such subclass roles are also not available, then the mediator would perform a breadth-first search down the FBI role-lattice to identify all subclasses of  $Agent$  and rewrites the subclass-roles with equivalent (preferred) or subclass roles (if equivalent not available) from the CIA role-lattice. For subclasses that do not have a mapping role, the search proceeds further down the role-lattice till there are no further subclass-roles. The mediator checks each rewritten query under each of these rewritten roles. If no subclass of  $Agent$  exists that has an equivalent or subclass role in the CIA role-lattice, the query is rejected. This process of role-refinement is similar to term-refinement shown in the next section.

#### Step 4: Semantic Access Control

After a query has been rewritten, PACT checks to verify if the translated role has the permissions to access the tables and columns in the rewritten queries. A novelty of PACT is that it does semantic access control at the mediator. The advantage is that the queries that are rejected are not sent to the responder. This aspect of PACT results in the following advantages: (1) Increased throughput because rejected queries are rejected early. (2) Reduced network traffic due to some queries not being forwarded to the responder. (3) Reduced denial-of-service attacks because only valid queries are forwarded to the responder. (4) Reduced processing load at the responder.

A main security concern in developing this access control scheme is whether it will violate our trust assumptions. The level of trust we assume in the mediator w.r.t metadata management and the level of trust we assume in the mediator w.r.t access control enforcement must be *consistent*, otherwise our claim that PACT greatly reduces the trust requirements on the mediator will not hold. To make our trust assumptions consistent, we have done the following:

- We do not trust that the mediator will always preserve the privacy and confidentiality of metadata. Correspondingly, we do not trust that the mediator will always preserve the privacy and confidentiality of access control policies. To achieve this aspect of consistency, (a) PACT lets the mediator use *encrypted* access control policies to do access control without knowing anything about the policies. The mediator does not know the encryption key used by the responder to “distribute” her policy to the mediator. (b) Due to schema obfuscation, the mediator does not have the actual schema terms in the access control table.
- We assume that the mediator will not purposely corrupt the encrypted ontology mapping table or the role mapping table (unless it is broken). Correspondingly, we assume the mediator will not purposely corrupt the encrypted access control policies.
- We assume that the mediator will not purposely distort the term mapping and query rewriting process (unless it is broken). Correspondingly, we assume the mediator will not collude with any organization  $A$  to get unauthorized access of data owned by organization  $B$ . Nevertheless, we assume that the mediator may abuse the way it enforces access control so that it can infer more information about the metadata of an organization. It should be noticed that the above assumption is not unrealistic, and such assumptions are often made in secure protocols that involve a third party (For example, in fair exchange protocols a semi-trusted third party can do bad things but he does not collude with any other parties). To protect organizations from mediator abuses, (1) PACT suggests the responder to *selectively* double-check the authorities of an incoming query. (2) PACT suggests auditing at each end. Both methods can effectively detect mediator abuses and greatly discourage the mediator to do “bad” things. For example, when a query does not pass authority double-checking, we know the mediator probably did something bad. Finally, note that auditing is a standard thing that every organization will usually do no matter it will help detect mediator misues or not. Also note that selective double-checking

of authorities is a light-weight procedure, especially when the responder is able to tolerate a small amount of confidentiality/privacy loss.

As indicated above, due to schema obfuscation, the mediator does not have the actual access control table from the responding database. Instead it has an access control table where the objects are the *synonyms* of the table or column names from the database, and the roles are the equivalent roles from the role-hierarchy used in the database. We refer such an access control table as the *semantic access control table*. For example, say, CIA's database-ontology mapping table indicates that the ontology term *staff* is synonymous to the database table-name *adminpersonnel*. Also, the corresponding role-table indicates the ontology role *field-agent* is synonymous with the database role *secretagent*. Let the role-privilege assignment table in the database initially contain the entry (*secretagent*,  $p_i(\text{adminpersonnel, select, +})$ ) indicating that the *secretagent* has select access to the table *adminpersonnel*. The semantic authorization table constructed and encrypted by CIA, and made available to the mediator will contain the semantically equivalent entry ( $E_{K_{CIA}}(\text{fieldagent}), p_i(E_{K_{CIA}}(\text{staff, select, +}))$ ), which means that  $E_{K_{CIA}}(\text{fieldagent})$  has read access to the attributes  $E_{K_{CIA}}(\text{pay})$ ,  $E_{K_{CIA}}(\text{firstName})$ , and  $E_{K_{CIA}}(\text{lastName})$  of the table  $E_{K_{CIA}}(\text{staff})$ . So the second rewritten query shown in Step 3 can be provided access. Upon successful access control check, the mediator forwards the query to the responder <sup>7</sup>.

