Privacy- Intensification in DaaS Composition

K.Cowsikk[1] Mrs. M. Jothi[2] Mr. M. Srinivasan[3]
[1]M. E. (CSE) [2, 3]Assistant Professor
[1, 2] RatnaVel Subramaniam College of Engineering & Technology
[3]Priyadarshini Engineering College, Vaniyambadi

Abstract---Data as a Service (DaaS) builds on service-oriented technologies to enable fast access to data resources on the Web. However, this paradigm raises several new privacy concerns that traditional privacy models do not handle. In addition, DaaS composition may reveal privacy-sensitive information. In this paper, we propose a formal privacy model in order to extend DaaS descriptions with privacy capabilities. The privacy model allows a service to define a privacy policy and a set of privacy requirements. We also propose a privacy-preserving DaaS composition approach allowing verifying the compatibility between privacy requirements and policies in DaaS composition. We propose a negotiation mechanism that makes it possible to dynamically reconcile the privacy capabilities of services when incompatibilities arise in a composition. We validate the applicability of our proposal through a prototype implementation and a set of experiments.

Keywords: Service Composition, DaaS Services, Privacy, Negotiation

I. Introduction

Web services have recently emerged as a popular medium for data publishing and sharing on the Web. Modern enterprises across all spectra are moving towards Service oriented architecture by putting their databases behind Web services, thereby providing a well-documented, and platform independent and interoperable method of interacting with their data. This new type of services is known as DaaS (Data-as-a-Service) Services where services' correspond to calls over the data sources. DaaS sits between services-based applications (i.e. SOA-based business process) and an enterprise's heterogeneous data sources. They shield applications developers from having to directly interact with the various data sources that give access to business objects, thus enabling them to focus on the business logic only. While individual services may provide interesting information/functionality alone, in most cases, users' queries require the combination of several Web services through service composition. In spite of the large body of research devoted to service composition over the last years, service composition remains a challenging task in particular regarding privacy. In a nutshell, privacy is the right of an entity to determine when, how, and to what extent it will release private information. Privacy relates to numerous domains of life and has raised particular concerns in the medical field, where personal data, increasingly being released for research, can be or have been, subject to several abuses, compromising the privacy of individuals.

PROBLEM DEFINITION

Web services have recently emerged as a popular medium for data publishing and sharing on the Web. Modern enterprises across all spectra are moving towards a service-oriented architecture by putting their databases behind Web services, thereby providing a well-documented, and platform independent and interoperable method of interacting with their data. This new type of services is known as DaaS (Data-as-a-Service) services where services correspond to calls over the data sources. DaaS sits between services-based applications (i.e. SOA-based business process) and an enterprise's heterogeneous data sources. They shield applications developers from having to directly interact with the various data sources that give access to business objects, thus enabling them to focus on the business logic only. While individual services may provide interesting information/functionality alone, in most cases, users' queries require the combination of several Web services through service composition. In spite of the large body of research devoted to service composition over the last years), service composition remains a challenging task in particular regarding privacy. In a nutshell, privacy is the right of an entity to determine when, how, and to what extent it will release private information. Privacy relates to numerous domains of life and has raised particular concerns in the medical field, where personal data, increasingly being released for research, can be or have been, subject to several abuses, compromising the privacy of individuals**.**

Two factors exacerbate the problem of privacy in DaaS. First, DaaS services collect and store a large amount of private information about users. Second, DaaS services are able to share this information with other entities. Besides, the emergence of analysis tools makes it easier to analyze and synthesize huge volumes of information, hence increasing the risk of privacy violation. In the following, we use our epidemiological scenario to illustrate the privacy challenges during service composition.

II. Existing Method

The existing work are suffering from two major shortcomings: The first one is the “take-it-or- leave-it” principle, i.e. a service can only accept or refuse the other service's proposal as a whole. The second is the “one-size-fits all” principle: once the service producer has designed its privacy policy, it will be proposed to all interested services no matter what their requirements are. A formal privacy model in order to extend DaaS descriptions with privacy capabilities. The privacy model allows a service to define a *privacy policy* and a set of *privacy requirements*. A privacy-preserving DaaS composition approach allowing to verify the compatibility between privacy requirements and policies in DaaS composition.

Disadvantage

– DaaS collect and store large amount of data about user and share to other entities.

– Increase the risk of privacy violation.

III. PROPOSED METHOD

We extend the previous composition approach to deal with the privacy-preserving issue within composition. We aim at designing techniques for protecting the composition results from privacy attacks before the final result is returned by the mediator. And also we extend the previous composition approach to deal with the privacy-preserving issue within composition. We propose a compatibility matching algorithm to check privacy compatibility between component services within a composition. A matching threshold is set up by services to cater for partial and total privacy compatibility.

