Semantic-based policy management for cloud computing environments

Hassan Takabi* and James B.D. Joshi

School of Information Sciences,
University of Pittsburgh,
135 N. Bellefield Avenue,
Pittsburgh, PA 15213, USA
Fax: (412)-624-2788
E-mail: hatakabi@sis.pitt.edu
E-mail: jjoshi@sis.pitt.edu
*Corresponding author

Abstract: Cloud computing environments do not allow use of a single access control mechanism, single policy language or single policy management tool for various cloud services. Currently, users must use diverse access control solutions available for each cloud service provider to secure their data. Access control policies may be composed in incompatible ways because of diverse policy languages that are maintained separately at every cloud service provider. Heterogeneity and distribution of these policies pose problems in their administration. The semantic web technologies can provide the solution to interoperability of heterogeneous cloud service providers. In this paper, we introduce a semantic-based policy management framework that is designed to give users a unified control point for managing policies that control access to their data no matter where the data is stored. We present the framework and describe its components. Furthermore, we present a proof of concept implementation and results of performance evaluation.

Keywords: access control; policy management; policy language; semantic web; policy management framework; semantic-based policy management; cloud computing; user centric; heterogeneity; interoperability.

Reference to this paper should be made as follows: Takabi, H. and Joshi, J.B.D. (2012) 'Semantic-based policy management for cloud computing environments', *Int. J. Cloud Computing*, Vol. 1, Nos. 2/3, pp.119–144.

Biographical notes: Hassan Takabi is currently working toward his PhD at the School of Information Sciences, University of Pittsburgh. He is a member of the Laboratory of Education and Research on Security Assured Information Systems (LERSAIS). Before joining the University of Pittsburgh, he was a research scholar in the E-Security Research Centre at the London South Bank University. His research interests include access control models, trust management, privacy and web security, usable privacy and security, security, privacy, and trust issues in cloud computing environments. He is a student member of the IEEE and the ACM.

James B.D. Joshi is Associate Professor and Director of the Laboratory for Education and Research on Security Assured Information Systems (LERSAIS) in the School of Information Sciences at the University of Pittsburgh. He received his MS in Computer Science and PhD in Computer Engineering from

Purdue University in 1998 and 2003, respectively. His research interests include access control, trust management, and secure interoperability. He is a member of the IEEE and the ACM.

1 Introduction

Cloud computing has recently raised significant interest in both academia and industry, but it is still an evolving paradigm. Essentially, it integrates the evolutionary development of many existing computing approaches and technologies such as distributed services, applications, information and infrastructure consisting of pools of computers, networks, information and storage resources (Mell and Grance, 2011; Cloud Security Alliance, 2011). It has shown tremendous potential to enhance collaboration, agility, scale, and availability (Takabi et al., 2010a).

Confusion still exists in IT communities about how a cloud differs from existing models and how these differences affect its adoption (Takabi et al., 2010b). The US National Institute of Standards and Technology (NIST) defines *cloud computing* as follows:

> "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models." (Mell and Grance, 2011)

The five key characteristics of cloud computing include *on-demand self-service*, *ubiquitous network access*, *location-independent resource pooling*, *rapid elasticity* and *measured service* (Mell and Grance, 2011). The three key cloud delivery models are *software as a service* (*SaaS*), *platform as a service* (*PaaS*), and *infrastructure as a service* (*IaaS*) (Takabi et al., 2010b). The cloud deployment models are *public cloud*, *private cloud*, *community cloud*, and *hybrid cloud* composed of multiple clouds (Cloud Security Alliance, 2011).

Cloud computing has become a very attractive paradigm, with the potential to significantly reduce costs through optimisation and increased operating and economic efficiencies (Cloud Security Alliance, 2011; Catteddu and Hogben, 2009). The architectural features of the cloud allow users to achieve better operating costs and be very agile by facilitating fast acquisition of services and infrastructural resources as and when needed. However, these unique features also give rise to various security concerns (Bruening and Treacy, 2009; Takabi et al., 2010b). Several surveys of potential cloud adopters also indicate that security and privacy is the primary concern hindering its adoption (Bruening and Treacy, 2009; Catteddu and Hogben, 2009), yet cloud computing appears to be growing fast because of its potential benefits (Takabi et al., 2010a). Hence, understanding the security and privacy risks in cloud computing and developing effective solutions are critical to its success.

1.1 Policy management in the cloud

In this paper, we focus on access control issues related to the cloud. The cloud computing environments do not allow use of a single authorisation mechanism, single policy language or single management tool for users who use various cloud service providers (CSPs). Each CSP has its own access control mechanism that typically has limited capability to support flexible security requirements of users (Takabi et al., 2010b). This approach is not user/customer centric and hence, can be a significant barrier to its widespread adoption. The specification of access control policies clearly is the responsibility of the organisations/users deploying the cloud service. However, not only do the CSPs dictate how these policies should be specified but each also does it in its own way (Takabi et al., 2010a). An ideal access control scheme must be able to work with all data regardless of where they are stored. Users should be able to manage policies to govern access to their information and resources from a central location.

Currently, users must use diverse access control mechanisms to secure their data stored at different CSPs. Access control policies may be specified in incompatible policy languages and maintained separately at every CSP. When such diverse mechanisms are used, they add considerable overhead, especially since they often lack flexibility with respect to functionality as well as user interfaces (Machulak and Van Moorsel, 2010). This may frustrate users and make them feel that they have no control on where their data ends up and how it is used. The challenge here is to design an integrated access control system that can be used across services from different providers.

Heterogeneity and distribution of access policies pose significant problems in managing them in cloud computing environments (Takabi et al., 2010a). Furthermore, traditional non-semantic-based access control approaches are inadequate for supporting interoperability for cloud computing environments.

Semantic web technologies when used for policy management, can provide runtime extensibility, adaptability and ability to analyse policies at different levels of abstraction. Hence, it can help address the crucial issues of interoperability of heterogenous CSPs (Pérez et al., 2011). The semantic web is an extension of the World Wide Web that allows knowledge, information and data to be shared on the web and reused across applications and enterprises (Takabi et al., 2009).

Web Ontology Language (OWL) is a standard knowledge representation and specification language for the semantic web, and hence it is a promising technology for addressing access control issues in cloud computing environments (Motik et al., 2009). Using OWL for specifying access policies provides some important advantages that are critical in cloud computing environments involving interoperation across multiple cloud providers (Takabi et al., 2009). It also enables involved entities to better understand and share the security policies (Hu et al., 2009).

In order to specify a policy, one should be able to precisely specify classes of subjects, objects, actions, etc. Having a centralised policy management helps the users to use same access policies in multiple CSPs; each provider must understand and enforce the policy rules even though they have their own schemas or data models. There needs to be a platform that provides access control to the resources and services of all the different CSPs. Using OWL gives us a natural and efficient way of specifying these classes and access policies. The second advantage is that OWL is based on description logic (DL) and we can translate access policies expressed in OWL to other policy languages and formalisms (Pérez et al., 2011).

1.2 Our contributions

Our contributions in this paper are as follows:

- we present lessons we have learned from a case study where we developed a unified policy management system for some real world cloud services
- based on the motivating scenarios and the limitations of existing approaches, we propose a semantic-based policy management framework for cloud computing environments
- we present a proof of concept implementation of the proposed framework
- we also present a performance evaluation of the implemented framework.

Our proposed semantic-based policy management framework enables users to control access to their resources that may be scattered across multiple CSPs. It is designed to give cloud customers a unified control point for specifying authorisation policies, no matter where the data is stored and distributed in the cloud. It enables users to specify, edit and manage access policies using a centralised policy manager which uses semantic web technologies for specifying access policies and providing a solution for interoperation of heterogenous cloud computing environments.