Now, let us assume that the responder does not allow access to the  $E_{K_{CIA}}(\text{manager})$  table for an user with the role  $E_{K_{CIA}}(\text{fieldagent})$ . The mediator will then perform a breadth-first search on the CIA ontology to identify *subClasses* of  $E_{K_{CIA}}(\text{manager})$  and check to see if the role  $E_{K_{CIA}}(\text{fieldagent})$  has access to them. If the access-control check succeeds for one or more subclasses, the search stops checking the subtree of these subclasses anymore. All queries for which the access control check succeeded are forwarded to the responder. Say,  $E_{K_{CIA}}(\text{manager})$  has two subclasses,  $E_{K_{CIA}}(\text{firstLevelManager})$  and  $E_{K_{CIA}}(\text{ExecutiveManager})$  and the access-control table allows  $E_{K_{CIA}}(\text{fieldagent})$  to access the table  $E_{K_{CIA}}(\text{firstLevelManager})$  but not  $E_{K_{CIA}}(\text{ExecutiveManager})$ . The mediator will refine the first rewritten query to:

```
select Merge( $E_{K_{CIA}}(\text{firstName}), E_{K_{CIA}}(\text{lastName})$ ),
DollarToEuro( $E_{K_{CIA}}(\text{comp})$ ) from  $E_{K_{CIA}}(\text{firstLevelManager})$ 
where  $E_{K_{CIA}}(\text{comp}) > \text{EuroToDollar}(E_{K_s}(70000))$ ;
```

Assuming that the attribute names are the same in the tables  $E_{K_{CIA}}(\text{firstLevelManager})$  and  $E_{K_{CIA}}(\text{manager})$  and sends it to the responder. The mediator continues walking down the subclass-hierarchy of the term  $E_{K_{CIA}}(\text{ExecutiveManager})$ . If there are no more subclasses of  $E_{K_{CIA}}(\text{ExecutiveManager})$ , the query refinement process is complete.

For the complex ontology-mapping rules, PACT replaces each table in the query by the SQL query given in the complex mapping rule and the attributes in that table are replaced using the attribute mapping rules associated with the complex mapping rule.

**Example 3.** Consider the query:

```
select  $E_{K_{FBI}}(\text{name}), E_{K_{FBI}}(\text{Mk})$  from  $E_{K_{FBI}}(\text{employee})$ ,
```

<sup>7</sup>See [15] for more details about our semantic access control scheme.

```
 $E_{K_{FBI}}(\text{LuxuryCar})$  where  $E_{K_{FBI}}(\text{employee.VehID}) =$ 
 $E_{K_{FBI}}(\text{LuxuryCar.VehID})$ ;
```

Using the complex mapping shown in Example 1, and the binary mappings used to rewrite Example 2, PACT will rewrite this query to the following query:

```
select Merge( $E_{K_{CIA}}(\text{firstName}), E_{K_{CIA}}(\text{lastName})$ ),
 $E_{K_{CIA}}(\text{Make})$  from  $E_{K_{CIA}}(\text{managers}), E_{K_{CIA}}(\text{Car}),$ 
 $E_{K_{CIA}}(\text{Prices})$  where  $E_{K_{CIA}}(\text{managers.VehID}) =$ 
 $E_{K_{CIA}}(\text{Car.VehID})$  and  $E_{K_{CIA}}(\text{Car.ID}) = E_{K_{CIA}}(\text{Prices.ID})$ 
and  $E_{K_{CIA}}(\text{Prices}).E_{K_{CIA}}(\text{Price}) > \text{DollarToEuro}(40,000)$ ;
```

Note that this query is obtained by replacing *FBI.employee* by its *subClass CIA.managers* and by replacing *FBI.LuxuryCar* by the SQL query in its complex rule in Example 1. A similar query is obtained by replacing *FBI.employee* by its other *subClass CIA.staff*.