Advantage

- The design techniques for protecting the composition results from privacy attacks before the final result is returned by the mediator.
- The previous composition approach to deal with the privacy-preserving issue within composition.

IV. THE PAIRSE PROJECT

The approach presented in this paper is implemented as a part of PAIRSE1 project which deals with the privacy preservation issue in P2P data sharing environments, particularly in epidemiological research where the need of data sharing is apparent for making better a health environment of people. To support the decision process, epidemiological researchers should consider multiple data sources such as the patient data, his social conditions, the geographical factors, etc. The data sources are provided by DaaS services and are organized with peers. DaaS services differ from traditional Web services, in that they are stateless; i.e. they only provide information about the current state of the world but do not change that state. When such a service is executed, it accepts from a user an input data of a specified format ("typed data") and returns back to the user some information as an output. DaaS services are modeled by RDF views. Figure 1 summarizes the architecture of this project. The Multi-Peer Query Processing component is in charge.

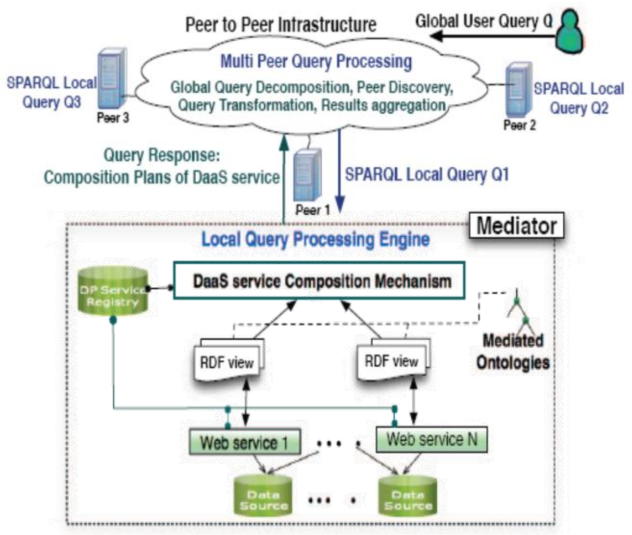
Peer to Peer Infrastructure
Global User Query Q
Multi Peer Query Processing
Global Query Decomposition, Peer Discovery,
Query Transformation, Results aggregation
SPARQL Local Query Q3
Peer 3
SPARQL Local Query Q2
Peer 2
Query Response:
Composition Plans of DaaS service
SPARQL Local Query Q1
Peer 1
Mediator
Local Query Processing Engine
DP Service Registry
DaaS service Composition Mechanism
Mediated Ontologies
RDF view
RDF view
Web service 1
Web service N
Data Source
Data Source

Fig. 1: PAIRSE global architecture

V. PRIVACY MODEL

In this section, we describe our privacy model for DaaS services. Each service S has a *privacy policy* (noted as PPS) specifying the set of privacy practices applicable on any collected data and *privacy requirements* noted as PRS) specifying the set of privacy conditions that a third-party service T must meet to consume S's data.

A. Privacy Level

We define two privacy levels: data and operation. The data level deals with data privacy. Resources refer to input and output parameters of a service (e.g., defined in WSDL). The operation level copes with the privacy about operation's invocation. Information about operation invocation may be perceived as private independently on whether their input/output parameters are confidential or not. For instance, let us consider a scientist that has found an invention about the causes of some infectious diseases, he invokes a service operation to search if such an invention is new before he files for a patent. When conducting the query, the scientist may want to keep the invocation of this operation private, perhaps to avoid part of his idea being stolen by a competing company. We give below the definition of the privacy level.

Privacy level on resource *rs* of S is defined as follows:

(i) *L*="data" if *rs* is an input/output of S operation;
(ii) *L*="operation" if *rs* is information about S's operation.

B. Privacy Rule

The sensitivity of a resource may be defined according to several dimensions called *privacy rules*. We call the set of privacy rules *Rules Set*(*RS*). We define a privacy rule by a *topic*, *domain*, *level* and *scope*. The *topic* gives the privacy facet represented by the rule and may include for instance: the resource recipient, the purpose and the resource retention time. The "purpose" topic states the intent for which a resource collected by a service will be used; the "recipient" topic specifies to whom the collected resource can be revealed. The *level* represents the privacy level on which the rule is applicable. The domain of a rule depends on its level. Indeed, each rule has one single level: "data" or "operation". The *domain* is a finite set that enumerates the possible values that can be taken by resources according to the rule's topic. For instance, a subset of domain for a rule dealing with the right topic is *{*"no-retention", "limited-use"*}*. The *scope* of a rule defines the granularity of the resource that is subject to privacy constraints. Two rules at most are created for each topic: one for data and another for operations.

