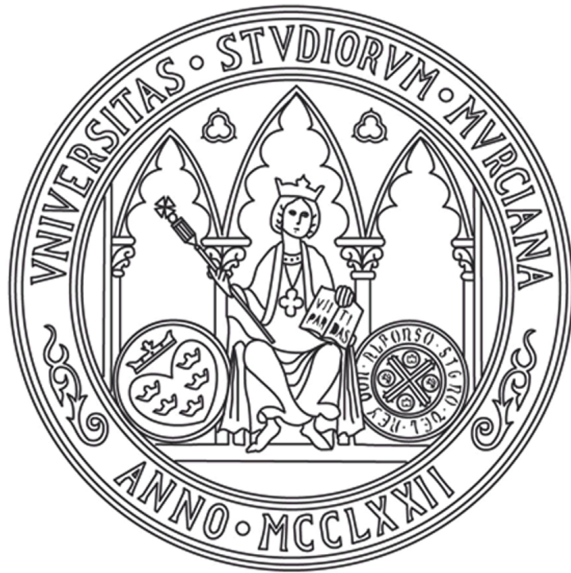




UNIVERSIDAD DE MURCIA

FACULTAD DE INFORMÁTICA

Advanced Authorization Model for Service and Resource
Protection in Distributed Systems and Cloud Computing

Modelo Avanzado de Autorización para Protección de Servicios
y Recursos en Sistemas Distribuidos y Computación Cloud

D. Juan Manuel Marín Pérez
2015



Universidad de Murcia

Departamento de Ingeniería de la Información y las Comunicaciones

**Advanced Authorization Model for Service and Resource Protection
in Distributed Systems and Cloud Computing**

PhD Dissertation

**Modelo Avanzado de Autorización para Protección de Servicios y
Recursos en Sistemas Distribuidos y Computación Cloud**

Tesis Doctoral

Presentada por:

D. Juan Manuel Marín Pérez

Supervisada por:

Dr. Gregorio Martínez Pérez

Dr. Antonio Fernando Skarmeta Gómez

A Fina...
A mis padres y hermano...
A mis abuelos...

Agradecimientos

Mi más sincero agradecimiento a todas aquellas personas que me han acompañado y apoyado durante todos estos años, haciendo que este trabajo de investigación sea hoy una realidad. En especial a Fina, por cambiar el argumento de mi vida, a mis padres y hermano, por su apoyo incondicional, por su cariño y por tanto que agradecer... vosotros me habéis permitido seguir adelante para cumplir mis sueños.

Mi mayor agradecimiento a Gregorio por su tutela durante esta tesis pero, sobre todo, por la confianza depositada en mí, por su apoyo y ayuda todo este tiempo y por ser un referente en mi vida, tanto profesional como personal, sin cuyo apoyo y confianza hoy no habría llegado hasta aquí.

Agradecer también a Antonio su tutela durante la tesis y la oportunidad de trabajar y colaborar en este grupo de investigación, donde he podido aprender y comenzar mi carrera profesional.

Gracias a todos los compañeros y amigos de la Universidad de Murcia por vuestra amistad y compañía, en particular al Departamento de Ingeniería de Información y las Comunicaciones y al grupo ANTS. Muy especialmente a Manolo y a Jorge, por tantos buenos ratos compartidos, por acompañarme en los buenos momentos y soportarme en los no tan buenos durante todos estos años en la Universidad.

Sin todos vosotros, este trabajo no habría sido posible...

Resumen

La computación en la Nube (computación Cloud o *Cloud computing*) es un paradigma de computación emergente que permite ofrecer servicios de tecnologías de la información bajo petición de los clientes que demandan un uso de dichas tecnologías. Muchas compañías están actualmente adaptando sus sistemas de información para utilizar servicios de computación Cloud, donde los servicios e infraestructuras son flexibles y dinámicos, y pueden escalar y distribuirse bajo demanda. Estos sistemas hacen uso de diversas capas de abstracción y virtualización para proporcionar una serie de recursos y servicios. Normalmente se consideran tres capas principales en estos sistemas, según el tipo de servicio que se proporciona. Concretamente, Infraestructura como Servicio (o IaaS por sus siglas en inglés), que proporciona una infraestructura virtual con recursos tales como máquinas virtuales, volúmenes de almacenamiento o infraestructura de red; Plataforma como Servicio (o PaaS), que proporciona servicios intermedios tales como plataformas de programación y servidores de aplicaciones; y Software como Servicio (o SaaS), que expone aplicaciones software para ser usadas por los usuarios finales. Este modelo posibilita un suministro eficiente de arquitecturas virtuales de TI a terceras partes, donde los recursos son creados y desmantelados dinámicamente de acuerdo con las necesidades del cliente.

Sin embargo, muchas compañías que estarían potencialmente interesadas en el Cloud son todavía reacias a la adopción de este paradigma debido principalmente a la preocupación y falta de confianza en cuanto a la seguridad y privacidad que actualmente ofrece esta tecnología. La mayoría de sistemas de seguridad actuales se basan en lo que se denomina seguridad perimetral, la cual consiste en implantar mecanismos de seguridad adecuados en las fronteras de la organización para proteger los recursos que se encuentran en su interior. Sin embargo, la computación Cloud rompe las barreras de la organización y el concepto de perímetro. Cuando los datos residen en el Cloud, estos residen fuera de los límites de la organización, concretamente en el proveedor de servicios de Cloud (o CSP por sus siglas en inglés). Esto provoca en el usuario una sensación de pérdida de control sobre sus datos y causa una razonable preocupación sobre la seguridad que ralentiza la adopción de la computación Cloud. Es por ello que el sistema de control de acceso a los recursos disponibles se convierte en una pieza clave y un aspecto crítico a la hora de proporcionar un control eficiente, confiable y seguro sobre el uso de los recursos del Cloud.

La gestión del control de acceso y la seguridad puede ser una tarea difícil y propensa a fallos en sistemas distribuidos como las infraestructuras Cloud. Las soluciones basadas en políticas pueden ayudar a controlar y gestionar la seguridad, ayudando a los administradores con esta tarea mediante la especificación de políticas de seguridad de alto nivel que son automáticamente aplicadas en el sistema. Sin embargo, las actuales soluciones de autorización para computación Cloud normalmente no son capaces de proporcionar suficiente expresividad para describir reglas avanzadas de autorización y federación que controlen un acceso seguro y confiable a los recursos. La disponibilidad de un sistema avanzado de autorización sería una característica diferenciadora para los proveedores de servicios Cloud, que demandan el diseño y desarrollo de nuevos modelos de autorización adecuados que mejoren el control de acceso a los recursos del Cloud.

Adicionalmente, el traslado de los datos al Cloud normalmente implica una confianza en el proveedor de servicio respecto a la protección de los datos, lo que también provoca una preocupación e incertidumbre en potenciales clientes y organizaciones usuarias de la computación Cloud. ¿Estará realmente el proveedor de servicio accediendo a los datos para su propio beneficio? ¿Estará aplicando fielmente las políticas de control de acceso definidas por el usuario? Aunque todos estos aspectos normalmente son gestionados legalmente en base a acuerdos de servicio (*Service Level Agreements* o SLAs en inglés), el CSP podría realmente acceder a los datos e incluso cederlos a terceras partes interesadas. Además, el cliente propietario de los datos debe confiar en que el CSP esté aplicando correctamente las políticas de control de acceso a los datos definidas por él para controlar el acceso de otros usuarios.

El problema se vuelve incluso más complejo cuando surgen alianzas y federaciones entre diferentes proveedores de servicio. Así, un CSP podría hacer uso de los servicios o la infraestructura virtual ofrecida por otros para proporcionara sus propios servicios a sus clientes. Este concepto de Clouds interconectados o *Inter-Cloud* incrementa los retos a la hora de abordar la seguridad y privacidad en estos entornos, ya que los datos pueden estar gestionados por distintos proveedores de servicio. Los usuarios pueden perder totalmente el control sobre sus datos en el Inter-Cloud. Incluso las relaciones de confianza entre los distintos CSPs de la federación que se intercambian los datos está fuera del control del usuario propietario de los datos. Esta situación lleva a replantearse las aproximaciones de seguridad y protección de datos actuales y empezar a considerar nuevas aproximaciones de seguridad centradas en los datos (*data-centric*) donde los datos están auto-protegidos con mecanismos criptográficos de seguridad donde quiera que se encuentren. El cifrado de datos es el método más comúnmente utilizado para proteger los datos, proporcionando confidencialidad y protección contra accesos no deseados cuando los datos están en el Cloud. Los trabajos de investigación más recientes en este campo intentan proporcionar diferentes aproximaciones hacia datos cifrados con capacidad de autorización que permitan que los datos estén auto-protegidos en el Cloud, asegurando técnicamente que no puedan ser accedidos por el CSP o cedidos a terceras partes no autorizadas.

Contribución

Esta tesis doctoral se centra en el control de acceso en sistemas distribuidos, con especial atención a su aplicación en sistemas de computación Cloud. Como principal resultado de este trabajo, se proporciona una solución de control de acceso altamente expresiva, confiable y segura, adecuada y adaptada a entornos de computación Cloud. Ha sido diseñada teniendo en cuenta las características de este tipo de entornos y permite una gestión de seguridad proporcionando características de autorización avanzadas.

Las principales contribuciones de la solución propuesta pueden resumirse en las siguientes:

- Una aproximación basada en políticas para control de acceso en entornos distribuidos que permite a los administradores mantener sistemas Cloud bajo control de manera segura, intuitiva y escalable.
- Un modelo de autorización avanzado con alta expresividad para control de acceso basado en semánticas comúnmente utilizadas y bien conocidas para las reglas de control de acceso siguiendo el esquema de Control de Acceso Basado en Roles (o RBAC por sus siglas en inglés).
- La aplicación de tecnologías de Web semántica al campo del control de acceso en sistemas distribuidos para proporcionar capacidades de razonamiento sobre políticas y descripciones de dominio.
- Una aproximación RBAC centrada en los datos (*data-centric*), en la que los datos son cifrados y auto-protegidos con la aplicación de un modelo basado en reglas con expresividad RBAC extendida.

Se define un modelo de control de acceso sofisticado y avanzado que permite la especificación y validación de políticas de alto nivel, proporcionando una alta expresividad en el lenguaje de definición de políticas y teniendo en cuenta información de dominio compleja. Esto simplifica las labores de gestión de seguridad, facilitando a los administradores el proceso de definición y despliegue de políticas, teniendo en cuenta la heterogeneidad de los distintos dominios de la computación Cloud. Para la definición de elementos del dominio se hace uso de estándares de representación como el Modelo Común de Información (*Common Information Model* o CIM) [1] desarrollado por el DMTF, con el que se obtiene una alta interoperabilidad en la representación que permite tratar con la heterogeneidad presentada por los diferentes dominios del Cloud. Junto con el modelo de autorización se proporciona también una arquitectura de seguridad para su aplicación en entornos de computación Cloud. Tanto el modelo como la arquitectura han sido especialmente diseñados teniendo en cuenta la naturaleza *multi-tenancy* de este tipo de entornos, donde múltiples usuarios hacen un uso compartido de una misma infraestructura, concretamente la infraestructura del CSP que ofrece sus servicios a los distintos clientes.

La solución proporciona soporte para *multi-tenancy*, incluyendo capacidades federativas para dar soporte a situaciones en las que distintos usuarios deseen compartir recursos en el Cloud, permitiendo una definición detallada de qué recursos pueden ser accedidos para cada usuario particular.

Además de la definición de reglas de autorización basadas en roles que proporciona el esquema RBAC, el modelo de autorización propuesto soporta las siguientes características de expresividad:

- RBAC jerárquico (*hierarchical RBAC* o hRBAC). Permite establecer jerarquías entre los roles definidos en el modelo de autorización.
- RBAC condicional (*conditional* o cRBAC). Permite establecer condiciones de contexto o entorno a las reglas de autorización definidas.
- Objetos jerárquicos (*hierarchical objects* o HO). Permite establecer jerarquías entre los objetos o recursos protegidos por el modelo de autorización.

El modelo se basa en tecnologías de la Web semántica [2], con lo que se consigue una alta expresividad para la definición de las políticas con el fin de reducir las diferencias entre la política abstracta definida y lo que el administrador puede especificar en el modelo. El formalismo lógico proporcionado por las ontologías y lenguajes de representación con semántica permite describir tanto la infraestructura Cloud subyacente como el propio modelo de autorización y las reglas empleadas para proteger el acceso a los recursos en el Cloud. La aplicación de estas tecnologías y lenguajes proporciona a la solución capacidades de razonamiento, tanto sobre el dominio como sobre las reglas del modelo de autorización. Esto habilita la aplicación de técnicas avanzadas para la gestión de políticas tales como la detección y resolución de conflictos en las reglas de autorización. Esta detección no se limita únicamente a conflictos modales basados en inconsistencias en la propia definición de reglas de autorización, sino que permiten ir un paso más allá y detectar conflictos semánticos que son más complejos de detectar ya que son dependientes del dominio.

También se proporciona una aproximación *data-centric* en la que la seguridad se centra en proteger los datos del usuario independientemente del proveedor de servicios Cloud que los tenga. Para ello se utilizan novedosas técnicas criptográficas basadas en identidad (*identity-based encryption*) y re-encryptación proxy (*proxy re-encryption*) que se utilizan para proteger tanto los datos como el propio modelo de autorización. Cada uno de los datos se cifra con su propia clave de cifrado que se enlaza con el modelo de autorización y las reglas de autorización son criptográficamente protegidas para preservar los datos del usuario de un posible acceso por parte del proveedor de servicios Cloud, así como un mal comportamiento del mismo a la hora de aplicar las reglas de autorización definidas por el usuario propietario de los datos. Se da soporte también para una aproximación centrada en el usuario (*user-centric*) respecto a las reglas de autorización, con la que el usuario puede definir una política de control de acceso unificada para todos sus datos. Este modelo de

protección de datos propuesto considera, además, un mecanismo seguro de distribución de claves y soporte para compatibilidad con infraestructura de clave pública (PKI), que permite poder utilizar certificados y claves X.509 estándar. En definitiva, esta propuesta *data-centric* proporciona una aproximación de autorización en Cloud basada en reglas, en la que las reglas están en todo momento bajo el control del usuario propietario de los datos, mientras que la evaluación del control de acceso en base a esas reglas es delegado al CSP de manera protegida, resultándole imposible acceder a los datos o cederlos a terceras partes no autorizadas.

Metodología y estructura

Para el desarrollo del trabajo de investigación llevado a cabo durante la realización de esta tesis doctoral se ha seguido una aproximación en la que se ha comenzado por proporcionar una primera propuesta del modelo de autorización en entornos distribuidos, que luego ha sido refinado, formalizado y contextualizado en entornos Cloud, y finalmente protegido criptográficamente, resultando en un modelo auto-protegido para la seguridad de recursos en el Cloud. Así, un primer trabajo proporciona una aproximación inicial a la aplicación de tecnologías de Web semántica para gestionar políticas de seguridad en entornos distribuidos que presentan características *multi-tenancy* como es la computación Grid, aportando una primera arquitectura y un primer modelo de autorización con capacidades de razonamiento avanzadas que permiten tareas de valor añadido tales como detección y resolución de conflictos entre políticas. En una segunda fase, se refina y formaliza el modelo de autorización, proporcionándole características de expresividad avanzadas basadas en RBAC, y se particulariza la arquitectura y el modelo para adaptarlo a las características específicas del Cloud, proporcionando también un modelo de confianza para dar soporte a posibles federaciones entre usuarios. Finalmente, se realiza una formalización matemática del modelo de autorización y se protege criptográficamente para conseguir una aproximación *data-centric* para que los datos queden auto-protegidos y puedan ser gestionados de manera segura y confiable por cualquier CSP sin riesgo a que se pueda acceder a ellos o cederlos a usuarios no autorizados. Cada una de estas tres fases corresponde a un trabajo de investigación completo en el que se incluye también una prueba de concepto y una serie de pruebas con resultados y estadísticas de rendimiento, con el objetivo de analizar la viabilidad de la propuesta en cada una de sus fases.

El presente documento de tesis se estructura en seis capítulos, de los cuales los capítulos tres a cinco componen el núcleo del trabajo donde desarrolla y expone la principal contribución de la investigación realizada. El capítulo 1 aporta una introducción con los conceptos necesarios para entender el trabajo realizado y proporciona una visión general de la problemática que contextualiza y motiva el desarrollo de esta tesis. El capítulo 2 presenta el actual estado del arte, realizando un análisis de otros trabajos e investigaciones relacionadas con los aspectos tratados en esta investigación. El capítulo 3 presenta el primer trabajo de investigación con la aproximación a la aplicación de tecnologías de

Web semántica para gestionar políticas de seguridad en sistemas distribuidos. El capítulo 4 expone el modelo refinado, formalizado y adaptado a las particularidades del Cloud con características de expresividad RBAC avanzadas, y la arquitectura de seguridad para entornos Cloud. El capítulo 5 proporciona la aproximación *data-centric* de autorización, donde se formaliza matemáticamente el modelo y se protege criptográficamente. Finalmente, el capítulo 6 sirve de conclusión a este documento y presenta una recopilación de los resultados conseguidos, así como una serie de vías futuras para la continuidad y mejora de la investigación realizada en esta tesis.

Durante la elaboración de esta tesis doctoral, el trabajo de investigación desarrollado se ha divulgado en una serie de publicaciones científicas internacionales. Concretamente, 8 artículos en revistas internacionales, de las cuales 7 están clasificadas en los primeros cuartiles del índice *Journal Citation Report* (JCR), 7 conferencias internacionales y 2 capítulos de libro. Cabe destacar también que cada uno de los tres capítulos principales que componen el núcleo de esta tesis doctoral (capítulos 3 a 5) corresponden directamente con tres artículos que han sido publicados o están siendo revisados en revistas internacionales con JCR Q1. En la sección 1.5 de este documento se proporciona una lista detallada de estas publicaciones, clasificadas en sus distintos tipos.

Abstract

Cloud computing is an emerging paradigm that allows to offer on demand IT services to customers which require provisioning of such services. Many businesses are currently adapting their information systems towards the Cloud computing paradigm where flexible and dynamic services and infrastructures are able to scale and be delivered on demand. These systems make use of virtualization and abstraction layers in order to provide a set of virtualized infrastructure, resources and services. Three main layers are usually considered in these environments, according to the kind of service that is provided. Namely, Infrastructure as a Service (IaaS), which provides virtual infrastructure resources such as virtual machines, storage volumes or networking; Platform as a Service (PaaS), which provides middleware services such as programming platforms and application servers; and Software as a Service (SaaS), which exposes software features to be used by end-users. This model enables an efficient provisioning of virtual IT architectures to third-parties, where resources are dynamically created and dismantled according to customer needs.

However, many potential businesses which would be interested in Cloud computing are still a bit reluctant to adopt this technology due to security and privacy concerns. Most current security solutions are based on what is called perimeter security, which mainly consists on deploying security mechanisms in the organization boundaries to protect the resources that reside within the organization. However, Cloud computing breaks the organization perimeters. When data resides in the Cloud, they reside outside the organizational bounds. Concretely, they are managed by the Cloud Service Provider (CSP), whose infrastructure and systems are not directly administered by the organization. This leads users to a lost of control over their data and raises reasonable security concerns that slow down the adoption of Cloud computing. Thus, the system to control the access to the available resources becomes a critical aspect in order to provide an efficient, reliable and secure control over the usage and access to Cloud resources.

The management of access control and security could become a difficult and error prone task in distributed systems like Cloud computing infrastructures. Policy-based solutions can help to control and manage security and to deal with this complexity. They can aid administrators with this task by enabling the specification of high-level security policies that are automatically enforced into the target system for this to behave as defined by the administrator. However, current authorization solutions for Cloud computing

usually lack of enough expressiveness to describe advanced authorization and federation rules that enable a secure and trustworthy access control over resources. The availability of advanced authorization capabilities can be a differentiating feature for Cloud providers, which demand the design of suitable authorization models to enhance the access control to the Cloud resources.

Moreover, moving data to the Cloud usually implies relying on the CSP for data protection, which also raises security concerns and questions in potential and current users. Is the Cloud service provider accessing the data for its own benefit? Is it legitimately applying the access control policy defined by the user? Although this is usually managed based on legal or Service Level Agreements (SLA), the CSP could potentially access the data or even provide it to third parties. Moreover, one should trust the CSP to legitimately apply the access control rules defined by the data owner for other users.

The problem becomes even more complex in scenarios where alliances and federations between different Cloud Service Providers arise. Thus, one service provider could make use of the services or virtual infrastructure offered by others in order to provide their own services to its customers. This concept of interconnected Clouds or Inter-Cloud increases the challenges on security and privacy since data can be managed by multiple CSPs. Users may lose control on their data in the Inter-cloud and even the trust on the federated CSPs is outside the control of the data owner. This situation leads to rethink about data security approaches and to start considering novel data-centric approaches where data are self-protected with cryptographic security mechanisms wherever they reside. Encryption of data is the most widely used method to protect data, providing confidentiality and protection against undesired accesses when the data is in the Cloud. Current research works in this field try to provide different approaches towards encrypted data with authorization capabilities that enables data to be self-protected in the Cloud, technically assuring that it cannot be accessed by the CSP or released to any third party not authorized by the data owner.

Contribution

This PhD thesis targets access control in distributed environments with special focus on its application to Cloud computing systems. As main result of this work, a highly expressive, reliable and secure access control approach suitable for Cloud computing is provided. It has been designed taking into account special characteristics of these environments and enables the management of security by providing advanced authorization features.

The main contributions of the proposed solution can be summarized as follows:

- A policy-based approach for access control in distributed environments that enables administrators to keep Cloud systems under control in a secure, intuitive and scalable manner.

- An advanced authorization model with high expressiveness for access control based on commonly used and well known semantics for access control rules following the Role Based Access Control (RBAC) scheme.
- The application of semantic Web technologies to the field of access control in distributed systems in order to enable reasoning capabilities about policy and domain descriptions.
- A novel data-centric RBAC approach in which data is encrypted and self-protected with the application of a rule-based approach with extended RBAC expressiveness.

An advanced and sophisticated access control model is defined to allow the specification and validation of high-level policies, providing high expressiveness in the language for policy definition and taking into account complex domain information. This makes the security management tasks easier, facilitating the policy definition and deployment processes to administrators and coping with integration issues caused by the domain heterogeneity of Cloud computing. Domain elements are defined by means of standard representations like the Common Information Model (CIM) [1] developed by the Distributed Management Task Force (DMTF), achieving a high interoperability in the representation that enables to tackle the heterogeneity of the different domains of the Cloud. Together with the authorization model, a security architecture is also provided for its application in Cloud computing environments. Both the model and the architecture have been specially designed taking into account the *multi-tenancy* nature of this kind of environments, where multiple users share the same infrastructure. Concretely, the CSP infrastructure that is used to offer its services to multiple customers. The solution provides multi-tenancy support, including federation capabilities to support situations in which different users share Cloud resources, allowing a fine-grained definition of what resources are available for each particular tenant.

Apart from the role-based authorization rules provided by the RBAC schema, the proposed authorization model supports the following expressiveness features:

- Hierarchical RBAC (hRBAC). Allows to establish hierarchies between the roles of the authorization model.
- Conditional RBAC (cRBAC). Allows to establish contextual or environmental conditions to the authorization rules.
- Hierarchical Objects (HO). Allows to establish hierarchies between the objects or resources protected by the authorization model.

The model relies on Semantic Web [2] technologies, achieving a high expressiveness for policy definitions to reduce the gap between the abstract policy and what the administrator can specify. The logic formalism provided by ontologies and languages with semantic representation allows to describe both the underlying Cloud infrastructure and

the authorization model, as well as the rules employed to protect the access to resources in the Cloud. The application of these technologies and languages endows the solution with reasoning capabilities about both the domain and rules of the authorization model. This enables advanced policy management techniques such as policy rule conflict detection and resolution. This detection is not only limited to modal conflicts, based on inconsistencies in the rule definition itself, but also allows to detect semantic conflicts that are domain dependent and more complex to detect.

A data-centric approach is also provided for reliable data protection in the cloud, in which security is focused on protecting user data regardless the Cloud service provider that holds it. Novel identity-based and proxy re-encryption techniques are used to protect both the data and the authorization model itself. Each piece of data is encrypted with its own encryption key linked to the authorization model and authorization rules are cryptographically protected to preserve user data against the service provider access or misbehavior when applying the authorization rules defined by the data owner. The solution also combines a user-centric approach for authorization rules, where the data owner can define a unified access control policy for his data. A secure key distribution mechanism is also considered within the solution and PKI compatibility is supported for using standard X.509 certificates and keys. The solution enables a rule-based approach for authorization in Cloud systems where rules are under control of the data owner and access control computation is delegated to the CSP, but making it unable to grant access to unauthorized parties.

Methodology and Structure

During the research work carried out within this PhD thesis, the followed approach started by providing an initial proposal of the authorization model for distributed environments that has been later refined, formalized and contextualized for Cloud environments. Then, it has been cryptographically protected, resulting in a self-protected model for Cloud resource security. Thus, a first work provides an initial approach to the application of semantic Web technologies to manage security policies in distributed systems with multi-tenancy features such as Grid environments. It provides an initial architecture and an initial authorization model with advanced reasoning capabilities that enable performing added value tasks such as policy conflict detection and resolution. In second step, the authorization model is refined and formalized, endowing it with enhanced expressiveness features based on the RBAC scheme. The architecture is particularized and the model adapted to the specific characteristics of the Cloud, including a trust model to support possible federations between tenants. Finally, the authorization model has been mathematically formalized and cryptographically protected to achieve a data-centric approach in which data are self-protected and can be securely managed by any CSP, being it unable to access data or to release it to unauthorized parties. Each one of these three stages correspond to a complete research work in which a proof of concept implementation is also

included, together with a set of test results and performance statistics in order to analyze the viability of the proposal in each stage.

The present PhD thesis document is structured in six chapters. Among them, chapters three to five conform the core of the work, where the main contribution of the performed research is exposed. Chapter 1 presents an introduction with the needed concepts to understand this work done and provides an overview of the main problems that contextualize and motivate this PhD thesis. Chapter 2 introduces current state of the art, analyzing other works and researches related to the main aspects tackled in this research. Chapter 3 presents the initial research work to the application of semantic Web technologies for security policy management in distributed environments. Chapter 4 exposes the model refined, formalized and adapted to the particularities of Cloud with advanced RBAC expressiveness, together with the security architecture for Cloud computing. Chapter 5 provides the data-centric authorization approach, where the model is mathematically formalized and cryptographically protected. Finally, Chapter 6 serves as conclusion to this document and presents a summary of the main achieved results, as well as a set of future works and research guidelines to continue and improve the research carried out during this PhD thesis.

During the realization of this PhD thesis, the research results have been published in international scientific publications. Concretely, 8 international journal papers, being 7 of them classified in the top four quarters of the Journal Citation Report (JCR), 7 international conferences and 2 book chapters. It should be also noticed that each one of the three main chapters that conform the core of this PhD thesis (chapters 3 to 5) directly correspond to three papers that have been published or are being reviewed in JCR Q1 international journals. Section 1.5 of this document provides a detailed list of these publications, classified according to their category.

Contents

1	Introduction	1
1.1	Cloud computing overview	1
1.2	Security and privacy in Cloud computing	3
1.3	Policy-based management of security	5
1.4	Data-centric approach to security	6
1.5	Contributions	8
1.6	Related publications	10
1.6.1	International Journals	10
1.6.2	International Conferences	12
1.6.3	Book Chapters	13
1.7	Structure	13
2	Related work	17
2.1	Access control in distributed systems	17
2.2	Access control models for Cloud computing	19
2.3	Policy specification approaches	21
2.4	Data centric access control	24
3	Semantic-based Authorization in multi-tenancy environments	27
3.1	Authorization policies and domain representations	27
3.1.1	Semantic Web representation for authorization policies	28
3.1.2	The Common Information Model	30
3.2	Authorization architecture for Grid services	33
3.2.1	Managing authorization information	35
3.2.2	Access request processing	36
3.3	Authorization policy management	37
3.3.1	Virtual Organization Management	41
3.3.2	The reasoning process	44
3.3.3	Policy conflict analysis	47
3.4	Implementation and statistics	49
3.4.1	Reasoner performance	50
3.4.2	Authorization request testbed description	53
3.4.3	Authorization request performance	55

4	Semantic-aware Authorization System for Cloud Architectures	59
4.1	Information model	60
4.2	Authorization Model	61
4.2.1	Authorization Models comparison	66
4.3	Architecture	68
4.4	Trust Management	70
4.5	Authorization Process	71
4.6	Implementation and Performance	73
4.6.1	Testbed description	74
4.6.2	Obtained results	77
5	SecRBAC: Secure data in the Clouds	83
5.1	Proxy Re-Encryption and Identity-Based Encryption	83
5.2	Authorization model formalization	86
5.3	Self-protected authorization model for data-centric security	89
5.3.1	Protecting the authorization model	89
5.3.2	Representation and evaluation	95
5.4	Data-centric solution for Cloud data protection	98
5.4.1	Key management	101
5.4.2	Security considerations	102
5.5	Implementation and performance	103
6	Conclusion and future work	111
6.1	Conclusions and main achievements	111
6.2	Future work and research guidelines	114
	Bibliography	117

List of Figures

1.1	Cloud stack layers	2
2.1	XACML 3.0 Policy Language Model	22
2.2	CIM Policy schema (reduced)	23
2.3	Ponder policy	24
3.1	CIM modeling levels	31
3.2	CIM Metaschema (reduced)	32
3.3	Authorization architecture	34
3.4	Organization WS-Resource lifecycle	36
3.5	Access request processing	37
3.6	UML diagram of concepts for privilege representation	38
3.7	Multiple authorization domains	42
3.8	Ontology and rule reasoning	46
3.9	Individuals distribution for populations	51
3.10	Reasoner performance	52
3.11	Testbed times description	53
3.12	Authorization request performance with continuous changes	56
3.13	Authorization performance with 10 requests before changes	56
3.14	Authorization performance with 100 requests before changes	57
3.15	Authorization request performance in static scenario	58
4.1	UML representation of the information model	61
4.2	Cloud authorization example	62
4.3	Authorization Models Comparison	68
4.4	Semantic Authorization Architecture for cloud.	69
4.5	Operation execution workflow	75
4.6	Describe instances overhead times without instances	77
4.7	Describe instances overhead times with 4 instances/node	78
4.8	Run instances overhead times	78
4.9	Terminate instances overhead times	79
4.10	Command overhead times	80
4.11	Operation overhead times	80

List of Figures

5.1	Ontology representing the authorization model	95
5.2	Architecture for deployment in a CSP	99
5.3	Times changing the number of re-encryptions I	105
5.4	Times changing the number of re-encryptions II	106
5.5	Times changing the length of identities I	106
5.6	Times changing the length of identities II	107
5.7	Times changing the length of encrypted data I	107
5.8	Times changing the length of encrypted data II	108
5.9	Authorization decision times	109

Chapter 1

Introduction

This chapter introduces the background needed to better understand the aim of this research and the work carried out during the realization of this PhD thesis. A short introduction to the research areas motivating the realization of this work and current challenges are described and analyzed.

This chapter also provides a description of the main contributions of the research work realized during this PhD thesis. A brief introduction to the scientific publications and contributions resulting from this PhD thesis is also presented. Finally, the structure of this document is depicted, providing an introduction and brief description to the content of the following chapters.

1.1 Cloud computing overview

Cloud computing is an important emerging technology which presents a vision of flexible and dynamic services, running on virtual IT infrastructure that scales on demand according to customer needs. The National Institute of Standards and Technology (NIST) provides the following official definition of this computing paradigm:

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.

— The NIST Definition of Cloud Computing [3]

This model enables Cloud Service Providers (CSP) to supply a set of computing resources as services, offering a “pay as you go” model for the usage of those resources. Customers are provided with a computing capacity adapted to their needs since resources

can be provisioned and released on-demand by the CSP. This is usually achieved thanks to virtualization and abstraction layers in the Cloud architecture.

Cloud computing describes a logical stack divided into three main different layers: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). Figure 1.1 presents an overview of these layers.



Figure 1.1: Cloud stack layers

The IaaS layer leverages virtual infrastructure resources such as virtual machines, storage volumes, network components and other fundamental computing resources. This layer represents an abstraction of the underlying physical hardware resources used to provide those functionalities. It provides the most extensible delivery model of Cloud resources. Customers using this layer are able to deploy and manage their own operating systems and applications. The services offered by this layer can be directly accessed by customers that make use of the virtual infrastructure or they can serve as basis for upper layers of the Cloud.

The PaaS layer provides middleware services that serve as deployment platform for customer applications. It offers a development environment and a set of software components that can be used by customer applications. This layer represents an abstraction of the underlying infrastructure, platform software and application resources. At this level, customers do not manage the infrastructure resources or operating systems that are used to run their applications, but they can control their applications and the configuration of the offered platform and development environment. These applications are constrained by the functionalities provided by this layer such as programming language, services of the operating system that can be used or restricted access to the filesystem.

The SaaS layer exposes application software which is used by customers, usually accessible through thin clients such as web browsers or lightweight applications for different devices. This layer represents a complete abstraction of the underlying computing architecture. At this level, customers do not develop applications and do not have control over

the infrastructure resources or the software components that are used by the applications. They can control some configuration and customization parameters of the application they are using as end users.