After the mediator rewrites the query and the roles and tries all possible refinements of the query by walking down the ontology tree, if the process fails to find an equivalent or contained query that is accessible to the given role, the query is not forwarded to the responder and sent back to the requestor with an error message indicating that access was denied. If a table or an attribute in a **where** clause does not have a mapping table in the responder, the query is rejected. If an attribute whose values are being selected does not have a mapping attribute in the responder, that attribute is dropped from the rewritten query and the rest of the query is forwarded to the responder. The mediator assigns a unique identifier to each rewritten query. For each rewritten query, the mediator sends an (identifier, bitmap) pair to the requestor. The bitmap indicates which attributes of the original query were dropped in the rewritten queries.

#### Step 5: Semantic to Syntactic Query Translation

At the responder, a semantic query is translated to a syntactic query by replacing ontology terms in the query with their equivalent terms that appear in the responder's database. For example, as indicated above, if the CIA database contains a table named *adminpersonnel* (equivalent to the ontology term *staff*) that has database attributes *fname*, *lname*, and *compensation* equivalent to the ontology terms *firstName*, *lastName*, and *pay* respectively, then the query:

```
select Merge( $E_{K_{CIA}}(\text{firstName}), E_{K_{CIA}}(\text{lastName})$ ),
DollarToEuro( $E_{K_{CIA}}(\text{pay})$ ) from  $E_{K_{CIA}}(\text{staff})$  where
 $E_{K_{CIA}}(\text{pay}) > \text{EuroToDollar}(E_{K_s}(70000))$ ;
```

is decrypted and translated to:

```
select Merge(fname,lname), DollarToEuro(compensation)
from adminpersonnel where compensation > EuroToDollar(70000);
```

Here, in order to preserve the simplicity of the example, we showed one-to-one binary mappings between database terms and ontology terms. However, PACT does not require the mappings between database terms and ontology terms to be one-to-one. In general, the database-ontology mappings can be any of the four types of mappings used by the mediator. However, note that if there are no database tables equivalent to the ontology tables, and instead we use database table terms that are *subClassOf* the ontology tables, the rewritten query is not equivalent to the original query but is contained in the original query.

**Step 6: Query Evaluation.** The query is then evaluated at the database and the results returned to the requestor.

**Step 7: Returning the Results.** The results of the query are sent back to the requestor after the data is translated using the split-merge attribute rules and the conversion functions. The data is sent back directly from the responder to the requestor bypassing the mediator. Consequently, the result of a query needs to be reformulated into the schema of the requestor before the results can be sent back to the requestor<sup>8</sup>. As shown above, the mediator sent the responder a query attached with hints on how to reformulate the data results using the merge/split attribute rules and data conversion functions. The responder applies the *Merge* function to merge the attribute values of the columns *firstName* and *lastName* and the function *DollarToEuro* to the columns *pay* and *comp* respectively from the two tables before sending out the results to the requestor.

We chose to send the data back via a secure and direct data channel because this choice provides maximum security. If all data goes via the mediator, the mediator becomes a bottleneck. Besides, any intruder who is able to record the history of all communication with the mediator or capture the mediator can infer characteristics of the data. Having separate secure data channels reduces the chances of such intrusion and avoids the capture of all the data by compromising one channel or one entity.

### 4.3 Correctness and Maintenance Issues

**Soundness & completeness:** The mediation algorithm (i.e., Algorithm 1) attempts to find all answers to the query posed by the end-user. If it fails to find an equivalent answer, the algorithm derives the maximally contained set of answers that are available.

**THEOREM 4.1.** *Given a query  $Q$  and target information sources  $T$ , the Mediator Algorithm used by PACT generates a sound and complete set of answers to  $Q$  for which the user posing  $Q$  has access permissions at any source in  $T$ .*

Due to lack of space, we omit the proof of the theorem.