The paper is organised as follows: In Section 2, we present motivating scenarios to show heterogeneity of access control mechanisms in the cloud. Section 3 presents lessons we have learned from a case study and based on these lessons, we discuss the limitations of existing approaches. In Section 4, we describe our proposed semantic-based policy management framework and explain its components. Section 5 explains the policy management process, reasoning and conflict analysis. In Section 6, we describe a proof of concept implementation of the proposed framework and present results of performance evaluation. Section 7 discusses related work and Section 8 presents conclusions and future work.

2 Motivating scenarios: heterogeneous access control mechanisms in the cloud

In this section, we present two use case scenarios to motivate the need for the proposed framework of a cloud-based policy management service and then discuss how individual users and organisations that use cloud services can benefit from it.

2.1 Individual user

Alice is a PhD student and uses different applications and services for various purposes. She is working on a project and wants to have access to the project files from anywhere. Sometimes she works at her office using her PC and sometimes she works at home or a coffee shop using her laptop. In order to properly synchronise the project files, she uses *Dropbox*, a file hosting service which uses cloud computing to enable users to store and share files and folders with others across the internet using file synchronisation (http://www.dropbox.com). Since she collaborates with some other researchers on the project, sometimes she needs to share some of the files she stores at *Dropbox* with her colleagues.

Alice also has other documents and spreadsheets that include important content like financial data; she uses *Google Docs*, a service to create and share various types of files online and access them from anywhere (http://docs.google.com). Moreover, she stores some of her older files at *Amazon S3*, an online storage service that provides unlimited storage through a simple web services interface (http://aws.amazon.com/s3). She occasionally shares some of these files with family members. Furthermore, she uses *Mint*, an online personal finance service, to manage her financial and budget planning (http://www.mint.com). Sometimes she may want to share some of these files with a family member or a close friend.

Some of the above mentioned applications are used for convenience and others for sharing and collaboration purposes. With the increasing amount of data that Alice puts online, managing access control for her resources becomes difficult and time consuming. Each of these applications has its own access policy mechanism forcing users to specify separate access control policies for each application. However, with resources scattered across multiple applications, it is difficult to manage access to them. Alice needs to understand the applications' policy mechanisms and specify access policies in their specific policy languages which is a challenging task for her and most users like her. These mechanisms often are either very simple and inflexible or complicated and difficult to use.

Moreover, introducing new access rules or modifying existing ones is problematic due to heterogeneity of these access policy mechanisms. Suppose Alice wants to modify an access rule to a set of resources, or a new colleague is added to her current project and she needs to share some resources with him. In order to do this, she needs to scan all her applications and services to modify access policy rules.

2.2 Small and medium businesses

Suppose an organisation wants to adopt cloud computing services. It uses *Amazon S3* and *FlexiScale*, which are examples of IaaS for storage and maintaining virtual servers respectively. The *Amazon S3* provides "a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web" (http://aws.amazon.com/s3) while the *FlexiScale* provides virtual dedicated servers to its customers (http://www.flexiant.com/products/flexiscale).

As instances of PaaS, *Google App Engine* and *LoadStorm* are used for running web applications and testing their performance respectively. The *Google App Engine* provides web applications on *Google*'s infrastructure. When using *Google App Engine*, there are no servers to maintain; the organisation just uploads its application to serve users (http://code.google.com/appengine) whereas the *LoadStorm* enables web developers to improve the performance of their web applications. Customers can create their own test plans, and generate concurrent users of HTTP traffic in realistic scenarios (http://loadstorm.com).

It also uses *Zoho*, *Zuora*, *Workday*, *Salesforce*, and *DocLanding* that are instances of SaaS for different purposes such as e-mail, billing, content management, human resource management, etc. *Zoho* offers a set of web applications geared towards increasing productivity and offering convenient collaboration (http://www.zoho.com) and *Zuora* offers online recurring billing and payment solutions for SaaS and subscription businesses (http://www.zuora.com). *Workday* provides human resource management, financial management, and payroll, and delivers the solutions on an SaaS

model (http://www.workday.com). *Salesforce* provide customer relationship management (CRM) service to its customers (http://www.salesforce.com). *DocLanding* offers on-demand web-based document management service (http://www.doclanding.com).

From the above two scenarios, we can see some of the key issues: applications and services have their own access policy mechanisms, users need to understand the applications' policy mechanisms, and are forced to specify separate access control policies for each application using different policy languages. A centralised policy management service could help users to better manage security and provide them with a better view on access policies applied to the their resources on different services.

Our proposed cloud-based policy management service not only allows users to centralise management of their access control policies but also enables them to find errors or inconsistencies in the specified access policies very quickly. Using the proposed service, the access control policies can be applied to a distributed set of resources hosted on various CSPs. It gathers information from all of the services users use and provides them with an interface to centrally manage access to their resources regardless of where they are stored at and what CSP they belong to.

3 Lessons learned from case study implementation

In this section, we present lessons we learned from a case study where we developed a unified policy management system for multiple cloud-based services. We analyse the scenarios described previously and discuss our implementation in order to identify limitations of existing access control mechanisms for the cloud and determine requirements that an access control mechanism should have, to be able to address those limitations and be an appropriate candidate for the cloud computing environments.

3.1 Case study implementation

Considering the aforementioned scenarios, we investigated access control mechanisms of some of the existing cloud services with the goal of developing a small scale unified policy management for cloud computing environments to enable users to manage access to their resources in a central location no matter where the resources are stored in the cloud. We analysed more than 20 services looking at various features for each, including the following:

- Authentication mechanism to determine whether it is identity-based, e-mail-based or some other mechanism.
- How users can share resources with other users.
- What privacy/access setting options it provides.
- What policy language and mechanism it uses.
- What APIs it provides.
- Whether it allows users to change privacy settings using an API or in some other ways.
- Whether we can discover users' resources stored in the service.
- Whether it supports XACML (XACML, 2010) or similar technologies.

Based on information gathered from this analysis, we picked the following five service providers for our purpose: *Amazon S3*, *Dropbox*, *LinkedIn*, *Flickr* (http://www.flickr.com), and *Twitter* (http://www.twitter.com/). Here is a summary of how each service deals with policy management.

- In *Amazon*, resources are represented by either bucket or object. An object is any data item stored in the system and a bucket is a top-level container in which objects are stored; for example, files are represented by objects and folders using buckets. In order to manage *S3* accounts, one should use Amazon web services (AWS) login credentials that are stored in an *AWSCredentials* object. *Amazon S3* uses access control lists (ACLs) to control who has access to buckets and objects in *S3*. By default, any bucket or object a user creates belongs to him and is not accessible to anyone else. The user can use JetS3t (a toolkit for Java programmers with an API for interacting with storage services) for access control lists to make buckets or objects publicly accessible, or to allow other *S3* members to access or manage objects.
- In *Dropbox*, one needs an app key, secret, username, and password to get a token for authentication. Function *getAccountInfo* is used to get an authentication token. It uses the key and secret to get the token, and then logs into the account using the username and password. A recursive function *getResource* can be used to get information about all the resources. Users can invite other users to share specific folders. Function *sharefile* is used for posting files to the *Dropbox* server while function *deletefile* is used for removing a file from the server. Function *getfile* is used for copying a file from server to the local machine. So, the sharing process could be done by copying the target file, deleting it from original folder, and then posting it to the public folder or a specific shared folder.
- In *LinkedIn*, we need an app key, secret, token key and token secret for authentication. Function *getProfile* is used to login and get all the information of the full profile from the user's account. Function *permissionGrant* is used to send an invitation to another user with his e-mail address, first name and last name.
- In *Flickr*, an app key and a secret are required to get a token for authentication. Function *showPeople* is used to get the photos of the user. Function *getPermissions* determines whether photos are public, shared or private. Function *publicPhoto* can be used for changing the ACL to public while function *privatePhoto* can be used to change the ACL to private.
- *Twitter* offers two discrete REST APIs (Fielding and Taylor, 2002). The REST API methods allow developers to access core Twitter data. This includes update timelines, status data, and user information. The Search API methods give developers methods to interact with the *Twitter* Search function. In order to use the *Twitter* API, one first has to register a client application that will be provisioned a consumer key and a secret. This key and secret scheme is similar to the public and private keys used in protocols such as SSH (SSH, 2006). Authentication can be achieved using OAuth (2010) and the consumer key and secret will be used, in conjunction with an OAuth library to sign every request you make to the API. The APIs allows developers to work with followers, friends, tweets, etc.