C. Privacy Policy

A service S will define a *privacy policy*, PPS, that specifies the set of practices applicable to the collected resources. Defining the privacy policy PPS of S is performed in two steps. First, the service S identifies the set (noted *Pp*) of all privacy resources. Second, S specifies assertions for each resource *rs* in *Pp*. Deciding about the content of *Pp* and the rules (from *RS*) to apply to each resource in *Pp* varies from a service to another. PPS specifies the way S treats the collected resources (i.e., received through the mediator). We give below a definition of privacy policy.

- *Algorithm 1: PCM*

Input : PRS=*{*(*Aj*(*Ri*, *rsk*)), *j* _ |PRS|, *i* _ |*RS*|, *k* _ |*Pc*|, *rsk* ∈ *Pc*, *Ri* ∈ *RS}* (assertion of privacy requirements).
Input : PPS'=*{*(*Aj_* (*Ri*, *rs_k*)), *j_* _ |PPS'|, *i* _ |*RS*|, *k* _ |*Pp*|, *rs_k* ∈ *Pp*, *Ri* ∈ *RS}* (assertion of privacy policy).
Output: InC (The set of incompatible assertion couple);

```
Step. 1 :   for each rsk = rs_k do
Step. 2 :   for i = 1, i _ |RS| do
            for j = 1, j _ |PRS|
Step. 3 :   do for j_ = 1, j_ _ |PPS'
Step. 4 :   do
Step. 5 :   if (Aj_ (Ri, rs_k) (Aj(Ri, rsk) then
Step. 6 :   Aj(Ri, rsk) is compatible with Aj_ (Ri, rs_k)
Step. 7 :   else InC ← (Aj(Ri, rsk), Aj_ (Ri, rs_k))
            privacy
```

D. *Privacy-aware Negotiation*

In services composition (cf. Section 2), a mediator selects one service from several candidate services to perform a sub-part of the user query. Several approaches in literature use non-functional (QoS i.e., quality of service) properties to select services, where the web services provide contracts that can guarantee a certain level of QoS. Contract compliance is usually assessed through a reputation mechanism. We use a similar notion to define a non-functional property called *composition reputation* as a criterion to select services during composition.

Composition reputation (or simply, reputation) is defined as the number of times that a service *S* has accepted to adapt its PPS, divided by the number of times *S* received PPS adaptation requests from the mediator. The more *S* is willing to adapt its PPS, the higher is its reputation:

$$\boxed{Reputation(S)= NAdapt(PPS)/ QAdapt(PPS)}$$

Where *NAdapt*(*PPS*) is the number of adoptions made by *S* on PPS and *QAdapt*(*PPS*) is the number adaptation requests received by *S* from the mediator. A service provider should generally be flexible when it specifies its *PP* (to attain better reputation). Moreover, a service may be willing to adapt some of its assertions in a *PP* while maintaining a minimum privacy level. The approach works as follows. If the PR*Sp* and PPS*c* are not compatible in a given *CPl* , the related service consumer *Sc* is informed by *PCM* about the assertions in its PPS*c* that are incompatible. The mediator starts the negotiation process with *Sc* with the objective of achieving adaptation of PPS*c* .

E. *Negotiation Protocol*

We propose a dynamic protocol called *ReP* (Algorithm 2), handled by the negotiator module. This protocol aims at automatically reconciling the mediator's and consumer's negotiation strategies related to consumer assertions in C.

In this regard, the negotiation protocol incorporates two state machine diagrams using the reconciliation algorithm, and finds the first alternative assertion from *STran An* that is compatible with *Au*. The algorithm *ReP* checks if an incentive *Iq*, from *MStat Ri* , is accepted by *STran An* and then checks the compatibility of the related alternative assertion *Aq* (instead of *Au_*) from *STran An* , (where the couple (*Au*, *Au_*)∈InC). Otherwise, if *Aq*, related to the acceptance of *Iq*, is not compatible with *Au*, the algorithm *ReP* will check the next incentive from *MStat R*∗; looks if it is accepted by *STran An* and the previous reasoning is observed.

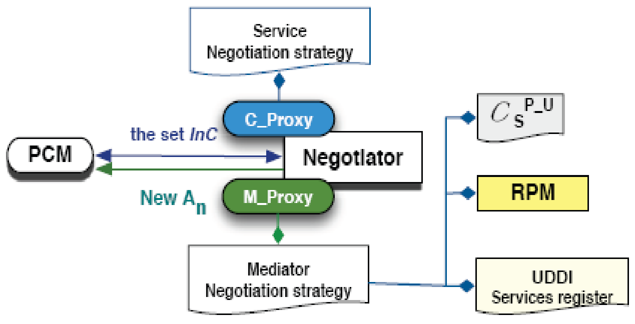

Fig. 2: The Negotiation Process overview

Thus, *ReP* is applied to all assertion couples (related to consumer services) of InC under the condition that there exist negotiation strategies specified for each assertion (of the corresponding privacy policy) of InC. The algorithm *ReP* returns Rec which contains the best alternative assertions that will be compatible.

