Secure and Privacy-preserving Execution Model for Data Services

Mahmoud Barhamgi[1], Djamal Benslimane[1], Said Oulmakhzoune[2], Nora Cuppens-Boulahia[2], Frederic Cuppens[2], Michael Mrissa[1], and Hajer Taktak[1]

[1] LIRIS Laboratory, Claude Bernard Lyon1 University
69622 Villeurbanne, France
{fisrtname.lastname}@liris.cnrs.fr
[2] IT/Telecom-Bretagne, 2 Rue de la Chataigneraie, 35576 Cesson Sevigne - France
{said.oulmakhzoune,nora.cuppens,frederic.cuppens}@telecom-bretagne.eu

Abstract. Data services have almost become a standard way for data publishing and sharing on top of the Web. In this paper, we present a secure and privacy-preserving execution model for data services. Our model controls the information returned during service execution based on the identity of the data consumer and the purpose of the invocation. We implemented and evaluated the proposed model in the healthcare application domain. The obtained results are promising.

Keywords: Data Services, Privacy Preservation, RDF Views

1 Introduction

Recently, Web services have started to be a popular medium for data publishing and sharing on the Web. Modern enterprises are moving towards service-oriented architectures for data sharing on the Web by developing Web service frontends on top of their databases, thereby providing a well-documented, interoperable method for interacting with their data [9, 8, 13, 19]. We refer to this class of services as *data services* in the rest of the paper. Data services are software components that encapsulate a wide range of data-centric operations over "business objects" in underlying data sources. They abstract data consumers from the details of where data pieces are located and how they should be accessed and allow data providers to restrain the way their business objects are manipulated and enforce their own business rules and logic. The growing importance of data services in the movement toward a hosted-services world is evidenced by the number of contexts within which they have been utilized in recent years: data publishing [9, 14], data exchange and integration [11], service-oriented architectures (SOA) [8], data as a service (DaaS) [19], and recently, cloud computing [5].

Most of the time data services are used to access privacy-sensitive information. For example, in the healthcare domain, data services are widely used to access and manipulate the electronic healthcare records [11]. Given the sensitive nature of the accessed information and the social and legal implications for its

disclosure [1], security and privacy are considered among the key challenging issues that still impede the widespread adoption of data services [4].

A considerable body of recent research works have been devoted to security and privacy in the area of Web services [10, 12, 20, 18]. Their focus was on providing mechanisms for ensuring that services act only on the authorized requests and for ensuring SOAP message confidentiality and integrity. However, this is not sufficient as control over who can invoke which service is just one aspect of the security and the privacy problem for data services. A fine-grained control over the information disclosed by data service calls is required, where the same service call, depending on the call issuer and the purpose of the invocation, can return more or less information to the caller. Portions of the information returned by a data service call can be encrypted, substituted, or altogether removed from the call's results. We explain the privacy and the security challenges for data services based on a concrete example.

1.1 Motivating Scenario

Let us consider a healthcare scenario in which a nurse *Alice* needs to consult the personal information (e.g., name, date of birth, etc.) of patients admitted into her healthcare organization NetCare for some medical purposes (e.g., to ensure that patients receive the right medical dosages corresponding to their ages, etc.). The NetCare organization involves many specialized departments (cardiology, nephrology, etc.) and laboratories, and follows a data service based approach [9, 11, 13] to overcome the heterogeneity of its data sources at their various locations; i.e. data services are created on top of heterogeneous data sources to mask their heterogeneity for data consumers. We assume that Alice works in the cardiology department, and that she issued the following query: "*Q: return the names and dates of birth DoB for all patients*". We also assume that she has the following service at her disposal: S_1($center, ?name, ?dob), where input parameters are preceded by "$" and output parameters by "?".

Obviously, the query Q can be resolved by simply invoking S_1 with the value $center = NetCare$. However, executing the service S_1 involves handling security and privacy concerns that could be associated with the service's accessed data. For example, nurses may be only allowed to access the information of patients from their own departments (e.g., nurses working at the cardiology department are not allowed to access the information about patients at the nephrology department); physicians may be only allowed to access the information of their own patients, etc. These are security concerns that are typically defined in security policies. Furthermore, the patients should also be allowed to control who can access their data, for what purposes and under what conditions. For example, two patients *Bob* and *Sue* whose data are accessed by S_1 may have different preferences regarding the disclosure of their ages to a nurse for medical treatment purposes. These are privacy concerns that relate to individuals and their requirements about their data. They are typically defined in privacy policies.

1.2 Challenges

Based on our scenario, we identify the following two challenges which are addressed in the paper. The first challenge is how to enable the service providers (e.g., NetCare) to handle the cited security and privacy constraints. A common approach in the database field to handle such constraints is to push them to the underlying DBMS by rewriting the query to include these constraints [16]. However, this may not be applicable to data services as the same service may access a multitude of heterogeneous data sources that may not necessarily have a DBMS (e.g., XML files, flat files, silos of legacy applications, external Web services, etc.). An alternative approach is to enforce privacy and security policies at the application level [7], by modifying, in our case, the source code of data services. However, this also may not always be applicable nor advisable as most of current data service creation platforms (e.g., *AquaLogic* [8]) provide data services as *black boxes* that cannot be modified; i.e., their internal data integration procedures and logics are not accessible. Even if the code was modifiable, this solution often leads to privacy leaks [16], as the dropped programming code may contain flaws; i.e., its correctness is hard to be proven (especially for complex queries), compared to declarative rewritten queries in the first approach. The second challenge is how to specify and model the security and privacy concerns associated with data services. There is a need for a model that provides explicit description of these concerns to ensure the correct execution of services (e.g., to make sure that services are executed by entitled bodies, etc.) and the proper usage of their returned data (e.g., if data were modified for some privacy concerns, the type of applied modifications needs to be declared so that users can interpret data correctly).