There are three major deployment models for Cloud computing architectures:

- *Private Cloud.* A deployment in which all the resources that are used to build and manage the Cloud are deployed for a single user organization. It is usually deployed and self-managed by the same organization on its own premises, although it could be administered by a third party or located off-premise.
- *Public Cloud.* A deployment made by a Cloud Service Provider that publicly offers its services to different customers. It is owned and managed by the CSP, usually on its own premises, and customers make use of the offered services according to their needs.
- *Hybrid Cloud.* A deployment in which some Cloud resources and services are deployed and managed following a private model and others are managed following a public model. This usually happens when an organization operates a private Cloud for some critical or specific services, while it also uses the services offered by a public CSP for other services.

Additionally, there are also Cloud architectures deployed by a group of organizations which share common goals or concerns and form a community to deploy and manage a Cloud system according to their common needs. This deployment model is also known as *Community Cloud*.

A key property that characterizes Cloud computing environments is what is called *multi-tenancy*. In essence, multi-tenancy is the ability to efficiently deal with multiple administrative domains (tenants) that are using the same service which, in turn, has isolated resources belonging to particular tenants. Many distributed systems like Grid and Cloud computing demand multi-tenancy support. This makes the management of this kind of environments more complex in order to provide isolated resources to different tenants by sharing the same architecture. Security and privacy also become critical issues when multiple tenants make use of shared services and facilities as is the case of Cloud computing systems.

1.2 Security and privacy in Cloud computing

The rising of Cloud computing supposes an innovation on IT systems deployment and management. However, its wide adoption by organizations for their business processes is slowed down by security and privacy concerns. Cloud computing solutions must provide a high level of flexibility to enable services running in the Cloud to scale up or down

dynamically in response to changing workloads. Enterprise customers demand Cloud services with a high level of security and protection. Service data and sensitive information must be protected at the same time that performance and quality of service are ensured. Therefore, Cloud solutions must be flexible, scalable, secure and resilient to failure to be considered enterprise-grade. However, Cloud computing makes use of a multi-tenanted infrastructure, where services offered to different customers share the same common IT resources. Due to this multi-tenancy nature, Cloud computing implies the design of strong security mechanisms in order to isolate and protect tenants when using Cloud services.

Current studies and research works are analyzing and providing diverse solutions to advance Cloud technology with regard to security and privacy. As exposed in [4], security in Cloud computing is a hot research area. This study analyzes different research works both in the industria and academia related to security and privacy in Cloud computing. A huge number of research is done in this area in the last years and according to authors this indicates an interest in solving security risks in Clouds before exploring their wide area of potential applications. Addressing these issues would increase the adoption of this technology and open new fields of research in this new paradigm of computing.

Moreover, security in Cloud computing is also included as a principal topic in current EU research activities and challenges in strategic programs like the EU Horizon 2020 Programme [5] and as a requirement for the adoption of this technology in institutional organizations as reflected in the NIST requirements for the adoption of Cloud computing in US Government institutions [6].

The European Union is fostering research on Cloud computing in several fields, including security and privacy issues. In its Horizon 2020 Work Programme for 2014-2015 on Information and Communication Technologies [5], the European Commission includes a research topic specifically defined to promote research on Cloud computing. Concretely, the *ICT 7 - Advanced Cloud Infrastructures and Services* topic includes the following Research & Innovation action: “Mechanisms, tools and techniques to increase trust, security and transparency of cloud infrastructures and services, including data integrity, localisation and confidentiality, also when using third party cloud resources”. The security in Cloud computing topic is also being continued in the Horizon 2020 Work Programme planned for 2015-2016 on Information and Communication Technologies [7]. In this case, the research topic *ICT-06-2016 - Cloud Computing* includes the following Research & Innovation action: “Techniques to deal with trust, security and privacy in decentralised cloud infrastructures and across multiple cloud providers, including aspects of data integrity, data localisation and data confidentiality”. This goes a step forward in Cloud computing security and raises the concern on security within federated and distributed Cloud scenarios, where data can be managed or accessed by multiple Cloud providers.

On another hand, the US Government Cloud Computing Technology Roadmap [6] developed by NIST also underlines the importance of security and privacy in Cloud computing systems. On its volume I related to High-Priority Requirements to Further USG Agency Cloud Computing Adoption, security of Cloud systems is depicted as one of

the main recommendations to take into consideration for the adoption of Cloud services. Concretely, Requirement 2 is entitled “Solutions for High-priority Security Requirements which are technically de-coupled from organizational policy decisions”. This requirement raises the issue that Cloud computing breaks the organization boundaries and traditional IT security mechanisms become less effective in providing protection in Cloud scenarios. It points out the need to take into consideration special characteristics of Cloud environments for security and privacy mechanisms such as multi-tenancy, which is stated as “an inherent cloud characteristic that intuitively raises concern that one consumer may impact the operations or access data of other tenants running on the same cloud”.

1.3 Policy-based management of security

Cloud systems are incredibly growing nowadays becoming more and more complex to be administrated. Managing huge systems is a very complex problem and the management of access control and security includes multiple aspects regarding authentication, authorization and communication protection, as well as managing user credentials and group membership information, among others. The configuration of these security aspects may become a complex and error-prone task, i.e., the administrator has to deploy a security policy by means of the proper configuration of security components. In most cases, an administrator receives a high-level security policy that must be previously refined before achieving the final security configuration that should be applied to the target elements. A security policy that could appear with a simple textual form like “if a service permits to perform query operations to user A and user B is located in the same laboratory as user A, then the service must also permit to perform query operations to user B” may imply a complex task for the administrator. Moreover, the administrator is limited to the parameters supported by the security model that could make it impossible to deploy that policy. This complexity is incremented in multi-tenancy environments like Cloud computing or Grid scenarios, where multiple organizations share the same infrastructure and resources. This increases the complexity of management and security administration due to the domain heterogeneity between the organizations which share resources on the system and which usually have their own access policy for their resources. For instance, in the previous example users A and B may belong to different organizations.

Policy-based solutions can help to control and manage security in distributed systems. They enable the specification of high level business policies which are refined into low-level policies suitable to be directly applied into final devices to achieve the high-level behaviour previously specified. Security policies can be described by administrators using a language designed for this purpose which enables the description of high level security policies. Researchers have proposed multiple approaches for policy specification that range from formal policy languages that a computer can directly process and interpret, to rule-based policy notation using an if-then-else format. The definition and the usage of high level policy languages ease the process of policy specification reducing significantly

errors therein. In fact, high-level policies require less effort to be written and maintained since a lot of details are hidden to writers. Note that this kind of policies does not need training or deep knowledge to be used by administrators. This is especially suitable in security field in which any specification error potentially may cause a security hole in the information system. In the policy definition process, the task of a policy manager is to transform the business policies into implementable policies using a formal language for this purpose. To do so, the manager uses a high level policy language that assures that the representation of security policies will be unambiguous and verifiable.

Access control solutions provide different methods to enable administrators to apply authorization policies. However, sometimes there is a gap between the policy the administrator has in mind and the policy he can describe using the expressiveness provided by the solution. In this situation, the administrator is usually able to refine the original policy, but resulting in complex policy definitions with lots of rules or, in the worst case, he may even be unable to enforce the policy in the system. XACML [8] is currently one of the most representative standards for access control in distributed systems. It provides a core schema and corresponding namespace for the expression of access control policies. However, the problem of using mere XML as a standard for policy expression is that its semantics are mostly *implicit*. Thereby, meaning is conveyed based on a shared understanding derived from human consensus. At the same time, this and other non-semantic solutions are rife with ambiguity, promote fragmentation into incompatible representation variations, and require extra manual work that could be saved by a richer representation.

Furthermore, these approaches are focused on describing security policies and they are not able to cope with the semantics related to the elements they are protecting. For example, a simple policy rule like “Only administrators can read files of the `/root` folder” entails including the semantics related to the structure of folders and files in order to correctly manage the protection of these resources. This capability of expressing domain semantics is rarely supported by current solutions. Moreover, these approaches do not properly deal with policy conflicts, providing detection capabilities to avoid undesirable or even security risky situations which may arise as a result of errors in complex policy definitions.

On the other hand, semantic approaches for policy representation enable runtime extensibility and adaptability of the system as well as the ability to analyze policies related to entities described at different levels of abstraction. The representation facilitates careful reasoning about policy disclosure, conflict detection, and harmonization about domain structure and concepts.

1.4 Data-centric approach to security

As stated before, security is one of the main user concerns for the adoption of Cloud computing. Although there exists different security solutions for data protection, they

are usually based on perimetral security. That is, they are focused on protecting data within the bounds of the organization that owns and manages the data. However, when data is moved to the Cloud, it no more resides within the organizational boundaries and it is totally under control of the Cloud Service Provider. This implies that data owners should rely on the CSP for data protection. Moreover, privacy is also an important aspect that slow down the adoption of Cloud solutions. Since data is actually moved to Cloud servers, the CSP could potentially access this data to obtain information and knowledge about the data owner, its organization or, even worse, its business sensible information. This problem becomes even more complex in federated or distributed Cloud scenarios, which is a current trend in Cloud technology as can be observed in the ICT-06-2016 EU Horizon 2020 topic as exposed in Section 1.2. In these scenarios data moved to the Cloud could potentially be managed or accessed by different CSPs and the data owner can loss the control over the real access to its data, raising security, privacy and confidentiality concerns.

Although these issues are usually managed based on legal or Service Level Agreements (SLA), there is no real technological enforcement of such agreements that actually prevent the CSP to access the data or even to provide it to third parties. Thus, there is a raising concern about loss of control over the data when it is moved to the Cloud. To solve these issues, the concept of data-centric security has raised, trying to find suitable and reliable mechanisms able to generate self-protected pieces of data which can be uploaded to the Cloud without risk of being disclosed by unauthorized parties.

Encryption is the most widely used method to protect data in the Cloud. In fact, the Cloud Security Alliance security guidance recommends data to be protected at rest, in motion and in use [9]. Encrypting data avoids undesired accesses. However, it also entails new issues related to access control management. Following a straightforward approach, one can encrypt data for the intended users. This is usually done when sending a file or document to a specific receiver and ensures that only the receiver with the appropriate key is able to decrypt it. From an authorization point of view, this can be seen as a simple rule where only the user with privilege to access the data will be able to decrypt it (i.e. the one owning the key). However, no access control expressiveness is provided by this approach. Only that simple rule can be enforced and just one single rule can apply to each piece of data. Thus, multiple encrypted copies of the data should be created in order to deliver the same data to different receivers. A possible quick solution for it could be to encrypt the data and deliver the same key for different users, but this would move the problem to a key management issue.

A rule-based approach would be desirable to provide expressiveness and achieve a feasible access control mechanism. But this supposes a big challenge for a data-centric approach since data has no computation capabilities by itself. It is not able to enforce or compute any access control rule or policy. This raises the issue of policy decision for a self-protected data package: who should evaluate the rules upon an access request? The first choice would be to have them evaluated by the CSP, but it could potentially apply any rule (even those not defined by the data owner) and get access to the data or provide

access to third parties. Another option would be to have rules evaluated by the data owner itself. This would reduce the CSP role to a mere storage service and either data could not be shared or the owner should be online in order to take a decision each time a user requests access to the data.

To overcome the aforementioned issues, several proposals [10] [11] [12] try to provide data-centric solutions based on novel cryptography mechanisms applying Attribute-based Encryption (ABE) [13]. These solutions are based on the Attribute Access Control model (ABAC), in which privileges are granted to users according to a set of attributes. There is a long standing debate in the IT community about whether Role-based Access Control (RBAC) or ABAC is a better model for authorization management [14] [15]. Without entering into this debate, both approaches have their own pros and cons and the availability of a self-protected data mechanism based RBAC will allow Cloud data owners to select one or another model according to their particular needs.

1.5 Contributions

Security in Cloud computing is a current major challenge and research field that needs more efforts to achieve a fully secure and trustworthy Cloud environment which encourages users and organizations to adopt this new paradigm and shift their business processes to the Cloud in a secure, trusted and reliable manner.

The research work carried out during this PhD explores the field of security in distributed systems with special focus on Cloud computing security and particular attention to managing and controlling the access to services and resources. The main result of this work is an advanced access control solution specially designed for distributed environments like Cloud computing that takes into account the particularities of these environments like their multi-tenancy nature, the heterogeneity of domains and the loss of control over the resources released to the Cloud from data owners point of view.

The main contributions of the proposed solution can be summarized as follows:

- A **policy-based approach for access control in distributed environments** that enables administrators to keep Cloud systems under control in a secure, intuitive and scalable manner. Policies are specified by means of rules based on domain concepts, which allow the representation of the resources to be protected and their semantics, easing the authorization management task. The solution also proposes an **authorization architecture** suitable for Cloud computing specifically designed to tackle the multi-tenancy and heterogeneity properties of this kind of environments. The architecture provides the components needed to support the authorization model also proposed in this PhD to manage and control resource sharing in order to establish who is allowed to perform what actions. The solution provides **multi-tenancy** support and federation capabilities allowing a fine-grained definition of what resources

are available for each particular tenant. Advanced federation capabilities among tenants have been successfully supported by means of a fine-grained trust model that determines the business alliances (coalitions and federation) among cloud tenants. A high level of **interoperability** to deal with the heterogeneity of Cloud domains is also provided thanks to the use of domain representation standards like the Common Information Model (CIM) and the ontological representation provided by the Semantic Web, which eases the interpretation and integration of the information. This enables advanced features like concept alignment between heterogeneous domains, allowing the combination of concepts from the different domains of the organizations which are part of the Cloud.

- An advanced **authorization model with high expressiveness** for access control based on commonly used and well known semantics for access control rules following the **Role Based Access Control (RBAC)** scheme. This approach is close to current access control methods used by most users and organizations, resulting more natural to apply for access control enforcement and easing the adoption by users moving their resources to the Cloud. The proposed authorization model improves RBAC expressiveness with the following enhanced features:
 - *Hierarchical RBAC (hRBAC)* extends RBAC with the aim of defining role hierarchies in order to simplify role management. These hierarchies establish privilege inheritance between roles, making a child role to inherit all the privileges defined for parent roles in the hierarchy.
 - *Conditional RBAC (cRBAC)* or *Generalized RBAC* is an advanced capability rarely supported by current Cloud access control systems. This feature supports the assignment of privileges to environment roles. These environment roles are activated according to the changes specified in environmental conditions defined by the administrators.
 - *Hierarchical objects (HO)* allows the privileges applied to a given object to be also applied to all its children objects in the hierarchy. This enables to achieve a high-level management of the access control system.
- The **application of semantic Web technologies to the field of access control** in distributed systems in order to enable **reasoning capabilities** about policy and domain descriptions. Both authorization rules and domain representations are specified in the form of an ontology, allowing reasoning processes to check constraints and query the information. This also allows the application of high-level and highly expressive meta-rules by means of Semantic Web Rule Language (SWRL) [16] that makes it possible to derive new knowledge and infer information. This mechanism provides the capability to describe the semantics of the resources which are to be protected, as well as extending these definitions to fit particular domain requirements. A policy execution engine allows for automated reasoning that supports advanced capabilities for **policy conflict analysis**. The proposal also represents one

step towards the automatic management of security services, considering not only authorization services, but also providing additional reasoning mechanisms to cope with issues such as detection and resolution of conflicts between different policies.

- A novel **data-centric RBAC approach** in which data is encrypted and self-protected with the application of a rule-based approach with extended RBAC expressiveness. This supposes a novel solution for data-centric security based on the RBAC model, offering an alternative to current ABE schemes based on the ABAC model and whose expressiveness is usually constrained by the expressiveness of ABE. The solution supports an RBAC scheme with role hierarchy and resource hierarchy that would be also closer to current access control methods, resulting more natural to apply for access control enforcement. The use of a **cryptographically secured access control** scheme enables access control rules to be computed in untrusted CSPs in a secure and reliable manner. Authorization rules defined by the data owner are protected by a cryptographic token and each piece of data is ciphered with its own encryption key linked to the authorization model. Novel encryption techniques like Proxy Re-encryption and Identity-Based Encryption schemes are applied. This enables access control computation to be delegated to the CSP, but making this unable to gain access to the protected data and even to grant access to unauthorized parties. The solution allows **separation of authentication** and authorization for self-protected data. Users only have to possess a standard credential or proof of identity, which is not tight to user status and does not need to be changed if policies or any attribute of the user changes over time. This significantly differs from current data-centric solutions based on ABE, in which user credentials need to be specifically generated for the purposes of the authorization model, containing user attributes and making them to be reissued if user attributes change. The solution supports a secure key distribution mechanism with **PKI compatibility** for using standard X.509 certificates and keys to access self-protected resources that are moved to the Cloud applying the proposed data-centric approach.

1.6 Related publications

The research work carried out along the realization of this PhD has been published -or is under revision- in several peer-reviewed international journals, conferences and book chapters. This section summarizes the most relevant scientific publications that have been published during this PhD.

1.6.1 International Journals

- [17] Juan M. Marín Pérez, Gregorio Martínez Pérez and Antonio F. Gómez Skarmeta. **SecRBAC: Secure data in the Clouds**. *IEEE Transactions on Services*

Computing, 2015 (Under revision). Special Issue on Security and Dependability of Cloud Systems and Services. (JCR Q1).

- [18] Jorge Bernal Bernabé, Juan M. Marín Pérez, José M. Alcaraz Calero, Félix J. García Clemente, Gregorio Martínez Pérez and Antonio F. Gómez Skarmeta. **Semantic-aware multi-tenancy authorization system for Cloud architectures.** *Future Generation Computer Systems*, 32: 154-167, 2014. DOI: 10.1016/j.future.2012.05.011. Special Issue on Exploiting Semantic Technologies with Particularization on Linked Data over Grid and Cloud Architectures. (JCR Q1).
- [19] Juan M. Marín Pérez, Jorge Bernal Bernabé, José M. Alcaraz Calero, Félix J. García Clemente, Gregorio Martínez Pérez and Antonio F. Gómez Skarmeta. **Taxonomy of trust relationships in authorization domains for Cloud computing.** *The Journal of Supercomputing*, 70 (3): 1075-1099, 2014. DOI: 10.1007/s11227-014-1117-x. (JCR Q2).
- [20] María A. Martínez Carreras, Juan M. Marín Pérez, Jorge Bernal Bernabé, José M. Alcaraz Calero, Gregorio Martínez Pérez and Antonio F. Gómez Skarmeta. **Towards a semantic-aware collaborative working environment.** *Computing and Informatics*, 30 (1): 7-30, 2013. (JCR Q3).
- [21] Félix J. García Clemente, José M. Alcaraz Calero, Jorge Bernal Bernabé, Juan M. Marín Pérez, Gregorio Martínez Pérez and Antonio F. Gómez Skarmeta. **Semantic Web-Based Management of Routing Configurations.** *Journal of Network and Systems Management*, 19 (2): 209-229, 2011. DOI: 10.1007/s10922-010-9169-6. (JCR Q3).
- [22] Juan M. Marín Pérez, Jorge Bernal Bernabé, José M. Alcaraz Calero, Félix J. García Clemente, Gregorio Martínez Pérez and Antonio F. Gómez Skarmeta. **Semantic-based authorization architecture for Grid.** *Future Generation Computer Systems*, 27 (1): 40-55, 2011. DOI: 10.1016/j.future.2010.07.008. (JCR Q1).
- [23] José M. Alcaraz Calero, Juan M. Marín Pérez, Jorge Bernal Bernabé, Félix J. García Clemente, Gregorio Martínez Pérez and Antonio F. Gómez Skarmeta. **Detection of semantic conflicts in ontology and rule-based information systems.** *Data & Knowledge Engineering*, 69 (11): 1117-1137, 2010. DOI: 10.1016/j.datak.2010.07.004. Special issue on contribution of ontologies in designing advanced information systems. (JCR Q2).
- [24] Manuel Gil Pérez, Jorge Bernal Bernabé, Juan M. Marín Pérez, Daniel J. Martínez Manzano and Antonio F. Gómez Skarmeta. **A policy-based dependability management framework for critical services.** *International Journal on Advances in Internet Technology*, 2 (4): 289-307, 2009.

1.6.2 International Conferences

- [25] Juan M. Marín Pérez, Antonio Moragón Juan, Jaime Arrazola Pérez, Javier Monge Rabadán and Antonio F. Skarmeta Gómez. **Security and Privacy in vehicular communications with INTER-TRUST**. In *Cybersecurity & Privacy Innovation Forum 2015. CSP FORUM 2015*, Brussels, Belgium, Apr 2015 *In press*. Springer.
- [26] Juan M. Marín Pérez, Jorge Bernal Bernabé, Gregorio Martínez Pérez and Antonio F. Skarmeta Gómez. **Managing identity-related information across borders**. In *Policies & Research in Identity Management (Future of Identity Workshop)*. *IFIP IDMAN 2013. 3rd IFIP WG 11.6 Working Conference on*, London, United Kingdom, Apr 2013.
- [27] Jorge Bernal Bernabé, Juan M. Marín Pérez, José M. Alcaraz Calero, Félix J. García Clemente, Gregorio Martínez Pérez and Antonio F. Gómez Skarmeta. **Towards an Authorization System for Cloud Infrastructure Providers**. In *Security and Cryptography. SECRYPT 2011. Proceedings of the International Conference on*, pp. 333-338, Seville, Spain, July 2011. IEEE Computer Society Press.
- [28] Jorge Bernal Bernabé, Juan M. Marín Pérez, Daniel J. Martínez Manzano, Manuel Gil Pérez and Antonio F. Gómez Skarmeta. **Towards a Policy-Driven Framework for Managing Service Dependability**. In *Dependability. DEPEND 2009. Second International Conference on*, pp. 66-72, Athens, Greece, June 2009. IEEE Computer Society Press. DOI: 10.1109/DEPEND.2009.16. Best Paper Award.
- [29] Juan M. Marín Pérez, Jorge Bernal Bernabé, Daniel J. Martínez Manzano, Gregorio Martínez Pérez and Antonio F. Gómez Skarmeta. **Towards the Definition of a Web Service Based Management Framework**. In *Emerging Security Information, Systems and Technologies. SECURWARE 2008. Second International Conference on*, pp. 362-367, Cap Esterel, France, Aug 2008. IEEE Computer Society Press. DOI: 10.1109/SECURWARE.2008.19.
- [30] Daniel J. Martínez Manzano, Juan M. Marín Pérez, Jorge Bernal Bernabé, Manuel Gil Pérez and Antonio F. Gómez Skarmeta. **Design-Time Learning for Operational Planning Improvement**. In *Emerging Security Information, Systems and Technologies. SECURWARE 2008. Second International Conference on*, pp. 326-331, Cap Esterel, France, Aug 2008. IEEE Computer Society Press. DOI: 10.1109/SECURWARE.2008.57.
- [31] Juan M. Marín Pérez, Jorge Bernal Bernabé, Jesús D. Jiménez Re, Gregorio Martínez Pérez and Antonio F. Gómez Skarmeta. **A proposal for translating from high-level security objectives to low-level configurations**. In *Systems and Virtualization Management: Standards and New Technologies. SVM 2007. 1st International DMTF Academic Alliance Workshop on*, Toulouse, France, Oct 2007.

1.6.3 Book Chapters

- [32] Juan M. Marín Pérez, Jorge Bernal Bernabé, José M. Alcaraz Calero, Jesús D. Jiménez Re, Félix J. García Clemente, Gregorio Martínez Pérez and Antonio F. Gómez Skarmeta. **Rule-Driven Architecture for Managing Information Systems**. In *Building Next-Generation Converged Networks*, pp. 417-435. 2013, CRC Press. DOI: 10.1201/b14574-21. ISBN: 978-1-4665-0761-6.
- [33] Jorge Bernal Bernabé, Juan M. Marín Pérez, José M. Alcaraz Calero, Jesús D. Jiménez Re, Félix J. García Clemente, Gregorio Martínez Pérez and Antonio F. Gómez Skarmeta. **Security Policy Specification**. In *Network and Traffic Engineering in Emerging Distributed Computing Applications*, pp. 66-93. 2013, IGI Global. DOI: 10.4018/978-1-4666-1888-6.ch004. ISBN: 978-1-4666-1888-6.

1.7 Structure

This document is structured into six main chapters that describe the work carried out during the realization of this PhD thesis. The document begins with an introductory chapter, followed by a description of current state of the art related to the main addressed topics. The next three chapters correspond to the core of this research work and they describe the main contributions and results. A final chapter provides some conclusions and future lines of direction.

A more detailed description of the different chapters that compound this document follows:

Chapter 1 describes the context of this PhD thesis and some background information for the reader to understand the main problems tackled in this research work. All the contributions resulted from this dissertation, as well as the main scientific publications produced during the realization of this research work, are also described in this chapter.

Chapter 2 analyses current state of the art related to access control in distributed environments and Cloud computing as well as policy-based approaches to access control and data-centric solutions, introducing the background needed for a better understanding of this work. This chapter describes some of the most relevant approaches related to the research activity conducted in this dissertation. These approaches present some deficiencies and open research questions that are addressed in the following chapters.

Chapter 3 presents a first approach to the application of Semantic Web technologies to manage and enforce security policies in multi-tenancy distributed systems, providing the initial authorization model with advanced reasoning capabilities that

allow performing added-value tasks such as conflict detection and resolution. An initial architecture for policy management in scenarios where resources are shared by multiple stakeholders is also provided. Due to the immaturity of Cloud computing technologies when this initial research was carried out, a first proof of concept was developed and applied to Grid scenarios, which are the precedent to Cloud computing and were mature enough to serve as realistic testbed in order to validate the feasibility of the proposal and to get a deep knowledge of the technologies used along the research conducted in this PhD thesis.

The results of the research described in this chapter have been published in the international journal *Future Generation Computer Systems* (JCR Q1) under the paper entitled “*Semantic-based authorization architecture for Grid*” [22].

Chapter 4 refines and formalizes the authorization model, endowing it with advanced features such as Role-based access control (RBAC), hierarchical RBAC (hRBAC), conditional RBAC (cRBAC) and hierarchical objects (HO). This chapter also adapts the information model and the architecture to the particularities of Cloud computing systems, providing multi-tenancy support and federation capabilities, enabling a fine-grained definition of what resources are available for each particular tenant. A Trust model is also provided to determine the possible business alliances or federations among cloud tenants. The proposed solution is validated by means of a proof of concept implementation of the access control system for OpenStack.

The results of the research described in this chapter have been published in the Special Issue on Exploiting Semantic Technologies with Particularization on Linked Data over Grid and Cloud Architectures of the international journal *Future Generation Computer Systems* (JCR Q1) under the paper entitled “*Semantic-aware multi-tenancy authorization system for Cloud architectures*” [18].

Chapter 5 cryptographically protects the authorization model to achieve a data-centric solution for self-protected data that can run in untrusted CSPs and provides extended Role-Based Access Control expressiveness. This chapter also provides a mathematic formalization of the authorization model that is then used to cryptographically protect the rules. This enables a rule-based approach where rules are under control of the data owner and access control computation is delegated to the CSP, but making it unable to access the data or to grant access to unauthorized parties. A secure key distribution mechanism with PKI compatibility for using standard X.509 certificates and keys is also provided. A proof of concept implementation has been developed and a working prototypical deployment of the proposal has been integrated within Google services.

The results of the research described in this chapter are under revision for publication in the Special Issue on Security and Dependability of Cloud Systems and Services of the international journal *IEEE Transactions on Services Computing* (JCR Q1) under the paper entitled “*SecRBAC: Secure data in the Clouds*” [17].

Chapter 6 provides a conclusion to this dissertation and overviews the more relevant results achieved during the realization of this PhD thesis. This chapter also describes some lines of future research to continue the advancement of current state of the art in Cloud computing security extending the research work carried out during this PhD thesis.

Chapter 2

Related work

This chapter provides some insights into related research works and current proposals in the field of access control for distributed systems, with special focus on Cloud computing authorization approaches. It gives a background overview and provides a brief analysis and discussion of related or similar studies.

A first section focuses on policy-based access control approaches for distributed systems, including some authorization solutions with semantic expressiveness. Then, the focus centers on solutions specifically designed for Cloud computing environments, providing a comparison among the analyzed approaches. An overview of some relevant policy specification approaches for access control is provided afterwards. Finally, a discussion about current data-centric solutions for self-protected data in Cloud computing is provided.

2.1 Access control in distributed systems

There is an important number of contributions in the field of access control for distributed systems, where several active working groups and projects are devoting their efforts to address some of the issues related to authorization.

A solution which provides policy-based authorization is the PrivilEge and Role Management Infrastructure Standards (PERMIS) [34]. PERMIS is a privilege management infrastructure that uses the principles of Role-Based Access Control (RBAC) to manage privileges and policies to make authorization decisions. This model uses X.509 attribute certificates to authenticate users and makes use of the attributes to check whether a user has access to the requested resources. The model supports hierarchical RBAC, allowing roles to inherit privileges from the hierarchy. The Argus authorization service [35] is another solution to provide consistent authorization decisions for distributed services. It is part of the European Middleware Initiative (EMI) [36]. Argus is an attribute-based system

which supports policies based on XACML. It uses attributes to identify the user attempting to perform an action, the resource on which the action is to be performed, the action itself, and other environmental information. Another approach is the Virtual Organization Membership Service (VOMS) [37]. VOMS is a system for managing authorization data within multi-institutional collaborations included in the Open Science Grid [38]. In this solution, user attribute assertions are issued in the form of X.509 attribute certificates. Then, these attributes are accessed and processed to take an authorization decision based on them.

These approaches are based on standardized attribute sets which may not fit the organization requirements to define the domain concepts needed by policy definitions. Although attribute sets are usually extensible by adding new types of attributes, these solutions do not allow the definition of semantics related to the elements they are protecting and administrators may find it difficult or they may even be unable to express the desired authorization policy based on these representations. It would be worth to use a model representation that enables to link with the protected resource definitions, providing more expressiveness and being able to better fit the organization domain representation.

Moreover, these solutions lack of semantic expressiveness and sometimes this may lead to a need for the administrator to previously refine the policy he has in mind in order to express it with the constructions provided by the language. Although administrators are usually able to do such a refinement, this may result in a complex policy definition with lots of rules and it can become a difficult task or they may even be unable to achieve an adequate policy definition. In this regard, ontology based approaches allow the administrator to combine the definition of semantics in the domain model together with the expressiveness of rules to specify policies using definitions which are closer to a natural representation of what the administrator has in mind and thus reducing the difficulty and the risk of mistakes when defining the policy.

A policy-based solution which supports semantic expressiveness is provided by KAoS [39][40]. KAoS is a set of platform-independent services, allowing fine-grained policy-based management of services. It enables the use of Semantic Web to manage services and provides a rich set of predefined ontologies that can be extended to accommodate specific application requirements. Moreover, there are some available graphical editing tools like KPAT which are quite useful in the policy definition phase. However, KAoS lacks of a complete solution with reasoning capabilities that facilitates advanced policy management features such as dealing with policy conflicts.

Some alternative approaches to represent and reason about policies for distributed systems include Rei [41] and Ponder [42]. Rei combines a logic language like Prolog within an ontological description like OWL for policy representation, making it a powerful logic-based policy language. However, this feature makes the solution difficult to be managed by users with limited expertise in logic languages. It lacks of any support in policy enforcement and requires extra effort from developers to integrate it into existing software to apply the policies. Moreover, Rei does not provide any policy specification tool which

is an important limitation to exploit this language for practical applications. Ponder is a declarative and object-oriented programming language. Its syntax is compiled at the end of the policy specification phase, making it impossible to change the policies at runtime. The fundamental policy types in Ponder are obligations and authorizations which could lead Ponder to be useful to manage access control. However, it is not an ontology-based solution and, therefore, the interoperability within other systems becomes cumbersome. It uses abductive reasoning techniques to analyse policy specifications and existing conflicts, but its level of expressiveness is limited. Moreover, its low level of abstraction for policies reduces its capacity to meet the requirements of dynamic environments.

2.2 Access control models for Cloud computing

Apart from the aforementioned access control solutions for distributed systems, there are also a set of access control models specifically designed for Cloud computing. Li et al. [43] provide a basic multi-tenancy access control model with discretionary access control (DAC) support. Later, Li et al. [44] also extend this model providing support for role management (RBAC) for cloud computing. Shirisha and Geetha [45] provide the next step supporting role hierarchies (hRBAC) in an access model designed to control the invocation of methods available in cloud computing APIs. Tsai and Shao [46] provide a semantic-aware multi-tenancy access control model with hRBAC and cRBAC support. Authors use an ontology for building up the role hierarchy for a specific domain. Ontology transformation algorithms are provided to compare the similarity of different ontologies.

Pereira [47] and Xu et al. [48] provide access control systems for the Cloud with analogous functionality. The main difference is that Pereira is focused on the IaaS layer whereas Xu et al is focused on the SaaS layer. They provide not only a multi-tenancy access control model with hRBAC and cRBAC support, but also a dynamic activation of roles in order to control a proper separation of duties in the Cloud. Danwei et al. [49] propose an access control system based on the *Usage CONtrol* access model (UCON) [50] which includes negotiation techniques in order to provide federation capabilities in Cloud computing. This model encompasses hRBAC, cRBAC, and attribute-based access control (ABAC) support. Fall et al. [51] provide a similar access control system for Cloud computing with the differentiating feature that federations between tenants are dynamically established, according to a risk management model in charge of deciding if two tenants can collaborate according to their previous interactions. Alcaraz et al. [52] provided an advanced multi-tenancy authorization model with RBAC, hRBAC, HO support for Cloud computing based on authorization statements defined by means of paths. This approach provides efficiency and performance making the system scalable. However, path-based representation could suffer of expressiveness limitations when authorization information needs to be expressed over information models that cannot be expressed using paths.