Figure 1 The screen shots of various cloud services, (a) Amazon (b) Dropbox (c) Flickr (see online version for colours)

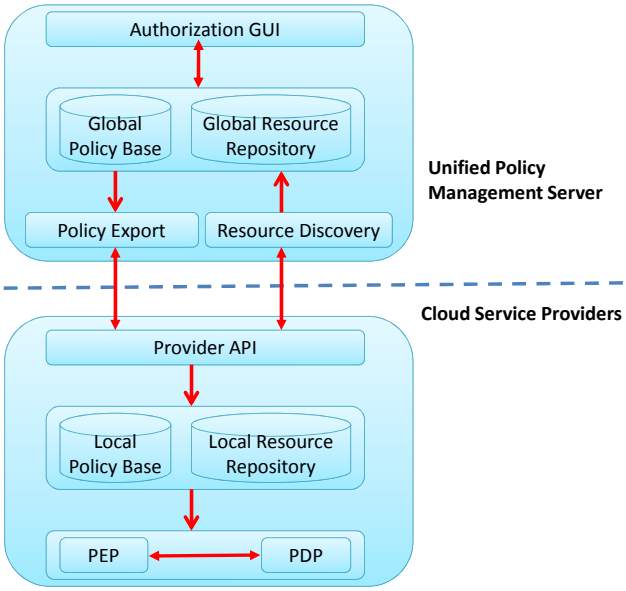
(a)

(b)

(c)

Figure 1 shows some screen shots of services and how users can modify their privacy settings using the APIs they offer. Figures 2(a) to 2(c) show resources and access control permissions related to *Amazon S3*, *Dropbox* and *Flickr*, respectively.

Figure 2 The architecture of the unified policy management framework (see online version for colours)



We developed a unified framework shown in Figure 2 to support policy management for all these services in one central place. The goal was to provide users with a unified way of specifying access policies and then export policies into the services on behalf of the users. It includes various components such as an interface that provides users with a means to register applications and services they use, discover the resources each user has on various application services, specify policies and a component to export the policies into the services through their APIs. The framework was developed to do the following tasks:

- get the name of service (i.e., *Dropbox*) with the username (i.e. Alice)
- connect to the service
- retrieve resources related to that username and display them
- provide appropriate interfaces to the user to specify access policies/privacy settings
- automatically change access control settings of the user in the services associated with the user based on user's modification of policies.

In order to integrate all five services, we used *servlet* and *jsp* to build a browser/server system so the users can access the framework using a browser. A servlet is the Java platform technology used for extending and enhancing the capabilities of web servers where host applications accessed via a request-response programming model. It does

not have the performance limitations of CGI programs and is platform-independent. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. Java server pages (JSP) technology enables Web developers and designers to develop and maintain dynamically generated web pages. In order to deploy and run a servlet, we need a web container to manage the life cycle of servlets and map a URL to a particular servlet. We also need a compatible web server with servlet to deploy and run a JSP.

The architecture is designed using model view controller (MVC). The *servlet* listens to the *jsp* page calls, runs the source code and redirects to the target page with the results inside the HTTP package. Based on the architecture, we developed a framework to integrate those five services. It displays all of users' resources associated with services and their access permissions in one central place. It also provides users with the ability to modify the permissions if they wish. Note that in some cases users' ability to specify/modify permissions depends on expressive power of the access control mechanism that each CSP uses. For example, in *Amazon S3*, users can only share a file with another user but not a group.

3.2 Limitations of the existing policy management systems

Based on what we have learned from our attempt and findings of the case study, here we discuss the limitations of the existing policy management mechanisms and requirements for a usable policy management solution that addresses those limitations and challenges of the cloud environments. We believe that approaches to access control for cloud computing environments should enable users to control how their information is shared and with whom. The users should be a core part of the access control system and they should be able to easily determine how the shared information will be used, and what the consequences of sharing this information are. However, the existing authorisation mechanisms do not support these requirements. The access control solution should provide the granularity, simplicity and usability required to respond to security and privacy challenges in the highly collaborative user-centred cloud environments.

- *Application centric vs. user centric*: Authorisation mechanisms in the existing services are bound to service providers; each CSP employs its own authorisation mechanism that is bound to its services and applications. In some cases, these access control mechanisms can only address simple scenarios where data is either made public or accessible only by a predefined set of users of the application. This limits the configuration of the application and it cannot be easily adapted to particular user's access control requirements. Alice, for example, must use the mechanisms provided by *Dropbox*, *LinkedIn*, *GoogleDocs*, *Amazon S3*, and *Mint* which may not necessarily meet all her security requirements. These mechanisms may not allow Alice to group users and assign access rights to such groups or may not support fine-grained access control policy rules. Most security novice users may choose their preferred services based on their functionality rather than based on their security features. However, security conscious users may decide to leave CSPs that do not support particular security features.

  However, we believe that users should be able to set their own access control policies for resources using their preferred policy management system. Moreover, the policy management system should enable users to apply a single access

control policy across a set of distributed resources. For example, it should be possible to specify a policy that offers access only to Alice's family members, and attach that policy to a document residing in *GoogleDocs* and *Mint*.

- *Unified policy management system*: Cloud services are controlled by different authorities and often use different policy specification mechanisms. This leads to access control policies that are composed using diverse and possibly incompatible policy languages. For example, *Dropbox* uses a different access control model than *LinkedIn* that supports a more flexible and expressive access control language. Therefore, Alice is unable to specify access control rules only once and apply these rules to her various resources such as photos, video clips and documents which are spread across different cloud services. Moreover, if Alice decides to move some of her resources from one CSP to another, for example from *Dropbox* to *Amazon S3*, then she may need to respecify the policies again.

  Different CSPs may deploy different access control mechanisms and may force users to use their specific policy management tools. This may result in an inconvenient and inconsistent User Experience (UX). For example, Alice, to be able to share her resources efficiently and securely, must learn how to use interfaces and management tools at all her cloud services which may differ significantly from one service provider to another. Some of these tools may not be designed with usability issues in mind which defeats the purpose of access control system and negatively impacts the intended use of the tool.

  In an ideal policy management system, users should be able to specify access control policies using a single policy management tool without being required to learn how to use each application's tool separately. Moreover, users should be provided with easy to use interfaces and tools to specify, edit, and manage access control policies.

- *Heterogeneity and interoperation*: Each CSP employs its own authorisation mechanism and the access control policies in existing solutions are distributed and heterogeneous. As a result, interoperation among service providers is not possible as services do not understand each other's authorisation mechanisms. Furthermore, users cannot have a consolidated view of the access control policies applied to their resources over the cloud. In our scenario, Alice does not have a holistic view of the access control policies applied to her information and resources at *Dropbox*, *LinkedIn*, *Google Docs*, *Amazon S3*, and *Mint*. Furthermore, with the increasing amount of resources that Alice may store on the various CSPs, another challenge is how to manage relations between access policies and resources. Users should be provided with a holistic view of the access control policies applied to all their resources stored at various CSPs in order to enable them to have a better understanding of the access policies and help them in managing access to distributed resources. Additionally, with the heterogeneity of access policies, in order to introduce new access control policies or modify existing ones, users need to go over all CSPs and configure access control policies appropriately which makes it a tedious task.

- *Privacy preservation*: For resources stored on different CSPs and owned by the same user, one service provider should not be able to discover the user's relationship with the other service providers.

4 The proposed semantic-based policy management framework

Based on the lessons learned and our analysis, we propose a framework which is designed on the concept of centrally expressing a users' security requirements that are applied to a user's resources regardless of where they are stored in the cloud. The framework should be able to address interoperability and heterogeneity issues as well as other requirements we discussed.