1.3 Contributions

In this paper, we propose a secure, privacy-preserving execution model for data services allowing service providers to enforce their privacy and security policies without changing the implementation of their data services (i.e., data services are considered as black boxes). Our model is inspired by the database approach to enforce privacy and security policies. It relies on a *declarative* modeling of data services using *RDF Views*. When a data service is invoked, our model modifies the RDF view of the corresponding service to take into account pertaining security and privacy constraints. Our model uses the mature query rewriting techniques to rewrite the modified view in terms of calls to data services (including the initial one). Services are then executed, and the constraints are enforced on the returned results. Our contributions are summarized as follows:

- We propose a semantic modeling for data services, privacy and security policies. The modeling is based on RDF views and domain ontologies.
- We propose a secure and privacy-preserving execution model for data services. Our model exploits the mature works in the areas of query rewriting

and modification, and defines new filtering semantics to protect the service's accessed data.
- We integrated our model in the architecture of the widely used Web services container AXIS 2.0, and carried out a thorough experimental evaluation.

The rest of the paper is organized as follows. In Section 2, we present our secure and privacy-preserving execution model for data services. We present also our modeling to data services, security and privacy policies. We evaluate our model in Section 3, survey related works in Section 4 and then conclude the paper in Section 5.

2 A Secure and Privacy-Preserving Execution Model for Data Services

In this section, we describe the proposed model for data service execution.

2.1 Model Overview

Our model is inspired by the database approach to "*declaratively*" handle the security and privacy concerns. Specifically, our model relies on modeling data services as *RDF Parameterized Views* over domain ontologies to explicitly define their semantics. An RDF view captures the semantics of the service's inputs and outputs (and their inter-relationships) using *concepts* and *relations* whose semantics are formally defined in domain ontologies. Views can be integrated into the service description files (e.g., WSDL) as annotations. Our model, as Figure 1 shows, enforces the privacy and the security constraints associated with data services "*declaratively*" as follows. Upon the reception of a service invocation request for a given service (e.g., S_i), it extracts the RDF view of the corresponding service from the service description file and the contextual information (e.g., the *recipient* of requested data, the *purpose*, the time and location, etc.) from the invocation request. Then, the RDF view is rewritten to include the security and privacy constraints that pertain to the data items referred in the view. These constraints are defined in the security and privacy policies and have the form of SPARQL expressions (which simplifies their inclusion in the RDF view). The generated extended view may include now additional data items necessary for the evaluation of the constraints (e.g., the consent of patients, the departments of nurses, etc.) that are not covered by the initial service. Therefore, the extended view is rewritten in terms of calls to (i) the initial service S_i and (ii) the services covering the newly added data items. Finally, the obtained composition (i.e., the rewriting) is executed, and the constraints are evaluated and enforced on the obtained results. The obtained results now respect the different security and privacy concerns, and can be returned safely to the service consumer. We explain and illustrate these steps in details in subsequent sections.

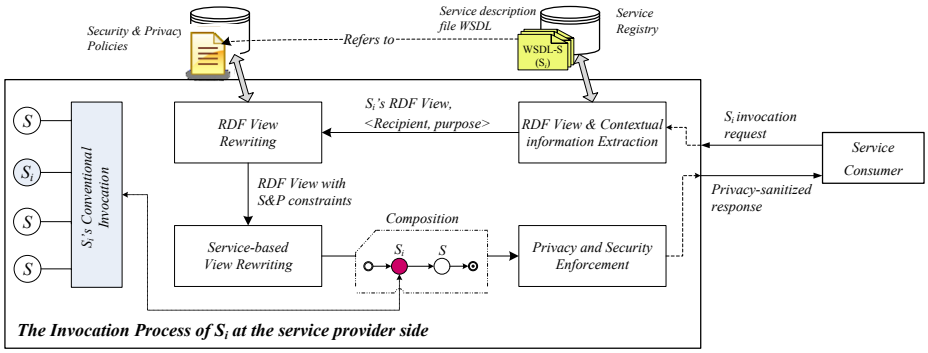

Fig. 1. Overview of the Privacy and Security aware Execution Model

2.2 Semantic models for data services and policies

Semantic model for data services: The semantics of data services should be explicitly defined to allow service consumers to correctly interpret and use the services' returned data. In this work, we model data services as *RDF Parameterized Views* ($RPVs$) over domain ontologies Ω. RPVs use *concepts* and *relations* from Ω to capture the semantic relationships between input and output sets of a data service.

Formally, a data service S_i is described over a domain ontology Ω as a predicate: $S_i(\$\overline{X_i}, ?\overline{Y_i}) :- < RPV_i(\overline{X_i}, \overline{Y_i}, \overline{Z_i}), C_i >$, where:

- $\overline{X_i}$ and $\overline{Y_i}$ are the sets of input and output variables of S_i, respectively. Input and output variables are also called as *distinguished variables*. They are prefixed with the symbols "$"and "?" respectively.
- $RPV_i(\overline{X_i}, \overline{Y_i}, \overline{Z_i})$ represents the semantic relationship between input and output variables. $\overline{Z_i}$ is the set of existential variables relating $\overline{X_i}$ and $\overline{Y_i}$. $RPV_i(\overline{X_i}, \overline{Y_i}, \overline{Z_i})$ has the form of RDF triples where each triple is of the form (subject.property.object).
- C_i is a set of data value constraints expressed over the $\overline{X_i}, \overline{Y_i}$ or $\overline{Z_i}$ variables.