All the previously described models represent good attempts to provide access control models adapted to cloud computing. However, there is still an important effort in the way

of providing efficient and highly expressive models. These models have been compared with respect to a set of features and characteristics mainly related to expressiveness capabilities, supported management features, including multi-tenancy and federation support as well as other desirable features for an authorization model suitable for Cloud computing environments.

1	RBAC	Role-Based Access Control. It allows to assign users to roles and define authorization statements over roles.
2	hRBAC	Hierarchical RBAC. It allows to manage sub-roles (children) which inherits the privileges of the parent roles.
3	PBAC	Conditional RBAC (cRBAC) or Policy-based Access Control (PBAC). It allows to dynamically assign privileges only when some conditions are fulfilled. These conditions are defined with information of the protected resources.
4	HO	Hierarchical Objects. It allows to create object hierarchies and inherit the privileges defined for the object on top of the hierarchy to its children.
5	SoD	Separation of Duties. It allows to define mutually exclusive role assignments to subjects.
6	CBAC	Context-based Access Control. It allows to dynamically assign privileges only when some conditions are fulfilled. These conditions are defined with contextual information of the system, for example, time.
7	ABAC	Attribute-based Access Control. It allows to define authorization statements for attributes and when these attributes are assigned to users, they receive the specified privileges.
8	SF	Simple Federation. It allows to share authorization information between tenants in order to enable a tenant to define authorization statements using the information of another tenant. It is simple because a tenant A trusts or does not trust another tenant B and it implies that A enables B to access all the information model of A.
9	MT	Multi-tenancy. It allows to manage different administrative domains using the same shared infrastructure.
10	FGF	Fine-grain Federation. It allows a selective choice of the information that a tenant A wants to share with another tenant B (rather than sharing the complete information as in SF).
11	CD	Conflict Detection. It allows to detect inconsistencies in the authorization information in order to validate the information.
12	US	Unified Semantics. A common problem in authorization systems is that there is usually a mismatch between the semantics used to define the authorization model and the semantics used to define the information model. That can potentially cause security holes. An homogenization of such semantics brings an added-value to the proposal.
13	VS	Virtualization Support. It determines whether the proposed model has direct support to define the information model needed in a virtualized Cloud computing environment.
14	PV	Performance Validation. It indicates if the research work has empirical results or not.
15	AMC	Authorization Model Comparison. It indicates whether the research work has a comparative analysis between the different authorization models which are supported.

Table 2.1: Features compared across access control models

The aspects to be compared are described in table 2.1 and the comparison is shown in table 2.2. Note that the columns of table 2.2 match with the identifiers specified in table 2.1. As can be seen in table 2.2, the analyzed access control models lack of any or several

features. Features present in some models are usually missing in others. Moreover, none of them are able to support federation between tenants or advanced reasoning capabilities for policy management, which would result of interest for Cloud environments. An access control model that combines the features present in previous proposals would be desirable in order to go a step forward towards a multi-tenancy authorization model suitable for Cloud computing, including several advantages with respect to its predecessors. The model should provide a high level of expressiveness as well as a fine-grained control over the definition of federations and coalitions between tenants.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Li et al. [43]									X						
Li et al. [44]	X								X						
Sirisha et al. [45]	X	X							X						
Tsai et al. [46]	X	X	X			X		X	X			X	X		
Pereira [47]	X	X	X		X	X		X	X						
Xu et al. [48]	X	X	X		X	X			X						
Danwei et al. [49]	X	X	X		X	X	X	X	X						
Fall et al. [51]	X		X		X	X	X	X	X						
Alcaraz et al. [52]	X	X		X				X	X				X	X	

Table 2.2: Comparison of the different access control models analyzed

2.3 Policy specification approaches

There are various security policy models and policy specification languages in the literature. This section briefly introduces XACML [8], the DMTF Common Information Model (CIM) [1] and Ponder [42] since they also provide associated management frameworks and they constitute the most representative policy specifications and standards for access control management in distributed systems.

The eXtensible Access Control Markup Language (XACML) [8] is being developed and maintained by the Organization for the Advancement of Structured Information Standards (OASIS) [53]. It is currently the main standard for access control definitions in distributed systems. The standard describes both an access control policy language and a request/response protocol. The policy language provides a common means to express subject-target-action-condition access control policies. Figure 2.1 shows the policy language model for XACML v3.0.

The main components of this model are policy sets, policies and rules. A rule is the elementary unit of a policy and it is composed of a target, which refers to the set of subjects, resources and actions to which the rule applies; an effect specifying the consequence (permit or deny); a condition, which refines the rule applicability; obligations that define specific actions that should be performed; and advices, which are similar to obligations, but not mandatory. A policy is a set of rules, which also have a target, a rule combining

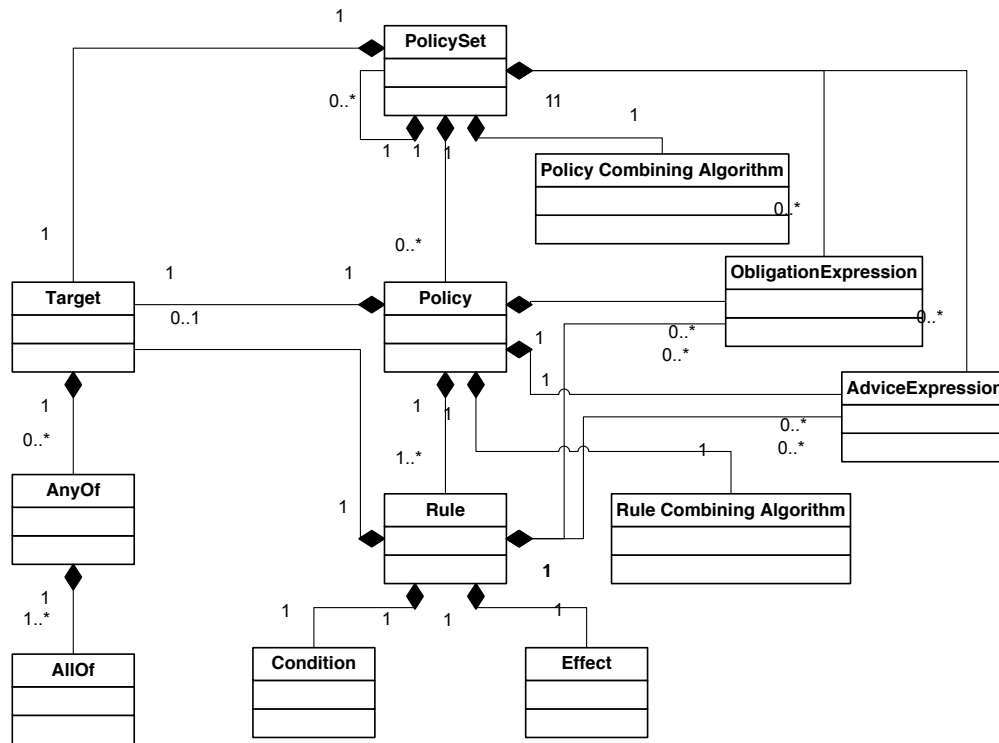


Figure 2.1: XACML 3.0 Policy Language Model

algorithm that specifies how to evaluate rules, a set of rules, obligations and advices. A policy set is a set of policies and it also has a target, a policy combining algorithm, a set of policies, obligations and advices. In turn, the request/response protocol expresses queries about whether a particular access should be allowed and describes answers to those queries. One of the main limitations of this approach is its capability to express system descriptions using a standard way since it lacks an information model to represent the managed domain ready to be used in policy definitions. Moreover, although XACML provides the possibility for a policy document to refer to others, it is not easy to express and interpret complex policies with this language.

The Common Information Model (CIM) is the main standard from the Distributed Management Task Force (DMTF) [54]. This standard provides a common definition of management-related information independent of any specification. Figure 2.2 provides an overview of the CIM Policy schema. In CIM, a policy rule is composed of a condition and an action and they can be aggregated in policy groups. The model includes concepts to define different kinds of policies such as authorization, authentication, packet filtering and privilege propagation for delegation. CIM is a model not linked to any representation or definition language. However, for an information model to be useful, it has to be mapped into some specification. For that, the DMTF also defines two standards: WBEM and WS-CIM. The Web-Based Enterprise Management (WBEM) [55] standard uses an

XML representation for CIM called CIM-XML, while WS-CIM provides a CIM mapping to WSDL and XML Schema. WS-CIM provides a representation more suitable than WBEM representation for tasks such as validation and semantic analysis, but WS-CIM representation is strongly linked to Web services specifications. CIM provides a wide support for management security policy and system concepts, but it must be extended to include domain specific concepts. Although it provides extension mechanisms, but this extension is not always straightforward for developers or administrators.

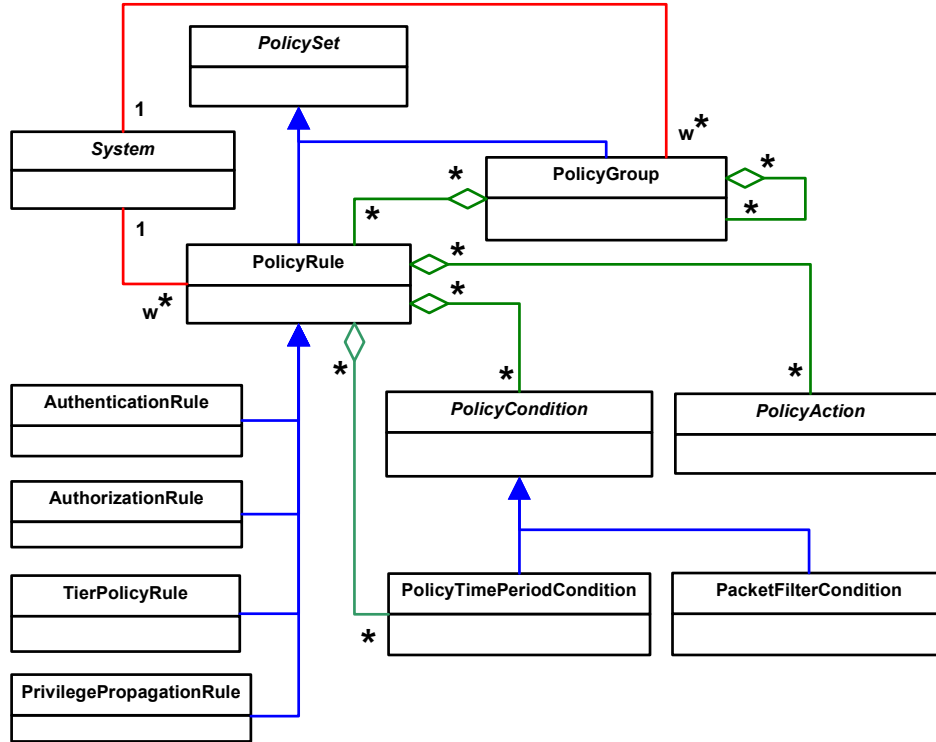


Figure 2.2: CIM Policy schema (reduced)

The PONDER Policy Based Management Toolkit [42][56] is the result of a research work in the Imperial College London. It is a declarative, object-oriented language developed for specifying management and security policies. Figure 2.3 overviews the Ponder policy structure. The language distinguishes between basic and compound policies. A basic policy is considered a rule that defines the behaviour for a set of subjects and targets. Ponder permits to express authorizations, obligations, information filtering, refrain policies, and delegation policies. Compound policies allow to group basic policies. Ponder can describe any rule to constrain the behavior of components, in a simple and declarative way. As for XACML, this solution also lacks an information model for domain representations, what supposes a limitation to express system descriptions in a standard way in order to be used in policies. Moreover, it is not an standard-based language and it has not been designed with interoperability or support for standard proposals with similar objectives (management and authorization policies), such as DMTF standards or XACML.

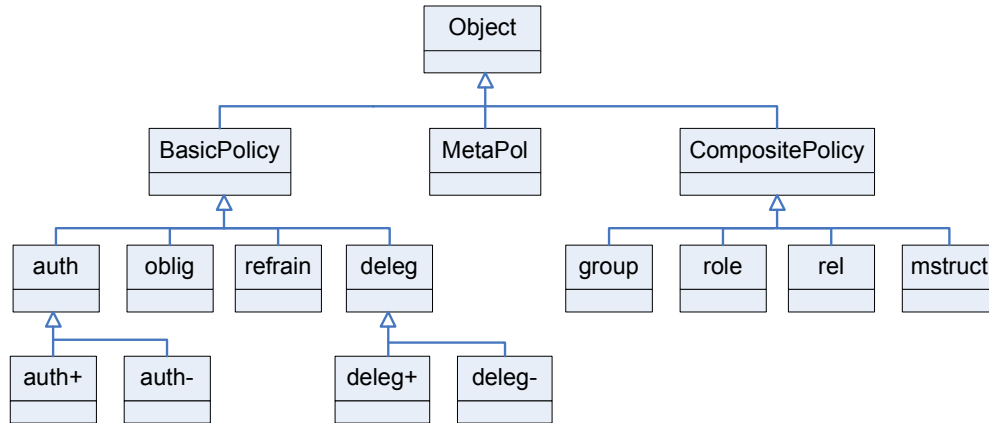


Figure 2.3: Ponder policy

From the perspective and goals of the research work presented in this PhD thesis, the limitations to express domain descriptions make XACML and Ponder weaker candidates for policy representation, whereas the CIM Model is suitable to define a common and standardized format to represent any domain system descriptions. Moreover, an advantage of the CIM-based approach to information management is that the model can be easily mapped to structured specifications such as XML, which can then be used for policy analysis as well as management of interoperable policies in distributed environments.

2.4 Data centric access control

Different approaches can be found in the literature to retain control over authorization in Cloud computing by the data owner. In [57] authors propose to keep the authorization decisions taken by the data owner. The access model is not published to the Cloud but kept secure on the data owner premises. However, in this approach the CSP becomes a mere storage system and the data owner should be online to process access requests from users. Another approach from [58] deals with this issue by enabling a plug-in mechanism in the CSP that allows data owners to deploy their own security modules. This permits to control the authorization mechanisms used within a CSP. However, it does not establish how the authorization model should be protected, so the CSP could potentially infer information and access the data. Moreover, this approach does not cover Inter-cloud scenarios, since the plug-in module should be deployed to different CSPs. Additionally, these approaches do not protect data with encryption methods and the CSP may be able to access it or release it to third parties bypassing the authorization mechanism.

Data encryption is a possible mechanism to protect data against CSP access. Following a straightforward approach, one can include data in a package encrypted for the intended users. This is usually done when sending a file or document to a specific receiver

and ensures that only the receiver with the appropriate key is able to decrypt it. From an authorization point of view, this can be seen as a simple rule where only the user with privilege to access the data will be able to decrypt it (i.e. the one owning the corresponding key). However, no access control expressiveness is provided by this approach. Only that simple rule can be enforced and just one single rule can apply to each data package. Thus, multiple encrypted copies should be created in order to deliver the same data to different receivers.

To cope with these issues, data-centric approaches, mostly based on Attribute-based Encryption (ABE) [13], have also arisen for data protection in the Cloud [12]. In ABE, the encrypted ciphertext is labeled with a set of attributes by the data owner. Users also have a set of attributes defined in their private keys. They would be able to access data (i.e. decrypt it) or not depending on the match between ciphertext and key attributes. The set of attributes needed by a user to decrypt the data is defined by an access structure, which is specified as a tree with AND and OR nodes. There are two main approaches for ABE depending on where the access structure resides: Key-Policy ABE (KP-ABE) [13] and Ciphertext-Policy ABE (CP-ABE) [11]. In KP-ABE the access structure or policy is defined within the private keys of users. This allows to encrypt data labeled with attributes and then control the access to such data by delivering the appropriate keys to users. However, in this case the policy is really defined by the key issuer instead of the encryptor of data, i.e. the data owner. So, the data owner should trust the key issuer for this to properly generate an adequate access policy. To solve this issue, CP-ABE proposes to include the access structure within the ciphertext, which is under control of the data owner. Then, the key issuer just asserts the attributes of users by including them in private keys. However, either in KP-ABE or CP-ABE, the expressiveness of the access control policy is limited to combinations of AND-ed and OR-ed attributes.

Several approaches have been developed to try to alleviate ABE expressiveness limitations. Authors in [59] propose a solution based on CP-ABE with support for sets of attributes called Ciphertext Policy Attribute Set Based Encryption (CP-ASBE). In CP-ASBE attributes are organized in a recursive set structure and access policies can be defined upon a single set or combining attributes from multiple sets. This enables the definition of compound attributes and specification of policies that affect the attributes of a set, independently of how they are combined. In [60] an approach named Hierarchical Attribute-Based Encryption is presented. It uses a hierarchical generation of keys to achieve fine-grain access control, scalability and full delegation. However, the solution proposed in this approach implies that attributes should be managed by the same root domain authority. In [61], authors extend CP-ASBE with a hierarchical structure to system users in order to improve scalability and flexibility. This approach provides a hierarchical solution for users within a domain, which is achieved by a hierarchical key structure. Another approach is Flexible and Efficient Access Control Scheme (FEACS) [10]. This solution is based on KP-ABE and provides an access control structure represented by a formula involving AND, OR and NOT, enabling more expressiveness for the KP-ABE approach.

The aforementioned ABE-based solutions proposed to deal with access control in Cloud computing are based on the Attribute Access Control (ABAC) model. As commented in Section 1.4, both ABAC and RBAC models have their own advantages and disadvantages. On one hand, RBAC may require the definition of a large number of roles for fine-grain authorization (role explosion problem in RBAC). ABAC is also easier to set up without need to make an effort on role analysis as needed for RBAC. On another hand, ABAC may result in a large number of rules since a system with n attributes would have up to 2^n possible rule combinations (rule explosion problem in ABAC). ABAC separates authorization rules from user attributes, making it difficult to determine permissions available to a particular user, while RBAC is deterministic and user privileges can be easily determined by the data owner.

Moreover, the cryptographic operations used in ABE approaches usually restrict the level of expressiveness provided by the access control rules. Concretely, role hierarchy and object hierarchy capabilities provided by the solution cannot be achieved by current ABE schemes. Moreover, private keys in ABE should contain the attributes of the user, which tightens the keys to permissions in the access control policy, that is, user privileges are not completely independent of their private key, which would be desirable in order to separate authentication and authorization in the access control scheme.

Chapter 3

Semantic-based Authorization in multi-tenancy environments

This chapter provides a first approach to security management based on Semantic Web technologies to define, manage and enforce security policies in a multi-tenancy scenario. These policies are defined by means of semantic-aware rules which help the administrator to create higher-level definitions with more expressiveness. The rules also permit performing added-value tasks such as conflict detection and resolution, which can be of interest in medium and large scale scenarios where different administrators define the authorization rules that should be followed before accessing resources. The proposed solution has been also tested providing some reasonable response times in the authorization decision process.

The research work described in this chapter has been published in the paper entitled *Semantic-based authorization architecture for Grid* [22].

This chapter is structured as follows. Section 3.1 introduces the concepts of Semantic Web technologies which are used to represent authorization policies in this proposal. Section 3.2 presents the proposed architecture to provide semantic-aware authorization for Grid and its integration within the Globus Toolkit. Details about how authorization is performed and how this approach deals with heterogeneous domains are provided in Section 3.3. Finally, this chapter concludes with some statistical results and details of the proof of concept implementation are exposed in Section 3.4.

3.1 Authorization policies and domain representations

In order to take policy-based authorization decisions, a representation of policy rules should be provided. Moreover, since policy rules are based on domain concepts, a rep-

resentation of such a domain is also necessary. These representations should be generic and flexible enough to be able to fulfil different domain modelling needs to cope with the heterogeneity of distributed systems. In this proposal, ontology languages are used to address these representations, since they endow with high expressiveness to model concepts and semantics for different domains. These languages provide constructors to define ontologies, where an ontology is defined as a set of axioms which provide a formal representation of a given domain, containing the schemas and vocabularies that define domain concepts, their properties and relationships among them, as well as instances of these concepts and relationships, representing a specific situation in the domain according to those schemas.

3.1.1 Semantic Web representation for authorization policies

Different approaches for security policy languages such as XACML, which make use of mere XML to represent policies, differ from those which take advantage of ontologies on the degree and manner of specifying the necessary relationships among terms. Ontologies are rigorously formalized following logical theories so that the degree of expressiveness is higher, enabling the automated reasoning quality. Thus, semantic features for policy representation such as transitivity, reflexivity or inverse and complement of properties are exclusive for ontologies and therefore are not available when dealing with non-semantic-based languages.

The Web Ontology Language 2 (OWL 2) [62] is a W3C standard which enables the specification of ontologies, defining class hierarchies and their relationships, associated properties and cardinality restrictions. Different kinds of OWL 2 ontologies can be distinguished, depending on the provided expressiveness and its computational complexity. In this context, the term expressiveness refers to the set of structural elements or constructors that can be employed in the axioms for the ontology description. Concretely, three different profiles [63] are defined for OWL 2: OWL 2 EL, OWL 2 QL and OWL 2 RL. OWL 2 EL is designed for ontologies with very large number of properties and classes, mainly providing existential quantification. OWL 2 QL is useful for ontologies with a large number of instances, where query answering is the most important reasoning task. These two profiles are targeted for particular applications, providing high performance reasoning algorithms for them, but limiting their expressiveness. In turn, OWL 2 RL is aimed at providing scalable reasoning, without sacrificing too much expressive power. This last profile has been selected in this research since it provides a wide variety of constructors to describe knowledge whereas it can be handled by reasoners implemented by means of rule-based engines, which offer good performance and scalability.

This representation allows the inclusion of semantics in the domain description, providing with 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. Moreover, the domain description may be composed of different ontologies which are significant for its representation. For example, the administrator can select an ontology based on CIM, such as [64], for applications, systems, networks, devices, and other technology-focused management domains, or SOUPA ontology [65] for representing ubiquitous and pervasive applications, or also SWRC ontology [66] for modelling entities of research communities such as persons, organizations, publications and their relationships.

Models defined using OWL 2 can be enriched by means of the Semantic Web Rule Language (SWRL) [16]. This language is used to represent rules on the Semantic Web and it extends OWL 2 in order to provide a way to express conditional knowledge. The language itself is not decidable, but a syntactic restriction called DL-Safe context [67] can be applied in order to restore the decidability. The combination of SWRL and OWL 2 can be used to express both the authorization policy and also more deductive processes on the system which may be relevant for authorization decisions. For instance, this combination enables the definition of rules to provide semantics related to the resources which are to be protected like “every file in the domain whose name ends with / is a directory” that makes the system to consider these files as directories for authorization purposes. Hence, the OWL 2 RL profile together with the SWRL DL-Safe have been selected to carry out the semantic-aware approach to Grid authorization. The OWL 2 RL profile and the SWRL with DL-Safe restriction will be indistinctly referred as OWL and SWRL henceforth.

The combination of OWL ontologies and SWRL to specify policy rules offers the advantage of allowing automated reasoning. This is carried out by what is called a reasoner, referring to a specific piece of software which performs reasoning processes. These processes constitute a remarkable added value of the usage of Semantic Web technologies, since they are able to infer new knowledge, that is, deriving additional information not explicitly specified in the ontology. Moreover, they also perform a formal validation and verification of the domain constraints which are specified in the ontology to assure they are fulfilled.

The representation of the policy and domain information used in our authorization architecture is based on these Semantic Web concepts and technologies, providing the following set of features as added value for our policy language:

- Separation between domain description and policy description. Our approach separates the concepts that are necessary to describe the domain to be protected and the rules which use such concepts to create policies expressing the desired security for the administered services. The separation of domain description and policy description permits us to manage both specifications individually using different techniques.
- Reasoning capabilities about domain descriptions. The domain representation is done in the form of an ontology, allowing reasoning processes to check constraints and query the information.

- Reasoning capabilities about policy description. Policy rules are created in the form of horn-like rules, enabling reasoning capabilities for policy conflict analysis techniques.
- Interoperability. The use of standards and ontological representation eases the interpretation and integration of the information. This enables advanced capabilities like concept alignment between heterogeneous domains, allowing the combination of concepts from different organizations.

Some of the concepts introduced in this section will be further explained and used including some exemplifications in the following sections of this chapter, with the aim of making them more clear to the reader.

3.1.2 The Common Information Model

The proposed ontology for domain representation is based on a well-recognized standard information model, namely the Common Information Model (CIM) [1] created by the Distributed Management Task Force (DMTF). CIM is an approach for providing consistent information models with well-defined associations that capture management content for applications, systems, networks, devices, and other technology-focused management domains. CIM models establish a common conceptual framework that enables both hardware and software providers to consistently represent management information across vendor boundaries. It is a common standard for system management supported in commonly used platforms such as Linux [68] and Windows [69].

The usage of such a common model enables the representation of policy information in a uniform manner and guarantees the necessary extensibility to support authorization for any domain. The main benefits of this model that have contributed to the motivation for the choice of using it in this research work are:

- CIM is an standard information model which is independent of the language used to represent it, thus it allows ontological representation in OWL
- CIM is a very complete information model covering almost all the different aspects required in information systems, including services, networks, applications, etc., so its OWL representation can be used as generic ontology
- There are related standards and technologies grouped under the WBEM [55] specifications which allow to dynamically gather the current state of the system by means of CIM, which may be used by organizations to get information about their systems for authorization purposes
- CIM is free, open source and extensible, ready for new information systems.

CIM is composed of a *Core Model* and *Common Models*, where the *Core Model* captures basic notions that are applicable to all areas of management and the *Common Models* are information models that capture notions that are common to particular management areas. The *Common Models* mainly considered in our work to represent security policies are:

- *CIM Policy Model*. The Policy Model provides a common framework for specifying system behaviors that are sufficiently abstract and scalable to configuring large complexes of computer systems, i.e. the Policy Model is a specific model for expressing policies in a general and scalable way.
- *CIM User and Security Models*. The objective of the User and Security Models is to provide a set of relationships between the various representations of users, their credentials, the managed elements that represent the resources, and the resource managers involved in system user administration.

The descriptions of CIM model are provided by the CIM Schema, but it is independent of any implementation or specific representation. However, for an information model to be useful, it has to be mapped into some encoding. As Figure 3.1 shows, CIM can be mapped to (or represented as) several structured specifications.

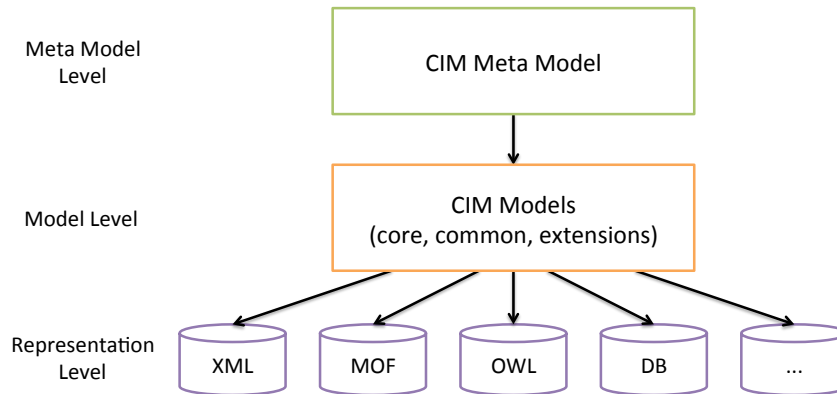


Figure 3.1: CIM modeling levels

CIM meta model provides a formal definition of the CIM models, i.e. it defines the terms used to express the models and its usage and semantics. Figure 3.2 shows the main elements of CIM Meta Schema such as *Schema* that is a group of classes with a single owner, and *Class* that defines the properties and the methods common to a particular kind of object. The model also defines the concept of *Association* that represent relationships between two or more classes using references, and *Qualifier* that provides additional information about classes, associations, methods, parameters, properties or references.

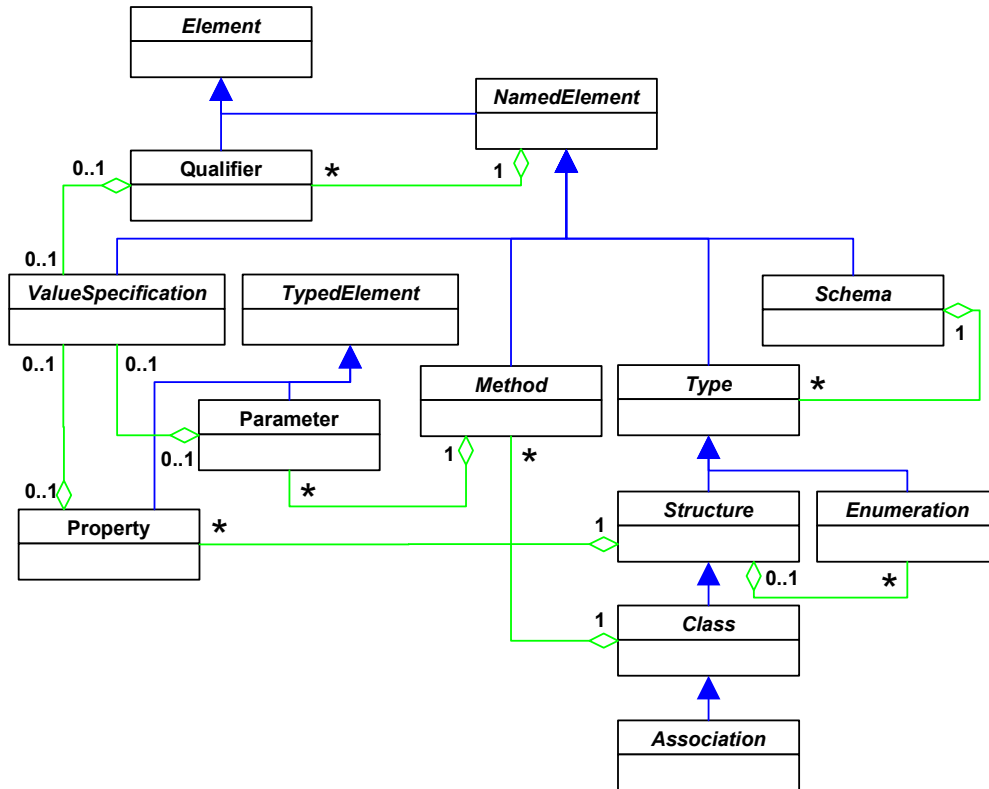


Figure 3.2: CIM Metaschema (reduced)

The DMTF defines a specific format called Managed Object Format (MOF) [70] in which CIM schema is specified. However, current versions also include their representation in XML since this representation is more interoperable and supposes a benefit for using the model in Web Services and other interoperable technologies. For mapping CIM objects using XML, there are two different alternatives: schema mapping and meta-schema mapping. The meta-schema mapping defines a DTD (Document Type Definition) to describe in a generic manner the notion of a CIM class or instance. CIM element names are mapped to XML attributes or element values, rather than to specific XML element names. On the contrary, the schema mapping defines an XML Schema to describe the CIM classes; in this approach CIM instances are mapped to valid XML documents for that schema. Essentially this means that each CIM class generates its own XSD fragment whose XML element names are the same that the corresponding CIM element names. The meta-schema mapping was firstly adopted by the DMTF, as it only required one standardized DTD for the whole CIM regardless the version of this information model used in one particular implementation. However, there are obvious benefits to employing a schema mapping (more validation power, and a more intuitive representation of CIM in XML) and in current versions the DMTF moved to a representation in XML based on schema mapping.

Regarding the mapping of CIM to OWL, several representations of CIM in OWL are available such as [64], [71], and [72]. The mapping proposed by [64] holds a high expressiveness in the translation to OWL in contrast to the approach provided on [71], which only preserves RDF expressiveness. Regarding the way in which relationships between CIM elements have been mapped, [72] proposes the mapping of CIM relationships to OWL classes, whereas [64] defends to map CIM relationships to OWL properties. This last approach confer simplicity to CIM models and, in turn, it allows to model semantics applied to these properties.

3.2 Authorization architecture for Grid services

In order to contextualize the authorization model to be developed during this research, a representative distributed environment has been chosen to enable a realistic design of the authorization model and the development of a proof of concept implementation to test the feasibility of the proposal. A Grid scenario where multiple organizations share the resources of the Grid services has been chosen since it provides a feasible environment for this research and the development of an authorization model with multi-tenancy capabilities.

The Globus Toolkit [73] is an open source software toolkit used for building grids developed by the Globus Alliance. It is considered the reference software system for generic grid implementations. In turn, the Globus A&A (Authentication and Authorization) architecture supports authentication and delegation through the use of X.509 certificates and public keys, and an authorization framework [74] which is able to process restricted delegation and can take the identity of intermediates into account. This authorization framework allows for a variety of authorization mechanisms, for instance, using a grid-map file, access control lists or accessing authorization services either by local calls or by means of remote callouts via the Security Assertion Markup Language (SAML) [75] or the SAML 2.0 profile of XACML [76].

The integration of our solution within the Globus Toolkit is achieved by means of a set of components which link with the authorization framework. This framework evaluates a chain of configured Policy Decision Points (PDPs for short) to determine if the client making the invocation can access the resource or service. The chain can also be configured with Policy Information Points (PIPs) which can be used to collect attributes about resources, operations, subjects, etc. to be used in the decision making process. While the toolkit provides some implementations for the supported authorization mechanisms, the framework is pluggable and custom mechanisms can be written and configured. Our solution makes use of this extension mechanism to connect the Globus Toolkit with the service which is in charge of taking the authorization decision in our approach.