**Maintenance:** Here, we address the overhead implied by metadata updates, when an update to the database schema or roles occurs. First, PACT works seamlessly in the presence of changes to data because PACT does not materialize any views at the mediator. Second, when an element is removed from the database schema, no update to the encrypted ontology mapping table or the ontologies (used by the Mediator) is needed.

Third, when a new element is added to the database schema, if the ontology mapping table only contains the synonyms of schema terms, the corresponding term (of the new element) needs to be added to the table. Nevertheless, to add this new term, light-weight incremental maintenance is enough (in both automated ways and semi-automated ways), and we do not need to redo the whole ontology-matching process. Moreover, to achieve more obfuscation, people may want to put additional term-mapping rules beyond schema terms into the ontology mapping table. In this case, if the new schema element is already covered by the additional term-mapping rules, no maintenance is needed.

Fourth, changes to the access-control policy at an information source mandates updates at the mediator but we expect such changes to be fewer than changes to the data and the schema of information sources. Finally, note that

<sup>8</sup>Note that in Step 3 the mediator already compiled the matched terms into the results' schema (see Figure 1.).

an organization may periodically refresh her master key for more privacy.

## 5. SECURITY AND PRIVACY ANALYSIS

In this section, we analyze the extent to which PACT can protect the privacy & confidentiality of metadata. Since both privacy and confidentiality of metadata are threatened by the attempts or attacks to *infer* the encrypted metadata stored on a mediator, we do the analysis in a uniform way. (Note that we have already investigated whether PACT's semantic access control scheme can allow unauthorized accesses in [15]).

We assume the attacker's goal is to infer 4 types of information: [High sensitivity] the data schemas; [Medium sensitivity] the access control policies; [Low sensitivity] the ontology and role hierarchy of each organization. We also assume the attacker uses a *dictionary* of size  $N$  to infer the above information.

Accordingly, we are interested in 3 representative types of *inference* attacks: [Mode A: ] The attacker breaks into the Mediator without any prior knowledge about the two organizations. [Mode B: ] The attacker breaks into the Mediator with a history (log) of the queries mediated by the Mediator some prior knowledge about the two organizations. [Mode C: ] Besides Mode B, the attacker breaks into one of the two organizations as well. Note that the case where the Mediator infers the 4 types of sensitive information is equivalent to Mode A.

In the following, we show that PACT can provide very good privacy. For simplicity, we assume the attacker's dictionary contains every term embedded in the 4 types of metadata together with many other terms.

**Mode A.** In Mode A, since no key is known by the Mediator and no key is used during any mediation procedure – we assume a secure key distribution protocol is used – decrypting the 4 types of metadata stored on the Mediator is almost impossible. Since the attacker does not have any prior knowledge about the two organizations, he can only rely on his dictionary.

First, one possible inference attack is to exploit the *equivalentClass* and *subclassOf* relations among encrypted concepts to infer an ontology. Although concepts are encrypted in every ontology, their relationships are not. Hence, a graph with the encrypted concepts as a node and the *equivalentClass* and *subclassOf* relationships as an edge can be constructed for one ontology or two ontologies “connected” by a mapping table. Similarly, such graphs can be constructed among the cleartext concepts in the attacker's dictionary. Then the attacker can match these two types of graphs and the matches can give the attacker better knowledge about the encrypted ontologies.

Nevertheless, since  $N$  (the size of the attacker's dictionary) is typically much larger than  $L$  (the size of an ontology), the *False Guess Probability* (the probability that the attacker makes a wrong guess about the ontology) is typically very high. Assume the graph built from an encrypted ontology matches  $X$  graphs built from the dictionary, the False Guess Probability is:  $P_{fg} = 1 - 1/X$ . Here  $X$  is a number between 1 and  $\binom{N}{L}$ . Note that  $X$  is usually a large number.