Figure 2 (Parts *a* and *b*) shows respectively the RDF view of S_1 and its graphical representation. The blue ovals (e.g., *Patient*, *Center*) are ontological concepts (ontological concepts and relations are prefixed by the ontology namespace "o:").

RDF views have the advantage of making the *implicit* assumptions made about the service's provided data *explicit*. These assumptions may be disclosed *implicitly* to service consumers. For example, the names and DoBs returned by S_1 are for patients "*who have diabetes*"; i.e., the service consumer will know -implicitly- in addition to the received names that these patients have diabetes. Modeling and explicitly describing this *implicit* knowledge is the first step to handle this unwanted implicit information disclosure. Note that RDF views can be integrated to the service description files as annotations (e.g., using the WSDL-S

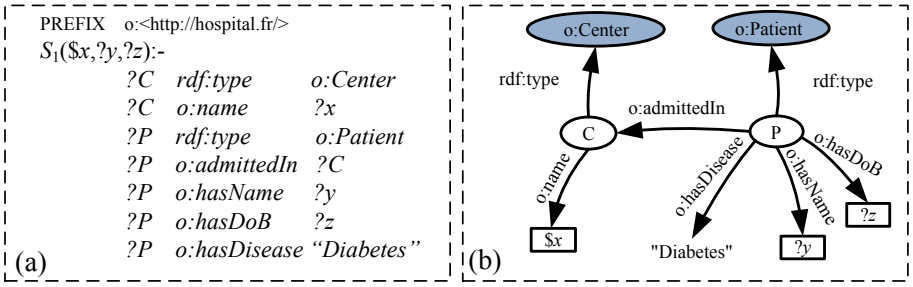

Fig. 2. Part-A: the RDF View of S_1; Part-B: its graphical representation

approach (www.w3.org/Submission/WSDL-S/).

Security and privacy policies: In this work, we suppose the accessed data are modeled using domain ontologies. We express therefore the security and privacy policies over these ontologies. We adopt the OrBAC [3] and its extension PrivOrBAC [6] to express the security and the privacy policies respectively. In the Organization-Based Access Control model (OrBAC), privileges are expressed in terms of *permissions*. A permission is a predicate *Permission(org, r, a, d, c)*; it is read as follows: the organization *org* grants permission to the role *r* to realize the activity *a* on the data item *d* and in the context *c*. In this work, *org* and *r* refer to ontological concepts in the ontology, *d* refers to the properties defined in the ontology. The context *c* is used to specify fine-grained access control constraints (e.g., constraints on the working hours, emergency, etc.).

The security rules corresponding to the motivating example, i.e. *nurses may be only allowed to access the information of patients admitted in the same department*, can be expressed in the OrBAC model as follows:

```
SecRule-1= Permission(NetCare, Nurse, Read, o:hasName, SameDepartment)
SecRule-2= Permission(NetCare, Nurse, Read, o:hasDoB, SameDepartment)
SecRule-3= Permission(NetCare, Nurse, Read, o:hasDisease, SameDepartment),
```

where the "*SameDepartment*" context is defined against domain ontologies as a SPARQL expression. It can be expressed in the Datalog notation as follows ("o:" denotes the ontology's namespace):

```
SameDepartment:- o:Patient(P), o:hasName(P,name), o:treatedIn(P,D),
                 o:Department(D), o:employedIn(recipient,D),
                 o:composedOf(NetCare,D)
```

The PrivOrBAC model [6] extends the OrBAC model with the privacy requirements specified by much of current privacy legislations [1, 2]. These requirements are *consent*, *data accuracy*, *provisional obligations* and *purposes*. *Consent* is the user agreement for accessing and/or processing his/her data. It is

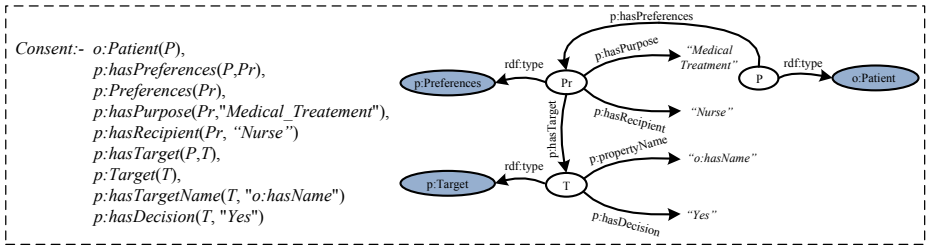

Fig. 3. The SPARQL and the graphical representations of the patient's consent

required before delivering personal data to third parties. *Accuracy* is the level of anonymity and/or level of accuracy of disclosed data. *Provisional obligations* refer to the actions to be taken by the requestors after the access. *Purpose* is the goal that motivates the access request. The proposed model in this paper considers only the *consent* and the *purpose* requirements. Expressions in PrivOrBAC have the form *Permission(org, r, p, a, d, c)*, where p denotes the *purpose*, the context c is used to represent the *consent*; org, r, a and d have the same semantics as above. As the consents of data owners can be regarded as any other data items in underlying data sources, we model them in the underlying ontology and include them in the *context* part of the PrivOrBAC's permissions.

The privacy rules of our example are as follows:

```
PrivRule-1= Permission(NetCare,Nurse,Medical_Treatment,Read,o:hasName,Consent),
PrivRule-2= Permission(NetCare,Nurse,Medical_Treatment,Read,o:hasDoB,Consent),
PrivRule-3= Permission(NetCare,Nurse,Medical_Treatment,Read,o:hasDisease,Consent)
```

where the "*Consent*" context is defined against domain ontologies. Figure 3 shows the *Consent* expressed as a SPARQL expression as well as its graphical representation (we factored out the concepts and properties that are needed to model the consent in a specialized ontology denoted by the prefix "p:").