Moreover, message level security in our solution is provided by means of the Globus GSI Secure Conversation mechanism [77], where the client establishes a context with

the server before sending any data. Once the context establishment is completed, the client can securely invoke an operation on the service by signing or encrypting outgoing messages using the shared secret captured in the context.

Finally, Globus Toolkit implements the WS-Resource framework (WSRF) [78] specifications, which allow to declare and develop the association between a Web service and one or more stateful resources. WS-Resources are defined as the composition of a Web service and a stateful resource. The WSRF specifications define different mechanisms to access, monitor and manage the lifecycle of such WS-Resources. This is used in the proposed solution for organizations to provide and manage their information in their own WS-Resource.

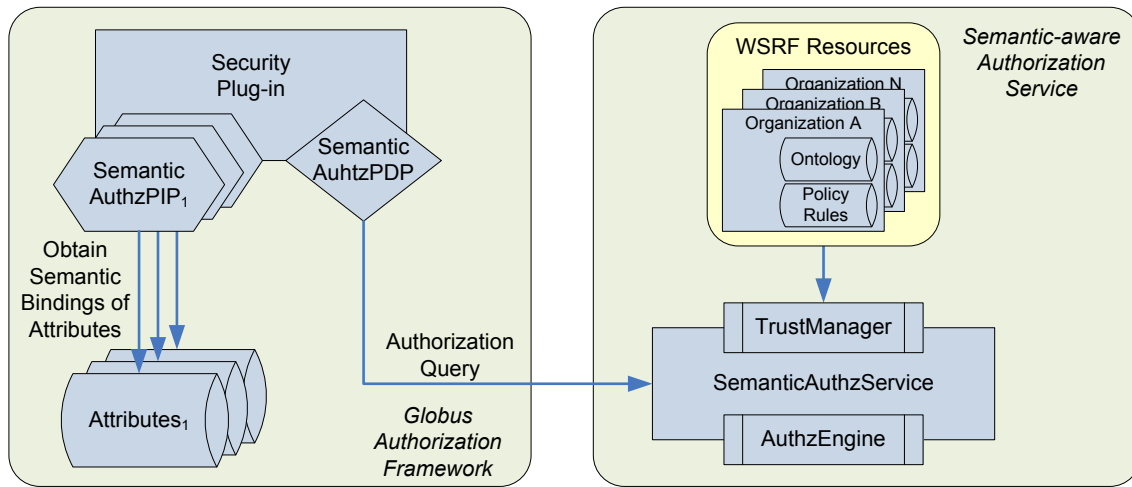


Figure 3.3: Authorization architecture

Figure 3.3 shows the proposed architecture to enable our authorization approach for Globus. The architecture provides a security plug-in which is integrated within the authorization framework, enabling the processing of authorization requests. This plug-in consists of a `SemanticAuthzPDP` and a set of `SemanticAuthzPIP` interceptors. The `SemanticAuthzPIPs` collect attributes associated with authorization requests such as subject, protected resources, action and so on, which are necessary for the authorization decision. Different PIPs can be used for different attributes. For example, there may be a PIP for attributes related to subjects and another one for resource attributes. These attributes are represented in terms of ontology instances, which is the expected format for our authorization service. In turn, the `SemanticAuthzPDP` is really a PDP proxy for Globus. It redirects the authorization request to our authorization service, named `SemanticAuthzService`, which is the one which really takes the authorization decision, acting as real PDP. The `SemanticAuthzPDP` provides the information retrieved by the PIPs and asks this service for an authorization decision. The service is responsible for reasoning about the request and then taking an authorization decision. It evaluates the request using the security policy and returns whether the given subject can perform the action on

the requested Grid service or not. This response is returned to Globus, for it to finally enforce the authorization policy by granting or denying the access to the requester.

It should be noted that this architecture design keeps the Globus authorization framework and our authorization service decoupled. This enables us to reuse the service in other architectures and implementations.

3.2.1 Managing authorization information

The main component of the architecture is the `SemanticAuthzService`. This service should handle the multiple ontologies and policies of the Grid participant organizations to make use of that knowledge when taking authorization decisions. Each organization has its own information system describing its users, services, resources and any other relevant domain information. The authorization service needs organizations to provide such information to use it for authorization purposes. This can be achieved by means of the WSRF specifications, enabling organizations to provide and manage their information in their own WS-Resource. The information is represented and kept up-to-date by organizations in OWL ontologies and SWRL rules in order to enable the service to perform the semantic-aware authorization process. Using the WSRF standard technology, information can be kept separate, using a different WS-Resource for each organization.

In this approach where multiple WS-Resources are used, an out-of-band mechanism is needed to create them. The WSRF specifications recommend to follow a factory pattern in which a different service is used to create the WS-Resources and the main service manages their information. Hence, although not depicted in Figure 3.3 for clarity reasons, there is also a factory service in our proposed architecture in charge of managing the WS-Resource creation.

Handling organization ontologies and policies in different WS-Resources also eases the lifecycle management of an organization within the Grid. Figure 3.4 shows a conceptual sequence diagram to illustrate a WS-Resource lifecycle, which is tied to the organization participation in the Grid.

When the organization joins the Grid, it uses the factory to create its own WS-Resource. This call returns an endpoint reference which is handled by the organization for future operations on the resource. Then, it makes use of the authentication service to manage its authorization data, accessing it through the appropriate WS-Resource-qualified endpoint reference. Initially, the organization has to define its own ontology and authorization policy, which is stored and handled by the authorization service in the WS-Resource of the organization. Although they are just depicted in the diagram as a single *operation()* call, different operations (get, add, update, etc.) can be made by the organization in order to manage its knowledge while still being part of the Grid. Finally, if the organization decides to end its participation in the Grid, it sends a request message for the service to destroy its WS-Resource.

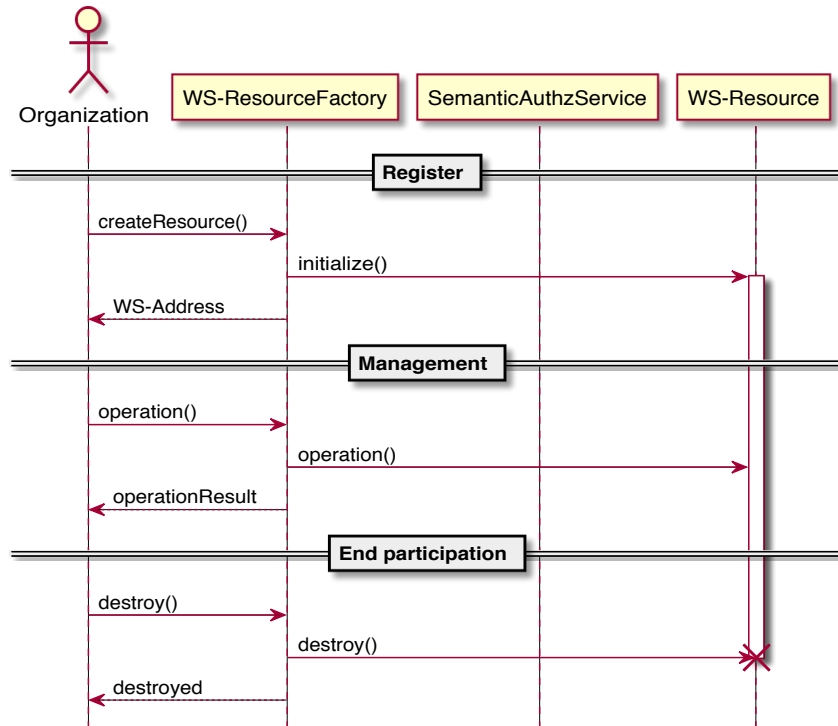


Figure 3.4: Organization WS-Resource lifecycle

3.2.2 Access request processing

Once the architecture has been set up and organizations have properly created their own WS-Resources, providing their ontologies and authorization policies, a resource access request made by a client will trigger the sequence depicted in Figure 3.5.

The client request is intercepted by the Globus Toolkit, which has been configured to use the proposed authorization architecture, registering the `SemanticAuthzPIPs` and the `SemanticAuthzPDP`. The toolkit uses PIPs to collect information relevant for authorization (e.g. subject and resource attributes). With that information, the toolkit queries the PDP for an authorization decision, forwarding to it that information. The PDP connects the `SemanticAuthzService` for it to actually take the decision. The call from the PDP provides request information in OWL, containing the information collected by the PIPs. The `SemanticAuthzService` uses a component of our architecture called `TrustManager` to get the organizations information to be used for authorization. The `TrustManager` is able to determine the organizations involved in the request and the trust relationships established among them. It accesses the WS-Resources and, based on the request and the trust relationship information, it returns the information in OWL and SWRL of the corresponding trusted organizations. This information, together with the one provided by the PDP is passed to the `AuthzEngine`, which performs the reasoning process.

The `AuthzEngine` uses an OWL reasoner to infer new knowledge based on the pro-

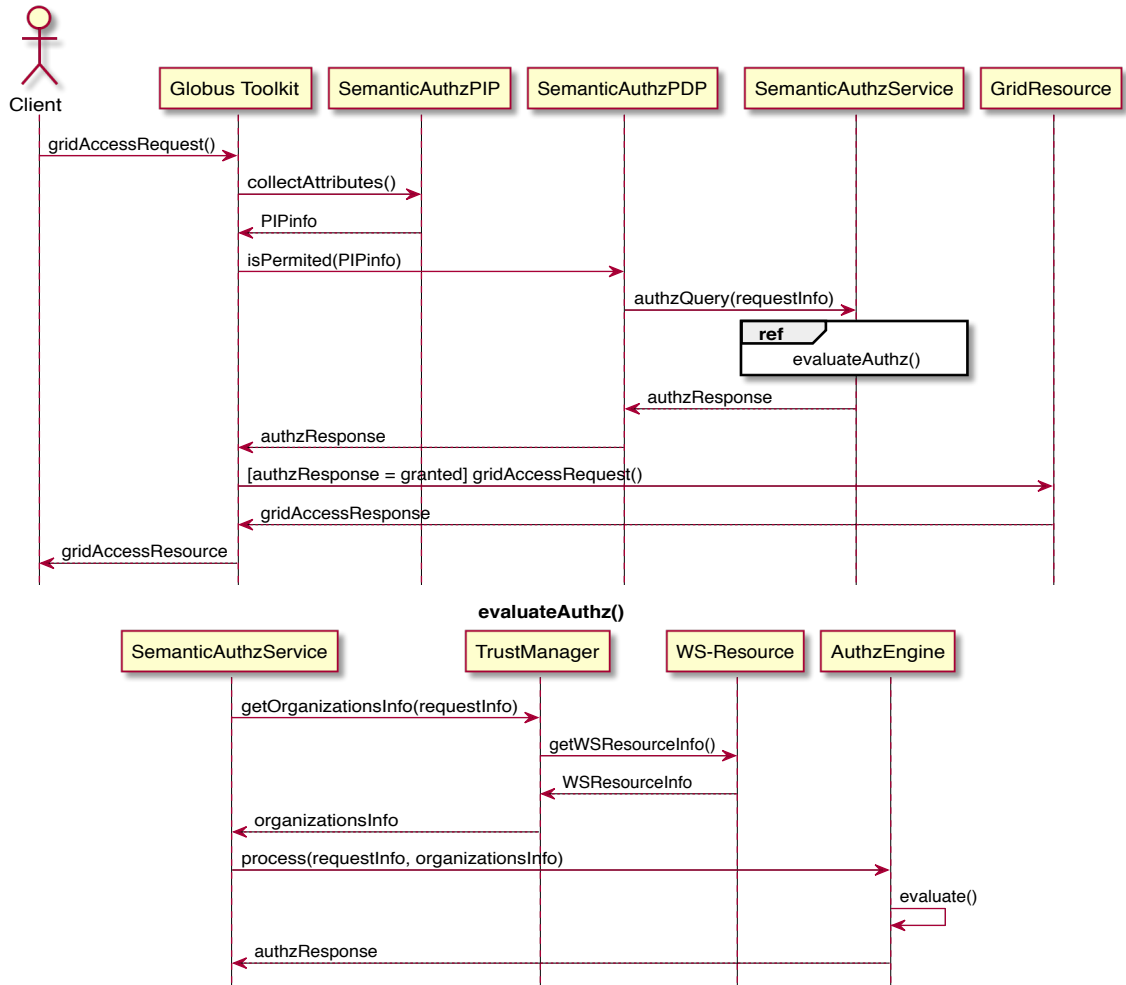


Figure 3.5: Access request processing

vided information, applying the authorization rules and determining whether they indicate a positive (permit) or negative (deny) authorization. Details about the reasoning process can be found in Section 3.3. The result of this process is used by the service to generate the authorization response, which is returned to the PDP and to the Globus authorization framework. In case of positive answer, the client accesses the requested resource.

3.3 Authorization policy management

The use of Semantic Web languages like OWL and SWRL provides a generic way of specifying domain descriptions and policies with powerful expressiveness, being able to satisfy the modelling requirements of different domains. However, for the authorization service to understand and enforce the specified policy, the definition of a minimum set

of concepts becomes necessary. Some concepts should be specified in order to provide a positive or negative authorization statement. It should be noticed that the definition of such concepts does not mean a restriction to the organizations ontologies. They can directly use this set of concepts to express authorization, which are already provided and so they do not need to design a new set. Or they can still define their particular concepts and then provide a few OWL constructors or SWRL rules to map these concepts to those which are needed by the authorization engine to perform the authorization process.

Moreover, a generic and extensible ontology to model the protected resources is provided based on the DMTF's CIM model as explained in previous section. Such an ontology defines concepts regarding system resources, which fits well with distributed environments. Like the aforementioned authorization concepts, this ontology can also be used by organizations which want to use our authorization solution without the need to define its own ontology from scratch. This approach has been followed for the proof of concept implementation exposed in Section 3.4.

The basic components of an authorization decision (or privilege) are subjects, actions, and targets. A sample authorization decision or privilege would be read as follows: it is permitted for subject(s) S to perform action(s) A in the target(s) T. To represent the privileges which are interpreted by the `SemanticAuthzService`, our solution makes use of the set of CIM concepts depicted in Figure 3.6.

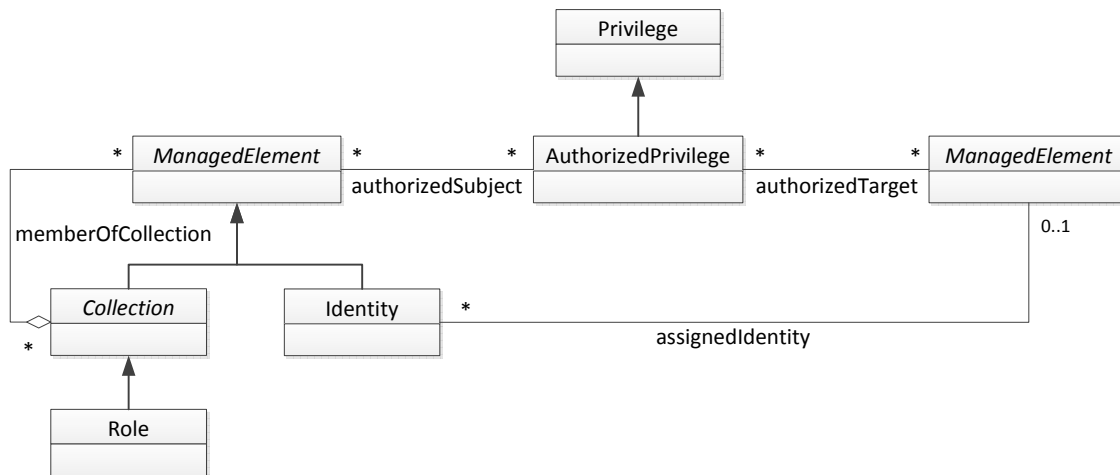


Figure 3.6: UML diagram of concepts for privilege representation

`Privilege` is the base concept for all types of activities which are granted or denied to a subject for a target. `AuthorizedPrivilege` is the specific subclass defining current privileges which result of applying the authorization policy rules. The association of subjects to `AuthorizedPrivilege` is accomplished via the `authorizedSubject` association. The entities that are protected (i.e. targets) can be similarly defined via the `authorizedTarget` association.

Subjects are represented by the `Identity` and `Role` concepts. The `Role` concept is used to represent a position or set of responsibilities within an organization, organizational unit or other scope, and it can be filled by people or non-human entities using the `memberOfCollection` association. An instance of an `Identity` represents an element that acts as a security principal, i.e., an entity which can be identified and verified by an authentication process. Elements with identities can be organizational entities, services, systems, etc. The element which is represented by an `Identity` is described using the `assignedIdentity` association. Both `authorizedSubject` and `authorizedTarget` associations connect with the `ManagedElement` concept, which is the common superclass (or top of the inheritance tree) for the non-association classes in CIM, thus enabling the definition of any element of the model as subject or target for authorization.

To illustrate these concepts, let us consider a Grid service allowing the execution of jobs for researchers, where the submission of jobs is limited by a quota based on CPU consumption for each organization in the Grid. The domain description for this example can be represented using a CIM-based ontology and it may be composed by a role that represent the set of researchers (`#Researcher`), a sample organization (`#OrgA`), the job submission service (`#JobSubmissionService`), the quota assigned to the organization (`#OrgAJobQuota`), the privilege to access the service (`#SubmitJob`) and finally the subject (`#JohnDoe`). Listing 3.1 shows a fragment of this domain description using RDF/XML [79] syntax, which is the primary exchange format for OWL 2 ontologies:

```
<rdf:RDF>
...
<Role rdf:about="#Researcher">
  <name>Researcher</name>
...
</Role>
<Organization rdf:about="#OrgA">
  <name>Organization A</name>
...
</Organization>
<Person rdf:about="#JohnDoe">
  <name>John</name>
  <surname>Doe</surname>
  <orgStructure rdf:resource="#OrgA"/>
...
</Person>
<Identity rdf:about="#JohnId">
  <instanceID>John Doe</instanceID>
  <memberOfCollection rdf:resource="#Researcher"/>
  <assignedIdentity rdf:resource="#JohnDoe"/>
</Identity>
<BatchService rdf:about="#JobSubmissionService">
  <name>Job Submission Service</name>
...
</BatchService>
<ServiceAffectsElementWithQuota rdf:about="#OrgAJobQuota">
```

```

<affectedElement rdf:resource="#OrgA"/>
<affectingElement rdf:resource="#JobSubmissionService"/>
<quota>65</quota>
<quotaUnits>Percent</quotaUnits>
<quotaUsed>24</quotaUsed>
...
</ServiceAffectsElementWithQuota>
<AuthorizedPrivilege rdf:about="#SubmitJob">
  <instanceID>jobSubmission:submitJob</instanceID>
  <authorizedTarget rdf:resource="#JobSubmissionService"/>
  ...
</AuthorizedPrivilege>
</rdf:RDF>

```

Listing 3.1: Sample domain description

In this context, the administrator may decide the following authorization policy: *if the subject is a researcher and he belongs to an organization whose CPU consumption quota for job execution is still not exceeded, then he is permitted to submit jobs*. Rule 1 shows the representation of this policy in SWRL abstract syntax.

$$\text{Identity}(?i) \wedge \text{memberOfCollection}(?i, ?r) \wedge \quad (3.1)$$

$$\text{Role}(?r) \wedge \text{name}(?r, \text{'Researcher'}) \wedge \quad (3.2)$$

$$\text{Organization}(?o) \wedge \text{Person}(?a) \wedge \text{orgStructure}(?a, ?o) \wedge \text{assignedIdentity}(?i, ?a) \wedge \quad (3.3)$$

$$\text{Service}(?s) \wedge \text{name}(?s, \text{'Job Submission Service'}) \wedge \quad (3.4)$$

$$\text{ServiceAffectsElementWithQuota}(?q) \wedge \text{affectingElement}(?q, ?s) \wedge \text{affectedElement}(?q, ?o) \wedge \quad (3.5)$$

$$\text{quota}(?q, ?v) \wedge \text{quotaUsed}(?q, ?u) \wedge \text{swrlb:lessThan}(?u, ?v) \wedge \quad (3.6)$$

$$\text{AuthorizedPrivilege}(?p) \wedge \text{authorizedTarget}(?p, ?s) \quad (3.7)$$

$$\rightarrow \quad (3.8)$$

$$\text{authorizedSubject}(?p, ?i) \quad (3.9)$$

Rule 1: Job submission for researchers

The first two lines of Rule 1 define an identity which belongs to the researcher role. Line 3.3 obtains the person represented by the `Identity` and its organization, which should not have exceeded its quota, as specified in lines 3.5 and 3.6. In this example the privilege is already defined for accessing the job submission service and the rule associates the privilege with subjects to grant them access to the service. This way, privilege instances may be reused for different subjects. Line 3.7 defines the privilege which allows accessing the job submission service and the rule consequent establishes the `authorizedSubject` association to grant the privilege to the identity which represents the researcher.

This rule will be applied to the domain definition in Listing 3.1 during the reasoning process described in Section 3.3.2, inferring new knowledge which grants the privilege

to submit a job to the researcher John. Listing 3.2 shows the OWL representation of the inferred knowledge in RDF/XML syntax.

```
<AuthorizedPrivilege rdf:about="#SubmitJob">
  <instanceID>jobSubmission:submitJob</instanceID>
  <authorizedTarget rdf:resource="#JobSubmissionService"/>
  <authorizedSubject rdf:resource="#JohnId"/>
  ...
</AuthorizedPrivilege>
```

Listing 3.2: Inferred authorization

There are several tools for supporting the definition of domain applications in OWL ontologies. For example, Protégé [80], CO-ODE [81] and SWedt [82] are editing tools to ease the description of ontologies under the Semantic Web languages. These tools successfully deal with OWL and SWRL languages. However, this kind of tools does not offer support for debugging processes when a policy is being inserted into the system. The debugging process during the designing of policies is crucial in order to check the correct behaviour of new policies. Thus, our solution uses the Ontology Rule Editor (ORE) [83], which is a stand-alone application created in our research group and aimed to graphically edit, test, debug and validate ontology rules in any domain. ORE contains a wizard to guide the administrator when editing rules and interpreting the reasoning results. It also incorporates a reasoner which allows debugging the high-level policy definition. It aids the administrator to test a given set of rules before they are applied to the actual system. If some of these policy rules are not behaving as expected, the administrator has the possibility to change the definition before the final policy is enforced.

3.3.1 Virtual Organization Management

A *Virtual Organization* (VO) is set up among different individuals and/or organizations which share a set of resources in a coordinated manner to reach a common goal. A VO is composed of several independent autonomous domains, resulting in an heterogeneous environment. The authorization architecture should be able to deal with this heterogeneity since such a resource sharing should be managed and controlled in order to establish who is allowed to perform what actions.

Usually, each resource provider may want to define its own authorization policy to specify the access to its resources. But also the VO itself may define some common policy which govern the behaviour of the resource sharing. This situation leads to multiple authorization policy specifications provided by different organizations which may also be defined in different terms due to the differences between the organization domains. For example, an organization may define their users using the concept *Researcher* and define its policies based on that concept. On the other hand, another organization may define its policies based on the concept *Scientist* to refer to its users. If both organizations are part of a Grid system, a simple join of their policies may not work, even though the concepts

Scientist and *Researcher* have the same meaning in both organizations (i.e., an user). This is because of the usage of different syntactic representations which will be considered as different concepts by the system.

Using a semantic-aware approach, each organization (as well as the VO itself) will represent its policies by means of an ontology, making use of its own authorization terms and concepts which are meaningful in its domain. Then, a concept alignment can be established in order to relate concepts between different domain ontologies. This alignment can be done using the constructors provided by OWL and SWRL rules. Taking up the previous example again, the concepts *Scientist* and *Researcher* may be aligned by means of the *equivalentClass* construct provided by OWL - i.e. specifying *equivalentClass(Scientist, Researcher)*.

Nevertheless, not all the organizations of the Grid should know about the rest and provide a mapping for all domains. The VO is established in order to solve a need by sharing resources and it may define its own ontology to represent the particular concepts which are significant in its domain, i.e. relevant for a successful resource sharing among the different organizations. This ontology is agreed between the participating organizations of the Grid and can be used as reference point for concept alignment. Any organization which wants to join and share resources provides the set of OWL constructs and SWRL rules to relate the concepts in its domain with the ones in the VO domain. Figure 3.7 depicts this approach.

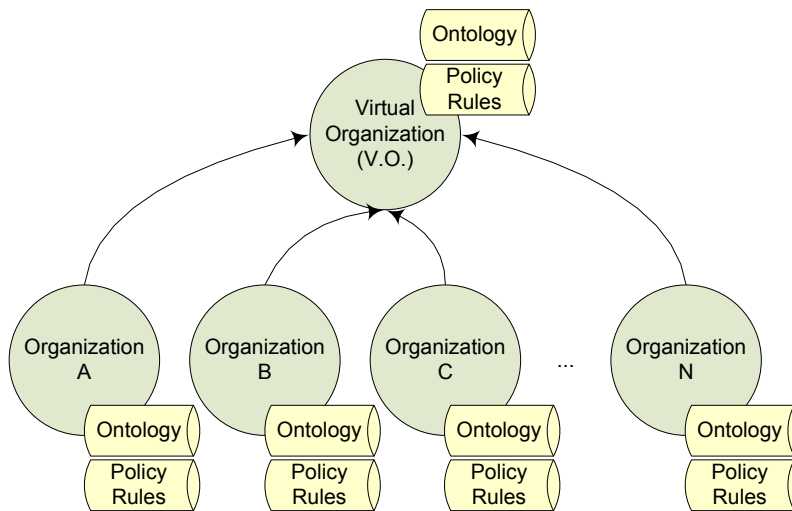


Figure 3.7: Multiple authorization domains

The VO defines authorization policies for resource sharing in its own domain, having no knowledge about the different domains of the particular organizations which take part of the Grid. At the same time, each organization knows the VO domain, but they do not either have to know about the domains of the remaining organizations. The reasoning process will automatically perform concept alignment for taking authorization decisions using all this heterogeneous information.

Using this approach, the VO may, for example, define some policies for users depending on their nationality. It can make use of ontology concepts such as *User* and, for instance, *SpanishUser* to refer to users from Spain. The concepts *Scientist* and *Researcher* may be defined as *User* by specifying *equivalentClass(Scientist, User)* and *equivalentClass(Researcher, User)*. Then, using SWRL rules, the unified concept *User* may be used to define a Spanish user as any user whose country is Spain. The rule may be something like the one specified in Rule 2.

$$User(?u) \wedge country(?u, 'Spain') \quad (3.1)$$

$$\rightarrow \quad (3.2)$$

$$SpanishUser(?u) \quad (3.3)$$

Rule 2: Definition of a Spanish user

Thus, policies which apply to users from Spain will apply to both researchers and scientist of that country, while the organizations that defined those concepts do not need to have knowledge of each other. They only have to know about the *User* concept defined by the VO.

This concept alignment enables support for legacy systems. The definition of the concepts to be used can be driven by the ones which are used by the current system. The semantic approach will allow the interoperation of the proposed architecture with these concepts. Concretely, an organization with a legacy system should define an ontology which enables the representation of the concepts employed in that system. For instance, if they use an attribute-based access control system which provides access according to some user attributes, the *User* concept may be defined, together with the attribute types. Then, current instances of those concepts present in the system should be added to the new ontology. In the previous example, the users which are defined in the system with their corresponding attributes. This conversion process can be automated and it can be quite straightforward if the ontology concepts directly represent the ones used by the legacy system. Once the ontology is defined, policies can be specified by means of SWRL rules. These policies can be directly derived from the ones present in the legacy system, since they can be specified using the legacy system concepts which are represented in the ontology. For instance, if there is a policy establishing that users with attribute *department* = '*deptA*' have access to certain resource, a rule can be defined using the *User* and *department* concepts. Finally, if the legacy system already uses its own concepts to define privileges, the same concepts can be also used with our approach. In this case, OWL constructors and SWRL rules can be used to align these concepts with the ones in Figure 3.6 for the authorization service to interpret the privileges.

For the concept alignment to be achieved, a joint of the information provided by organizations becomes necessary. In our authorization architecture, information of different organizations is kept separate for privacy purposes by means of the WSRF specifications.

However, each participating organization is also able to specify a set of other organizations which are trusted by it and to which its information can be safely shared. This serves to model a trust relationship among organizations, meaning that an organization of the Grid trusts on the knowledge provided by another one. This relationship is used by the `TrustManager` upon a request to only use the knowledge of those organizations which trust on each other when taking the authorization decision. Trust on the VO is assumed since the VO arises from an agreement between the participating organizations to form the Grid and concept alignment with the VO should be performed to take its authorization policy into account. The rest of organizations are considered untrusted unless explicitly defined by this relationship. Moreover, for the knowledge of two organizations A and B to be joint, they should trust on each other, i.e., A should specify its trust on B and B should specify its trust on A.

The `TrustManager` is in charge of providing the information to be used during reasoning for a given authorization request. The information managed by the reasoner is kept in what is called a knowledge base (or KB for short), which is a data structure in which both the original ontology and the new inferred information is stored. A different KB is maintained for each organization, containing its own knowledge together with the knowledge of its trusted organizations. The `TrustManager` keeps these KBs up to date by updating them when a trusted organization changes its domain information. Then, since each resource provider defines the access to its resources, the reasoning process is performed using the KB of the organization to which the request target belongs.

It should be noticed that at this time the trust relationship is not transitive, i.e., if organization A trusts on B and B trusts on C, it does not imply that A trusts on C. Management of transitivity may become a complex task and it is still not supported. Although the authorization architecture has been designed with trust management support and a first approach is provided, the development of a complete solution including trust management will be further investigated and presented in Section 4.4.

3.3.2 The reasoning process

Authorization decisions are taken during a reasoning process which is performed by an OWL and SWRL reasoner in the `SemanticAuthzService`. The main operations for this reasoner are:

- **Inference.** New knowledge about the domain is inferred using the information which is available in the ontology. An inference process over the SWRL rules that represent the authorization policy derives new individuals or OWL instances that are used to make the authorization decision. An example of inference for authorization can be found in Section 3.3, when the policy rule shown in Rule 1 is applied to the domain defined in Listing 3.1. The knowledge that is inferred is shown in Listing 3.2.

- **Validation.** The OWL language allows constraints to be expressed; the validation operation is used to detect when such constraints are violated by some data set. In other words, the validation consists in a global check across the schema and instance data looking for inconsistencies.
- **Querying the ontology,** including instance recognition and inheritance recognition. The former consists in testing if an individual is an instance of a class expression and the latter in testing if a class is a subclass of another class or if a property is a sub-property of another. This enables the formulation of generic queries referring to abstract concepts, being the system able to recognize instances belonging to concrete subclasses or sub-properties of such a concept. For instance, let us suppose that the concept *Entity* is defined in the domain shown in Listing 3.1 as a superclass of *Person* and *Service* to group these concepts in order to refer them as entities which can make use of other services or resources. Thus, if a query is defined to look for entities, John Doe will be returned, since he is a *Person* (instance recognition) which, at the same time, inherits from *Entity* (inheritance recognition).

Figure 3.8 depicts the elements which take part during the reasoning process. The domain knowledge and constraints are specified in form of OWL ontology. Different ontologies can be used and constructs and rules for concept alignment can be specified as exposed in Section 3.3.1. Two different kinds of rules can be also distinguished: policies and meta-policies. The former represent the authorization policies defined by the administrators, like the one shown in Rule 1. These rules guide the inference during the reasoning process to make a positive or negative authorization decision. Meta-policies are policies which provide additional information for authorization. These rules can be used to define advanced semantics of the protected resources which need to go beyond what can be specified by means of OWL constructs, as the one specified in Rule 2 of Section 3.3.1. This is an added value which provides an extensible mechanism for describing the semantics of the domain model, as well as enhanced constraints which can be used for conflict detection and resolution. Details about the usage of meta-policies for conflict detection, including an example, can be found in Section 3.3.3.

During reasoning, the reasoner uses the knowledge base which contains the OWL ontology together with the policies and meta-policies defined in form of SWRL rules. Two kinds of reasoning can be identified: OWL reasoning and SWRL reasoning. The former implies the aforementioned operations of querying, validation and inference on the ontology, the latter consists in performing inference with SWRL rules. The inference process (about both the OWL ontology and the SWRL rules) generates new knowledge which is used to update the KB. Validation is also performed, checking the ontology for inconsistencies or constraint violations. Finally, when the *AuthzEngine* queries the reasoner for an authorization decision, this is taken based on the presence of certain instances and associations on the KB. Those instances represent an authorization definition according to the CIM-based model depicted in Figure 3.6 and they may be present due to statical definitions provided by the administrator or they may be inferred during reasoning.