Using the ontology mapping table to correlate or “merge” the two ontologies for Organizations A and B before the

matching may help reduce  $P_{fg}$  by decreasing the upper bound of  $X$  to  $\binom{N}{L_A + L_B}$ . However, the attacker would have difficulty in telling which concepts belong to which ontology when a good match is obtained.

Second, a role hierarchy can be inferred in a similar way because an encrypted role hierarchy tells the true relationships among encrypted roles.

Third, the probability that the attacker can guess a permission  $p_i$  (of an access control policy) correctly is dependent upon the probability that he can guess an ontology correctly, namely  $1 - P_{fg}$ .

Finally, the probability that the attacker can guess an attribute of the database schema of an organization correctly is:  $P_{ds} = (1 - P_{fg}) \times P_{so}$ . Here  $1 - P_{fg}$  is the probability that the attacker can guess the organization’s ontology correctly. However, due to schema obfuscation, even if the attacker guesses the ontology correctly, he still needs to guess the “real” schema attribute involved in the original query among a set of equivalent terms (or concepts) that are used to obfuscate the attribute; and  $1 - P_{so}$  is the corresponding false guess probability. Assume the average size of an equivalence class (of concepts) in an ontology is  $K$ , then  $P_{so} = 1/K$ .

**Mode B.** In Mode B, we assume the attacker knows the identity of each organization. Furthermore, the attacker has good pre-knowledge about the role hierarchy and ontology of the organization. In addition, we assume the attacker has a log of the previous queries.

Now, inferring an ontology (or a role hierarchy) is much easier not only because the attacker’s pre-knowledge can allow him to do better ontology graph matching, but also because the attacker can do frequency-based attacks. By monitoring a history of requests across organizations, the attacker knows the most frequently used terms (though encrypted) and roles, among others. Such frequencies can be matched against the attacker’s pre-knowledge about term (or role) usage frequencies to infer the cleartext of an encrypted term. After a couple of key terms are inferred, the whole ontology (graph) can be inferred with good accuracy, since the conditional inference probability of a term based on some known, relevant terms is typically much higher than  $P_{fg}$  in Mode A.

Moreover, even after the ontologies (and role hierarchies) are disclosed, it is still difficult to infer the database schema because of schema obfuscation. Every request arriving at the Mediator must have been obfuscated by the requestor; and all results schema are in the obfuscated format. Hence the attackers false guess probability will be  $1 - 1/K$ .

**Mode C.** In Mode C, after breaking into organization A and the Mediator, the attacker knows all the metadata about A and his goal is to infer the metadata about organization B. Frequency-based attacks are still effective so that B’s ontology (and role hierarchy) can be inferred with good accuracy. In addition, inferring B’s access control policy is possible because the attacker can know if a request is authorized or denied. However, due to schema obfuscation, it is still difficult for the attacker to infer the syntactic access control policy of B. Similarly, the false guess probability in inferring B’s database schema is still  $1 - 1/K$  (the same as Mode B).

## 6. EXPERIMENTS AND RESULTS

We have implemented a prototype of PACT. We programmed PACT 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 of the experimental testbed are in Table 1. 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.

**Metadata and data sets:** We generated the database schema and data sets for both FBI and CIA databases based an information sharing scenario. In general, each table contains 50 to 10000 records depending on its functionality, and the corresponding access control rules are stored in a separate authorization table.

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 [1]. Each organization has 5 roles. Each role has different privileges to access the tables in database.

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 query:* These queries only involve semantic translations. For example, when an FBI agent poses a remote query (using FBI’s ontology):

```
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';
```

(2) *Query using complex mapping rules:* These queries involve query folding. For example, an FBI agent’s across-

**Table 1: System specifications**

Tier	FBI	Middleware	CIA
CPU (GHz)	P4 2.53	Dual Xeon 1.8	P4 2.4
RAM	768MB	512MB	512MB
Linux Kernel	2.6.9	2.6.9	2.6.8
Database tables	15	-	18
Roles	5	-	5
Ontology (triples)	289	-	325

organization query can be originally written as:

```
select phonenumber from PersonDescription where ID = 101;
```

After semantic translation and expansion by the mediator, the query becomes:

```
select number from PersonProfile, Cell where PersonProfile.ID = Cell.ID and PersonProfile.ID = 101;
```