F. *Prototype Architecture*

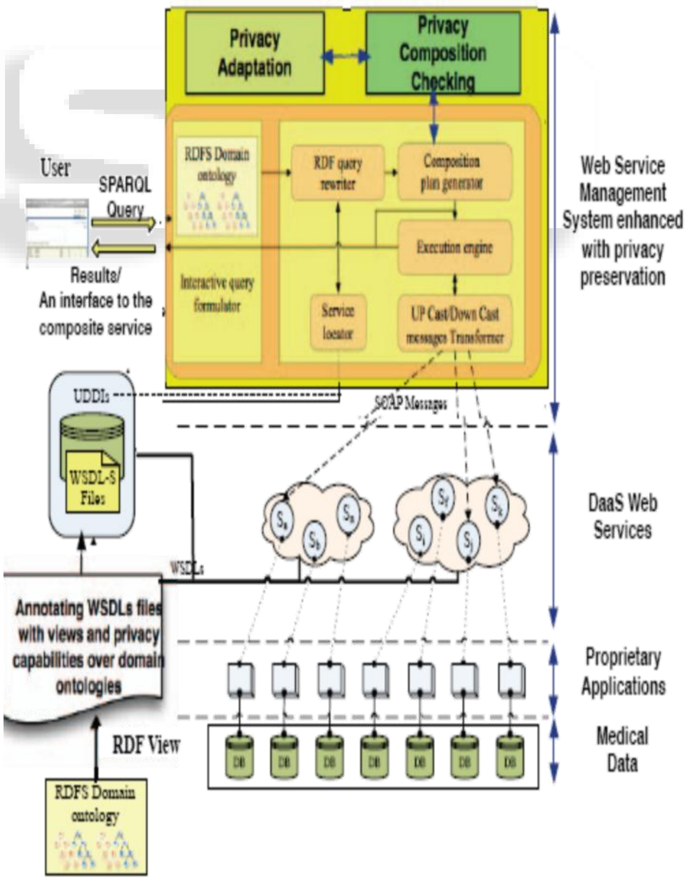
Privacy Adaptation
Privacy Composition Checking
RDFS Domain ontology
RDF query rewriter
Composition plan generator
Execution engine
Interactive query formulator
Service locator
UP Cast/Down Cast messages Transformer
User
SPARQL Query
Results/ An interface to the composite service
Web Service Management System enhanced with privacy preservation
UDDIs
WSDL-S Files
SOAP Messages
DaaS Web Services
WSDLs
Annotating WSDLs files with views and privacy capabilities over domain ontologies
Proprietary Applications
Medical Data
RDF View
RDFS Domain ontology
DB

Fig. 3: Prototype Architecture

Our prototype allows querying and composing DaaS according to the architecture depicted in Figure 3, which is organized into four layers. The first layer contains a set MySQL databases that store medical data. The second layer includes a set of proprietary applications developed in Java;

each application accesses databases from the first layer. These proprietary applications are exported as DaaS services. These services constitute the third layer, and their description files (i.e., WSDLs) are annotated with RDF views and published via registries (we use Openchord DHT to this end). The upper layer includes a Graphical User Interface (GUI) and a *Web Service management system* (WSMS). The GUI component is composed of two basic interfaces: Requester-Interface and Administrator-Interface. Users access the system via Requester-Interface of the GUI to submit queries to the composition system. Administrator accesses the system to develop and manage Web services through the *Privacy Composition Checking* and *Privacy Adaptation* components, which implement our *PCM* algorithm and negotiation process respectively.

VI. Conclusion

Web services are becoming a popular technology to implement Cloud computing to provide services over the Internet. As more and more personal data will be exchanged and processed by Web services, data privacy becomes an issue. Our privacy-enhanced design helps the achievement of minimal disclosure of personal data and tighter control over the access to such data. In any case, privacy policies always reflect the usage of private data as specified or agreed upon by service providers.

VII. Future Enhancement

As future work, we are planning to conduct an extensive evaluation of our Java-based prototype. First, we will evaluate its performance with respect to the number of candidate Web services, the complexity of the privacy policies of the orchestrator and component services, and to the (re)delegation depth. Then, we will conduct a controlled experiment.

References

[1] M. Alrifai, D. Skoutas, and T. Risse. “Selecting skyline services for qos-based web service composition” in 2010
[2] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. “Qos-aware middleware for web services composition” in 2004
[3] S. Ran. “A model for Web services discovery with QoS” in 2003
[4] H. Kargupta, K. Das, and K. Liu. “Multi-party, privacy-preserving distributed data mining using a game theoretic framework” in 2007
[5] Machanavajjhala, J. Gehrke, and M. G otz. Data publishing against realistic adversaries. In 2009.