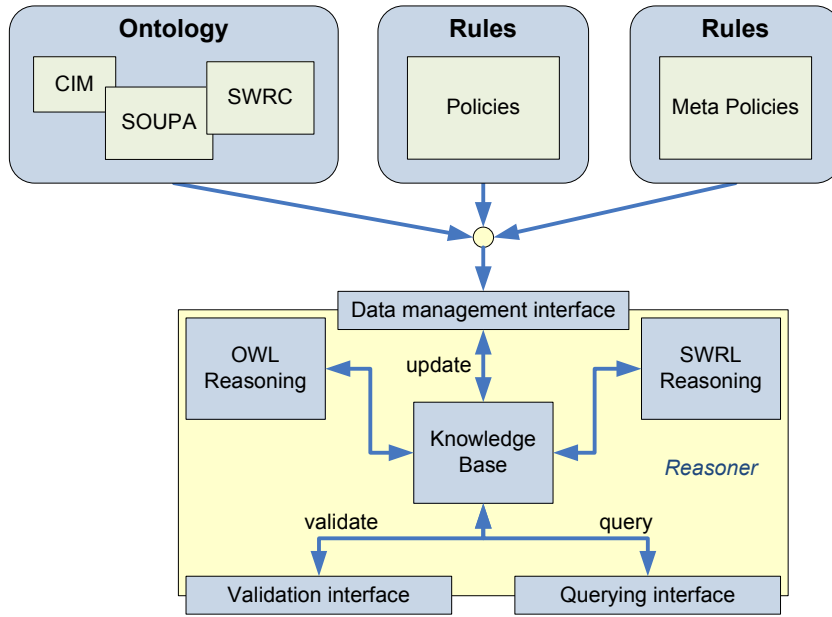


Figure 3.8: Ontology and rule reasoning

Listing 3.3 shows the authorization query in abstract syntax which gets the privilege allowing a subject to access to a given target. Namely, it explores the KB looking for an `AuthorizedPrivilege` instance which is associated with the given subject and target. The query has two parameters: the subject or identity which is trying to access; and the target representing the Grid resource which is being accessed. If the query does not return any privilege instance, the `AuthzEngine` denies the subject accessing the target (deny by default).

```
PARAMS (subject, target)
SELECT(?p)
WHERE AuthorizedPrivilege(?p) ∧
Identity(?s) ∧ assignedIdentity(?s, #subject) ∧
authorizedSubject(?p, ?s) ∧ authorizedTarget(?p, #target)
```

Listing 3.3: Authorization query

For this reasoning process to be efficient, an incremental approach is used. Some OWL reasoners like Pellet [84] support incremental consistency. This functionality enables the reasoner to just take into account the new knowledge which may have been added since the last reasoning performed with the same KB. In our approach, a KB is maintained for each organization with its knowledge and the one of its trusted organizations. This KB is updated on every request and the new knowledge inferred during reasoning is also kept in it. Since this KB remains the same for different requests, incremental consistency can be used to make the reasoning process faster. If the organization

domain and the authorization policy is not changed, the new knowledge between reasoning processes is in many cases only the information provided by the PIPs for the current request. For initial requests, this information may be unknown for the system, but when the authorization system has been up for some time, the KB will probably contain information about most of the system and few new knowledge will really be included as new for reasoning. Moreover, the reasoner is also able to detect knowledge which is already present in the KB. Thus, if the information provided by a PIP is already in the KB (probably provided by another PIP from a previous request), it will not be validated again, since it was validated in a previous consistency checking process.

3.3.3 Policy conflict analysis

An important functionality of the reasoner is that it enables the detection and resolution of conflicts [85]. Conflicts raised by evaluating policy rules may be categorized into two main types, depending on whether the conflict depends on the application domain or not. In the first case, the conflicts obtained are called semantic conflicts, because they are generated by using information related to the current state of the system. They are really difficult to detect, and their appearance is conditioned to the dynamic state of the application domain. In the second case, there are syntactic conflicts as they can be detected by simply looking at the rules structure. This second type of conflict occurs irrespective of the state of the particular application domain and may be the result of specification errors in the policy or they may also be legitimately derived from other rules. Moreover, the heterogeneous environment of a Grid composed by different organizations may lead to the rising of policy conflicts. Conflicts may appear between different organization policies or even between the policies of an organization and the policies of the VO.

Focusing on authorization policies, modality conflicts are syntactic conflicts that can be detected from the rule syntax that defines these policies. A modality conflict occurs when two or more policies with modalities of opposite sign refer to the same subjects, activities and targets. An example of this type of conflict is observed when two different policies about a subject (e.g. John) forbid and permit to access the job submission service at the same time. Thanks to the automatic generation performed by the reasoner, the system is able to identify a policy conflict when two disjoint properties appear simultaneously (i.e. inconsistency). Several techniques may be applied to make this inconsistency for conflict detection. As a first approach, our proposal defines an `unauthorizedSubject` association which is declared as disjoint with `authorizedSubject`, using the *disjoint-DataProperties* construct provided by OWL. The inconsistency appears when, due to some policy, the reasoning process infers an instance `unauthorizedSubject` between John and the job submission privilege (meaning that John is not authorized to submit a job) and, at the same time, due to some other conflicting policy, the rule reasoner also infers a new instance of `authorizedSubject` between them (meaning that John is authorized to submit a job). The inconsistency is detected by the reasoner during validation because an individual cannot contain to two disjoint properties simultaneously. Listing 3.4

shows the inconsistency in OWL using RDF/XML syntax.

```
<AuthorizedPrivilege rdf:about="#SubmitJob">
  <instanceID>jobSubmission:submitJob</instanceID>
  <authorizedTarget rdf:resource="#JobSubmissionService"/>
  <authorizedSubject rdf:resource="#JohnId"/>
  <unauthorizedSubject rdf:resource="#JohnId"/>
  ...
</AuthorizedPrivilege>
```

Listing 3.4: Privilege with inconsistency

On the other hand, semantic conflicts are domain dependent and their detection is based on some meta-policies describing undesirable situations which may appear in the domain, in order to detect them and avoid their occurrence. An example is observed when an user can change its own privileges, being able to use it on his own benefit to perform actions which were initially forbidden for him. For instance, if a subject (e.g. John) is able to change the job submission quota, he may be able to submit a job exceeding the allowed amount of CPU consumption. A meta-policy may be defined to detect such a situation, forbidding the subject to submit the job. Rule 3 shows the representation of this meta-policy.

$$Identity(?i) \wedge \quad (3.1)$$

$$AuthorizedPrivilege(\#SubmitJob) \wedge authorizedSubject(\#SubmitJob, ?i) \wedge \quad (3.2)$$

$$AuthorizedPrivilege(\#ChangeQuota) \wedge authorizedSubject(\#ChangeQuota, ?i) \quad (3.3)$$

$$\rightarrow \quad (3.4)$$

$$unauthorizedSubject(\#SubmitJob, ?i) \quad (3.5)$$

Rule 3: Job submission and quota modification

Once the conflict has been detected, the rule consequent states that the subject is not authorized to submit the job by establishing the `unauthorizedSubject` association between the privilege and the subject. This property will cause an inconsistency in the knowledge base with the already existent `authorizedSubject` property, since they are defined as disjoint properties.

Finally, conflict resolution is necessary to provide an automatic solution for detected conflicts. Prioritization of policies is a suitable technique for resolving the authorization conflicts [85][86]. Conflicts may appear between policies defined by different organizations. In this case, assigning a priority to each organization may resolve the conflict. For instance, the VO may be defined with a higher priority and, thus, its policy should apply in case of conflict. Conflicts between policies of the same organization should also be resolved prioritizing the rules or even prioritizing the authorization decision which should be taken. For instance, specifying that negative policies have priority over positive ones

(deny by default). Once detected, a decision can be made to resolve the conflict, taking the decision with higher priority.

It should be noticed that this section has introduced how both semantic and syntactic conflict detection and resolution are supported by the proposed authorization approach. However, conflict detection and resolution are quite complex tasks, including conflict analysis and source of conflict discovery. For more information and details regarding semantic conflict detection research refer to [23].

3.4 Implementation and statistics

This section provides some details about the proof of concept implementation which has been developed to evaluate feasibility and performance of the proposal. Some statistical results achieved are also exposed herein. Thus, as will be shown later, these statistics deal with the evaluation of the average authorization response time when a user tries to access to a resource in the Grid, as well as with the evaluation of the reasoner behaviour in charge of taking such a decision.

With regards to the implementation used to perform the testbed, it is worth taking into account some considerations. Firstly, the components `SemanticAuthzService` and the `WS-ResourceFactory` described in Section 3.2 have been developed as Globus Web services making use of `Introduce` [87] as a tool for aiding on Web services develop affairs.

Moreover, in order to abstract the `SemanticAuthzService` from the KB management, a Java library has been developed. This library defines a simple interface which allows the service to manage the organization ontologies, as well as to perform the authorization query in charge of making the authorization decision. The query is executed using SPARQL [88], the query language fostered under the Semantic Web technologies.

Currently, there are several suitable implementations of DL reasoners, i.e., reasoners using Description logics, such as Pellet [84], Jena [89], KAON [90] and the one proposed by FaCt++ [91], among others. Our Java implementation makes use of Jena as a general Java API to manage ontologies and Pellet as DL reasoner. Namely, it uses Jena version 2.6.2 and Pellet in its version 2.0.0. The main reason by which Pellet has been chosen as reasoner is due to the fact that it supports high expressiveness dealing with OWL 2 ontologies and it is also able to perform incremental consistency checking. Likewise, Jena is nowadays the standard de facto Java library to manage ontologies.

Statistics have been achieved making use of virtualization technology. An Intel Xeon machine with a 4 core processor at 2GHz holding 4 Gb of RAM has been used as host to install a VMware Server, in its version 2.0.2. The VMware virtual machine where Globus Toolkit 4.0 has been installed to execute the testbed has been configured with 3 Gb of RAM, using Ubuntu 9.10 as Operating System. Besides, the Globus Toolkit Java WS Core container has been customized to 2.5 Gb of RAM to hold the services.

3.4.1 Reasoner performance

An important part of our architecture which widely affects the performance is the reasoner in charge of making the authorization decision which is taken based on ontology instances present on the KB. In order to check the performance and behaviour of such a reasoner when it is under different work loads, several tests have been done using different amount of individuals, i.e. OWL instances, present in the KB.

A way of measuring the performance of the proposal is making different complexity executions. These executions are performed using different sets of work loads in the KB which make the test to become more and more complex. Thereby, the complexity is achieved by introducing new knowledge in the ontology. An ontology can be divided into *TBox* (terminological) and *ABox* (assertional) components. The former contains the schemas and vocabularies that define domain concepts, whereas the latter is populated with instances of these concepts. Thus, the complexity of executions is incremented by increasing the number of individuals present in the *ABox* component of the ontology what, in turn, increases the number of statements hold in the knowledge base. In order to add new instances to the KB, the set of operations provided by the `SemanticAuthzService` to manage the ontology are used as exposed in Section 3.2.1.

The number of different kinds of individuals contained in the *ABox* for an specific execution is referred as population. Thus, a population represents the knowledge provided by an organization at a given moment. The addition of new instances leads the KB to reach the next population state, making it necessary to perform the KB consistency checking again. It should be noticed that the number of axioms or statements that are hold in the *ABox* differs from the number of individuals which are present in the ontology, since more than one axiom is usually necessary to represent one individual.

As stated in Section 3.3, the CIM to OWL representation proposed by [64] has been used as ontology to model the domain concepts in this implementation. Each population is composed by individuals which are represented by instances of different OWL classes defined in the model described in Figure 3.6, i.e., `CIM_Identity`, `CIM_Role`, `CIM_AuthorizedPrivilege` and `CIM_ManagedElement`. Figure 3.9 depicts the percentages of every kind of individual which have been used to make up each population of the testbed. These percentages have been selected trying to get an individual distribution as closest as possible to the one which may be present in a real system. Thus, the individuals for a given population have been randomly generated, but in a driven way in order to achieve the desired distribution for the testbed to better fit a real scenario.

Another issue to be considered is the way the populations sizes are established since this size will be used to evaluate the reasoner scalability. In this sense, it is very common to make use of an exponential function to establish the size of these populations. It should be notice that the individuals percentage distribution described above is maintained for each population regardless its size.

The testbed makes use of 10 incremental populations. The following function, which

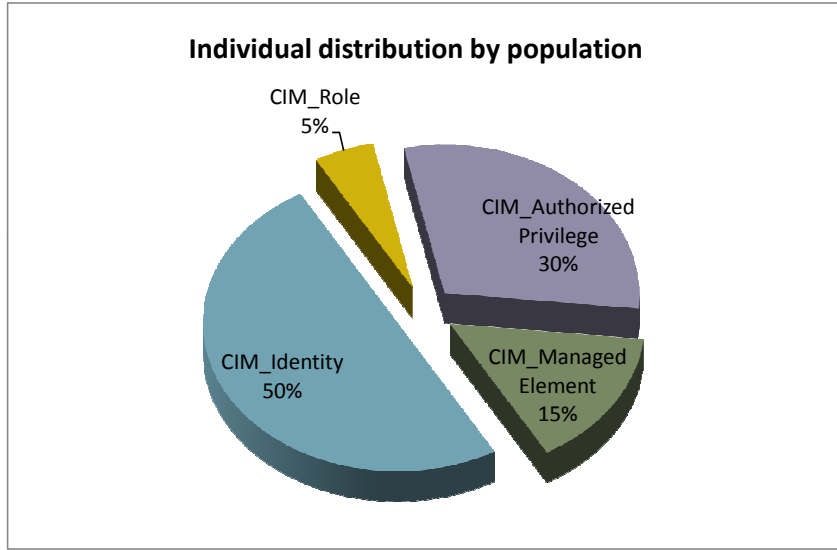


Figure 3.9: Individuals distribution for populations

determines these ten populations size, has been empirically achieved, starting on the basis that, for our testbed, the biggest population must be the one which shows unrealistic performance results. Thus, the biggest population in this testbed has about 27000 individuals, since for this amount the reasoning process needs more than 10 seconds to perform the reasoning. It could be considered that 10 seconds added to other additional times required to give an authorization response such as the spend by Globus or the required by the authorization query, is not an affordable response time for an authorization system.

$$f(x) = 300 \cdot e^{(x/2)}$$

The variable x takes integer values from the interval $[0, 9]$. Table 3.1 shows the relationship between the number of individuals for each population used in the testbed and the corresponding number of statements representing such individuals. The procedure to obtain the statements is not shown because it does not follow a mathematical expression, thereby, the number of statements are consequence of the amount of axioms required by the reasoner to represent the individuals and its properties.

As stated before, these populations are generated with individual distributions trying to represent realistic scenarios. For instance, according to Figure 3.9, the population number five which has 3700 individuals, could represent a scenario where a Institute of Technology with 1500 researchers and 350 professors (50%) have rights to access to 555 different managed elements (15%), such as, PCs, services, resources, etc.

These populations will be used to quantify the time that our authorization service takes to check the Knowledge Base consistency and perform the authorization query on the KB. It should be noticed that the reasoning time is determined not only by the number

Population	0	1	2	3	4
Individuals	300	500	900	1,400	2,300
Statements	1,500	2,800	4,700	8,300	14,100
Population	5	6	7	8	9
Individuals	3,700	6,100	10,000	16,400	27,100
Statements	25,200	46,200	86,200	167,500	336,000

Table 3.1: Number of individuals and statements by population

of statements hold in the ontology, but also by the *TBox* complexity itself. However, the schema and vocabularies that define the domain concepts, making up the *TBox*, are the ones present in [64] ontology and they remain the same for all the executions. Thus, this complexity is not explicitly shown in statistics since the *Tbox* is not changed and this time component is constant.

As exposed in Section 3.3.2, for each authorization request processing, the reasoner performs the validation and query operations on the KB. Figure 3.10 depicts the Pellet reasoner performance for both the time it spends making the validation or consistency checking process and the time spent carrying out the SPARQL authorization query once the KB is valid. This query searches instances in the KB which may indicate a positive (permit) or negative (deny) authorization for the requested Grid resource, as stated in Section 3.3.2. Moreover, the statements hold in the KB are added incrementally, i.e., populations grow by adding new individuals, keeping the ones present in the previous population what emulates an organization in the grid system adding new information to its existing domain model.

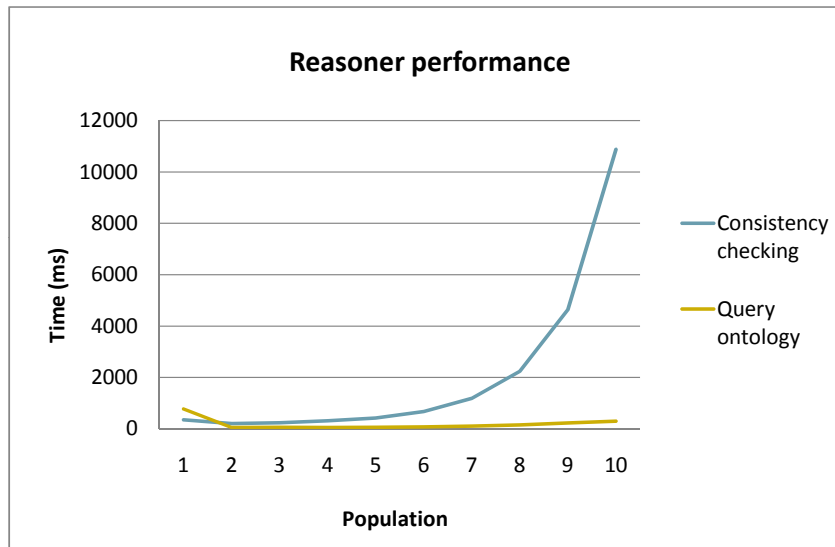


Figure 3.10: Reasoner performance

As shown in Figure 3.10, the greater number of statements the KB holds the more time it takes for the reasoner to perform the consistency checking task. However, except for the first query, the time the reasoner needs to query the KB grows quite slower compared with the time spent in the consistency checking. The first query takes more time since the reasoner has to perform the *Classification* reasoning process which computes the *subClassOf* and *equivalentClass* inferences between all ontology named classes and properties. This time appears just in the first execution, since the *TBox* is not changed as stated before and it remains the same for the rest of queries.

3.4.2 Authorization request testbed description

Different simulations have been carried out with the aim of evaluating the average time needed to get an authorization response from our authorization architecture in Globus Toolkit. The test is performed from the moment when a user tries to access to a resource shared in the Grid, until the authorization response is given. It makes one or more requests evaluating the KB when it is under incremental work loads. The overall time has been split in the test in four different times in order to be analysed and understood properly. Figure 3.11 depicts these four times and the main tasks performed during such periods.

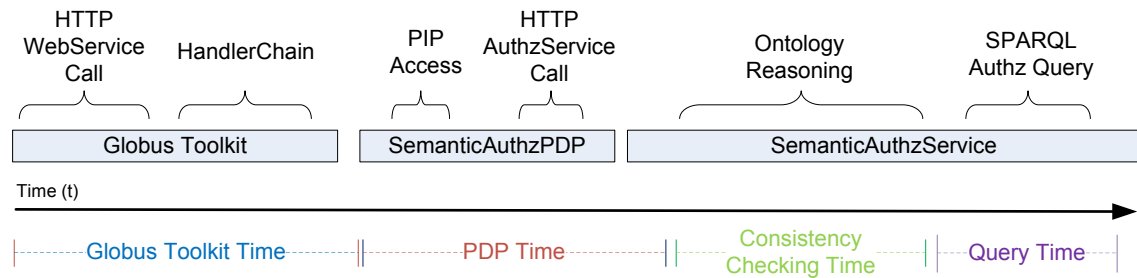


Figure 3.11: Testbed times description

The first time, referred as *Globus Toolkit Time*, deals with the period spent by the Globus Toolkit to process a user request and redirect it to our *SemanticAuthzPDP*. A user registered in the Grid, with his own certificate, tries to access to a dummy Grid Web service called *HelloWorldService*. In the testbed, the user has been previously authorized to access to this Web service, and therefore, the ontology individuals which grant the user to access the Web service are included in the KB.

This time includes the elapsed time introduced by the Toolkit for authenticating and securing the HTTP communication. Hence, the time required to establish the context between the client and the server by means of GSI Secure Conversation mechanism is also included. When the HTTP message arrives from the client, as stated in Section 3.2, the Globus SOAP engine invokes several security related handlers in what it is called handler chain. Thus, the *WS-Security handler* searches the message for any WS-Security headers extracting from them any keying material populating a peer subject object with princi-

pals. Next, the *Security policy handler* checks that incoming messages fulfil any security requirements the service may have. Afterwards, the *Authorization handler* verifies that the principal established by the *WS-Security handler* is authorized to invoke the service.

At this point, since the `HelloWorldService` has been previously configured in the security descriptor to make use of our `SemanticAuthzPDP`, the Globus authorization request reaches our PDP implementation. It is important to remark that the *Globus Toolkit Time* presented in this testbed is out of our implementation. However, it is taken in order to evaluate the whole time required to obtain an authorization request response in the Globus Toolkit when making use of our proposal.

The period of time depicted as *PDP Time* in Figure 3.11 represents the time spent by our `SemanticAuthzPDP`. This component implements the Globus Toolkit interface `org.globus.security.authorization.PDP` and behaves like a gateway that redirects the Globus authorization request to the `SemanticAuthzService` through an HTTP call. This connection is also secured with message level security by means of the GSI Secure Conversation mechanism explained above. It is also worth mentioning that, in this testbed, the *PDP Time* does not include the time spent by third organizations PIP implementations needed to obtain attributes related to the user request. This time does not depend on our implementation since each organization should provide its own PIPs and therefore the time is quite implementation specific.

The elapsed time introduced by the `SemanticAuthzService` can be mainly split into two periods. On one hand, the time spent by the reasoner to check the consistency of the KB, shown as *Consistency Checking Time* in Figure 3.11. On the other hand, the time spent carrying out the SPARQL authorization query, depicted as *Query Time* in the figure.

Listing 3.5 shows the SPARQL query that has been used in our proof of concept implementation which is a realization of the abstract one exposed on Section 3.3.2. As can be seen, the query is self-configurable and looks for any *CIM_AuthorizedPrivilege* individual in the KB which allows a subject to access to a given target. In this case, the target is a Web service modelled by the CIM class *CIM_Service*. Moreover, since Globus Toolkit identifies the subject by means of certificates, and therefore, such certificates information is available in our implementation, the SPARQL query identifies the subject with the CIM property *CIM_PublicKeyCertificate__Subject*. The *CIM_PublicKeyCertificate* is, at the same time, assigned to the *CIM_Identity* which represents this subject in the system.

```
PREFIX cim: <http://gom.kwfgrid.net/ontology/cim/2006/10/02/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?privilege
WHERE {
    ?certificate rdf:type cim:CIM_PublicKeyCertificate .
    ?certificate cim:CIM_PublicKeyCertificate__Subject ' subject ' .
    ?identity rdf:type cim:CIM_Identity .
    ?identity cim:associationCIM_AssignedIdentity ?certificate .
    ?privilege rdf:type cim:CIM_AuthorizedPrivilege .
    ?privilege cim:associationCIM_AuthorizedSubject ?identity .
```

```

?service rdf:type cim:CIM_Service .
?privilege cim:associationCIM_AuthorizedTarget ?service .
?service cim:CIM_Service__Name ' target ' .
}

```

Listing 3.5: SPARQL authorization query

In order to set up the SPARQL authorization query, the implementation makes use of the default Globus PIP X509BootstrapPIP. It collects attributes from the X509 user certificate and the Globus Toolkit makes them available for the `SemanticAuthzPDP` and, in turn, for the `SemanticAuthzService` which actually replaces the values in the query and executes it. Therefore, the *subject* variable is replaced by the Distinguished Name (DN), e.g. `"/O=Grid/OU=GlobusTestbed/ OU=simpleCA-john-desktop/CN=John"`. Likewise, the variable *target* is replaced in the testbed by the corresponding identifier. In this case, it is replaced by `"http://org.umu.security.grid.semantic/HelloWorldService"` since it is the name given to the *HelloWorldService* which acts as target in our testbed.

3.4.3 Authorization request performance

It is important to remark that the reasoner always performs the consistency checking for each authorization decision request, but if the KB has not been modified this process does not take any time because the KB is already reasoned and consistent. Therefore, the average authorization time response can be relatively lower in systems characterized by few changes in the KB, where the consistency checking operation takes no time for many consecutive authorization requests. Such static systems could be those which are in production during a long time and therefore the system is relatively steady. On the other hand, an example of a system under frequent changes could be the one where new organizations joins the Grid or leave it very often.

Figures 3.12, 3.13, 3.14 and 3.15 depict four different stats representing four different scenarios that could take place in a Grid, arranged from dynamical to more statical systems. It permits to show the behaviour of the proposal for different scenarios.

Figure 3.12 represents a dynamic system where the KB is changing continuously. In such systems, the reasoner is forced to spend time checking the KB consistency in every authorization request. As it can be seen in the graph, the *Consistency Checking Time* gets a high percentage of the overall time required by a request. Since it is assumed that for each scenario the same user try to access to the service throughout the ten populations, when the request is executed for the first time, the *Globus Time* and the *PDP Time* gain prominence over the whole request time. This is due to the fact that the first time the user makes the HTTP call to the `HelloWorldService` it is necessary to establish the GSI Secure Conversation context. Therefore, it has to configure the security communication context between the user and the Globus Toolkit and also the communication between the `SemanticAuthzPDP` and the `SemanticAuthzService`.

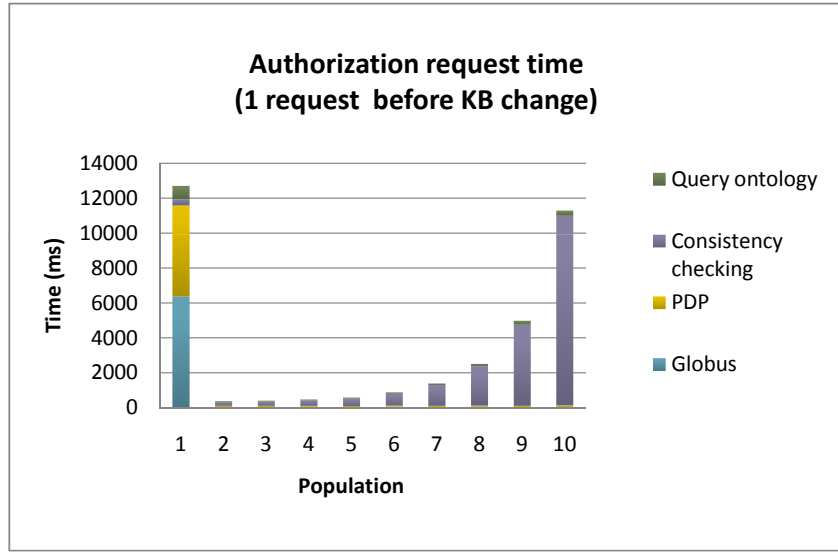


Figure 3.12: Authorization request performance with continuous changes

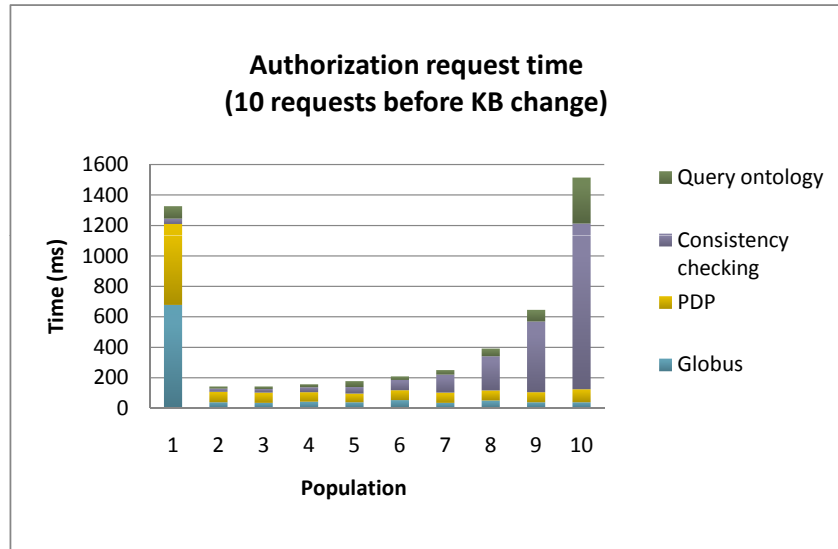


Figure 3.13: Authorization performance with 10 requests before changes

Figure 3.13 denotes a kind of system where 10 authorization requests occur before the KB changes. The results showed in this graph represent the average times for these 10 requests. During 9 of these 10 requests the *Consistency checking Time* is zero, therefore, the average *Consistency checking Time* for these 10 executions is very low. This leads to the reduction of the overall average authorization request time compared with the system situation depicted in the first graph. Moreover, this implies that the *Query Ontology Time* and the *PDP Time* represent a greater percentage of the total request time regarding to the previous system situation. In addition, as expected, the time required to perform the

SPARQL query is greater when the number of statements hold in the KB grows up.

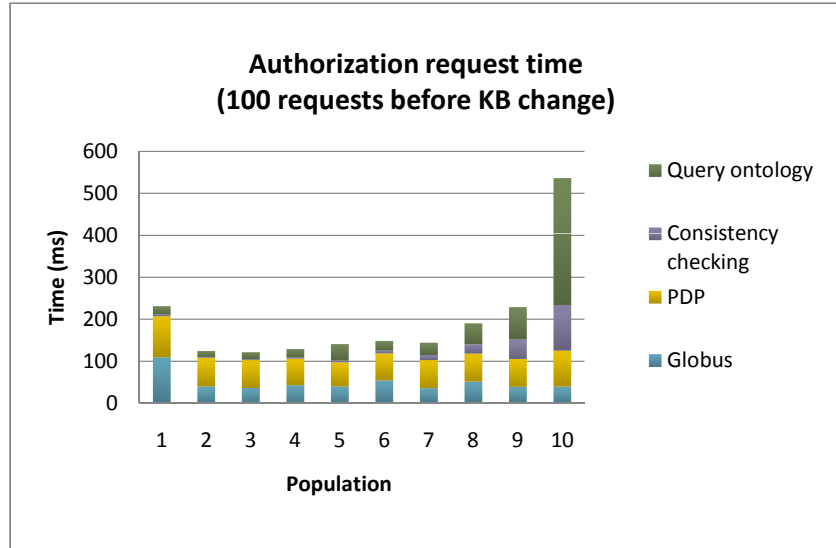


Figure 3.14: Authorization performance with 100 requests before changes

Figure 3.14 corresponds with a common scenario where up to 100 authorization requests can be performed before the KB changes. An scenario like this could be the aforementioned one regarding the Institute of Technology, where there are a great amount of researcher and their rights to access to the resources hardly ever change, making the KB to be fairly steady. A general deduction looking at this graph is that the average *Consistency checking Time* is decreased when the system situation become more static. Moreover, it is also important to bear in mind that, although it is not showed in the graphs, the first SPARQL query done just after the KB changes spends more time than the following ones as explained previously. Additionally, with the exception of the first population, the *Globus Time* and the *PDP Time* are quite stable during all the executions for every population.

Figure 3.15 symbolizes a more static scenario where the KB hardly ever changes. Thus, up to 1000 authorization requests can be done without the need of perform reasoning on the KB. The average time spent checking the consistency when the system is in such a situation is almost irrelevant, comparing with the average time spent on the *Query Time*. Thus, as it can be seen in the graph, the percentage of the *Query Time* is the greatest time in such a system. This last fourth graph shows that the overall average time in a static system like this, is very low compared with the dynamic one shown in the first graph.

To sum up, this section has shown that the reasoning and query times required by our solution to take the authorization decision is quite acceptable when high populations are hold in the KB. Furthermore, the statistics prove that the time required by our solution to give an authorized response compared with the overall time the Globus Toolkit requires to process the authorization request is also quite acceptable. This statement be-

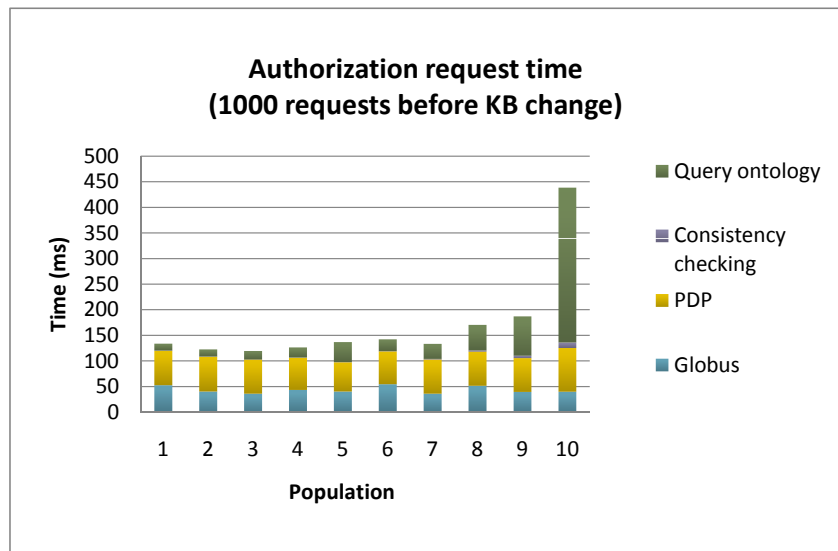


Figure 3.15: Authorization request performance in static scenario

comes especially evident in a static scenario where the KB is stable and our authorization approach reduces the time required to reasoning over the KB, and therefore, to make the authorization decision.

Chapter 4

Semantic-aware Authorization System for Cloud Architectures

This chapter presents a step forward in the improvement and refinement of the authorization model and the proposed architecture exposed in Chapter 3. A formalization of the model based on the RBAC scheme is provided with high expressiveness capabilities such as hierarchical RBAC (hRBAC), conditional RBAC (cRBAC) and hierarchical objects (HO). Both the model and security architecture have been also adapted to Cloud computing, taking into account the particularities of this kind of environments. The authorization model is also based on semantic Web technologies to describe both the underlying infrastructure and the authorization rules. A trust model is also introduced to enable the definition of possible business alliances like federation among cloud tenants. The proposed model has been validated by means of a proof of concept implementation of the access control system for OpenStack with promising performance results.

The research work described in this chapter has been published in the paper entitled *Semantic-aware multi-tenancy authorization system for Cloud architectures* [18].