(3) *Query with data conversion:* These queries contain values or columns that PACT 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 translated query will be:

```
select ID from PersonProfile where height > 179.83;
```

(4) *Query using complex mapping rules and data conversion:* These queries need to be translated using complex mapping rules and data must be converted using 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;
```

## 6.1 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 requested a query until she got the response to the query from PACT. The response time includes both computation and communication costs. encryption and decryption are also required.

To compare the overhead of PACT processing, we have implemented a simple direct query system using the same architecture as PACT where the requestor sends a query to the responder via the mediator but does not perform semantic mediation and access control because the requestor uses the database schema of the responding organization. Access control is done by the responding organization by checking the role's privilege.

**Result data size:** The results in Fig. 2(a) show that, in general, when the result size increases, the response time of PACT increases. In this figure and in the rest of the figures in this paper, we refer to PACT's algorithm as SACE (semantic access control enabling algorithm). The results show that in the absence of encryption, PACT only adds 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. However, despite the overhead of key management and encryption, we show that the performance of PACT was not seriously degraded and we believe that the response times we observed are acceptable given the security benefits.

**The Number of users:** We fix the size of each query's result to about 100KB. That is about 50 to 100 records in our database depending on which tables are queried. Fig. 2(b) shows the average response time when the number of users in the system, we see that in this case, the increases

response time in all systems increases. This result shows that PACT has good scalability in a multi-user environment.

**The size of database and ontology:** The results in both Fig. 2(a) and Fig. 2(b) show that the effect of doubling the database size on the response time is very small and can be neglected. The results show that PACT can easily be extended to large database systems in practical applications.

**Types of queries:** Finally, 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. 2(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 conversion increases with the size of the result data. This explains why the response time for type 3 and type 4 queries are higher than type 1 and type 2.

## 6.2 Component Throughput

In this section, we focus on the performance of the three key components of PACT, 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. We get the throughput measurements by injecting 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  $n/t$ .