In order to deal with the heterogeneity of cloud computing environments, representations of concepts and access policy rules should be generic and flexible enough to fulfil different cloud providers' modelling requirements. We use semantic web technologies to address these representations and model concepts and semantics of different cloud providers with high expressiveness. The proposed framework should provide capabilities to users to manage access policies for services and products running on a cloud infrastructure.

In the following, we first discuss why semantic web technologies are appropriate for our purpose and then describe our proposed framework and its components. We then present the proposed *semantic-based policy management framework* for cloud computing environments that delivers policy management services.

4.1 Semantic web and policy management

In the semantic web, ontologies are used to specify a domain of interest that consists of terms representing individuals, classes of individuals, properties, and axioms that assert constraints over them. It provides a structured vocabulary that describes concepts and relationships between them as well as a specification of the meaning of terms used in the vocabulary (Takabi et al., 2009). In a policy management system, access rules are specified based on representations of concepts and policy rules and these representations should be able to make policy-based authorisation decisions. In order to deal with the heterogeneity of cloud computing environments, these representations should be generic and flexible enough to fulfil different cloud providers' modelling requirements.

We use ontology languages to address these representations, since they provide high expressiveness to model concepts and semantics of different cloud providers. Different ontology languages provide different facilities (Pérez et al., 2011). The W3C standards for ontology languages are based on RDF. The OWL 2 is a family of standard knowledge representation languages for the semantic web based on DL with a representation in RDF. Using a reasoner, we can check whether all of the statements and definitions in the ontology are mutually consistent (Motik et al., 2009). There are different kinds of OWL 2 ontologies that differ in terms of expressiveness and computational complexity. There is a tradeoff between expressiveness and efficient reasoning. The more expressive the language is, the more difficult and less efficient the reasoning is.

W3C's Web Ontology Working Group defines three different subsets of OWL 2: OWL 2 EL, OWL 2 QL and OWL 2 RL (Motik et al., 2009). OWL 2 EL is particularly useful for ontologies with very large number of properties and/or classes. It captures the expressive power used by many such ontologies and performs basic reasoning problems in time that is polynomial with respect to the size of the ontology. OWL 2 QL is useful for applications that use very large number of instance data, where query answering is the most important reasoning task. OWL 2 QL can perform sound and

complete conjunctive query answering in logspace with respect to the size of the data and it can be used to implement the ontology consistency and class expression subsumption reasoning problems using polynomial time algorithms. These two subsets target particular applications that need high performance reasoning algorithms but limit their expressiveness. On the other hand, OWL 2 RL is able to provide scalable reasoning, without sacrificing too much expressive power (Pérez et al., 2011). OWL 2 RL reasoning systems can be implemented using rule-based reasoning engines and most of the problems like ontology consistency, class expression satisfiability, class expression subsumption, instance checking, and conjunctive query answering can be implemented in polynomial time with respect to the size of the ontology.

We use OWL 2 RL in our approach to describe the concepts in the CSPs. This OWL 2 subset provides constructors to define property semantics like *inverseOf*, *transitiveProperty* or *reflexiveProperty*; constructors to define object semantics like *equivalentClass*, *disjointWith* or *unionOf*; as well as cardinality and existential restrictions like *minCardinality*, *allValuesFrom* or *someValuesFrom* among other features (Motik et al., 2009). We can also use the Semantic Web Rule Language (SWRL) to enrich the models defined using OWL 2. SWRL is used to represent rules on the semantic web and extends OWL 2 in order to provide a way to express conditional knowledge (Horrocks et al., 2004). SWRL is not decidable so we use the DL-Safe context (Motik et al., 2005) that is a restriction of SWRL and is decidable. The combination of SWRL and OWL 2 can be used to express the authorisation policy as well as more deductive processes on the system which may be relevant for authorisation decisions.

In our proposed approach, we use the combination of OWL 2 RL and SWRL with DL-Safe restriction, referred as OWL and SWRL respectively in the rest of this paper, to express and manage authorisation policies as it has the following advantages:

- it offers high expressiveness by providing a wide variety of constructors to describe knowledge
- its reasoning could be implemented using rule-based engines which offer good performance
- it provides scalable reasoning without sacrificing too much expressive power
- it provides heterogeneity management and interoperability among different CSPs
- it provides separation between domain description and policy description.

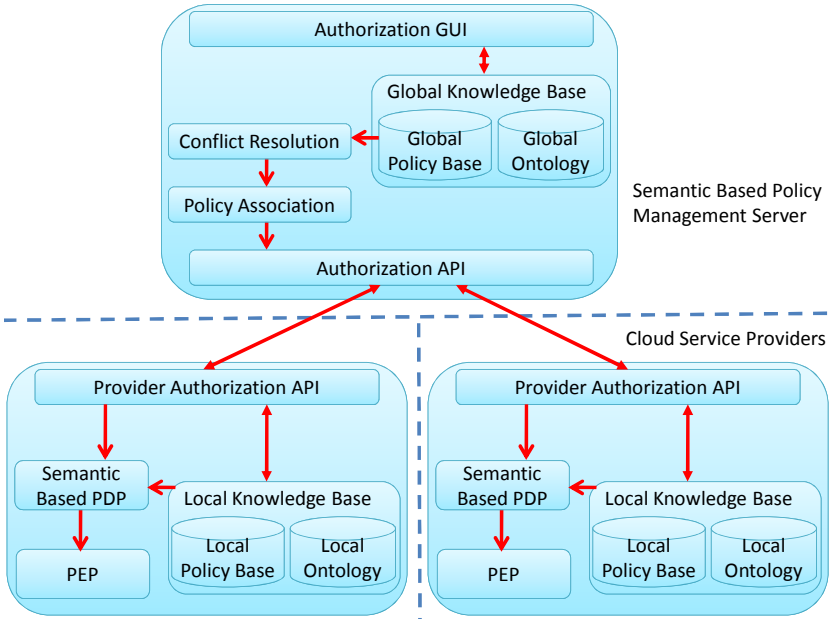
4.2 The proposed architectural framework

Our proposed semantic-based policy management framework is built on the concept of centrally expressing a user's security requirements that are applied to a user's resources scattered across the cloud. The customers do not manage or control the underlying cloud infrastructure, network, servers, operating systems, storage, or even individual application capabilities. We extend and generalise the earlier framework presented in Figure 2. Figure 3 illustrates the framework and its components which are briefly described in the followings.

At a high level, the two components of the proposed architecture are the *CSP* and the *semantic-based policy management service*.

- A CSP offers one or more cloud services that are used by cloud customers and controls access to the protected resources. It evaluates access requests made by a requester against applicable policies and is in charge of enforcing access decisions when a requester attempts to access the protected resources. Therefore, a CSP acts as a policy decision point (PDP) and policy enforcement point (PEP) (AAA, 2000).
- A semantic-based policy management service (SBPMS) enables the cloud users to specify, edit and manage their access policies. It also conducts a conflict resolution on the policies to find and resolve possible conflicts and finally exports the policies into target CSPs. Therefore, SBPMS acts as policy administration point (PAP) and a policy information point (PIP) (AAA, 2000).

Figure 3 The proposed semantic-based policy management framework (see online version for colours)



Each CSP includes a *semantic-based PDP*, a *PEP* and its own *local knowledge base*. The local knowledge base stores all the ontologies and the policy rules of the CSP. The semantic-based PDP component is in charge of making authorisation decisions while the PEP module is in charge of enforcing those decisions.

The SBPMS provides authorisation services and interfaces for cloud users to manage access policies in a single centralised location. Once the policies are specified by cloud users, it detects and resolves possible conflicts among access policies. Then, it exports policies into the CSPs. In order to do this, it first separates the policies related to each CSP based on the resource-provider association and then exports them into the associated CSP via the *provider authorisation API*. The SBPMS has several components. The *global knowledge base* is a central repository that stores all the ontologies and the policy rules gathered from different CSPs' local knowledge bases. This is done using the *provider authorisation API*s.