This chapter is organized as follows. Section 4.1 describes the languages and models that are used to define the underlying system infrastructure. Then, Section 4.2 formalizes the authorization model and describes its capabilities in terms of expressiveness. Afterwards, Section 4.3 delves into the authorization architecture for Cloud computing. The proposed trust model is presented in Section 4.4. The workflow carried out during the authorization process is presented in Section 4.5. Finally, a proof of concept implementation and some performance results are shown in Section 4.6.

4.1 Information model

The proposed access control system uses a model to represent the resources that are managed and protected by the system. As exposed in Section 3.1.1, this model is represented by means of an ontology defined using the Ontology Web Language 2 (OWL 2) [62] and the Semantic Web Rule Language (SWRL) [16]. These two languages are based on formal methods and they constitute a remarkable added value to the proposal since they enable the usage of reasoners to infer new knowledge. The reasoner is able to derive additional information not explicitly specified in the ontology, and to perform a formal validation and verification of the model. The Description Logics formalism in which these languages are based is kept within the decidability bounds, so that inference processes are performed in a finite time.

Several standard models like the Common Information Model (CIM) [1] and the Open Information Model (OIM) [92] have been designed to model information systems. As explained in Section 3.1.2, the Common Information Model (CIM) has been selected as base model for our access control system due to its advantages. Namely, wide coverage of information systems, extensibility mechanisms, independency of the representation language, and widely used in several research works like [93] and [94], as well as in a variety of large systems such as SAP, Microsoft Windows and VMWare, among others. The representation of CIM into OWL results in a semantically enriched information model as exposed in the previous chapter.

For the shake of simplicity, only managed resources at the IaaS layer are exposed in this section but the model is easily extensible to provide support for the PaaS and SaaS layers. Common concepts managed in cloud infrastructures include virtual machines (VMs), virtual networks which interconnect the VMs and volumes attached to the VMs. In conclusion, the model should contain enough concepts and relationships to define the different elements supported by the virtual infrastructure. Figure 4.1 shows a UML representation of the main CIM classes employed in our information model.

Among others, `VirtualComputerSystem` is an important class which allows to manage the virtual machines of the cloud. Thanks to the `installedOS` and `runningOs` associations, the model is able to deal with the operating system of each virtual machine. `VirtualSystemSettingData` defines the aspects of a virtual system through a set of virtualization specific properties (e.g. `SnapshotDataRoot`, `ConfigurationFile`, `AutomaticStartupAction`, `VirtualSystemType`, `SwapFileDataRoot`...).

Each cloud tenant in our model is represented by an instance of the `AdminDomain` class. The class `AutonomousSystem` is used to segregate the system by routing into a set of separately administered domains, having each one its own independent set of policies. `LogicalDevice` is an abstraction or emulation of a hardware entity which enables the management of devices at the IaaS layer. The `StorageVolume` class, which in turn extends `LogicalDevice`, is used for the representation and management of volumes in the Cloud.

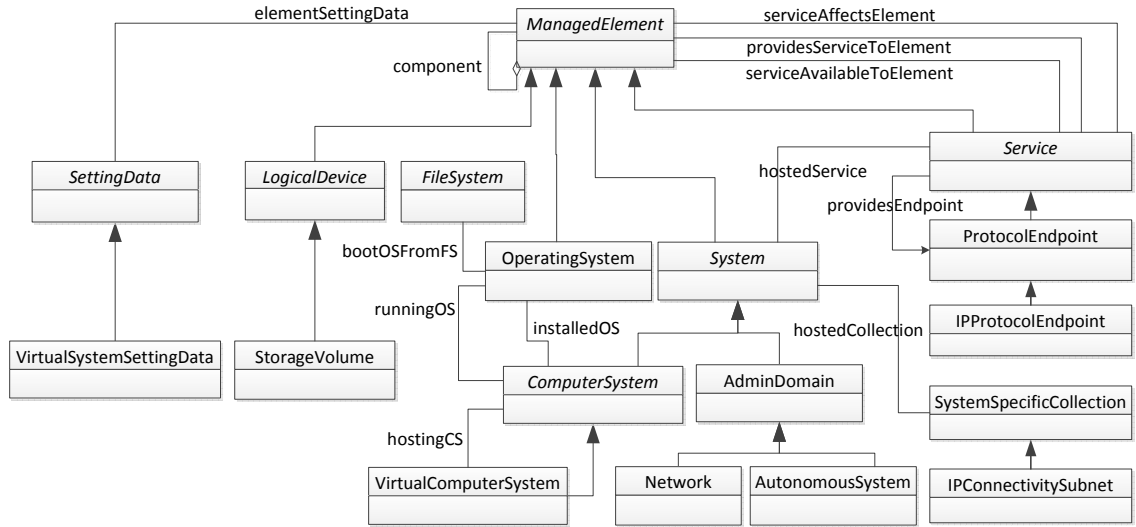


Figure 4.1: UML representation of the information model

4.2 Authorization Model

The access control system must determine whether a given subject is granted with a privilege over a given resource or not. The authorization decisions are based on the definition of authorization statements. An authorization statement is defined by a 3-tuple (*Subject, Privilege, Resource*) which establishes that a subject has a privilege on a resource. When a user attempts to access a Cloud resource, the access control system seeks along the available authorization statements for the requested user (*Identity*), action (*Privilege*) and resource (*ManagedElement*). By default, if there is no matching authorization statement in the model, the access to the resource is denied.

These authorization statements are represented in the ontological authorization model by means of the set of CIM concepts and relationships depicted in Figure 3.6 and explained in Section 3.3. In sum, an *Identity* represents a user that can be validated during the authentication stage, whereas a *Role* represents a set of responsibilities within an organization. Identities can be assigned to a *Role*. *Privilege* is the base concept for all types of activities/actions which are granted or denied to a subject for a given resource. *AuthorizedPrivilege* extends *Privilege* to describe current privileges in the authorization model. In order to increase the readability, *AuthorizedPrivilege* class will be henceforth referred to as *Privilege*, indistinctly.

With the aim of clarifying the authorization capabilities supported by the access control system, Figure 4.2 depicts an example where different cloud tenants share different resources. In this example, a user of tenant A (belonging to the role Lab Head) tries to access resources (in this case to the VMs) of tenant B. Likewise, a Lab Head user of tenant B tries to access to the network of tenant A.

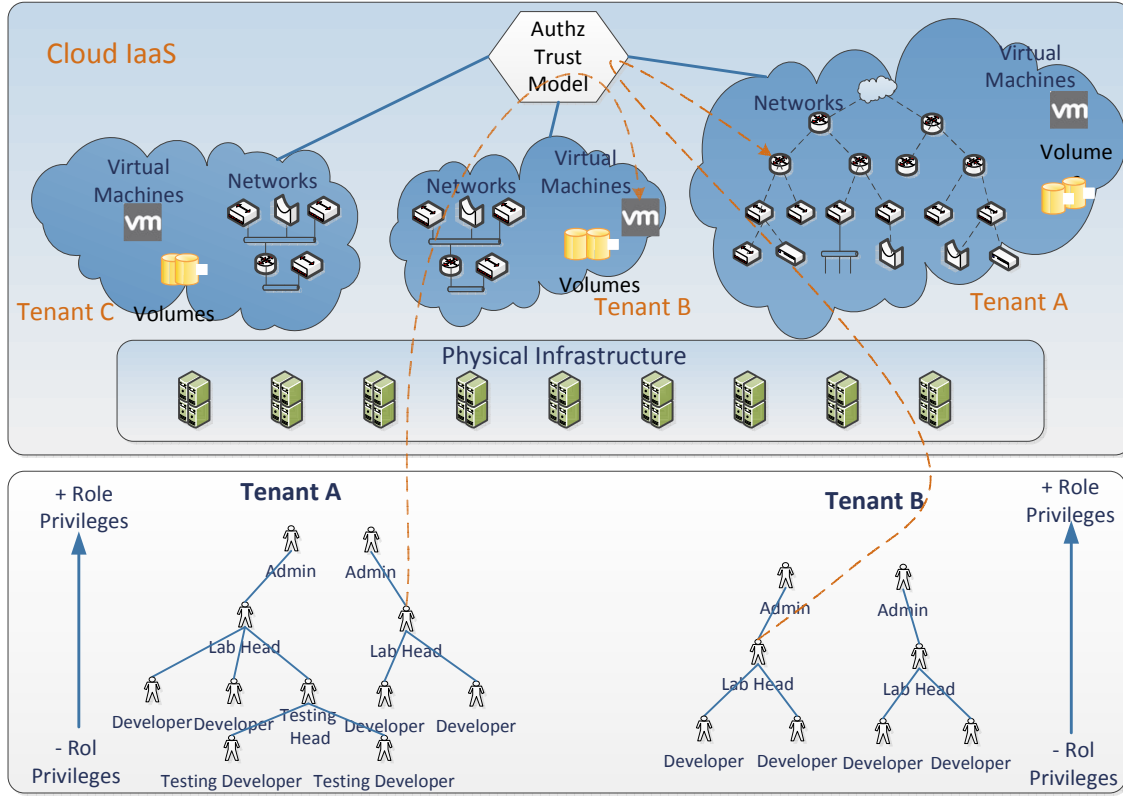


Figure 4.2: Cloud authorization example

To support Role-based Access Control (RBAC) in the authorization model, the component *Subject* defined in the 3-tuple can be either represented by an *Identity* or a *Role*. In Figure 4.2, tenant A describes different roles such as Admin, Lab Head and Developer in order to assign later different privileges to the role as a whole. Role-based Access Control (RBAC) capability is supported in the authorization model by means of a rule (expressed in SWRL language) which propagates the privileges assigned to a role to the identities belonging to that role. This rule is depicted in Rule 4.

$$Role(?r) \wedge AuthorizedPrivilege(?p) \wedge authorizedSubject(?p, ?r) \wedge \quad (4.1)$$

$$Identity(?i) \wedge memberOfCollection(?i, ?r) \quad (4.2)$$

$$\rightarrow \quad (4.3)$$

$$authorizedSubject(?p, ?i) \quad (4.4)$$

Rule 4: Role privileges propagation meta-rule

The rule selects any role having some associated privileges. It also selects any identity which is member of that role (line 4.2). As a result, any privileges granted to the role is also granted to the identities belonging to such role. Notice the power of using this

ontological approach for the authorization model, since just one rule is enough to provide RBAC capabilities to the whole access control system.

Multi-tenancy is an important aspect which characterizes a cloud environment. It implies a new extension to the authorization statements, adding the concept of Issuer. This concept is introduced to represent the tenant who defines the statement. It is required in order to recognize the owner of the statement and therefore to limit the information model that is taken to perform the authorization decision. Thus, authorization statements are now represented by the 4-tuple (*Issuer, Subject, Privilege, Resource*). This tuple can be read as '*The tenant Issuer states that the user or role Subject has the permissions defined by Privilege over the element Resource*'. The *Issuer* is represented using the CIM *AdminDomain* class in the information model.

Conditional RBAC (or Generalized RBAC) is an advanced capability rarely supported by current cloud access control systems. This feature supports the assignment of privileges to environment roles. These environment roles are activated according to the changes specified in environmental conditions defined by the administrators. Thus, the permissions defined in the authorization statements only apply when the specified conditions are fulfilled, which usually refer to context information of any nature. It requires the extension of the current authorization statements to be formed by a 5-tuple (*Issuer, Subject, Privilege, Resource, Conditions*). Note that conditional RBAC includes many previous research work in authorization models such as attribute-based accessing control (aBAC) - using user attributes as conditions - or Generalized spatial time RBAC (GST-RBAC) - using temporal and spatial context information as conditional -, including all the proposals in which authorization statements are granted when certain conditions are fulfilled.

The usage of SWRL to represent the authorization statements enables to cope with this capability in the access control system. The antecedent of the SWRL rule represent the *Conditions* of the tuple. It includes references to elements of the information model as conditions. This means that only when the conditions are fulfilled, the consequent is triggered granting the privileges. When the administrator does not provide *Conditions* in the authorization statement, it means that the rule will have an empty antecedent. Thus, the rule will be always triggered since no conditions have been specified and the permissions defined by the statement always apply. This is the way in which the authorization statements are stored in the model.

The enablement and disablement of roles is achieved by adding and removing identities associated to the Role. This approach allows that two users with the same role can keep it activated or deactivated in different periods of time. For instance, a given user can keep the *Admin* role activated only from Monday to Friday, while another user can keep it activated on weekends. If the role needs to be completely deactivated, it is just required to remove all the users associated to the Role. The antecedent of a SWRL rule describes the activation conditions and the consequent creates or removes the *memberOfCollection* association between *Subjects* and *Roles* to activate or deactivate them.

In the example represented in Figure 4.2, a user *Bob* acting as *Developer* of tenant A tries to perform an action on one of the virtual machines of his own organization. Listing 4.1 shows a fragment of the model managed by the access control system using RDF/XML [79] syntax, which is the primary exchange format for OWL 2 ontologies.

```
<rdf:RDF>
...
<Identity rdf:about="#Bob">
  <instanceID>Bob</instanceID>
...
</Identity>
<Role rdf:about="#Developer">
  <instanceID>Developer</instanceID>
...
<Role>
<memberOfCollection rdf:about="#BobDeveloper">
  <collection>Developer</collection>
  <member>Bob</member>
</memberOfCollection>
<VirtualComputerSystem rdf:about="#VM1">
  <name>VM1</name>
  <operatingStatus>In Service</operatingStatus>
...
</VirtualComputerSystem>
</rdf:RDF>
```

Listing 4.1: Sample model fragment

In this context, the administrator may decide the following authorization statement: “*Developers are allowed to stop VM1 only if that machine is not performing a snapshot*”. This statement can be represented by the 5-tuple represented in (4.5).

$$(CompanyX, Developer, Stop, VM1, VM1.OperatingStatus \neq Snapshotting) \quad (4.5)$$

$$VirtualComputerSystem(?vm) \wedge name(?vm, ?n) \wedge equal(?n, 'VM1') \wedge \quad (4.1)$$

$$operatingStatus(?vm, ?st) \wedge notEqual(?st, 'Snapshotting') \wedge \quad (4.2)$$

$$Role(?r) \wedge instanceID(?r, ?id) \wedge equal(?id, 'Developer') \quad (4.3)$$

$$\rightarrow \quad (4.4)$$

$$AuthorizedPrivilege(?p) \wedge activities(?p, 'Stop') \wedge \quad (4.5)$$

$$authorizedSubject(?p, ?r) \wedge authorizedTarget(?p, ?vm) \quad (4.6)$$

$$(4.7)$$

Rule 5: Sample authorization statement rule

Rule 5 shows the representation of this authorization rule in SWRL abstract syntax. The first two lines of Rule 5 specify the corresponding virtual machine (*VM1*) and check the condition that it is not performing a snapshot. Line 4.3 selects the subject of the authorization statement, i.e. the role *Developer*. The consequent of the rule establishes the corresponding privilege in terms of user concepts and relationships. Since the status of the virtual machine *VM1* described in Listing 4.1 is “In Service” - which is different from “Snapshotting” - the conditions of the statement are fulfilled. Thus, Rule 5 grants the stop privilege to the role *Developer*.

Thanks to the usage of Rule 4, the access control system also infers the authorization fragment which allows *Bob* - as identity belonging to the *Developer* role - to access the virtual machine. The inferred fragment of the model is depicted in Listing 4.2.

```
<rdf:RDF>
...
<AuthorizedPrivilege rdf:about="#StopPriv">
  <instanceID>example:stopPriv</instanceID>
  <activities>Stop</activities>
...
</AuthorizedPrivilege>
<authorizedSubject rdf:about="#DevelopersStop">
  <privilege rdf:resource="#StopPrivilege"/>
  <privilegedElement rdf:resource="#Developers"/>
</authorizedSubject>
<authorizedTarget rdf:about="#StopVM1">
  <privilege rdf:resource="#StopPrivilege"/>
  <targetElement rdf:resource="#VM1"/>
</authorizedTarget>
...
  <authorizedSubject rdf:about="#Bob">
    <privilege rdf:resource="#StopPrivilege"/>
    <privilegedElement rdf:resource="#Bob"/>
  </authorizedSubject>
</rdf:RDF>
```

Listing 4.2: Inferred information example of an authorization rule

Hierarchical Role-based Access Control (hRBAC) extends RBAC with the aim of defining role hierarchies. These hierarchies establish privilege inheritance between roles, making a child role to inherit all the privileges defined for parent roles in the hierarchy. The major motivation for adding role hierarchy to RBAC is to simplify role management. A child role A can be defined as member of another parent role B. Privilege inheritance is achieved by the definition in OWL of the `memberOfCollection` property as transitive. It makes the system to consider instances belonging to any parent role, to be also belonging to its child roles in the hierarchy. In the example depicted in Figure 4.2, this feature allows any new privilege assigned to a role, for instance *Testing Developer*, to be automatically propagated to the upper role and so on. Thus the privilege is automatically assigned to the *Testing Head* role, the *Lab Head* role and the *Admin* role, respectively.

Hierarchical objects (HO) capability allows the privileges applied to a given object to be also applied to all its children objects in the hierarchy. This enables to achieve a high-level management of the access control system. In the example shown in Figure 4.2, the access control to the virtual network of tenant A can be effectively managed using this capability. A network represents an inherent hierarchical structure. In our domain model, different levels of networks and subnetworks can be defined as parents or children. So that, privileges affecting to a given network can be applied to its subnetworks and even to the virtual resources located in such subnetworks. The CIM allows the representation of parent-children relationships between resources by means of the `Component` association. This association is defined in the model as transitive, enabling the authorization system to recognize all subcomponents and dependent objects of another object through the whole hierarchy. A SWRL rule can be easily established to propagate the privileges throughout these objects.

Separation of duties (SoD) enables the protection of the fraud that might be caused by users. Since it is possible that two given privileges should not be associated to the same role at the same time and in the same place, the Static Separation of Duty feature is quite important in any access control system. This situation can be detected in our access control system by looking for a provoked inconsistency in the knowledge Base (KB).

As explained in Section 3.3.3, the reasoning capabilities available in the Semantic Web enable advanced features like conflict detection and even conflict resolution. The usage of OWL and SWRL as modeling languages for information systems provides different ways to detect conflicts that may appear in the KB. A customized set of SWRL rules can be used to include conflictive privileges in the antecedents and force a conflict by including contradictory facts in the KB. Since our models are described in OWL, a reasoner can be used to detect an inconsistency in the KB. When two contradictory facts are hold on the KB, it becomes inconsistent and reasoners are able to detect this situation. Thus, the consistency checking process of DL reasoners can be used to detect semantic conflicts. More information and details about these topics can be found in Section 3.3.3 and [23].

4.2.1 Authorization Models comparison

The previous section has defined different authorization capabilities that are supported by the access control system. This subsection provides some experimental results to evaluate and compare the authorization models which result from the application of the aforementioned capabilities, i.e. DAC, RBAC, hBAC and HO. These authorization capabilities can be compared since all of them can represent the same scenario and system behavior. However, conditional RBAC (cRBAC) can not be analyzed because the absence of that capability in the system can not be replaced by static definitions. That is, it affects not only to the ontology definition but rather to the behavior of the authorization system.

As explained before, the authorization capabilities are implemented by means of SWRL rules. Since the time required to evaluate the SWRL rules varies according to

the nature and number of the statements, the results depend on the selected scenario. Thus, the obtained results can change substantially according to different factors. Among them, the depth in the hierarchical system structure (e.g. Subnets and AdminDomains), the number of Roles and its levels of hierarchical depth or the number of Identities and privileges defined.

Nevertheless, describing a generic scenario can be useful to figure out the time required by the system to evaluate the rules, and therefore to support certain authorization capabilities. The chosen reference scenario is composed of 100 Identities, 21 Roles, 30 Authorization Privileges and 200 Managed Elements. Roles are arranged in a three tier fashion where 3 of them are declared on top of the hierarchy, 6 in the medium level and 12 in the bottom. The Managed elements are composed of 6 AdminDomains, 24 Subnets associated to them and 170 virtual machines associate to these subnets. The 100 identities are assigned to the different Roles. Finally, authorization statements are defined to grant the privileges to both Roles and Identities.

Privileges are assigned to the upper level in the system hierarchy i.e. to AdminDomains. Thus, after executing the HO SWRL rule, all the identities obtain the grants to access to the virtual machines defined in the bottom of the hierarchy. If hRBAC and RBAC are used, privileges assigned to the upper level Roles are inherited by the identities in the bottom of the hierarchy.

Five different configurations with five different populations have been made up to compare the authorization models. All the populations represent the same described scenario. Differences lie in the amount of rules as well as the ontology model statements hold at each one. Although the amount of statements at each population is different, executing the SWRL rules lead populations to be equivalent to each other. This is due to the fact that SWRL rules derive the equivalent statements as if they were created beforehand in the ontology model.

The first population is envisaged to evaluate HO in an isolated way. This population initially includes all the required model statements of the scenario as if they were generated by rules of hRBAC and RBAC. The second population is used to evaluate HO, hRBAC and RBAC authorization capabilities all together. This population requires a minimum initial amount of statements since many of them are derived by the rules. The third population is used to evaluate the behavior of hRBAC and RBAC. Notice that statements that may be generated by the HO rule are included beforehand in the population to make it comparable. The fourth population is used to evaluate RBAC independently. Similarly, the statements generated by the SWRL rules of hRBAC and HO are previously included in the population. The fifth population represents a reference population without any rule, where all the statements have been previously generated. This population is equivalent to having a Discretionary Access Control (DAC) system without hRBAC, RBAC nor HO.

Figure 4.3 shows the achieved results. There are three different data series in the graphic. *Initial Statements* represents the percentage of statements, with respect to the total, that are required to evaluate the authorization capabilities depicted in the x-axis. The

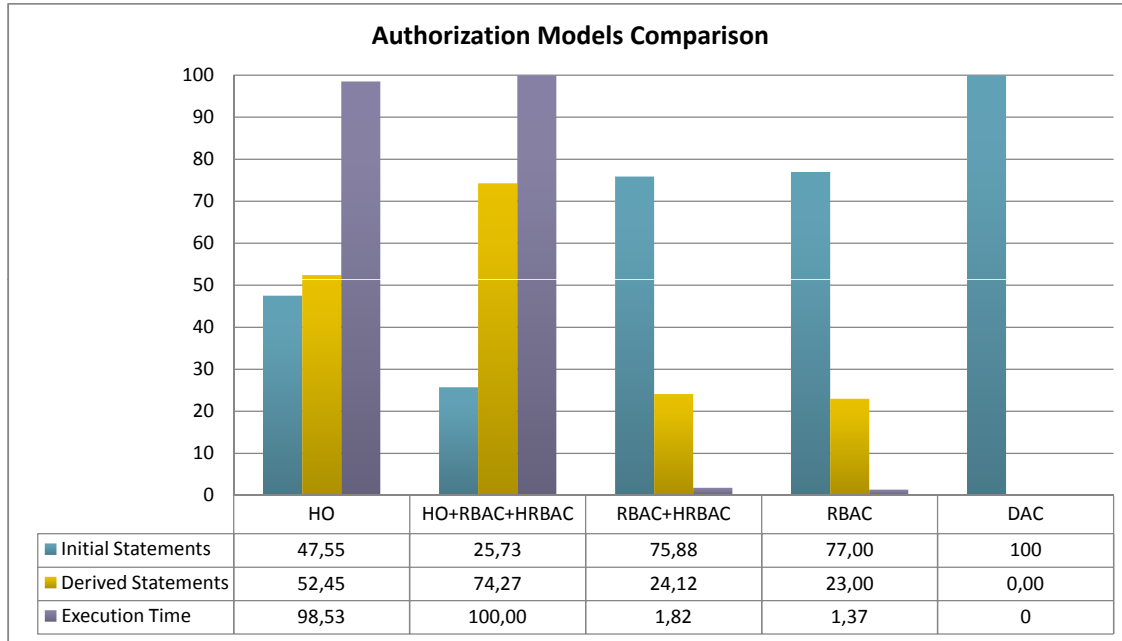


Figure 4.3: Authorization Models Comparison

total amount of statement in the scenario is 8600. This number coincides with the amount of *Initial Statements* in the DAC configuration, where everything is statically defined in the model and nothing is derived by rules. *Derived statements* represents the amount of statements which are inferred by SWRL rules. It should be noticed that *Initial Statements* plus *Derived statements* always sum up to the 100% of the statements, regardless the authorization configuration.

The configuration (HO + hRBAC + RBAC) provides the authorization system with the maximum level of expressiveness, so that it requires the highest computational time to be enforced. For this reason, this *Execution time* has been taken as reference (100% value in the chart) to be comparable with the rest of configurations. As can be seen from the figure, the evaluation of HO alone, requires a lot of time to be completed when compared with RBAC or HRBAC. It can be explained because it is a complex SWRL rule that derivates a lot of statements, more than 50% of the total. In DAC, since there is no SWRL rule, there is nothing to evaluate, and the amount of derived statements is 0.

4.3 Architecture

The proposed architecture is illustrated in Figure 4.4. Although the proposed access control system is potentially suitable for all the layers of the cloud computing stack, its adaptation needs to consider some specifics of each layer, such as relationships with other

layers, multi-vendor support, information model, etc. Thus, it has been decided to validate this research work over the IaaS layer. This layer constitutes the first logical step of the cloud stack and there is also a clear lack of advanced access control systems for this layer. It takes authorization decisions for cloud resource access attempts according to the authorization statements described by both tenants and cloud providers.

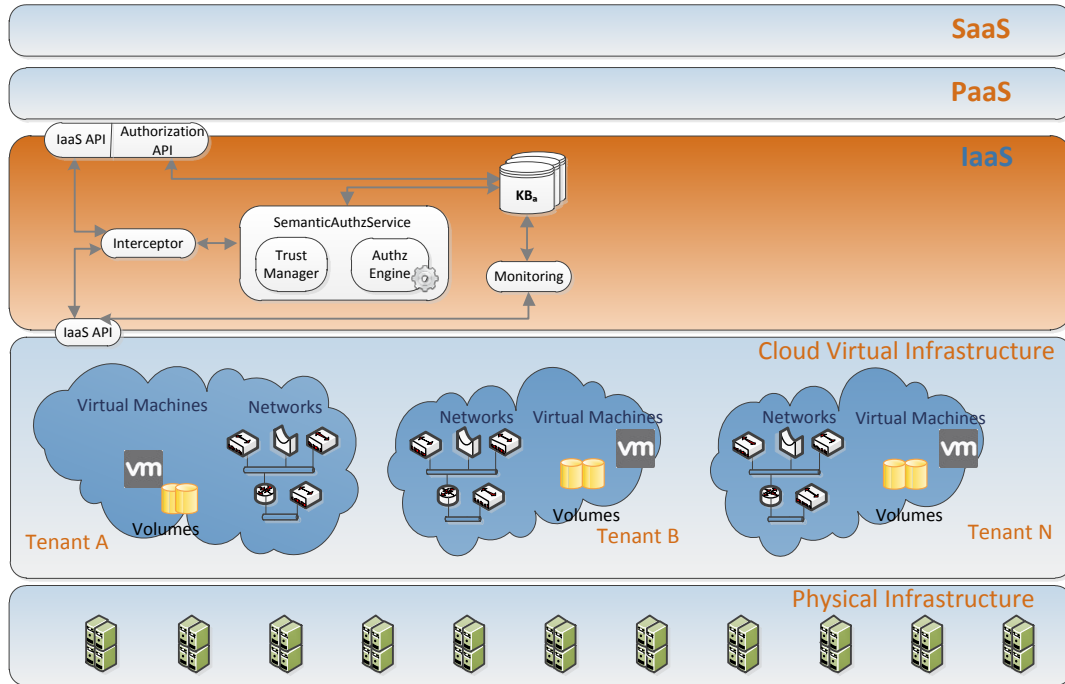


Figure 4.4: Semantic Authorization Architecture for cloud.

The IaaS API is the API already provided by all current IaaS cloud solutions like Eucalyptus, OpenStack, Amazon EC2 or RackSpace, among others. The *Interceptor* module captures the requests coming from users through the IaaS API in order to redirect them to the access control system. The *Interceptor* component is also responsible for returning the request back to the IaaS API, according to authorization decision carried out by the access control system.

The access control system is implemented by means of the *SemanticAuthzService*. The main aim of this component is to evaluate whether user requests have enough grants to perform the requested actions on the resources. It is composed of two main different sub-modules: the *Authorization Engine* and the *Trust Manager*. The *Authorization Engine* performs authorization reasoning based on the SWRL rules and the OWL ontology. In turn, the *Trust Manager* manages the privacy of tenants information according to established trust relationships.

The `Authorization API` component extends the `IaaS API` by enabling the description and insertion of authorization statements in the system. Such an API translates the statements to authorization rules in SWRL format in order to be later stored in the Knowledge Base. This component keeps the authorization information needed to carry out the authorization decisions. It includes both the authorization rules and the information model instances that represent the virtual infrastructure being managed.

Note that customers using the `Authorization API` do not require to understand neither OWL nor SWRL. The API helps to deal with rule definition, abstracting users from these languages. The concepts of the ontology are presented to the user so that he is able to define the conditions of the rules based on these concepts.

The access control system requires being up to date with the events that take place in the virtual infrastructure in order to achieve proper authorization decisions. Thus, the `Monitoring` component obtains information about the underline virtual environment and updates the KB accordingly. This update is done thanks to a subscription mechanism enforced in the cloud `IaaS API`. The module translates the source event messages into our ontological information model representation which is maintained in the KB.

4.4 Trust Management

A collaboration agreement between two tenants can be represented by means of a trust relationship. Trust relationships define the level of security and privacy that should be ensured between tenants. Our solution provides a fine grained trust model where tenants can describe which specific information from its information model is shared with other tenants. If a tenant *A* trusts another tenant *B* - denoted as $A \prec B$ -, it implies that *B* has available certain information of *A* to describe its authorization rules. Thus, a trust relationship can be seen as a 3-tuple (*Trustor*, *Trustee*, *ContextInfo*) describing that a *Trustor* trusts a *Trustee* by exposing the *ContextInfo* information. Then, the trustee is able to use that context information belonging to the trustor in order define its authorization statements. This mechanism enables tenants to control the access to their the resources from other tenants.

A common situation is a tenant *B* allowing another tenant *A* to use its subjects (indicating the subjects in the *ContextInfo*). It would permit *A* to describe authorization rules taking into account subjects of *B*. This approach enables *A* to grant subjects of *B* to access to *A*'s resources.

It could also happen that the trust relationship $A \prec B$ implies that *B* allows *A* to use objects and resources (not only subjects) in *A*'s authorization rules as context information, i.e. as conditions. To do so, *B* only needs to specify that information in the *ContextInfo* of the trust relationship. As an example of this situation, a tenant *A* could describe an authorization rule stating that a user of tenant *B* could access some of *A*'s resources only if *B*'s virtual machines of a given network are not available. It is important to remark

that this feature does not imply that A can grant privileges for his users to access B 's resources. This can only be specified by B . In other words, a tenant is the only one which can determine its permissions over its resources.

Trust relationships are not transitive, i.e. if $A \prec B$ and $B \prec C$, it does not imply that $A \prec C$. It has been decided to not automatically manage such a feature in order to keep the federation model simple. Obviously, this is supported just by explicitly inserting such trust relationships. Moreover, trust relationships are not symmetric, i.e. if $A \prec B$, it does not imply that $B \prec A$. This enables more control over the definition of federation scenarios. Analogously, in case this feature is needed, the inclusion of the corresponding symmetric trust relationship is enough to support it into the system. Finally, by default nobody trusts anyone else unless there is an explicit statement for this.

The trust information is used by the `TrustManager` module available in the architecture. Upon a request, this module is in charge of providing only the knowledge of those organizations which trust each other to take the authorization decision. It controls the privacy of all the information stored in the `Knowledge Base`. The module selects the information to be used to make the authorization decision taking into account the issuer requesting authorization in order to provide multi-tenancy in the cloud environment.

Our architecture keeps a different KB for each tenant. Each of these KBs contains the information model and the authorization statements for a particular tenant. Each KB also keeps the knowledge allowed by its trusted tenants according to the trust models defined with the aforementioned 3-tuple. This approach is safer and more secure than having just one KB for all the information since it allows to physically isolate sensitive information from different tenants.

The `TrustManager` module also manages the life cycle of the different KBs keeping them up-to-date according to the changes in the trust relationships between tenants. This update implies removing knowledge from the KB when a tenant cancels a trust relationship or updating the KB when a new trust relationship is established. Holding different KBs with information for different tenants implies that the `Monitoring` component also has to be aware of these relationships in order to update the KBs according to the information coming from the virtual infrastructure.

4.5 Authorization Process

Considering a symmetric trust relationship between two tenants A and B defined by $A \prec B$ and $B \prec A$, when a user of A tries to access to a resource belonging to B , the request first reaches the `IaaS API`. Then, the `Interceptor` takes the incoming request and forwards it to the `SemanticAuthzService`. The `Trust Manager` chooses the proper KB according to the organization to which the request resource belongs. When the `Authorization Engine` is aware of the KB which should be used, it reasons using the information model and authorization statements and derives the authorization decision. Finally, the autho-

rization response is sent back to the `Interceptor` which forwards it to the IaaS API, granting or denying the access to the resource.

The `AuthzEngine` makes authorization decisions based on the reasoning process performed by an OWL and SWRL reasoner. The underlying information model and the authorization statements are specified as SWRL rules and an OWL ontology stored in the KB. The reasoner uses this KB together with the semantic rules and the authorization model information to carry out the reasoning process.