**Result data size and number of users:** In Fig. 3(a), the non-encrypted system outperforms the encrypted system. Fig. 3(b) shows the impact of the number of users on the throughput of each component. The throughput of mediators in both systems are very close. Because encryption takes significant time, the mediator becomes the bottleneck in the non-encrypted PACT system. Since PACT is a very flexible system, we can add more mediators to the system to improve the overall performance. Load balancing can be applied when there are multiple mediators are available.

**The number of access control rules:** Fig. 3(c) 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

There is a rich literature on access-control in information interoperation systems. For example, Gong and Qian [12, 11] have discussed the complexity and composability issues in secure interoperation. Ahn and Mohan [5] have im-

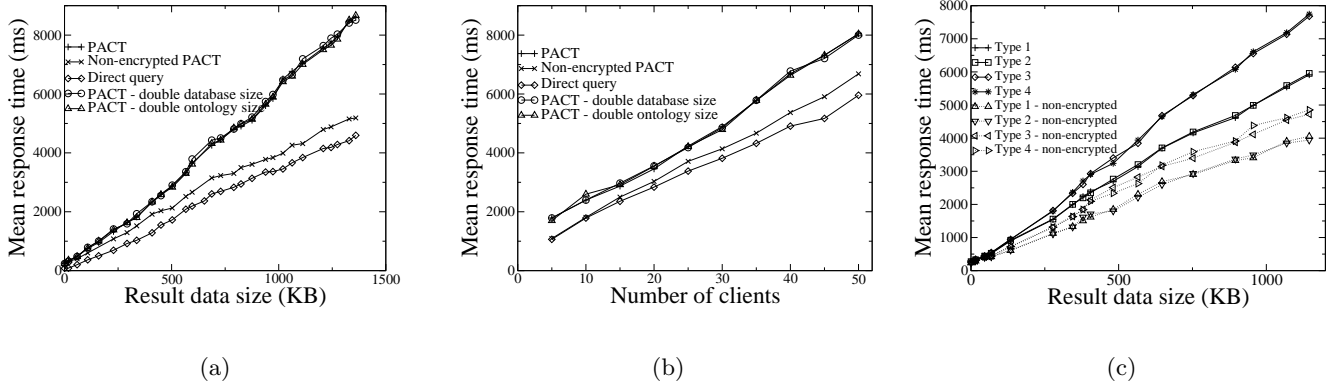


Figure 2: End-to-end response time of PACT, non-encrypted PACT, and the direct query system. (a) The effects of result data size on response time. (b) The effects of number of users on response time at peak traffic. (c) The effects of query types on response time.

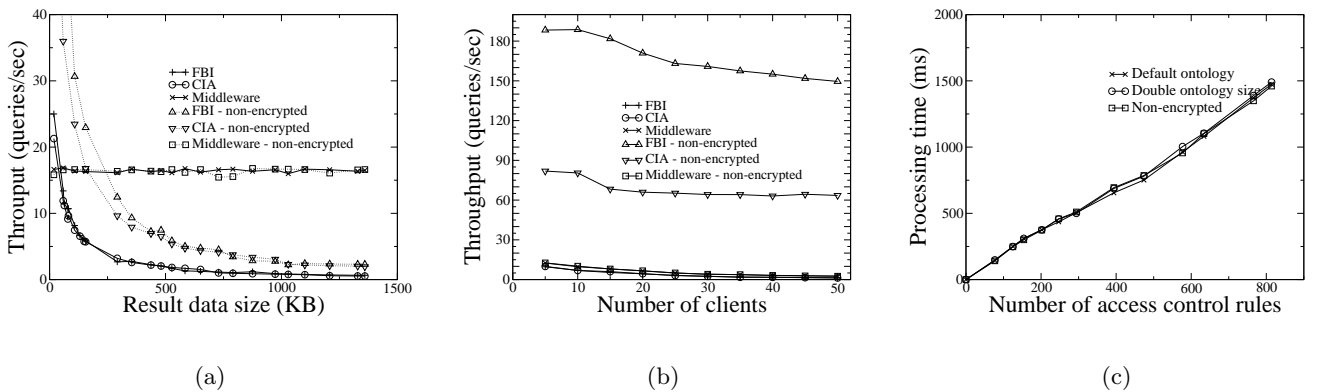


Figure 3: Component throughput of PACT and non-encrypted PACT. (a) The effects of result data size on the throughputs of FBI, CIA and mediator. (b) The effects of number of users on the throughputs of FBI, CIA and mediator at peak traffic. (c) Mediator performance evaluated by access control rules.

plemented 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 [9]. Dawson, Qian, and Samarati, discuss how security can be provided while enabling interoperation of heterogeneous systems [8]. However, their system does not have provisions to preserve the privacy & confidentiality of the metadata of the information sources.

There has been substantial work on querying databases with encrypted data [13]. They tackle data inference attacks while PACT tackles metadata inference attacks. They are complimentary to PACT since we can simply plug in a database with encrypted data in the PACT infrastructure, and enable interoperation in such a setting with simple modifications.

Regarding semantic access control, Qin and Atluri introduced concept-level semantic access control for the semantic web [17]. 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. Qu, et al. [18], have presented an ontology-based rights expression language built on top of OWL. Damiani, et al. [7] have discussed how policy languages can be extended for

the semantic web. Agarwal and Sprick [2, 3], and Yague and Troya [21, 20] have presented frameworks for access control policies for semantic web services.

## 8. CONCLUSION

In this work, we outline PACT, a novel mediation based solution to provide maximum privacy and confidentiality for metadata, queries and data while enabling interoperation among heterogeneous databases. PACT incurs only a minor performance degradation in comparison to existing interoperation systems.

## 9. REFERENCES

- [1] Resource description framework(rdf) model and syntax specification, w3c recommendation <http://www.w3.org/tr/rec-rdf-syntax>. 1999.
- [2] S. Agarwal and B. Sprick. Access control for semantic web services. In *International Conference on Web Services (ICWS '04)*. IEEE Computer Society Press., July 2004.
- [3] S. Agarwal, B. Sprick, and S. Wortmann. Credential based access control for semantic web services. In *2004 AAAI Spring Symposium Series*, March 2004.

- [4] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proc. ACM SIGMOD 2003*, pages 86–97, 2003.
- [5] G-J. Ahn and B. Mohan. Secure sharing role-based delegation. *Journal of Network and Comp. Applications*, 2004.