The *authorisation GUI* provides users with information that is required for specifying their access policies and retrieved from the global knowledge base. The *conflict resolution* module then detects and resolves possible conflicts among the specified access policies while the *policy association* module is responsible for associating the policies with their target CSPs and services. Finally, an *authorisation API* provides all the services related to the management of authorisation services and it is accessed via the *provider authorisation API*.

4.3 Authorisation knowledge management

The main component of the framework is the SBPMS. This service handles the multiple ontologies and policies of the CSPs to make use of that knowledge in policy specification. Each CSP has its own information system describing its users, services, resources and any other relevant domain information. The SBPMS requires CSPs to provide such information for authorisation purposes. The information is represented and kept up to date by CSPs in OWL 2 ontologies and SWRL rules in order to enable the policy management service to perform the semantic-based authorisation process.

Whenever a cloud user changes his policies, the SBPMS applies the required updates and communicates them to the target CSPs. This could be done by *push* and/or *pull* strategies. In the *push* strategy, whenever there is a change, the SBPMS updates the policies and exports them to target CSPs while in the *pull* strategy, the CSP initiates the communication and checks for possible policy changes/updates in certain time periods. We believe that *push* strategy is more efficient in this situation. Since the associations among policies, resources, and CSPs have been already identified, the SBPMS only needs to relate the changes to target CSPs and export them.

Similarly, whenever cloud users add/remove resources to/from CSPs, appropriate updates need to be done. In this case, we believe that *pull* strategy is better because resources are stored in the CSPs and whenever there is a change, CSPs can inform the SBPMS to get updated policies.

A question may arise regarding the privacy of cloud user's identity and privacy of his policies. The concern here is that cloud users have to trust SBPMS to provide their identifications in different CSPs and specify their access policies using the SBPMS. We assume that there is enough level of trust between cloud user and the SBPMS to deploy the service. In order to avoid privacy concerns, however, we can use cryptographic methods. We can use encrypted ontologies, encrypted ontology-mapping and encrypted queries (Mitra et al., 2006). Alternatively, the SBPMS could be deployed as private cloud within an organisation's premise or be fully controlled by an individual user.

4.4 Access request processing

The access requests are processed locally in each CSP. The request is sent to the CSP's PEP. After that, the PEP forwards the request to the semantic-based PDP for an access decision. The semantic-based PDP connects to the up-to-date local knowledge base to get the information it needs for decision making. The local knowledge base returns the information in OWL and SWRL to the semantic-based PDP. This information is used in the reasoning process to infer new knowledge based on the information provided and make an access decision. The result of this process is used by the service to generate the access response, which is returned to the PEP. Finally, the PEP enforces the decision and returns an access response to the requester.

The key advantage of our proposed framework is that with existing systems, the cloud user is limited to the functionality provided by the CSPs' policy engine. However, our proposed framework may be able to apply additional policies by transforming them into the provider's policies. For instance, in *Amazon S3* or *Dropbox*, if Alice wants to share a file with a group of users, she has to specify different policies for each user. In our system, however, she can specify a policy to share the file with a group of users called colleague and our system exports required policies (one policy per each user of the group) into the *Amazon S3* or *Dropbox*.

5 Reasoning and conflict analysis

Using semantic web languages like OWL 2 and SWRL provides us with powerful expressiveness to specify access policies and to satisfy the modelling requirements of different CSPs. However, for the policy management service to be able to understand the specified policies, we need to define a minimum set of concepts (Pérez et al., 2011). These concepts should be specified so the policy management service can provide a positive or negative authorisation statement. Note that the definition of such concepts does not restrict the CSPs' ontologies. The CSPs can either directly use this set of concepts or define their own concepts. In the latter case, however, they should provide a few OWL constructors or SWRL rules to map these concepts to those which are required by the policy management service to perform the authorisation process.

There are several tools for definition of domain applications in OWL ontologies and their modification (Pérez et al., 2011). We use *Protege* which is an open source ontology editor and knowledge-base framework to describe and edit ontologies (Protege, 2011). The *Protege* successfully deals with OWL and SWRL.

5.1 The reasoning process

The OWL and SWRL reasoner in the semantic-based PDP component performs a reasoning process to make access decision. The reasoner includes three main operations: *inference*, *validation*, and *querying the ontology* as described below.

- Inference: Using the information which is available in the ontology we infer new knowledge about the CSP. The results of inference on SWRL rules are new individuals or OWL instances which are used to make the access decision.
- Validation: The validation operation is used to detect whether constraints expressed using OWL 2 language are violated by some dataset. In other words, the validation process looks for possible inconsistencies.
- Querying the ontology: This operation is used for instance recognition and inheritance recognition. The instance recognition tests whether an individual is instance of a class expression whereas the inheritance recognition examines if a class is subclass of another class or if a property is sub-property of another property. Using this operation, we can formulate generic queries that refer to abstract concepts and consequently the system would be able to recognise instances that belong to concrete subclasses or sub-properties of such a concept.

5.2 Policy conflict analysis

One of the important functionalities of a reasoner is the ability to detect and resolve conflicts (Pérez et al., 2011). There are two main categories of conflicts in policy evaluation: semantic conflicts and syntactic conflicts. The semantic conflicts are results of using information related to the current state of the system. So their appearance depends on the dynamic state of the application domain and they are really difficult to detect. The syntactic conflicts, however, are results of specification errors in the policy or they may also be legitimately derived from other rules. Their appearance does not depend on state of the application domain and can be detected by simply looking at structure of the rules.

Furthermore, the heterogeneous environments of cloud computing composed of different CSPs may lead to more policy conflicts among different CSPs' policies. One of the syntactic conflicts is modality conflicts that can be detected from the rule syntax that specifies access policies. A modality conflict occurs when two or more policies with modalities of opposite sign refer to the same subjects, objects and actions.

Semantic conflicts, on the other hand, can be detected using some meta-policies that describe undesirable situations which may appear in the domain. One situation where this kind of conflicts may occur is when a subject can change its own permissions. For instance, if *Subject1* is allowed to change his own permissions, he may be able to perform *Action2* on *Object1* which initially was forbidden for him. We can define a meta-policy to detect such a situation.

Once the system detects conflicts, a conflict resolution is needed to provide an automatic solution for resolving detected conflicts. We use policy prioritisation as a solution to resolve the authorisation conflicts. If the conflicts occur between policies specified by different CSPs, we assign a priority to each CSP to resolve the conflict. If the conflicts are between policies of the same CSP, we assign priorities to the rules or the authorisation decisions. For example, we can state that negative policies have priority over positive ones.

6 Implementation

In this section, we explain a proof of concept implementation of our proposed system and present our performance evaluation results.

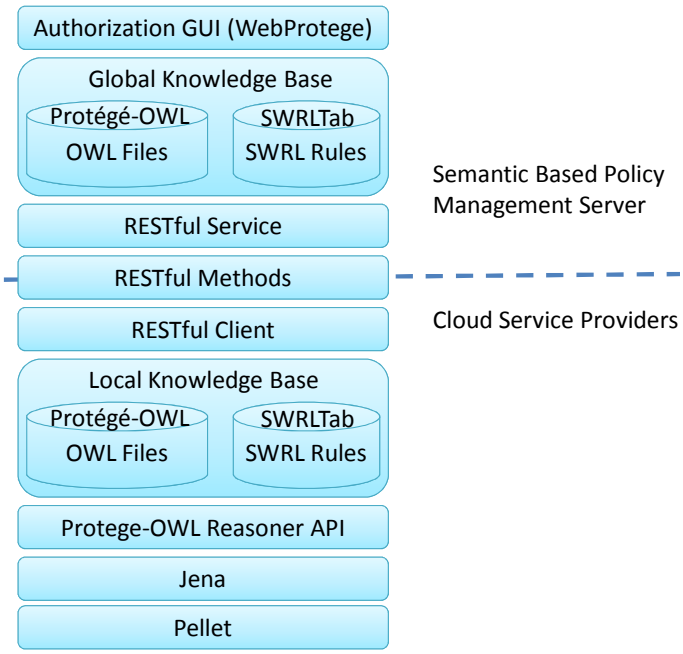
6.1 The implementation architecture

Our implementation includes developing a Java library to abstract the proposed framework. The library provides a simple interface which allows the cloud users to manage the ontologies, as well as to specify the authorisation policies. Our implementation contains two different Java-based components: the CSP and the SBPMS. The SBPMS contains authorisation API, knowledge management and authorisation GUI and the CSP contains the provider authorisation API, knowledge management and semantic-based PDP.