As explained in Section 3.3.2, three main operations are available in the reasoner. Firstly, the reasoner performs an **Inference** process that allows new knowledge to be derived using the information available in the ontology. Note that the SWRL rules which compose the authorization model derive new information that is used to make the authorization decision. An example of inferred knowledge is shown in Listing 4.2. Secondly, the reasoner also performs a **Validation** process to detect possible violations of the constraints expressed in the ontology. It mainly consists of a global checking across the schema and the current instances looking for inconsistencies. Thirdly, the reasoner allows to **Query the ontology**, what includes instance recognition and inheritance recognition. The former consists in testing if an individual is an instance of a class and the latter if a class is a subclass of another class - or if a property is a sub-property of another one. This enables the formulation of generic queries referring to abstract concepts, being the system able to recognize instances belonging to concrete subclasses or sub-properties of such a concept.

To make the authorization decision, the `SemanticAuthzEngine` queries the reasoner to look for OWL instances in the KB which compose the authorization statement. Listing 4.3 shows the authorization query in SPARQL that examines the KB looking for a privilege that allows an identity to perform a certain action on a given resource. This query is a realization of the original abstract query exposed on Listing 3.3.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?privilege
WHERE {
    ?privilege rdf:type AuthorizedPrivilege .
    ?privilege AuthorizedPrivilege_activity #action# .
    ?identity rdf:type Identity
    ?identity Identity_name #identity# .
    ?privilege authorizedSubject ?identity .
    ?privilege authorizedTarget ?target .
    ?target ManagedElement_ID #target# .
}
```

Listing 4.3: SPARQL authorization query

The query searches an `AuthorizedPrivilege` instance which is associated with the given identity, target and action. It can be applied to the example of inferred knowledge provided in rule 5. The query has three input parameters: the identity that is performing the request, which in the example is *Bob*; the target that is being accessed, which in this

case is the virtual machine *VM1*; and the action performed against the target, which in this case is a *Stop* operation over the virtual machine *VM1*. In case the query does not return any privilege, the `AuthzEngine` denies the identity to perform the requested action over the resource.

4.6 Implementation and Performance

A proof of concept prototype of the `SemanticAuthzService` has been developed (see Figure 4.4). It has been integrated into an OpenStack [95] IaaS cloud provider. OpenStack is an open initiative to provide an IaaS stack for cloud computing. It is becoming one of the most widely used IaaS stacks solutions, with an important and increasing list of companies and organizations involved, such as Rackspace, Nasa, Citrix, AMD, Intel, NEC or Hewlett Packard, to name a few. This is the main reason for which OpenStack has been chosen for this research work.

OpenStack provides three main components to enable an IaaS platform. Namely, the OpenStack Compute (Nova), the OpenStack Image Service (Glance) and the OpenStack Object Storage (Swift). Nova is the cloud fabric controller, in charge of controlling virtual machine instances and networking. Glance can be seen as an image repository which enables lookup and retrieval of virtual machine images. Swift provides a high capacity, scalable and reliable storage system for the cloud infrastructure. It can also be used by Glance as storage service.

Currently, some of these Open Stack components have their own authentication and authorization mechanisms (e.g. `swauth` for Swift). Leveraging the access control to the IaaS in several different modules increases the complexity of authorization management. This may result in inconsistencies, conflicts or even security flaws. The implementation developed in our proposal is offered to the rest of system components as an autonomous service. This approach enables the use of this service not only by the Nova component, but also by Glance or Swift. These components can be adapted to make use of the `SemanticAuthzService`. Thus, providing it as a service enables a unified authorization mechanism.

The access control system (`SemanticAuthzService`) has been developed in Java, using the Java API for XML Web Services (JAX-WS) [96] stack to provide a Web service interface. This component uses a DL reasoner (i.e. a reasoner using Description Logics). There are several suitable implementations, such as Pellet [84], Jena [89], and the one proposed by FaCt++ [91], among others. Pellet has been chosen as reasoner due to the fact that it supports high expressiveness dealing with OWL 2 ontologies and it is also able to perform incremental consistency checking. All the rules described in the Section 4.2 are loaded in the reasoner as part of the bootstrapping process. Jena is also used to perform ontology operations and to manage the knowledge base since it is nowadays the de facto standard Java library to manage ontologies. The *Trust Manager* component described in

our architecture uses Jena to enforce the trust relationships in the KB. Concretely, Jena version 2.6.2 and Pellet version 2.0.0 have been used. The trust relationships are specified using an external API that enables to define such relationships. They are stored in a trust database managed by the *Trust Manager* component itself.

For the proof of concept implementation, an interceptor has been placed in the EC2 IaaS API interface of the OpenStack Nova component. This IaaS API is the entry point to the IaaS services and receives all the requests to be served by the IaaS. Each request is checked against the `Authorizer` class of the EC2 *nova-api* module in order to determine if the user is or is not authorized to invoke the requested action in the IaaS. The `Authorizer` class has been modified to intercept the authorization requests and to forward them to the `SemanticAuthzService` which provides the authorization decision, accordingly. The modified python code of OpenStack modules has been adapted to execute the Java Web-service client library provided to connect to the `SemanticAuthzService`. For this purpose, JPyype has been used, which allows python programs to access java class libraries. Thus, the Java Web-Service client library is instantiated into Python and then the appropriate methods are executed from the Python code, accordingly. It should be noticed that this approach has been adopted for testing the client library purposes, as well as for simplicity during the development to achieve the performance metrics. However, a native Python client could be also developed since the `SemanticAuthzService` interface is provided as a Web service, which is platform independent and highly compatible with different languages and platforms.

The monitoring module conceptually keeps the KB up-to-date with the current state of the IaaS. To do so, this module is continuously monitoring the IaaS to discover the status of the system and then updates the information model into the KB. In this proof of concept implementation, this module has been developed by attaching it to the message bus of Open Stack, which is the communication middleware used to coordinate all the components of the Open Stack architecture. This attachment enables to receive all the messages exchanged by Open Stack and update the KB of the `SemanticAuthzService` accordingly.

4.6.1 Testbed description

A testbed scenario has been deployed to test the proposed architecture and obtain some performance metrics. An OpenStack-based IaaS system has been deployed with 8 compute nodes to run virtual machine instances. Each node runs in a Hewlett Packard xw8400 Workstation with an Intel Xeon CPU at 3.00 GHz and 6 GB of RAM. The cloud controller, a Glance image service and the authorization service have been placed in one of these nodes. The purpose of using this configuration is to obtain tests for the worst case, in which the cloud controller and the access control system have to deal with the overload produced by running instances in the same host. Having a dedicated host for these components will result in lower execution times.

Three different operations have been considered in the tests. Namely, *describe instances*, *run instances* and *terminate instances*. They have been chosen as they represent the main operations related to virtual machine instance management. Moreover, they perform tasks with a different level of complexity. Thus, the *describe instances* operation is a lightweight operation, while the *run* and *terminate instances* perform some costly tasks, being the former even more costly. This makes the test offering more significant and different results. The aforementioned operations have been executed by invoking the corresponding *euca2ools* command, which uses the EC2 API to interact with the cloud controller.

In order to obtain statistics significant results, operations have been performed in sets of 100 executions, whose average is used as result value. The tests simulate the following use case. Firstly, a set of *describe instances* operations are executed, while the system is still not running any virtual machine instance, i.e. without instance overload. This enables to obtain the overhead produced by our authorization service in a dedicated system for the worst case, i.e. for the simplest operation with no instances. Then, the system is driven to a status in which each node is executing 4 instances (virtual machines). The rest of operation sets are then executed. These operations are executed alternating the set of *run instances* with another set of *describe instances* and the set of *terminate instances*. Operations are alternated in such a way that the system keeps in average the overload of 4 instances per node. These executions provide results considering a constant system overload for the considered operations.

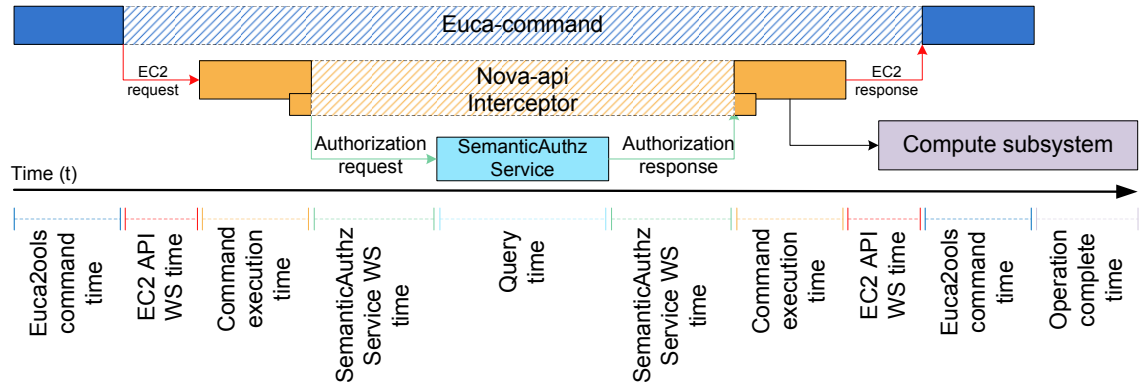


Figure 4.5: Operation execution workflow

Figure 4.5 shows the workflow followed by every operation and the average times taken for the entire executions. Firstly, the corresponding *euca2ools* command is executed. It accesses the EC2 API of the *nova-api* module, which is offered as a Web service. The authorization method is invoked in the *nova-api* and the interceptor redirects the request to the *SemanticAuthzService*. It performs the authorization decision and returns an access response to the interceptor. The interceptor enforces the authorization decision by allowing or forbidding the request in the *nova-api* module. Finally, the rest of

the command is executed and a response is provided to the *euca2ools* command. At this point, some operations may still be pending due to its asynchronous nature. This is the case of the *run instances* and *terminate instances*. For these operations, the corresponding command finalizes just after launching the request to the compute system, but before the actual operation has been completed. For instance, the *run instances* command will trigger the operation and return before the instance is actually in the running state. For this reason, another time has been included in the test, which considers the time when the operation goal is achieved (e.g. the virtual machine is running).

The bottom side of the figure shows the times taken into consideration for analysis purposes. It should be noticed that the two times related to the same command execution or Web service invocation have been added and considered as a single time in the results. This is the case of the *euca2ools* and *nova-api* execution times, as well as the EC2 and Authorization request and response times. It should be also mentioned that only positive authorizations are considered for the tests. Otherwise, negative authorizations would not execute the command. This would distort the statistics significance of the results due to the short time of the executions.

The above scenario takes into account the overload of the IaaS system. But, there is also an internal overload the SemanticAuthzService should deal with. This is related with the number of elements the system should take into account to make authorization decisions, i.e. number of roles, users, permissions, objects, etc. These elements are contained in the KB used by the SemanticAuthzService. The elements contained in the KB are called individuals and the set of individuals for a specific execution is referred to as population.

In order to study the scalability of the proposal, it has been taken a similar approach as the one taken for Grid environments and exposed in Section 3.4. Thus, the above-described scenario has been executed 10 times using incremental populations. This allows getting results with an increasing order of authorization elements. A simulated monitoring component is in charge of producing different populations which represent different system statuses. In order to quickly increment the number of individuals in the populations, this increment follows an exponential distribution. Finally, each individual is represented in the KB by a set of statements (not to be confused by authorization statements).

Population	1	2	3	4	5
Individuals	500	900	1,400	2,300	3,700
Statements	1,889	3,166	5,035	8,845	16,187
Population	6	7	8	9	10
Individuals	6,100	10,000	16,400	27,100	44,600
Statements	33,404	72,384	166,495	407,442	1,031,942

Table 4.1: Number of individuals and statements by population

Table 4.1 shows the number of individuals and its corresponding statements for each

population considered during the tests. Note that the real size of the KB is determined by these statements and not by the number of individuals.

4.6.2 Obtained results

Figures 4.6, 4.7, 4.8 and 4.9 show the times obtained during the tests for each population. It shows times for the execution of the first *describe instances* operation with no instances running and the times for the three operations with 4 instances running in each node. Six different times are depicted in each figure. They correspond with the times taken from the workflow exposed in Figure 4.5.

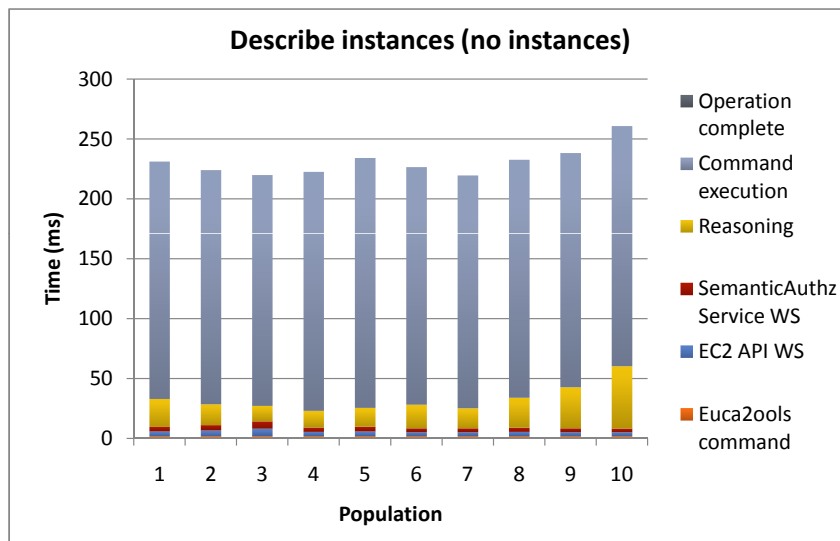


Figure 4.6: Describe instances overhead times without instances

As can be seen, the most part of the time is taken by the *Command execution* and *Operation complete* times. The *Reasoning* time spent by the authorization service is also outstanding in the *describe instances* operations. But it should be taken into account that these operations are quite simple and the whole operation times are small. It should be noticed the different scales of the four figures. *Describe instances* operations take around 250 milliseconds, while *run instances* are about 5,000 and *terminate instances* around 1,500. The fluctuations observed in the different executions (and populations) are due to the fluctuating nature of the Cloud environment. Moreover, note that populations are following an exponential trend whereas the reasoning times are almost following a linear trend.

The figures also show the difference between the *Command execution* and *Operation complete* times. While the *run instances* command finishes near the 1,900 ms (the time it takes to process the request), the underlying system takes around 3,000 ms more to

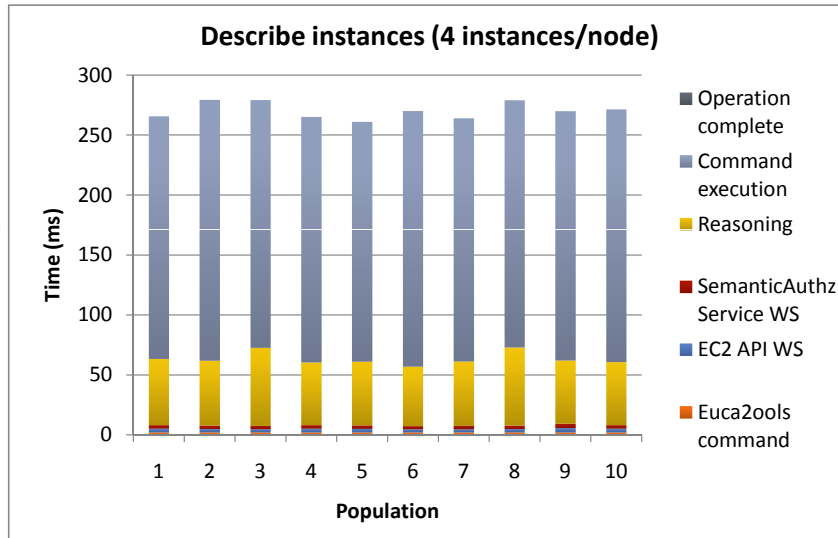


Figure 4.7: Describe instances overhead times with 4 instances/node

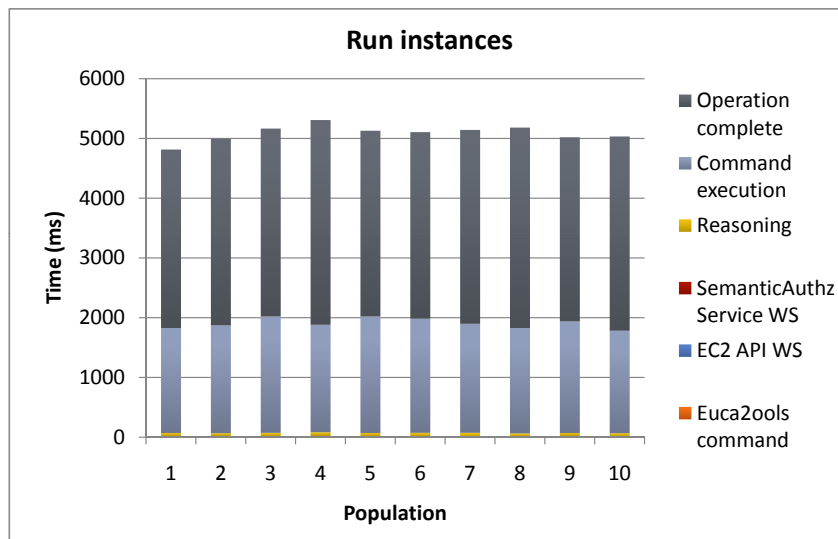


Figure 4.8: Run instances overhead times

actually get the virtual machine running. At this point, it should be noticed that these times have been taken considering a running system, i.e. discarding the first time an image is taken from the Glance image server, which takes much longer. For the rest of executions, the image is already cached in the node's hard disk and a copy-on-write image is quickly deployed. On the other hand, it can be also seen that the *describe instances* operation has no *Operation complete* time. That is because it is a get operation to retrieve information. Thus, the operation ends at the same time that the command terminates (synchronous

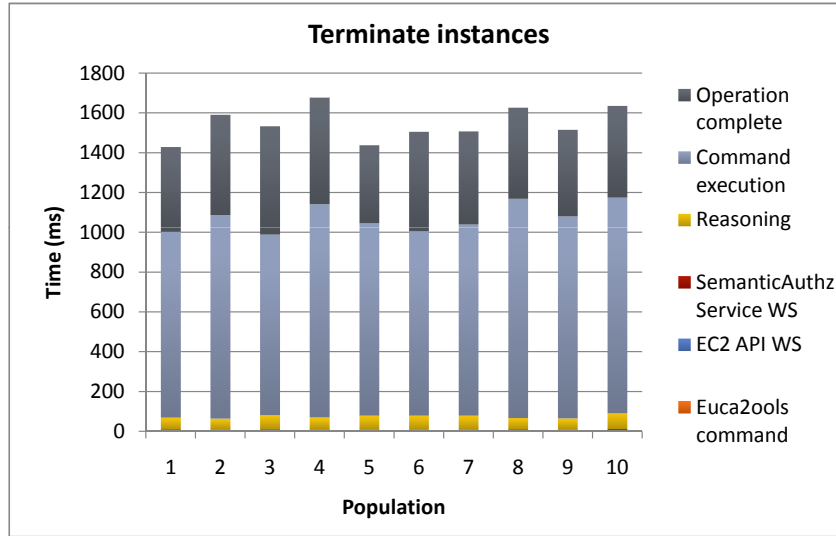


Figure 4.9: Terminate instances overhead times

execution), when it shows the requested information to the user.

In any case, it can be observed that the command and operation times are higher than the time taken by the access control system to perform an authorization decision. Trying to measure this difference, Figure 4.10 and Figure 4.11 show the overhead introduced by the access control system when compared with command and operation times, respectively. That is, the portion of the whole command or operation time that is due to the authorization process. Figure 4.10 shows the overhead against the command time, while the Figure 4.11 shows it against the whole operation. The authorization time considered to calculate this overhead contains the *SemanticAuthzService WS* and *Reasoning* times.

As can be observed, the overhead highly depends on the executed operation. This is due to the different ending times for each operation, as noticed in the previous figures. The average overhead is just around 2% and 5% for operations that manage the underlying infrastructure (5% and 9% if considering just the command execution).

The *describe instances* command shows the worst case for our authorization mechanism. For such a simple command - which just performs a single quick database query - the overhead is still around 22%. The reason of this overhead is that this command takes very little time and it makes the authorization time to be more significant. However, note that this percentage is with respect to the total execution time which, in turn, is significantly lower for the *describe instances* than for others commands as the reader can see in the figures. This fact causes that bigger overhead percentages can be tolerated in the system.

It can be observed that in an empty system with no instances running, the overhead trend is linear whereas it is almost constant for the rest of the cases. This is due to the

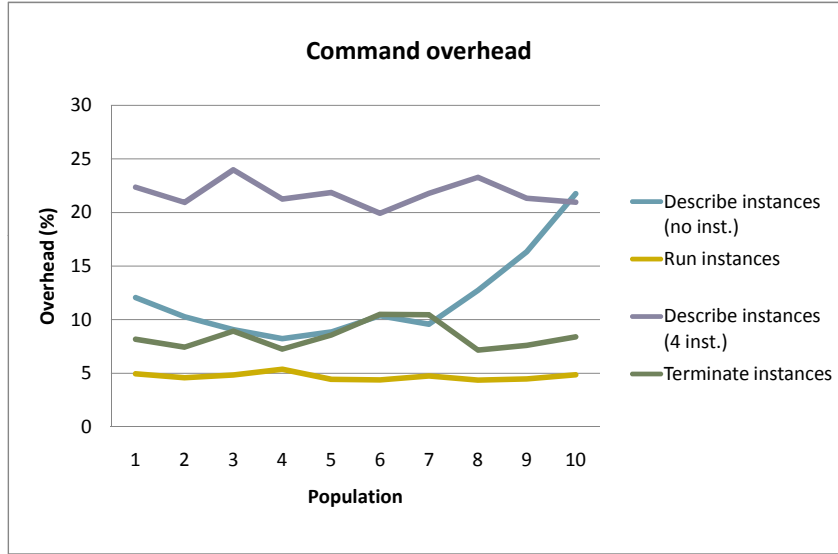


Figure 4.10: Command overhead times

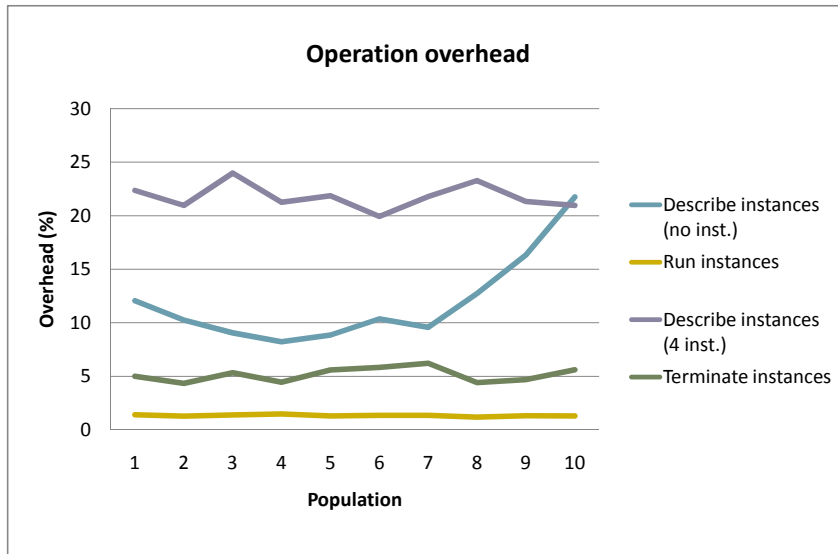


Figure 4.11: Operation overhead times

external overload suffered by the host machine, which directly affects to the authorization service execution. Note that the execution of rest of the command is done while the VMs are being executed and this causes an important CPU overhead in the system which makes the overhead of the authorization system almost neglected, causing the constant trend shown. However, the *describe instances* command with no instances running do not suffer of this overhead and it causes a linear trend, according the increasing of the

population.

The low execution time of the *describe instances* command with no instances running, enables to appreciate how increasing the population size affects the reasoning time. It can be seen that the last populations with a higher number of statements, also produce a higher overload. The exponential form of the graph is due to the population increments, which also follow an exponential distribution. Note that the worst case (22%) is for systems in which hundreds of thousands of VMs are being managed with a single instance of the access control system (more than 1 million of statements). For this kind of scenarios we'd probably have to adopt a replication of the access control system in order to balance the load. Anyway, for this big scenario, the authorization time is still provided in terms of milliseconds, which is a clear sign of the scalability of the proposed system.

It should be noticed that these overheads are provided without taking into account the time of the current Nova authorization mechanism. The actual overhead produced by the proposed access control system when compared with a pure OpenStack deployment would be smaller than the figures shown in this section, since just the increase over the Nova authorization time should be considered. That is, in this section the current Nova authorization time is being considered as 0 in order to analyze the worst case. However, the actual Nova authorization time should be subtracted from the times presented in this section in order to get the actual overhead. This would result in even smaller times than the ones presented herein.

Finally, a constant tendency can be observed in the majority of the results shown in these figures. This tendency shows a good scalability of the proposed access control system, even if the population is increased following an exponential distribution.

Chapter 5

SecRBAC: Secure data in the Clouds

This chapter presents a data-centric access control solution with enriched RBAC expressiveness in which security is focused on protecting user data regardless the Cloud service provider that holds it. The authorization model presented in Chapters 3 and 4 is mathematically formalized and the authorization statements are then cryptographically protected to avoid any misbehavior from the CSP when evaluating the rules. Data is encrypted and protected, resulting in a trustworthy access control mechanism that can run in untrusted CSPs without risk of the CSP accessing the data or releasing it to unauthorized parties.

The research work described in this chapter has been submitted for publication in the paper entitled *SecRBAC: Secure data in the Clouds* [17].

This chapter is structured as follows. Section 5.1 introduces the cryptographic techniques that are used to protect the authorization model in this proposal. Then, the authorization model with enriched RBAC expressiveness is formalized in Section 5.2 as base for the data-centric model. Detailed specification of the data-centric authorization model and its representation and evaluation are exposed in Section 5.3.1. Section 5.4 provides some deployment guidelines for an access control solution and its integration within a Cloud Service Provider. Finally, some statistical results and details of a prototypical implementation that has been developed are exposed in Section 5.5.

5.1 Proxy Re-Encryption and Identity-Based Encryption

The proposed solution makes use of cryptography in order to protect data when moved to the cloud. Advanced cryptographic techniques are used to protect the authorization model in order to avoid the CSP being able to disclose data without data owner consent. Concretely, the proposed solution is based on Proxy Re-Encryption (PRE). A PRE

scheme [97] is a cryptographic scheme that enables an entity called *proxy* to re-encrypt data from one key to another without being able to decrypt the data. That is, given a couple of key pairs α and β , the proxy could re-encrypt a ciphertext c_α encrypted under α public key to another ciphertext c_β that can be decrypted using β private key. Using this kind of cryptography, a user u_α can encrypt a piece of data m using his own public key pub_α to obtain a ciphertext c_α . A re-encryption key $rk_{\alpha \rightarrow \beta}$ can be generated for a proxy to re-encrypt from α to β , thus transforming c_α to another ciphertext c_β . Then, another user u_β can use his own private key $priv_\beta$ to decrypt c_β and obtain the plain piece of data m .

Several works in this direction have arisen, resulting in diverse Proxy Re-Encryption schemes that present different features. The solution presented is not tight to a concrete PRE scheme or implementation. However, not all the available PRE schemes are suitable to achieve the goals of this research. In order to characterize and compare the different schemes, [97] provided a set of features relevant to proxy re-encryption. Based on this characterization, the following set of features are required by the Proxy Re-Encryption scheme used for the solution proposed:

- *Unidirectionality.* A unidirectional scheme enables the generation of a re-encryption key $rk_{\alpha \rightarrow \beta}$ without allowing re-encryption from β to α .
- *Non-interactivity.* A non-interactive scheme enables a user u_α to construct a re-encryption key $rk_{\alpha \rightarrow \beta}$ without the participation of u_β or any other entity.
- *Multi-use.* A multi-use scheme enables the proxy to perform multiple re-encryption operations on a single ciphertext. That is, to re-encrypt from c_α to c_β , from c_β to c_γ and so on.

On the other hand, Identity-Based Encryption (IBE) [98] is a type of public key cryptography in which key pairs for a given entity are generated based on the identity of that entity. Using this kind of cryptography, a piece of data m can be encrypted using the identity id_α of a user u_α to obtain a ciphertext c_α . Then, user u_α can use his private key $priv_\alpha$ to decrypt c_α and obtain the plain piece of data m . Note that no public key pub_α is used for encryption, but the identity of the user id_α is applied instead. IBE schemes usually require the generation of a master key pair that is used to derive user keys based on their identities. The master public key is publicly known and can be directly employed by users to generate the public key of another user based on his identity. In turn, the master private key should be kept private and users can obtain their private keys from a trusted entity that owns the master private key. This entity is called Private Key Generator (PKG).

As will be seen in following sections, an authorization model is composed of several elements (e.g. subjects, roles, grants, etc). The usage of IBE avoids the need to generate and manage a public and private key pair for every authorization element. Instead, the identities of such elements can be directly used within the cryptographic operations.

In [99], an Identity-Based Proxy Re-Encryption (IBPRE) approach is proposed. It combines both IBE and PRE, allowing a proxy to translate a ciphertext encrypted under a user's identity into another ciphertext under another user's identity. In this approach, a Master Secret Key (MSK) is used to generate user secret keys from their identities. These secret keys are equivalent to private keys in IBE. No public keys are needed, since identities are directly used in the cryptographic operations. With this approach, a user u_α can encrypt a piece of data m using his own identity id_α to obtain a ciphertext c_{id_α} encrypted under id_α . A re-encryption key $rk_{\alpha \rightarrow \beta}$ can be generated to re-encrypt from id_α to id_β . Then, a proxy can use $rk_{\alpha \rightarrow \beta}$ to obtain another ciphertext c_{id_β} under the identity of another user u_β . This can then use his own secret key sk_β to obtain the plain piece of data m . As for IBE approaches, the MSK should be kept private and users can obtain their secret key from the PKG.

This IBPRE scheme is the one selected for the authorization solution proposed. It has been selected because it combines both PRE and IBE. It fulfills the three aforementioned required features of proxy re-encryption and it supports IBE, what allows to use the identities of the authorization elements for cryptographic operations, avoiding the need to generate and manage a key pair for each element.

As mentioned before, the proposed solution is not tight to any PRE scheme or implementation. For the purpose of providing a comprehensive and feasible solution, the rest of this chapter is based on the IBPRE approach and notation. However, the proposal could be applied to use other Proxy Re-Encryption schemes that fulfill the three aforementioned required features. This includes current or future schemes that could improve performance or security. It could be even a pure PRE scheme without combination with IBE, although that could imply the generation and management of extra key pairs. Moreover, some advanced functionality provided by this solution might be lost, like compatibility with PKI, which is supported by an extension of IBPRE and avoids the usage of a PKG.

The following set of functions is provided by IBPRE, and it constitutes the cryptographic primitives of the proposed solution:

$$setup(p, k) \rightarrow (p, msk) \quad (5.1)$$

$$keygen(p, msk, id_\alpha) \rightarrow sk_\alpha \quad (5.2)$$

$$encrypt(p, id_\alpha, m) \rightarrow c_\alpha \quad (5.3)$$

$$rkgen(p, sk_\alpha, id_\alpha, id_\beta) \rightarrow rk_{\alpha \rightarrow \beta} \quad (5.4)$$

$$reencrypt(p, rk_{\alpha \rightarrow \beta}, c_\alpha) \rightarrow c_\beta \quad (5.5)$$

$$decrypt(p, sk_\alpha, c_\alpha) \rightarrow m \quad (5.6)$$

Details about the cryptographic operations that are performed by these functions can be found in [99]. A brief description of each function follows. (5.1) Initializes the cryptographic scheme. It takes as input a security parameter k to initialize the cryptographic scheme (e.g. parameters to generate an elliptic curve) and outputs both the Master Secret

Key msk and a set of public parameters p that is used as input for the rest of functions. (5.2) generates Secret Keys. It takes as input the msk and an identity id_α ; and outputs the Secret Key sk_α corresponding to that identity. (5.3) encrypts data. It takes as input an identity id_α and a plain text m ; and outputs the encryption of m under the specified identity c_α . (5.4) generates Re-encryption Keys. It takes as input the source and target identities id_α and id_β as well as the Secret Key of the source identity sk_α ; and outputs the Re-encryption Key $rk_{\alpha \rightarrow \beta}$ that enables to re-encrypt from id_α to id_β . (5.5) re-encrypts data. It takes as input a ciphertext c_α under identity id_α and a Re-encryption Key $rk_{\alpha \rightarrow \beta}$; and outputs the re-encrypted ciphertext c_β under identity id_β . (5.6) decrypts data. It takes as input a ciphertext c_α and its corresponding Secret Key sk_α ; and outputs the plain text m resulting of decrypting c_α .