2.3 RDF views rewriting to integrate security and privacy constraints

In this step, the proposed model extends the RDF view of the queried service with the applicable security and privacy rules (from the policies) as follows.

Our model extracts the RDF view of the invoked service from the service description file, and consults the associated security and privacy policies to determine the applicable rules for the given couple of (recipient, purpose). With respect to security policies, our model applies the access rules associated with each of the data items declared in the view to remove unauthorized data items. In some cases, the access to a given data item is granted only under certain conditions. For example, the security rules in our example restrict the access to the patient's personal information to the nurses working in the department where

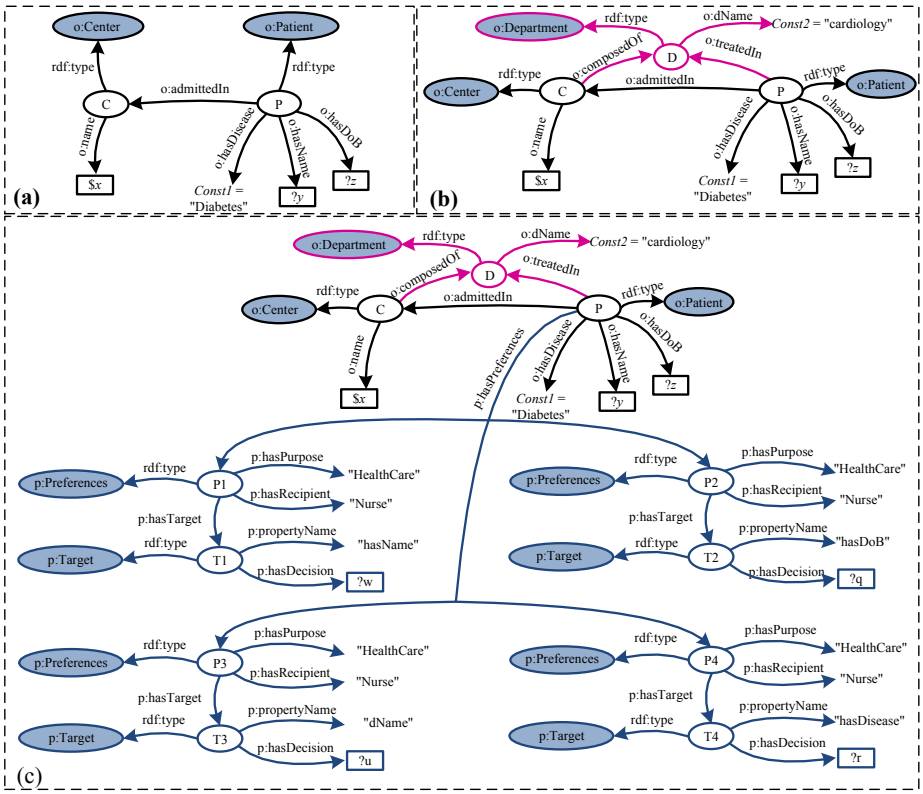

Fig. 4. (a) The original view of S_1; (b) The extended view after applying the security policy; (c) The extended view after applying the privacy policy

the patients are treated. These conditions (which have concretely the form of SPARQL expressions) are accommodated in the RDF view. The parts (a) and (b) of Figure 4 shows respectively the initial and the extended view; the added RDF triples are marked in red. Similarly, our algorithm rewrites the extended view to integrate the privacy rules. Returning to our example, the condition related to the patient's consent are added to the view. Figure 4 (Part-c) shows the extended view, where the added RDF triples are marked in blue.

2.4 Rewriting the extended view in terms of data services

The extended RDF view $v_{extended}$ may include additional data items (denoted by $\Delta v = v_{extended} - v_{original}$) required to enforce security and privacy constraints. These data items may not be necessary covered by the initial service. In our example (Figure 4, Part-c), Δv includes the RDF triples ensuring that the patients and the nurse have the same departments, and the RDF triples querying the patient's consent relative to the disclosure of his personal and medical data.

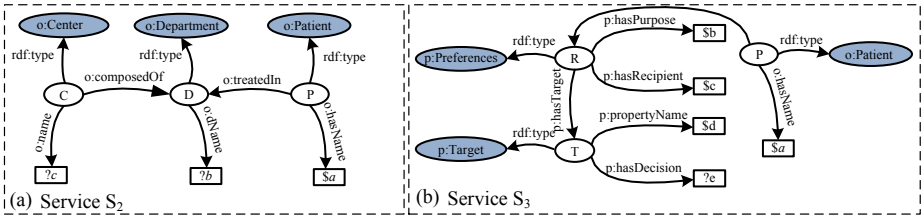

Fig. 5. A graphical representation of the services S_2 and S_3

In this step, we find the data services covering Δv to prepare for the enforcement of privacy and security conditions (in a later step), and rewrites $v_{extended}$ in terms of these services along with the initial service. In this work, we assume the data items necessary for the evaluation of the security and privacy constraints (e.g., consent, time, location, etc.) are also provided as data services.