As shown in Figure 4, the authorisation API and the provider authorisation API are published by means of RESTful web service technology, which enables the remote invocation of the methods for the different parties involved. A RESTful web service is

implemented in the SBPMS side and the CSPs are implemented as RESTful clients. The provider authorisation API establishes a secure channel to the semantic-based policy management server using *SSL*. Authentication of the provider and server is done using *OAuth* which is an open protocol to allow secure API authorisation in a simple and standard method from web applications and a simple way to publish and interact with protected data (OAuth, 2010). The authorisation API provides methods to *insert*, *remove*, *update*, *access* and *search* information in the knowledge base.

Figure 4 The implementation architecture of the proposed framework (see online version for colours)



Our proposed framework uses *Protege* to provide an optimised solution to efficiently manage a large knowledge base with authorisation information (Protege, 2011). *Protege* is a Java tool that provides an extensible architecture and a suite of tools for the creation of customised knowledge-based applications with ontologies. We use *Protege-OWL* (http://protege.stanford.edu/overview/protege-owl.html) that is an editor to enable users to build ontologies for the semantic web, in particular in the OWL. It implements a rich set of knowledge-modelling structures and actions that support the creation, visualisation, and manipulation of ontologies in various representation formats. For SWRL rules, our implementation makes use of SWRLTab (http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTab) which is a development environment for working with SWRL rules in Protege-OWL. It supports the editing and execution of SWRL rules. Further, Protege can be extended by way of a plug-in architecture and a Java-based API for building knowledge-based tools and applications.

In order to reason over the ontologies represented in Protege-OWL, we use *Jena* (Carroll et al., 2004) and *Pellet* (Sirin et al., 2007). *Jena* is used as a general Java API to manage ontologies and *Pellet* is used as DL reasoner. *Jena* is

the standard de facto Java library to manage ontologies and *Pellet* supports high expressiveness dealing with OWL 2 ontologies and performs incremental consistency checking (Perez et al., 2011). Our implementation uses the *Protege-OWL Reasoner API* (http://protegewiki.stanford.edu/wiki/ProtegeReasonerAPI) that provides programmatic access to a reasoner. It provides methods for consistency checking, classification, etc. of an ontology as well as methods for getting the inferred information for a particular OWL entity. The *ProtegePelletJenaReasoner* implementation converts a *Protege-OWL* model into a *Jena* model and then it uses the existing *Pellet* reasoner connection available in *Jena* for the inference.

The authorisation GUI is a graphical web user interface that enables cloud users to access authorisation services in a friendly and usable way. Our implementation uses *WebProtege* (http://protegewiki.stanford.edu/wiki/WebProtege) which is a lightweight, web-based ontology editor developed to support the process of ontology development in a web environment. The cloud users can use this interface to browse the knowledge base, as well as specify authorisation policies and request authorisation proofs.

6.2 Performance evaluation

After implementing our prototype based on the architecture discussed above, we present its performance evaluation. We used two different machines to do our experiments: a server and a client.

The server is an Intel Core i5-520M 2.40 GHz with 8 Gbytes RAM and Windows 7 Ultimate running the SBPMS server. The client is an AMD Opteron 252 2.60 GHz with 8 Gbytes RAM and Windows 7 Ultimate running an application that simulates CSPs that utilise the SBPMS server using the authorisation client API. The SBPMS is launched as a web service and the simulator starts 100 threads each representing a CSP using the policy management system. All the threads are started in parallel to stress the policy management system with concurrent invocations.

6.2.1 Semantic-based specification language and policy generation process

Since we did not have any real data, we simulated OWL files and SWRL rules. We generated 100 separate OWL files, one for each CSP and 200 SWRL rules using those OWL files. In order to do this, we used a semantic-based specification language based on the meta-model shown in Figure 5. All the entities in the system are modelled as OWL classes and are as follows:

Subject rdfs:subClassOf owl:Thing

Role rdfs:subClassOf owl:Thing

Object rdfs:subClassOf owl:Thing

Action rdfs:subClassOf owl:Thing

Attribute rdfs:subClassOf owl:Thing

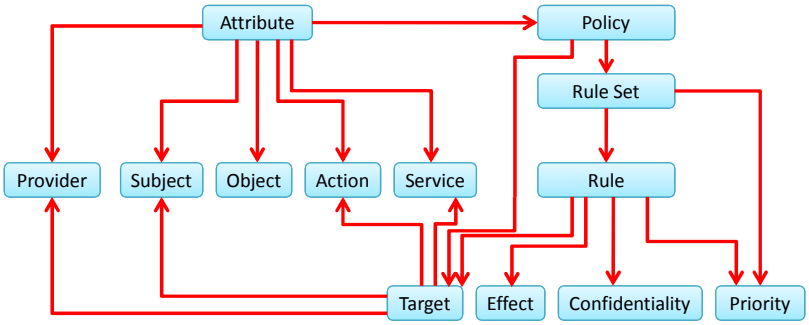
Provider rdfs:subClassOf owl:Thing

Service rdfs:subClassOf owl:Thing

In the data simulation process, we defined all entities in OWL ontology, their properties, relationships, etc. We generated instances of these classes including 100 providers,

500 subjects, 2,000 objects, eight actions and between one and ten services for each CSP. These services are randomly assigned to the CSPs and each of the 100 providers has at least one and at most ten services. The actions we used in data generation process are *read*, *write*, *execute*, *parallel read*, *parallel write*, *parallel execute*, *parallel read and write*, *parallel read and execution*. This data was generated and stored at 100 separate OWL files each associated with one CSP. Note that these OWL files are not completely different and there are some similarities among them as in real world scenarios where users have similar resources stored at multiple CSPs.

Figure 5 The policy specification language meta model (see online version for colours)



Our next step in data generation was to generate SWRL rules. As shown in Figure 5, a typical access rule in our system includes a combination of zero or one target, an effect, zero or one priority and confidentiality. The target of the rule is presented by a 5-tuple [*Provider, Subject, Object, Action, Service*] where provider represents the CSP that owns the object and assigns the access permission to subjects and service represents cloud service which is associated with the object. This 5-tuple is interpreted as follows: the *Provider* states that the *Subject* is allowed to perform the *Action* over the *Object* associated with the *Service*. The provider part of the rule could be any CSP that owns some objects and offers some services. The subject of the rule can be a single user, role or user group of any CSP. The object part of the rule can be a single object or object group controlled by CSPs. If the action part of the rule is empty, the rule is applied to all the actions of the subject and object. The service part of the rule can be a service or service group provided by any CSP. The effect part of a rule defines an authorisation result and could have one of the following values: allow, deny, oblige, refrain. We use confidentiality of a rule for feedback of the denied request. We generated 200 SWRL rules based on the data generated in OWL files.