- [6] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, and L.A. Stein. Owl web ontology language reference. Technical report, W3C.
- [7] E. Damiani, S. De Capitani di Vimercati, C. Fugazza, and P. Samarati. Extending policy languages to the semantic web. In *ICWE*, pages 330–343, 2004.
- [8] S. Dawson, S. Qian, and P. Samarati. Providing security and interoperation of heterogeneous systems. *Distribute Parallel Databases*, 8(1):119–145, January 2000.
- [9] S. De Capitani di Vimercati and P. Samarati. Authorization specification and enforcement in federated database systems. *Journal of Comp. Security*, 5(2):155–188, 1997.
- [10] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, V. Vassalos, J. D. Ullman, and J. Wisdom. The tsimmi approach to mediation: data models and languages. *J. Intelligent Information Systems*, 8(2):117–132, 1997.
- [11] L. Gong and X. Qian. The complexity and composability of secure interoperation. In *IEEE Symp. Security and Privacy*, 1994.
- [12] L. Gong and X. Qian. Computational issues in secure interoperation. *IEEE Trans. Soft. Eng.*, 22(1):43–52, 1996.
- [13] H. Hacigumus, B. R. Iyer, C. Li, and S. Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *ACM SIGMOD Conference*, pages 216–227, 2002.
- [14] M. Kantarcioglu and C. Clifton. Privacy preserving data mining of association rules on horizontally partitioned data. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1026–1037, 2004.
- [15] Peng Liu, Prasenjit Mitra, and Chi-Chun Pan. Privacy-preserving semantic access control across heterogeneous information sources. available at <http://ist.psu.edu/s2/paper/sace.pdf>. Technical report, Pennsylvania State University, Nov. 2004.
- [16] P. Mitra, P. Liu, and C-C. Pan. Privacy-preserving ontology matching. In *AAAI Workshop on Context and Ontologies*, July 2005.
- [17] L. Qin and V. Atluri. Concept-level access control for the semantic web. In *Workshop on XML Security, held in conjunction with the 10th ACM Conf. on CCS*, Oct. 2003.
- [18] Y. Qu, X. Zhang, and H. Li. An ontology-based rights expression language. In *13th Int. World Wide Web Conf. on Alternate track papers & posters Poster, (WWW, Alt. 04)*, pages 324–325. ACM Press, 2004.
- [19] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4), 2001.
- [20] M. Yague, A. Mana, Lopez J., and J. M. Troya. Applying the semantic web layers to access control. In *Web Semantic Workshop, DEXA 2003 Conference*, Sept. 2003.
- [21] M. Yague and J.M. Troya. 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, W3C, pages 483–494, December 2002.
- [22] A. C. Yao. How to generate and exchange secrets. In *Proc. 24th Annual Symposium on Foundations of Computer Science*, Oct. 1986.

## APPENDIX

### A. FBI AND CIA ONTOLOGIES

FBI-Ontology:

```
Class: employee
Type: Table
Prop: name
Prop: salary
Prop: address
Prop: VehID
```

```
Class: LuxuryCar
Type: Table
Prop: Mk
Prop: VehID
```

CIA-Ontology:

```
Class: manager
Type: Table
subClass: executiveManager
subClass: firstLevelManager
Prop: firstName
Prop: lastName
Prop: comp
Prop: VehID
```

```
Class: staff
Type: Table
Prop: firstName
Prop: lastName
Prop: pay
Prop: VehID
```

```
Class: car
Type: Table
Prop: Make
Prop: VehID
Prop: PriceID
```

```
Class: prices
Prop: ID
Prop: Price
```

Metadata:

comp and pay are in Dollars

car.PriceID is the foreign key to the

primary key Prices.ID

(Equivalent CIA-ontology.Staff CIA-DB.adminpersonnel)