Table 1. The sample services along with the covered parts of the extended view V'

Service	**Partial Containment Mapping**	**Covered nodes & object properties**
$S_1(\$x, ?y, ?z)$	$V'.P \rightarrow S_1.P$, $V'.D \rightarrow S_1.D$, $V'_C \rightarrow S_1.C$ $x \rightarrow x$, $y \rightarrow y$, $z \rightarrow z$, $const1 \rightarrow const1$	P(y,z,const1), admittedIn(P,C), C(x)
$S_2(\$y, ?x, ?b)$	$V'.P \rightarrow S_2.P$, $V'.D \rightarrow S_2.D$, $V'_C \rightarrow S_2.C$ $x \rightarrow c$, $y \rightarrow a$, $const2 \rightarrow b$	composedOf(C,D),D(const2) treatedIn(P,D), C(x), P(y)
$S_3(\$y, \$b, \$c, \$d, ?w)$	$V'.P \rightarrow S_3.Pa$, $V'.P_1 \rightarrow S_3.P$, $V'.T_1 \rightarrow S_3.T$, $y \rightarrow a$, $b \rightarrow$ "$HealthCare$", $c \rightarrow$ "$Nurse$", $d \rightarrow$ "$hasName$", $w \rightarrow e$	$P(y), hasPreferences(P, P_1)$ P_1("HealthCare","Nurse"), hasTarget(P_1, T_1), T_1("hasName", w)
$S_3(\$y, \$b, \$c, \$d, ?q)$	$V'.P \rightarrow S_3.Pa$, $V'.P_2 \rightarrow S_3.P$, $V'.T_2 \rightarrow S_3.T$, $y \rightarrow a$, $b \rightarrow$ "$HealthCare$", $c \rightarrow$ "$Nurse$", $d \rightarrow$ "$hasDoB$", $q \rightarrow e$	$P(y), hasPreferences(P, P_2)$ P_2("HealthCare","Nurse"), hasTarget(P_2, T_2), T_2("hasDoB", q)
$S_3(\$y, \$b, \$c, \$d, ?u)$	$V'.P \rightarrow S_3.Pa$, $V'.P_1 \rightarrow S_3.P$, $V'.T_3 \rightarrow S_3.T$, $y \rightarrow a$, $b \rightarrow$ "$HealthCare$", $c \rightarrow$ "$Nurse$", $d \rightarrow$ "$dName$", $u \rightarrow e$	$P(y), hasPreferences(P, P_3)$ P_3("HealthCare","Nurse") hasTarget(P_3, T_1), T_3("dName", u)
$S_3(\$y, \$b, \$c, \$d, ?r)$	$V'.P \rightarrow S_3.Pa$, $V'.P_1 \rightarrow S_3.P$, $V'.T_4 \rightarrow S_3.T$, $y \rightarrow a$, $b \rightarrow$ "$HealthCare$", $c \rightarrow$ "$Nurse$", $d \rightarrow$ "$hasDisease$", $r \rightarrow e$	$P(y), hasPreferences(P, P_4)$ P_4("HealthCare","Nurse"), hasTarget(P_4, T_4), T_4("hasDisease", r)

Our rewriting algorithm that implements this step has two phases:
Phase 1: Finding the relevant services: *In this phase, the algorithm compares $v_{extended}$ to the RDF views of available services and determines the parts of $v_{extended}$ that are covered by these views.* We illustrate this phase based on our example. We assume the existence of a service S_2 returning the centers and the departments where a given patient is treated, and a service S_3 returning the privacy preference of a given patient regarding the disclosure of a given property (e.g., name, DoB, etc.) relative to a given couple of recipient and purpose. The RDF views of these services are shown in Figure 5. Table 1 shows our sample services and the parts they cover of $v_{extended}$. The service

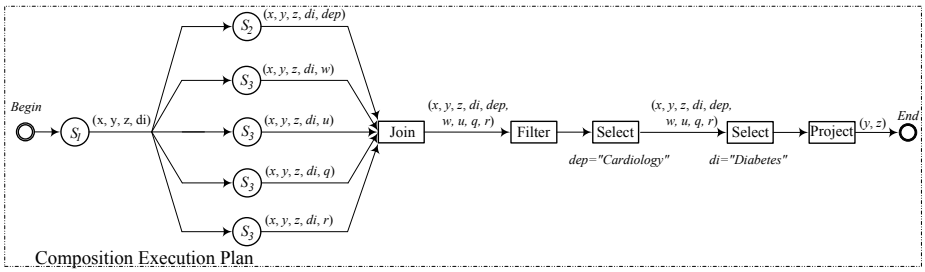

Fig. 6. The Obtained Composition

S_2 covers the properties $composedOf(C, P)$ and $treatedIn(P, D)$ and the node $D(const2 =$ "$cardiology$"$)$, and covers from the nodes P and C the functional properties (i.e., identifiers properties) $hasName$ and $dName$ that could be used to make the connection with the other parts of $v_{extended}$ that are not covered by S_2. The service S_3 covers the identical sub graphs involving a node of a $Preferences$ type (e.g., P_1, P_2, P_3, P_4), a node of $Target$ type (e.g., T_1, T_2, T_3, T_4) and the object properties $hasPreferences$ and $hasTarget$, hence its insertions in the third, fourth, fifth and sixth rows of the Table 1.

Phase 2: Combining the relevant services: *In the second phase, the algorithm combines the different lines from the generated table (in the first phase) to cover $v_{extended}$ entirely.* In our example we need to combine all of Table-1's lines to cover $v_{extended}$. $v_{extended}$ is written in the Datalog notation as follows

```
V_extended($x,?y,?z,?w,?q,?u,?r):-  S1($x,?y,?z)$ ∧ const1="Diabetes"
                                    ∧ S2($y,?x,const2)∧ const2="cardiology"
                                    ∧ S3($y,"HealthCare","Nurse","hasName",?w)
                                    ∧ S3($y,"HealthCare","Nurse","hasDoB",?q)
                                    ∧ S3($y,"HealthCare","Nurse","dName",?u)
                                    ∧ S3($y,"HealthCare","Nurse","hasDisease",?r)
```

2.5 Enforcing security and privacy constraints

The services selected in the previous step are orchestrated into a composition plan to be executed. The composition plan defines the execution order of services and includes filters to enforce privacy and security conditions. Figure 6 shows the execution plan of the running example. The service S_1 is first invoked with the name of the healthcare center ($x=$ "NetCARE"); the patient names obtained (denoted by the variable y) are then used to invoke the service S_3 which returns the patients' preferences relative to the disclosure of their properties (name, DoB, department, and disease). In parallel, the service S_2 is invoked to retrieve the departments where the patients are treated. The results of these services are then joined. Figure 7 gives the outputs of the join operator.