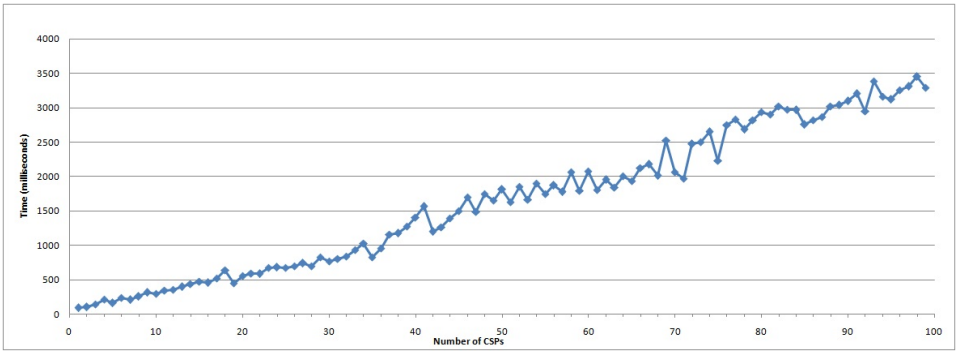
6.2.2 Performance of the ontology construction

First, we looked at performance of the authentication process. Our results show that it takes the SBPMS only about 25 milliseconds to authenticate itself to a CSP. Our experiments also show that the number of CSPs does not have any significant effect on this as expected.

In order to enable users to specify their access policy rules, the SBPMS first needs to provide them with information about all their resources. In this process, that is done only once at the beginning of the deployment of the framework, the SBPMS connects to knowledge bases of all the CSPs and retrieves information about all the resources the

user has stored at the CSPs. This is essentially a construction of the global ontology in which the SBPMS gathers information from OWL files of all the CSPs and constructs the global ontology to be used for policy specification by the user. Figure 6 shows the performance of this process. As we can see, for the first CSP it takes less than 100 milliseconds to fetch required information to the server. For ten CSPs, it takes about 300 milliseconds and for 40 CSPs the time it takes to fetch the ontologies is one second. For 100 CSPs, it takes less than 3.5 seconds to fetch all the ontologies stored at all the CSPs to the server. This may seem hight but considering the large number of CSPs and size of the OWL files, it is an acceptable and reasonable performance.

Figure 6 Global ontology construction process (see online version for colours)



6.2.3 Performance of the authorisation API

Next, we looked at performance of the authorisation API and the provider authorisation API. Whenever there is a change in one of the CSPs, for example, user adds a new object, adds a new subject, etc, we need to update the OWL file in the SBPMS side to reflect these changes. This experiment simulates the *pull* strategy that was discussed in Section 4.3. The results of this experiment are shown in Figure 7. As it can be seen, when there is only one CSP it takes only 75 milliseconds to update the ontology base of the SBPMS server. If we increase the number of CSPs to 40, it takes 150 milliseconds to update the server. With 100 CSPs, the time it takes to update the server when there is a change in the CSPs' side is always less than 300 milliseconds (with the exception of two points as demonstrated).

Similar to OWL files, when there is a change in SWRL rules at the SBPMS side, for example user adds a new rule, modifies an existing rule, deletes a rule, etc, these changes need to be reflected at the CSPs side too. The SBPMS should update SWRL rules files of the CSPs that are related to the change (determined by policy association module of the framework). This experiment simulates the *push* strategy that was discussed in Section 4.3. The results of this experiment are shown in Figure 8. As it can be seen, for ten CSPs it takes less than 200 milliseconds to update the policy base of the CSPs when there is a change in the policy base of the SBPMS server and the number of the CSPs does not have much effect on performance of this process. reasonable.

Figure 7 OWL ontology update (see online version for colours)

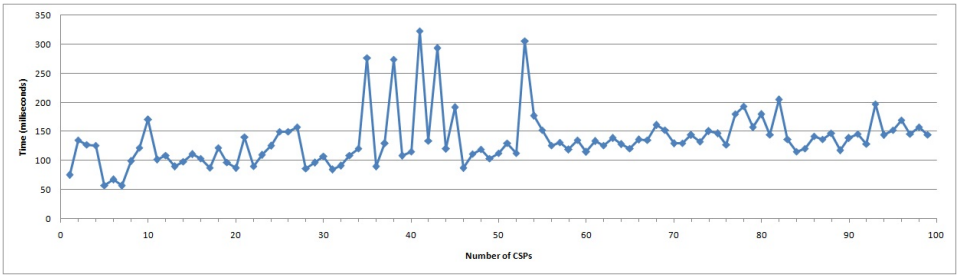
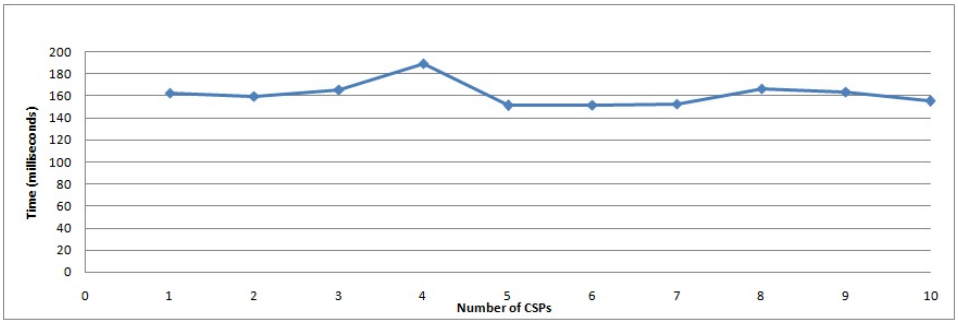


Figure 8 SWRL rules update (see online version for colours)



Note that for all the experiments, we ran each of them five rounds and the results reported here are average of those five rounds of experiments. Overall, the results show that our proposed framework performs well and the time required for policy management is reasonable.

7 Related work

The NIST cloud efforts intend to promote the effective and secure use of the technology within government and industry by providing technical guidance and promoting standards. The NIST has released a definition of cloud computing and also documents on how effectively and securely use the cloud computing paradigm (Mell and Grance, 2011). The cloud security alliance is an effort to facilitate the mission to create and apply best practices to secure cloud computing (Cloud Security Alliance, 2011). Its report, 'Security guidance for critical areas of focus in cloud computing', outlines areas of concern and guidance for organisations adopting cloud computing. The goal is to provide security practitioners with a comprehensive roadmap for being proactive in developing positive and secure relationships with cloud providers (Cloud Security Alliance, 2011).

Takabi et al. (2010a) discuss security and privacy challenges in cloud computing environments. They present unique issues of cloud computing that exacerbate security and privacy challenges in clouds, discuss these challenges and provide some possible approaches and research directions to address them. They have also proposed SecureCloud, a comprehensive security framework for cloud computing environments

(Takabi et al., 2010b). The framework consists of different modules to handle security, and trust issues of key components of cloud computing environments. These modules deal with issues such as identity management, access control, policy integration among multiple clouds, trust management between different clouds and between a cloud and its users, secure service composition and integration, and semantic heterogeneity among policies from different clouds.

Jaeger and Schiffman (2010) discuss security challenges in the cloud, foundation of future systems' security and key areas for improvement of cloud computing environments. Kandukuri et al. (2009) present security issues that have to be included in service layer agreement (SLA) in cloud computing environments. Jensen et al. (2009) provide an overview on technical security issues of the cloud. They start with real-world examples of attacks performed on the *Amazon EC2* service, then give an overview of existing and upcoming threats to the cloud. They also briefly discuss appropriate countermeasures to these threats, and further issues to be considered in future research.

Yague et al. (2003) have developed the semantic access control (SAC) model that applies semantic web technologies to the access control in open, heterogeneous and distributed systems. The model makes use of different layers of metadata to take advantage of the semantics of different components relevant for access decision purposes. The semantic approach of the SAC model helps to achieve semantic interoperation among different components of access control systems. The authors have developed the semantic policy language (SPL) for specification of access policies as an application of the SAC model.

Pérez et al. (2011) propose a Grid middleware based on semantic web technologies to specify, manage and enforce security policies in a grid computing. They use Globus Toolkit which is an open source software toolkit and a reference implementation for grid systems as base middleware. The authors use ontologies to provide a representation of the underlying information system and the resources. They specify access policies using semantic-aware rules which enables administrators to create higher-level definitions with more expressiveness. Their proposed architecture supports multiple authorisation domains, deals with heterogeneity of Grid systems and enables organisations to use their own domain concepts without having knowledge about the rest of participant organisations. They also developed a proof of concept implementation and tested in Globus to show the feasibility of the solution.