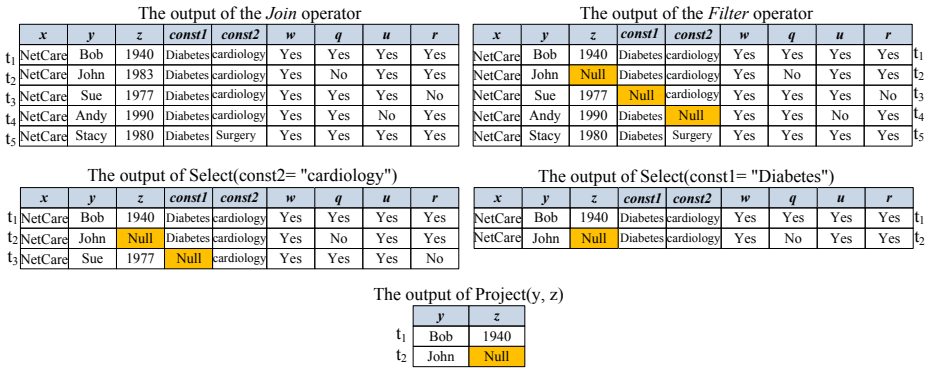

Fig. 7. The intermediate and final results

After the join operation has been realized, the obtained results are processed by a privacy filter that uses the values of the properties that were added to the initial view to evaluate the privacy constraints of the different properties that are subject to privacy constraints in the view. Null values will be returned for properties whose privacy constraints evaluate to False.

Privacy filters are added to the outputs of services returning privacy sensitive data. The semantics of a privacy filter is defined as follows:

Definition 1. *Let t (resp., t_p) be a tuple in the output table T (resp., T_p) of a service S returning privacy sensitive data, let $t[i]$ and $t_p[i]$ be the projected datatype properties that are subject to privacy constraints, and let $constraint(t[i])$ be a boolean function that evaluates the privacy constraints associated with $t[i]$. A tuple t_p is inserted in T_p as follows:*

For each tuple $t \in T$
For i = 1 to n / n is the number of columns in T */*
if const(t[i]) = true Then t_p [i] = t[i]
else t_p [i] = null
Discard all tuples that are null in all columns in T_p

Continuing with our running example, the added filter computes the values of y, z, $const1$ (i.e, department) and $const2$ (i.e, disease) as follows:

$y = y$ if $w =$ "Yes", otherwise $y =$ Null
$z = z$ if $q =$ "Yes", otherwise $z =$ Null
$const1 = const1$ if $u =$ "Yes", otherwise $const1 =$ Null
$const2 = const2$ if $r =$ "Yes", otherwise $const2 =$ Null

After applying the privacy filter, the composition execution plan applies the predicates of the extended view (e.g., dep ="cardiology", and di="Diabetes") on the filter's outputs. This operation is required for two reasons: (i) to remove the tuples that the recipient is not allowed to access according to the security policy, and (ii) to remove the tuples that the recipient has access to, but whose disclosure would lead to a privacy breach.

Figure 7 shows the output of the *Select*(*dep*= "cardiology") operator. The tuples t_4 and t_5 have been removed. t_5 has been removed in compliance with the security policy which requires the patient and recipient to be in the same department - the patient *Stacy* is treated in the surgery department, whereas the recipient*Alice* works in the cardiology department). t_4 was removed despite the fact that the patient and the recipient are in the same department. Note that if t_4 were disclosed, then the recipient *Alice* would infer that the patient *Andy* is treated in the cardiology department which violates *Andy*'s privacy preferences.

The *Select*(*di*= "Diabetes") operator removes the tuple t_3 by comparing the value "Null" with the constant "Diabetes". Note that if t_3 was disclosed, then the recipient *Alice* would infer that the patient *Sue* has *Diabetes* which violates *Sue*'s privacy preferences.

3 Implementation and Evaluation

3.1 Implementation

In order to validate and evaluate our proposal, we exploited the extensibility support provided by Axis 2, specifically the ability to deploy user modules, to implement our privacy-preserving service invocation model. As shown in Figure 8, we extended the AXIS 2.0 architecture with a `privacy` module consisting of two handlers: the *Input* and *Output handlers*, detailed as follows.

Input Handler: This handler intercepts incoming SOAP messages and uses the AXIOM API (`http://ws.apache.org/axiom/`) to extract context information and the request contents, which are then stored into an XML file. The context information of the request is extracted from the SOAP header and contains the recipient identity and the purpose of the invocation. The business service in then invoked by our Axis2 engine.

Output Handler: The output handler intercepts the output SOAP response message before it is sent out of the service engine and makes sure that it complies with the applicable privacy and security policies. To do so, the *RDF View Modification component* parses the security and privacy policies associated with the invoked service using the DOM API and extracts the rules that apply to the accessed data items for the recipient and the purpose at hand. It rewrites the RDF view to take into account these extracted rules as explained in the previous sections. Then, the *RDF View Rewriting component* decomposes the obtained extended view into a set of calls to data services that retrieve the different data items requested by the extended view. The obtained composition is then executed. As a final step, the *Result Filtering component* enforces the privacy and the security constraints on the obtained results. The output SOAP message is built and the filtered results are sent to the service consumer.