Hu et al. (2009) present a new Semantic Access Control Policy Language (SACPL) in order to overcome to the limitations of traditional access control systems in the cloud computing environments. They introduce Access Control Oriented Ontology System (ACOOS) as the semantic basis of SACPL that aims to solve the interoperability issue of distributed access control policies. The ACOOS is used to annotate some syntax elements of XACML, such as subject, object, action and attribute variables with semantic information. The authors also add some syntax elements such as priority and confidentiality.

Calero et al. (2010) propose a multi-tenancy authorisation system for cloud computing that is suitable for middleware service in the PaaS layer. Their proposed authorisation model supports multi-tenancy, RBAC, hRBAC, path-based object hierarchies, and federation. The authors also present an architecture for implementing the authorisation model, describe a prototype implementation and its performance evaluation.

8 Conclusions

The access control issues in cloud computing environments particularly heterogeneity and interoperation are the main focus of this paper. CSPs use diverse access control mechanisms that may result in access control policies being composed in incompatible ways. The application of semantic web to access control field provides a lot of benefits making it a natural solution to interoperability of heterogenous CSPs.

We presented lessons we learned from a case study where we implemented a unified policy management system for various cloud services. Based on those lessons and motivated by limitations of existing approaches, we proposed a semantic-based policy management framework that is designed to help cloud users to specify and manage security policies using semantic web technologies. We presented our proposed framework and described its components. Furthermore, we explained reasoning process and conflict detection and resolution approaches supported by our framework.

Finally, we explained a proof of concept implementation of the proposed framework to show its applicability and reported results of the experiments we performed to evaluate performance of the framework.

Acknowledgements

This research has been supported by the US National Science Foundation Award IIS-0545912. The authors would like to thank anonymous reviewers for their helpful comments.

References

The AAA Authorization Framework (2000) (online) Available at http://tools.ietf.org/html/rfc2904 (accessed on 10 November 2011).

Bruening, P.J. and Treacy, B.C. (2009) *Cloud Computing: Privacy, Security Challenges*, Bureau of National Affairs, Inc., available at www.hunton.com/files/tbl_s47Details/FileUpload265/2488/CloudComputing_Bruening-Treacy.pdf (Accessed 10 November 2011).

Calero, J.M.A., Edwards, N., Kirschnick, J., Wilcock, L. and Wray, M. (2010) 'Toward a multi-tenancy authorization system for cloud services', *IEEE Security and Privacy*, Vol. 8, No. 6, pp.48–55.

Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A. and Wilkinson, K. (2004) 'Jena: implementing the semantic web recommendations', *Proceedings of the 13th international World Wide Web Conference*, pp.74–83.

Catteddu, D. and Hogben, G. (2009) 'Cloud computing: benefits, risks and recommendations for information security', ENISA, available at http://www.enisa.europa.eu/act/rm/files/deliverables/cloud-computing-risk-assessment/at_download/fullReport (accessed on 10 November 2011).

Cloud Security Alliance (2011) 'Security guidance for critical areas of focus in cloud computing V3.0', available at https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf (accessed on 10 November 2011).

Fielding, R.T. and Taylor, R.N. (2002), 'Principled design of the modern web architecture', *ACM Transactions on Internet Technology (TOIT)* Vol. 2, No. 2, pp.115–150.

Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B. and Dean, M. (2004) 'SWRL: a semantic web rule language combining OWL and RuleML', Technical report, W3C.

Hu, L., Ying, S., Jia, X. and Zhao, K. (2009) 'Towards an approach of semantic access control for cloud computing', *Proceedings of the 1st International Conference on Cloud Computing (CloudCom'09)*, Beijing, China, pp.145–156.

Jaeger, T. and Schiffman, J. (2010) 'Outlook: cloudy with a chance of security challenges and improvements', *IEEE Security and Privacy*, Vol. 8, No. 1, pp.77–80.

Jensen, M., Schwenk, J., Gruschka, N. and Iacono, L.L. (2009) 'On technical security issues in cloud computing', *Proceedings of the 2nd IEEE International Conference on Cloud Computing (Cloud 2009)*, Bangalore, India, pp.109–116.

Kandukuri, B.R., Paturi, R. and Rakshit, A. (2009) 'Cloud security issues', *Proceedings of the 6th IEEE International Conference on Services Computing (SCC '09)*, Bangalore, India, pp.517–520.

Machulak, M.P. and Van Moorsel, A. (2010), 'Architecture and protocol for user-controlled access management in Web 2.0 applications', *Proceedings of the First Workshop On Security And Privacy In Cloud Computing*, Genoa, Italy, pp.62–71.

Mell, P. and Grance, T. (2011) *The NIST Definition of Cloud Computing*, (online) NIST Special Publication 800-145, available at http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf (accessed on 10 November 2011).

Mitra, P., Pan, C.C., Liu, P. and Atluri, V. (2006), 'Privacy-preserving semantic interoperation and access control of heterogeneous databases', *Proceedings of the ACM Symposium on InformAtion, Computer and Communications Security (ASIACCS '06)*, pp.66–77.

Motik, B., Sattler, U. and Studer, R. (2005) 'Query answering for OWL-DL with rules', *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 3, No. 1, pp.41–60.

Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A. and Lutz, C. (2009) 'OWL 2 web ontology language: profiles', W3C recommendation, W3C.

OASIS eXtensible Access Control Markup Language (XACML), available at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml (accessed on 10 November 2011).

OAuth (2010) (online) Available at http://oauth.net/ (accessed on 10 November 2011).

Pérez, J.M.M., Bernabé, J.B., Calero, J.M.A., Clemente, F.J.G., Pérez, G.M. and Skarmeta, A.F.G. (2011) 'Semantic-based authorization architecture for grid', *Journal of Future Generation Computer Systems*, Vol. 27, No. 1, pp.40–55.

Protege (2011) (online) Available at http://protege.stanford.edu (accessed on 10 November 2011).

Prud'hommeaux, E. and Seaborne, A. (2008) 'SPARQL query language for RDF', W3C recommendation, W3C, January.

Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A. and Katz, Y. (2007) 'Pellet: a practical OWL-DL reasoner', *Journal of Web Semantics*, Vol. 5, No. 2.

The Secure Shell (SSH) Protocol Architecture (2006) (online) Available at http://tools.ietf.org/html/rfc4251 (accessed on 10 November 2011).

Takabi, H. and Joshi, J.B.D. (2012) 'Policy management as a service: an approach to manage policy heterogeneity in cloud computing environment', *Proceedings of the 45th Hawaii International Conference on System Sciences (HICSS)*, Maui, HI, USA.

Takabi, H., Kim, M., Joshi, J.B.D. and Spring, M.B. (2009) 'An architecture for specification and enforcement of temporal access control constraints using OWL', *Proceedings of the 2009 ACM Workshop on Secure Web Services (SWS '09)*, Chicago, IL, USA, pp.21–28.

Takabi, H., Joshi, J.B.D. and Ahn, G.J. (2010) 'Security and privacy challenges in cloud computing environments', *IEEE Security and Privacy*, Vol. 8, No. 6, pp.25–31.

Takabi, H., Joshi, J.B.D. and Ahn, G.J. (2010) 'SecureCloud: towards a comprehensive security framework for cloud computing environments', *Proceedings of the 1st IEEE International Workshop Emerging Applications for Cloud Computing (CloudApp 2010)*, Seoul, South Korea, pp.393–398.

Yague, M.I., Mana, A., Lopez, J. and Troya, J.M. (2003) 'Applying the semantic web layers to access control', *Proceedings of the 14th International Workshop on Database and Expert Systems Applications (DEXA '03)*, pp.622–626.