3.2 Evaluation

To evaluate the efficiency of our model, we applied it to the healthcare domain. In the context of the PAIRSE Project (http://picoforge.int-evry.fr/cgi-bin/twiki/view/Pairse/), we were provided with a set of /411/ medical data

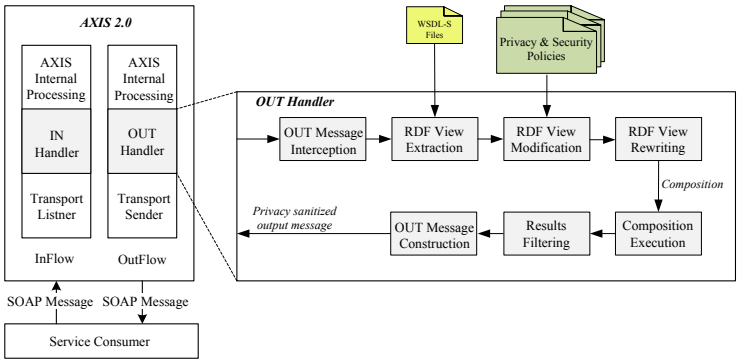

Fig. 8. The extended architecture of AXIS 2.0

services accessing synthetic medical information (e.g., diseases, medical tests, allergies, etc) of more than /30,000/ patients. The access to these medical data was conditioned by a set of /47/ privacy and security rules. For each patient, we have randomly generated data disclosure preferences with regard to /10/ medical actors (e.g., researcher, physician, nurse, etc.) and different purposes (e.g., scientific research). These preferences are stored in an independent database and accessed via /10/ data services, each giving the preferences relative to a particular type of medical data (e.g., ongoing treatments, allergies, etc.). We deployed all of these services on our extended AXIS server running on a machine with 2.2GHz of CPU and 8GB of memory.

We conducted a set of experiments to measure the cost incurred in the enforcement of security and privacy policies during the service invocation. Specifically, we evaluated: (i) the cost c_1 incurred in computing the extended view and writing it in terms of services, and the (ii) the cost c_2 incurred in enforcing the security and privacy constraints on retrieved data (i.e., the cost incurred in the filters). For that purpose, in the first set of experiments we executed the services to return the medical information about one given patient (e.g., the patient Bob). In the second set, we executed the same services to return the medical information for all patients. In the first set of experiments, as the services return the information of one patient only, c_2 can be neglected and remains only c_1. In the second set, c_2 is amplified by the number of processed patients. The executed services in our experiments were selected such that they have different sizes of RDF views (namely, /1/ class-node, /3/ class-nodes, and /5/class-nodes). The invocations were made by the same actor (a researcher) and for the same purpose (medical research). Figure 9 depicts the results obtained for the invocations in Sets 1 and 2. The results for Set 1 show that security and privacy handling adds only a slight increase in the service invocation time. This could be attributed to the following reasons: (i) the time needed to inject the security and privacy constraints in the service's RDF view is almost negligible, (ii) rewriting the $v_{extended}$ in terms of services is not expensive, as most of $v_{extended}$'s graph is

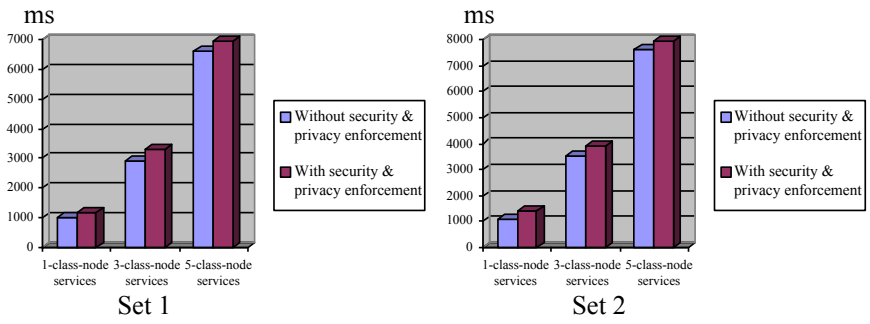

Fig. 9. The experimental results

already covered by $v_{original}$ and the size of (Δv) does not exceed generally 20% of the size of $v_{original}$, and finally (*iii*) there is no network overhead incurred in invoking the additional services as they are already deployed on the server. The results for Set 2 show that c_2 is still relatively low if compared to the time required for executing the services without addressing the security and privacy concerns.

4 Related Work

Most approaches in the area of Web service security have focused on providing mechanisms for ensuring that services act only on authorized requests as well as ensuring the confidentiality and the integrity of exchanged messages [10]. These works range from proposing new public- and private-key encryption mechanisms to protect exchanged SOAP messages [20], to proposing secure communication protocols and architectures [12]. We consider these works as complementary to our proposal as we focus on a different security aspect which is limiting the service's disclosed information based on the identities of services' consumers (i.e., the recipients), their purposes and the data queried at the service endpoint.

Some works have addressed the privacy of service consumers as they may release sensitive information (e.g., credit card numbers, etc.) when they interact with Web services. Hamadi et al. [15] proposed a formal technique to allow Web service providers describe "faithfully" the way they use and store the data collected from service consumers. The description is integrated into Web service standards using an extended state machine model, and could be used in the service selection process. Meziane et al. [18] proposed a system to monitor the compliance of service providers with the privacy agreements that define the consumers' privacy rights. Malik et al. [17] proposed a reputation-based approach to enable interacting Web services to have a better understanding of their privacy practices, to help them preserve users' privacy when interacting with other services. In our work, the focus is on data subjects whose data are accessed by the data services rather than on service consumers. Therefore, our work complements these works by addressing a new dimension of the privacy problem.

In addition, the main aspect that differentiates our contribution from existing work is the transparent integration and management of privacy and security concerns into existing architectures, without modifying either business services, protocols or query languages that give access to the services. Therefore, the integration of our contribution into existing business applications should require minimal changes, leaving the original business infrastructure unchanged.

5 Conclusion and Perspectives

In this paper, we proposed a secure and privacy-preserving execution model for data services. Our model exploits the services' semantics to allow service providers enforce locally their privacy and security policies without changing the implementation of their data services (i.e., data services are considered as black boxes). We integrated our model to the architecture of Axis 2.0 and evaluated its efficiency in the healthcare application domain. The obtained results are promising. As a future work, we plan to address data privacy concerns when composing autonomous data services with conflicting privacy policies.

Acknowledgments: This research work is funded by the French National Research Agency under the grant number ANR-09-SEGI-008.

References

1. Rindfleisch, T.C.: Privacy, Information Technology, and Health Care. Communications of the ACM, 40(8), pp. 92–100, (1997).
2. US Department of Health and Human Services: Standards for privacy of individually identifiable health information; Final rule. `http://www.hhs.gov/ocr/privacy/hipaa/administrative/privacyrule/privrulepd.pdf`, August (2002).
3. Abou El Kalam, A., Benferhat, S., Miege, A., El Baida, R., Cuppens, F., Saurel, C., Balbiani, P., Deswarte, Y., Trouessin, G.: Organization based access control. In: 4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2003), IEEE Computer Society 2003, ISBN 0-7695-1933-4 (2003).
4. Agrawal, D., El Abbadi, A., Wang, S.: Secure and privacy-preserving data services in the cloud: A data centric view. In: PVLDB 2012, 5(12), pp. 2028-2029, (2012).
5. Agrawal, D., El Abbadi, A., Antony, S., Das. S.: Data management challenges in cloud computing infrastructures. In: Databases in Networked Information Systems, 6th International Workshop, DNIS Proceedings 2010, LNCS vol. 5999, pp. 1–10, Springer (2010).
6. Ajam, N., Cuppens-Boulahia, N., Cuppens, F.: Contextual privacy management in extended role based access control model. In: Data Privacy Management and Autonomous Spontaneous Security, 4th International Workshop, DPM 2009 and Second International Workshop, SETOP 2009, LNCS vol. 5939, pp. 21–35, Springer, (2009).
7. Ashley, P., Moore, D.: Enforcing privacy within an enterprise using IBM Tivoli privacy manager for e-business. In IBM Developer Domain, May, (2003).

8. Carey, M. J.: Declarative data services: This is your data on SOA. In: IEEE International Conference on Service-Oriented Computing and Applications, SOCA 2007, California, USA, pp.4, IEEE Computer Society, (2007).
9. Carey, M J., Onose, N., Petropoulos, M.: Data services. Communications of the ACM, 55(6), pp. 86–97 (2012).
10. Damiani, E.: Web service security. In: Encyclopedia of Cryptography and Security (2nd Ed.), pp. 1375–1377, Springer, (2011).
11. Dogac, A.: Interoperability in ehealth systems (tutorial). In: PVLDB, 5(12), pp. 2026–2027, (2012).
12. Durbeck, S., Fritsch, C., Pernul, G., Schillinger, R.: A semantic security architecture for Web services. In: Fifth International Conference on Availability, Reliability and Security (ARES 2010), Poland, pp. 222–227, IEEE Computer Society, (2010).
13. Dustdar, S., Pichler, R., Savenkov, V., Truong, H L.: Quality-aware service-oriented data integration: requirements, state of the art and open challenges. SIGMOD Record, 41(1), pp. 11–19, (2012).
14. Gilpin, M., Yuhanna, N., Smillie, K., Leganza, G., Heffner, R., Hoppermann, J.: Information-as-a-service: What's behind this hot new trend?. Forrester Research, Research Report, March 22, (2007).
15. Hamadi, R., Paik, H., Benatallah, B.: Conceptual modeling of privacy-aware web service protocols. In: 19th International Conference on Advanced Information Systems Engineering (CAiSE 2007), LNCS vol. 4495, pp. 233–248, Springer, (2007).
16. LeFevre, K., Agrawal, R., Ercegovac, V., Ramakrishnan, R., Xu, Y., DeWitt, D.J.: Limiting disclosure in hippocratic databases. In: the Thirtieth International Conference on Very Large Data Bases VLDB 2004, pp. 08–19, (2004).
17. Malik, Z., Bouguettaya, A.: RATEWeb: Reputation assessment for trust establishment among Web services. VLDB Journal, 18(4), pp. 885–911, (2009).
18. Meziane, H., Benbernou, S., Zerdali, A.K., Hacid, M.S., Papazoglou, M.P.: A view-based monitoring for privacy-aware web services. In: the 26th International Conference on Data Engineering (ICDE 2010), pp. 1129–1132, IEEE, (2010).
19. Vu, Q. H., Pham, T. V., Truong, H. L., Dustdar, S., Asal, R.: DEMODS: A description model for data-as-a-service. In: IEEE 26th International Conference on Advanced Information Networking and Applications (AINA 2012), pp. 05–12, IEEE, (2012).
20. Yau, S. S., Yin, Y.: A privacy preserving repository for data integration across data sharing services. IEEE Transactions on Services Computing, 1(3), pp. 130–140 (2008).