5.2 Authorization model formalization

This section presents the authorization model with enriched Role-based Access Control expressiveness used for this proposal. This serves to provide the formalization basis for the data-centric solution that will be specified in following sections.

An authorization model determines the privileges granted to subjects for accessing specific resources or system objects.

Definition 1 (Privileges). *Let $\mathbb{O}$ be the set of objects that are managed and $\mathbb{A}$ the set of actions that may be performed over those objects. A privilege specifies a given action that can be performed over a specific object. It is defined as an ordered pair (a, o) , $a \in \mathbb{A}$, $o \in \mathbb{O}$ and the set of privileges in the authorization model determines the binary relation*

$$P := (\mathbb{A}, \mathbb{O}, \mathbb{P}), \mathbb{P} \subseteq \mathbb{A} \times \mathbb{O} = \{(a, o), a \in \mathbb{A}, o \in \mathbb{O}\} \quad (5.7)$$

Role-Based Access Control (RBAC) is an authorization scheme supported by most of the current authorization solutions. In this approach, the authorization model makes use of the *Role* concept to assign privileges to subjects. A set of subjects can be assigned to one or more roles which, in turn, can be associated to a set of privileges. This approach provides more expressiveness to the authorization model making it easier to manage privilege assignments through roles.

Definition 2 (Grants). *Let $\mathbb{S}$ be the set of subjects and $\mathbb{T}$ the set of roles. A grant assigns a privilege to a given subject or role. It is defined as an ordered pair (u, p) , $u \in \mathbb{S} \cup \mathbb{T}$, $p \in \mathbb{P}$ and the set of grants in the authorization model determines the binary relation*

$$G := (\mathbb{S} \cup \mathbb{T}, \mathbb{P}, \mathbb{G}), \quad \mathbb{G} \subseteq \mathbb{S} \cup \mathbb{T} \times \mathbb{P} = \{(u, p), u \in \mathbb{S} \cup \mathbb{T}, p \in \mathbb{P}\} \quad (5.8)$$

Definition 3 (Subject-role assignment). *A subject-role assignment is defined as an ordered pair (s, r) , $s \in \mathbb{S}$, $r \in \mathbb{T}$. This assignment makes s to inherit all privileges granted*

to r by (5.8). The set of subject-role assignments in the authorization model determines the binary relation

$$D := (\mathbb{S}, \mathbb{T}, \mathbb{D}), \mathbb{D} \subseteq \mathbb{S} \times \mathbb{T} = \{(s, r), s \in \mathbb{S}, r \in \mathbb{T}\} \quad (5.9)$$

This authorization model can be extended to hierarchical RBAC (hRBAC). Hierarchical RBAC enables the definition of role hierarchies, where child roles inherit all the privileges defined for their parent roles in the hierarchy. It should be noticed that privilege inheritance in this hierarchy is transitive. That is, if $r_1 \in \mathbb{T}$ is a child of $r_2 \in \mathbb{T}$ and r_2 is a child of $r_3 \in \mathbb{T}$, then r_1 will inherit all the privileges from both r_2 and r_3 .

Definition 4 (Parent-role assignment). A parent-role assignment is defined as an ordered pair (r_1, r_2) , $r_1, r_2 \in \mathbb{T}$, where r_1 is the child role and r_2 the parent role. This assignment makes r_1 to inherit all the privileges granted to r_2 by (5.8). The set of parent-role assignments in the authorization model determines the binary relation

$$E := (\mathbb{T}, \mathbb{T}, \mathbb{E}), \mathbb{E} \subseteq \mathbb{T} \times \mathbb{T} = \{(r_1, r_2), r_1, r_2 \in \mathbb{T}\} \quad (5.10)$$

To endow the model with more expressiveness, a hierarchical approach can be also applied to the protected objects. In this case, a child object will be affected by all the privileges that affect its parent objects in the hierarchy. This hierarchy is also transitive and it enables the specification of a single grant affecting a whole a set of objects - those which are children of the object specified in the grant.

Definition 5 (Parent-object assignment). A parent-object assignment is defined as an ordered pair (o_1, o_2) , $o_1, o_2 \in \mathbb{O}$, where o_1 is the child object and o_2 the parent object. This assignment makes o_1 to be affected by all the privileges that affect o_2 by (5.7). The set of parent-object assignments in the authorization model determines the binary relation

$$F := (\mathbb{O}, \mathbb{O}, \mathbb{F}), \mathbb{F} \subseteq \mathbb{O} \times \mathbb{O} = \{(o_1, o_2), o_1, o_2 \in \mathbb{O}\} \quad (5.11)$$

Since both role and object hierarchies are transitive, the transitive closures of the binary relations E and F are considered on their respective sets $\mathbb{T}$ and $\mathbb{O}$.

Definition 6 (Transitive closure). The transitive closure of a binary relation R on a set $\mathbb{A}$ is the minimal transitive relation R' on $\mathbb{A}$ that contains R . That is, $R' := (\mathbb{A}, \mathbb{A}, \mathbb{R}')$ with

$$\begin{aligned} \mathbb{R}' \subseteq \mathbb{A} \times \mathbb{A} = \{ & (x, y) \in \mathbb{A} \times \mathbb{A} \mid \exists (a_0, a_1, \dots, a_n), \\ & a_0 = x, a_n = y \wedge (a_i R a_{i+1}) \forall 0 \leq i < n \} \end{aligned} \quad (5.12)$$

The set of binary relations defined above are determined by the authorization rules that are defined by the data owner to administrate the access to the data. For instance, the definition of a rule stating that “subject s_1 belongs to role r_1 ” results in the element $d_1 = (s_1, r_1) \in \mathbb{D}$ in the binary relation D of the authorization model. Hence, this authorization

information is specified by the data owner and it is what determines whether a resource can be accessed or not by a given user. When a user requests to access a resource, the Cloud Service Provider queries the authorization model to check whether the requesting user (subject) is granted to perform the requested operation (action) over the protected resource (object).

Theorem 1 (Privilege granted). *A subject $s_1 \in \mathbb{S}$ is granted to perform action $a_1 \in \mathbb{A}$ over the object $o_1 \in \mathbb{O}$ iff $\exists g_1 = (u_1, p_1) \in \mathbb{G}$ such that*

$$u_1 \in \mathbb{S} \Rightarrow u_1 = s_1 \quad (5.13)$$

$$u_1 \in \mathbb{T} \Rightarrow \exists r_1 \in \mathbb{T} \mid (s_1 D r_1) \wedge ((r_1 = u_1) \vee (r_1 E' u_1)) \quad (5.14)$$

$$p_1 = (a_2, o_2) \mid (a_1 = a_2) \wedge ((o_1 = o_2) \vee (o_1 F' o_2)) \quad (5.15)$$

where E' and F' are the transitive closures of E and F on $\mathbb{T}$ and $\mathbb{O}$, respectively.

Proof. ($\Rightarrow$) Suppose subject $s_1 \in \mathbb{S}$ is granted to perform action $a_1 \in \mathbb{A}$ over the object $o_1 \in \mathbb{O}$. Then, there should exist a grant that relates the subject with the corresponding privilege. So, by Def. 2 there should exist $g_1 = (u_1, p_1) \in \mathbb{G}$. Since $u_1 \in \mathbb{S} \cup \mathbb{T}$, this grant can apply to a subject or to a role. If $u_1 \in \mathbb{S}$, then $u_1 = s_1$ as stated in (5.13). Otherwise, $u_1 \in \mathbb{T}$ and s_1 should inherit p_1 through role assignment.

Let us assume $u_1 = r_2 \in \mathbb{T}$ is the role to which the grant applies. For s_1 to inherit the privileges of a role, there should exist such a role and the corresponding subject-role assignment. So, there should exist $r_1 \in \mathbb{T}$ and by Def. 3 there should exist $(s_1, r_1) \in \mathbb{D}$ or, equivalently, $(s_1 D r_1)$ as stated in the left part of the conjunction in (5.14). Moreover, r_1 should acquire the privileges granted to $r_2 = u_1$, either being the same role or through the role hierarchy, as stated in the right part of the conjunction in (5.14). If $(s_1 D r_2)$, then $r_1 = r_2 = u_1$ as stated in the right part of the disjunction in (5.14). Otherwise, for s_1 to inherit the granted privilege, r_1 should be a child of r_2 in the role hierarchy. Hence, there should exist a sequence $(t_0, t_1, \dots, t_n), t_i \in \mathbb{T} \forall 0 \leq i < n, t_0 = r_1, t_n = r_2$ and t_i being child of $t_{i+1} \forall 0 \leq i < n$. For t_i be a child of t_{i+1} , there should exist the corresponding parent-role assignment. So, by Def. 4 there should exist $(t_i, t_{i+1}) \in \mathbb{E}$ or, equivalently, $(t_i E t_{i+1}) \forall 0 \leq i < n$ and by Def. 6, $(t_0 E' t_n) \Rightarrow (r_1 E' r_2) \Rightarrow (r_1 E' u_1)$ as stated in the right part of the disjunction in (5.14).

In turn, $p_1 = (a_2, o_2)$ by Def. 1. For s_1 to be granted to perform a_1 , the actions should be the same ($a_1 = a_2$) as stated in the left part of the conjunction in (5.15). Moreover, a_2 should be performed over o_1 , either by o_1 being the same object as o_2 or through the object hierarchy, as stated in the right part of the conjunction in (5.15). If a_2 can be performed directly over o_1 , then $o_2 = o_1$ as stated in the left part of the disjunction in (5.15). Otherwise, for a_2 to be performed over o_1 , this should be a child of o_2 in the object hierarchy. Hence, there should exist a sequence $(q_0, q_1, \dots, q_n), q_i \in \mathbb{O}$ with $q_0 = o_1, q_n = o_2$ and q_i being child of $q_{i+1} \forall 0 \leq i < n$. For q_i be a child of q_{i+1} , there should exist the corresponding parent-object assignment. So, by Def. 5 there should exist $(q_i, q_{i+1}) \in \mathbb{F}$

or, equivalently, $(q_i F q_{i+1}) \forall 0 \leq i < n$ and by Def. 6, $(q_0 F' q_n) \Rightarrow (o_1 F' o_2)$ as stated in the right part of the disjunction in (5.15).

($\Leftarrow$) Suppose there exists $g_1 = (u_1, p_1) \in \mathbb{G}$ such as (5.13), (5.14) and (5.15) hold. Then by Def. 2 p_1 is granted to u_1 , which may be either a subject or a role. If $u_1 \in \mathbb{S}$ and $u_1 = s_1$ as stated in (5.13), then p_1 is granted to s_1 . Otherwise, if $u_1 \in \mathbb{T}$ and there exists a role $r_1 \in \mathbb{T}$ such as $(s_1 D r_1)$ and $(r_1 = u_1)$ as stated in (5.14), then $(s_1 D u_1)$ and by Def. 3 s_1 inherits all the privileges granted to u_1 . Concretely, s_1 inherits p_1 . Alternatively, if $(s_1 D r_1)$ and $(r_1 E' u_1)$ as also stated in (5.14), then by Def. 6 there exists a sequence $(t_0, t_1, \dots, t_n), t_0 = r_1, t_n = r_2$ and $(t_i E t_{i+1}) \forall 0 \leq i < n$. By Def. 4 t_i is a child of t_{i+1} and t_i inherits all privileges granted to t_{i+1} . So, r_1 inherits all privileges assigned to u_1 . Concretely, r_1 inherits p_1 and by Def. 3 s_1 inherits p_1 . Hence, either by (5.13) or (5.14), p_1 is granted to s_1 . Now, if $p_1 = (a_2, o_2)$ such as $(a_1 = a_2)$ and $(o_1 = o_2)$ as stated in (5.15), then $p_1 = (a_2, o_2) \Rightarrow p_1 = (a_1, o_1)$. So, by Def. 1 a_1 can be performed over o_1 by all subjects to which p_1 is granted. Alternatively, if $p_1 = (a_2, o_2)$ such as $(a_1 = a_2)$ and $(o_1 F' o_2)$ as also stated in (5.15), then by Def. 6 there exists a sequence $(q_0, q_1, \dots, q_n), q_0 = o_1, q_n = o_2$ and $(q_i F q_{i+1}) \forall 0 \leq i < n$. By Def. 5 q_i is a child of q_{i+1} and q_i is affected by all privileges that affect q_{i+1} . Concretely, o_1 is affected by p_1 and a_1 can be performed over o_1 . Since p_1 is granted to s_1 and p_1 states that a_1 can be performed over o_1 , then s_1 is granted to perform a_1 over o_1 . $\square$

5.3 Self-protected authorization model for data-centric security

This section describes a protected authorization model for a data-centric solution. It is achieved by the application of the cryptographic techniques described in Section 5.1. Then, a representation and evaluation mechanism based on Semantic Web technologies is also proposed.

5.3.1 Protecting the authorization model

The authorization model presented in the previous section determines the privileges that are granted to subjects. It should be evaluated by the Cloud Service Provider upon an access request in order to decide whether such a request is permitted or not. However, one should trust the CSP to legitimately evaluate the model and enforce the authorization decision. A data-centric protection mechanism is provided to assure that data can only be accessed by authorized subjects according to the data owner rules. The CSP cannot access or disclose data to unauthorized parties.

Following a data-centric security approach, data should be encrypted to avoid undesired access. Then, the access control mechanism should control who will be able to

decrypt the data and get access to its content. In terms of authorization, this means that the set of objects $\mathbb{O}$ should be encrypted before being uploaded to the Cloud. Moreover, the set of actions $\mathbb{A}$ is formed by the *access* action, meaning being able to decrypt the data and get access to its content. That is, $\mathbb{A} = \{access\}$. The general authorization model defined in Section 5.2 is able to support other actions over protected data (e.g. modify or delete). However, actually protecting the data against those actions in a data-centric approach imply advanced and more complex cryptographic techniques to provide such protection at data level. This is currently out of the scope of this research, although it is being considered as part of on-going research work.

The reduction of the set $\mathbb{A}$ to a single element allows a simplification of the protected authorization model regarding privileges and grants. A new binary relation G_a can be defined to represent the grants with access privileges.

Definition 7 (Access grant). *An access grant assigns a privilege to perform the access action to a given subject or role and it is defined as*

$$\begin{aligned} G_a &:= (\mathbb{S} \cup \mathbb{T}, \mathbb{O}, \mathbb{G}_a), \\ \mathbb{G}_a &\subseteq \mathbb{S} \cup \mathbb{T} \times \mathbb{O} = \{(u, o), u \in \mathbb{S} \cup \mathbb{T}, o \in \mathbb{O} \mid \\ &\quad \exists g \in \mathbb{G}, p \in \mathbb{P}, g = (u, p), p = (access, o)\} \end{aligned} \quad (5.16)$$

It should be noticed that when $\mathbb{A} = \{access\}$ both relations are equivalent, i.e. $G \equiv G_a$. Since $\mathbb{A} = \{access\}$, then there is only one possible privilege for each object. That is, $\mathbb{P} = \{(access, o), a \in \mathbb{A}, o \in \mathbb{O}\}$ and $\mathbb{G} = \{(u, (access, o)), u \in \mathbb{S} \cup \mathbb{T}, o \in \mathbb{O}\} \equiv \mathbb{G}_a$.

This reduction also simplifies the conditions of Theorem 1 when considering only the *access* action. The following Corollary can be deduced from Theorem 1.

Corollary 1 (Access granted). *A subject $s_1 \in \mathbb{S}$ is granted to access the object $o_1 \in \mathbb{O}$ iff $\exists g_1 = (u_1, o_2) \in \mathbb{G}_a$ such that*

$$u_1 \in \mathbb{S} \Rightarrow u_1 = s_1 \quad (5.17)$$

$$u_1 \in \mathbb{T} \Rightarrow \exists r_1 \in \mathbb{T} \mid (s_1 D r_1) \wedge ((r_1 = u_1) \vee (r_1 E' u_1)) \quad (5.18)$$

$$(o_1 = o_2) \vee (o_1 F' o_2) \quad (5.19)$$

As exposed in Section 5.1, a cryptographic scheme is used to protect the authorization model. This is achieved by encrypting the objects using the *encrypt()* function (5.3) and assigning a set of keys to the different elements of the authorization model. In practical terms, objects are not directly encrypted using the *encrypt()* function. Since data objects may be large documents or files, a digital envelope approach should be rather used as explained in Section 5.4.

In order to apply the cryptographic operations, identities of the authorization elements should be used. Let us denote as id_x the identity of any authorization element x , being $x \in \mathbb{O} \cup \mathbb{S} \cup \mathbb{T}$. The actual assignment of identities to the elements of the authorization

model is domain specific. For IBPRE functions, an identity can be any string identifying the element as far as it is unique within the domain.

Definition 8 (Encrypted object). *Let $\mathbb{O}$ be the set of objects managed by the system as specified in Def. 1. The set $\mathbb{O}'$ of encrypted objects is defined as*

$$\mathbb{O}' = \{c_o \mid c_o = \text{encrypt}(p, id_o, o) \forall o \in \mathbb{O}\} \quad (5.20)$$

where $\text{encrypt}()$ is the function (5.3).

Once data objects are encrypted, a mechanism is provided for authorized users to be able to access the data based on the authorization model. Intuitively, when a user s_1 is able to access object o_1 according to Corollary 1, a path can be found through the binary relations of the authorization model that links s_1 with o_1 . In general terms, the path will be something like the following:

$$\begin{array}{ccccccc} s_1 & \xrightarrow[\text{rk}_{r_1 \rightarrow s_1}]{D} & r_1 & \xrightarrow[\text{rk}_{r_i \rightarrow r_1}]{E} & \dots & \xrightarrow[\text{rk}_{r_2 \rightarrow r_i}]{E} & r_2 \\ r_2 & \xrightarrow[\text{rk}_{o_2 \rightarrow r_2}]{G_a} & o_2 & \xleftarrow[\text{rk}_{o_i \rightarrow o_2}]{F} & \dots & \xleftarrow[\text{rk}_{o_1 \rightarrow o_i}]{F} & o_1 \end{array} \quad (5.21)$$

Labels above arrows represent the binary relations of the authorization model. This path can be shorter as some relations may not be present, depending on the conditions of Corollary 1. For instance, if $u_1 \in \mathbb{S}$, then by (5.17) $u_1 = s_1$ and $g_1 = (s_1, o_2)$, which directly links s_1 to o_2 without intermediate roles. So, no roles would appear in the path.

This link between o_1 and s_1 enables to construct a chain to re-encrypt protected objects under the identity of users. That is, the above path determines the re-encryption chain needed to re-encrypt the protected object from c_{o_1} to c_{s_1} , which can be directly decrypted by using the Secret Key of user s_1 . Labels below arrows in the path represent the re-encryption keys that should be applied. Note that re-encryption keys assigned to each binary relation should be generated in reverse order to the one determined by the relation, except for the F relation of parent object assignments.

In the protected model, a Secret Key is assigned to every element of the authorization model based on its identity. Although these keys are not used to decrypt data (only users need to access the data), but they are employed to generate re-encryption keys. It should be noticed that this represents the formal definition of the protected authorization elements. However, when actually implementing the model, not all keys need to be computed and stored. They can be generated on demand when required to generate a particular re-encryption key, as explained in Section 5.4.

Definition 9 (protected subject, role and object). *Let $\mathbb{H}$ be the set of secret keys and $\mathbb{S}$, $\mathbb{T}$ and $\mathbb{O}$ the sets of subjects, roles and objects as specified by Def. 1 and Def. 2. The*

following sets are defined

$$\mathbb{S}'' = \{(s, sk_s), sk_s \in \mathbb{H}, \forall s \in \mathbb{S} \mid$$

$$sk_s = \text{keygen}(p, msk, id_s)\} \quad (5.22)$$

$$\mathbb{T}'' = \{(r, sk_r), sk_r \in \mathbb{H}, \forall r \in \mathbb{T} \mid$$

$$sk_r = \text{keygen}(p, msk, id_r)\} \quad (5.23)$$

$$\mathbb{O}'' = \{(o, sk_o), sk_o \in \mathbb{H}, \forall o \in \mathbb{O} \mid$$

$$sk_o = \text{keygen}(p, msk, id_o)\} \quad (5.24)$$

where $\text{keygen}()$ is the function (5.2).

The binary relations of the authorization model should also be protected, since they represent the authorization rules defined by the data owner. They are also the basis for the computations done by the CSP in order to check whether a user can access a piece of data. The assignment of re-encryption keys that only the data owner can generate guarantees that the CSP is not able to apply authorization rules that have not been legitimately defined by the data owner.

Definition 10 (Protected subject-role assignment). *Let $\mathbb{K}$ be the set of re-encryption keys and $\mathbb{D}$ be the set of subject-role assignments as specified in Def. 3. The set $\mathbb{D}''$ of protected subject-role assignments is defined as*

$$\mathbb{D}'' = \{(d, rk_{r \rightarrow s}), rk_{r \rightarrow s} \in \mathbb{K}, \forall d \in \mathbb{D} \mid$$

$$rk_{r \rightarrow s} = \text{rkgen}(p, sk_r, id_r, id_s),$$

$$d = (s, r), (r, sk_r) \in \mathbb{T}''\} \quad (5.25)$$

where $\text{rkgen}()$ is the function (5.4).

Definition 11 (Protected parent-role assignment). *Let $\mathbb{E}$ be the set of parent-role assignments as specified in Def. 4. The set $\mathbb{E}''$ of protected parent-role assignments is defined as*

$$\mathbb{E}'' = \{(e, rk_{r_2 \rightarrow r_1}), rk_{r_2 \rightarrow r_1} \in \mathbb{K}, \forall e \in \mathbb{E} \mid$$

$$rk_{r_2 \rightarrow r_1} = \text{rkgen}(p, sk_{r_2}, id_{r_2}, id_{r_1}),$$

$$e = (r_1, r_2), (r_2, sk_{r_2}) \in \mathbb{T}''\} \quad (5.26)$$

Definition 12 (Protected parent-object assignment). *Let $\mathbb{F}$ be the set of parent-object assignments as specified in Def. 5. The set $\mathbb{F}''$ of protected parent-object assignments is defined as*

$$\mathbb{F}'' = \{(f, rk_{o_1 \rightarrow o_2}), rk_{o_1 \rightarrow o_2} \in \mathbb{K}, \forall f \in \mathbb{F} \mid$$

$$rk_{o_1 \rightarrow o_2} = \text{rkgen}(p, sk_{o_1}, id_{o_1}, id_{o_2}),$$

$$f = (o_1, o_2), (o_1, sk_{o_1}) \in \mathbb{O}''\} \quad (5.27)$$

Definition 13 (Protected grant). Let $\mathbb{G}_a$ be the set of access grants as specified in Def. 7. The set $\mathbb{G}_a''$ of protected grants is defined as

$$\begin{aligned} \mathbb{G}_a'' = \{ & (g, rk_{o \rightarrow u}), rk_{o \rightarrow u} \in \mathbb{K}, \forall g \in \mathbb{G}_a \mid \\ & rk_{o \rightarrow u} = rkgen(p, sk_o, id_o, id_u), \\ & g = (u, o), (o, sk_o) \in \mathbb{O}'' \} \end{aligned} \quad (5.28)$$

Given the above sets with the corresponding secret and re-encryption keys, an encrypted object can only be accessed (i.e. decrypted) if there exists an adequate re-encryption chain. That is, an user s_1 , which only has access to its own secret key sk_{s_1} , can only decrypt an object o_1 initially encrypted under id_{o_1} by finding a re-encryption chain that transforms the object encrypted under id_{o_1} to the same object encrypted under id_{s_1} . This is only possible if the access to object o_1 is granted to the subject s_1 according to the protected authorization model.

Theorem 2 (Protected access). A subject $s_1 \in \mathbb{S}$ is granted to access the object $o_1 \in \mathbb{O}$ iff $\exists (r_0, \dots, r_n), r_i(x) = reencrypt(p, rk_i, x), rk_i \in \mathbb{K}, (y_i, rk_i) \in \mathbb{D}'' \cup \mathbb{E}'' \cup \mathbb{F}'' \cup \mathbb{G}_a'' \forall 0 \leq i \leq n$ such that

$$m = decrypt(p, sk_{s_1}, (r_0 \circ \dots \circ r_n)(c_{o_1})) \quad (5.29)$$

where $decrypt()$ is the function (5.6) and $\circ$ denotes function composition.

Proof. ($\Rightarrow$) Suppose subject s_1 is granted to access the object o_1 . By Corollary 1 there exists $g_1 = (u_1, o_2) \in \mathbb{G}_a''$ that fulfills (5.17), (5.18) and (5.19). Then, by Def. 13, there exists $g'' = (g_1, rk_{o_2 \rightarrow u_1}) \in \mathbb{G}_a''$ and by (5.5) $reencrypt(p, rk_{o_2 \rightarrow u_1}, c_{o_2})$ can be applied to get a re-encrypted ciphertext c_{u_1} from c_{o_2} . Let us denote this re-encryption as $r1_{o_2 \rightarrow u_1}$.

Since $u_1 \in \mathbb{S} \cup \mathbb{T}$, the grant should apply either to a subject ($u_1 \in \mathbb{S}$) or to a role ($u_1 \in \mathbb{T}$). If the grant applies to a subject, then by (5.17) $u_1 = s_1 \Rightarrow id_{u_1} = id_{s_1} \Rightarrow r1_{o_2 \rightarrow u_1} = r1_{o_2 \rightarrow s_1}$. Otherwise, the grant applies to a role and by (5.18) there exists $r_1 \in \mathbb{T} \mid (s_1 D r_1) \wedge ((r_1 = u_1) \vee (r_1 E' u_1))$. By Def. 10, there exists $d'' = (d, rk_{r_1 \rightarrow s_1}) \in \mathbb{D}''$. Then, by (5.5) $reencrypt(p, rk_{r_1 \rightarrow s_1}, c_{r_1})$ can be applied to get a re-encrypted ciphertext c_{s_1} from c_{r_1} . Let us denote this re-encryption as $r2_{r_1 \rightarrow s_1}$.

Additionally, if $u_1 = r_1$, then $id_{u_1} = id_{r_1}$ and thus $r1_{o_2 \rightarrow u_1} = r1_{o_2 \rightarrow r_1}$. Otherwise, by (5.18) $(r_1 E' u_1)$ and by Def. 6 there exists a sequence $(a_0, a_1, \dots, a_n), a_0 = r_1, a_n = u_1 \wedge (a_i E a_{i+1}) \forall 0 \leq i < n$. For each $(a_i E a_{i+1})$, by Def. 11 there exists $e_i'' = (e_i, rk_{a_{i+1} \rightarrow a_i}) \in \mathbb{E}''$. So, by (5.5) $reencrypt(p, rk_{a_{i+1} \rightarrow a_i}, c_{a_{i+1}})$ can be applied to get a re-encrypted ciphertext c_{a_i} from $c_{a_{i+1}}$. If this set of re-encryptions is applied in the order determined by the sequence $(a_n, \dots, a_0)$, a re-encrypted ciphertext c_{r_1} can be obtained from c_{u_1} . Let us denote this re-encryption as $r3_{u_1 \rightarrow r_1}$.

On the other hand, by (5.19) either $(o_1 = o_2)$ or $(o_1 F' o_2)$. In the first case, $o_1 = o_2 \Rightarrow id_{o_1} = id_{o_2} \Rightarrow r1_{o_2 \rightarrow u_1} = r1_{o_1 \rightarrow u_1}$. In the second case, $(o_1 F' o_2)$ and by Def. 6 there exists a sequence $(b_0, b_1, \dots, b_n), b_0 = o_1, b_n = o_2 \wedge (b_i F b_{i+1}) \forall 0 \leq i < n$. For each $(b_i F b_{i+1})$,

by Def. 12 there exists $f_i'' = (f_i, rk_{b_i \rightarrow b_{i+1}}) \in \mathbb{F}''$. So, by (5.5) $reencrypt(p, rk_{b_i \rightarrow b_{i+1}}, c_{b_i})$ can be applied to get a re-encrypted ciphertext $c_{b_{i+1}}$ from c_{b_i} . If this set of re-encryptions is applied in the order determined by the sequence $(b_0, \dots, b_n)$, a re-encrypted ciphertext c_{o_2} can be obtained from c_{o_1} . Let us denote this re-encryption as $r4_{o_1 \rightarrow o_2}$.

Finally, by Def. 9, there exists $(s_1, sk_{s_1}) \in \mathbb{S}''$ and by (5.6) $decrypt(p, sk_{s_1}, c_{s_1})$ can be applied to get the plain (decrypted) value m . Then, $(r_0 \circ \dots \circ r_n)$ is formed by the composition of the functions r_1, r_2, r_3 and r_4 , depending on the different cases previously considered. That is, $r_4 \circ r_1 \circ r_3 \circ r_2$ in the general case, $r_1 \circ r_3 \circ r_2$ if $(o_1 = o_2)$, $r_4 \circ r_1 \circ r_2$ if $(r_1 = r_2)$, $r_4 \circ r_1$ if $(u_1 = s_1)$, $r_1 \circ r_2$ if $(o_1 = o_2 \wedge r_1 = r_2)$ and r_1 if $(o_1 = o_2 \wedge u_1 = s_1)$. In any case, these function compositions re-encrypt from o_1 to s_1 . Thus, $decrypt(p, sk_{s_1}, c_{s_1})$ can be applied and $m = decrypt(p, sk_{s_1}, (r_0 \circ \dots \circ r_n)(c_{o_1}))$.

($\Leftarrow$ proof by contradiction) Suppose that there exists a sequence $(r_0, \dots, r_n)$ with $r_i(x) = reencrypt(p, rk_i, x)$ and $rk_i \in \mathbb{K}$, $(y_i, rk_i) \in \mathbb{D}'' \cup \mathbb{E}'' \cup \mathbb{F}'' \cup \mathbb{G}_a''$ such that $m = decrypt(p, sk_{s_1}, (r_0 \circ \dots \circ r_n)(c_{o_1}))$ and subject s_1 is not granted to access the object o_1 . By (5.6), in order to apply $decrypt(p, sk_{s_1}, x)$, x should be under id_{s_1} . So, the result of $(r_0 \circ \dots \circ r_n)(c_{o_1})$ should be c_{s_1} . Moreover, since $(y_i, k_i) \in \mathbb{D}'' \cup \mathbb{E}'' \cup \mathbb{F}'' \cup \mathbb{G}_a''$, by (5.25), (5.26), (5.27) and (5.28), rk_i should be either $rk_{r \rightarrow s}$ or $rk_{r_a \rightarrow r_b}$ or $rk_{o_a \rightarrow o_b}$ or $rk_{o \rightarrow u}$, with $s \in \mathbb{S}$, $r, r_a, r_b \in \mathbb{T}$, $u \in \mathbb{S} \cup \mathbb{T}$, $o_a, o_b \in \mathbb{O}$. Additionally, by (5.5) for any $r_i(x)$ to re-encrypt from c_α to c_β , the corresponding re-encryption key should be $rk_{\alpha \rightarrow \beta}$. So, for $(r_0 \circ \dots \circ r_n)(x)$ to re-encrypt from c_{o_1} to c_{s_1} , the re-encryption key rk_0 of r_0 should be either $rk_{o_1 \rightarrow u_1}$ or $rk_{o_1 \rightarrow o_2}$, $u_1 \in \mathbb{S} \cup \mathbb{T}$, $o_2 \in \mathbb{O}$.

If $rk_0 = rk_{o_1 \rightarrow u_1}$, then $(y_0, rk_{o_1 \rightarrow u_1}) \in \mathbb{G}_a''$ and by (5.28) $y_0 = g = (u_1, o_1) \in \mathbb{G}_a$. So by Def. 7 u_1 is granted to access o_1 . If $u_1 \in \mathbb{S}$, then $u_1 = s_2 \neq s_1, s_2 \in \mathbb{S}$, otherwise s_1 would be granted to access o_1 . But if $u_1 = s_2 \neq s_1$ then $r_0(x) = reencrypt(p, rk_{o_1 \rightarrow s_2}, x)$, which produces c_{s_2} and there is no possible subsequence $(r_1 \circ \dots \circ r_n)$ that re-encrypts from id_{s_2} to id_{s_1} because $\nexists (y_1, rk_{s_2 \rightarrow \alpha}) \in \mathbb{D}'' \cup \mathbb{E}'' \cup \mathbb{F}'' \cup \mathbb{G}_a''$. Hence, it is not possible that $u_1 \in \mathbb{S}$. So $u_1 = r_1 \in \mathbb{T}$, being $r_0 = rk_{o_1 \rightarrow r_1}$ and the subsequence $(r_1 \circ \dots \circ r_n)$ should re-encrypt from id_{r_1} to id_{s_1} . Then, rk_1 should be either $rk_{r_1 \rightarrow s_3}$ or $rk_{r_1 \rightarrow r_2}$, $s_3 \in \mathbb{S}$, $r_2 \in \mathbb{T}$. If $rk_1 = rk_{r_1 \rightarrow s_3}$, then $(y_1, rk_{r_1 \rightarrow s_3}) \in \mathbb{D}''$ and by (5.25) $y_1 = d = (s_3, r_1)$. If $s_3 = s_1$, then by Def. 3 and Def. 7 s_1 would be granted to access o_1 . So, $s_3 = s_4 \neq s_1, s_4 \in \mathbb{S}$, but then $r_1 = reencrypt(p, rk_{r_1 \rightarrow s_4})$, which produces c_{s_4} and there is no sequence $(r_2 \circ \dots \circ r_n)$ that re-encrypts from id_{s_4} to id_{s_1} because $\nexists (y_2, rk_{s_4 \rightarrow \alpha}) \in \mathbb{D}'' \cup \mathbb{E}'' \cup \mathbb{F}'' \cup \mathbb{G}_a''$. So, it is not possible that $rk_1 = rk_{r_1 \rightarrow s_3}$. So, $rk_1 = rk_{r_1 \rightarrow r_2}$ and the subsequence $(r_2 \circ \dots \circ r_n)$ should re-encrypt from r_2 to s_1 . Recursively, $rk_2 = rk_{r_2 \rightarrow r_3}$, $r_3 \in \mathbb{T}$ and the sequence would be always re-encrypting from one role to another. Therefore, it is not possible for rk_0 to be $rk_{o_1 \rightarrow u_1}$.

Then, $rk_0 = rk_{o_1 \rightarrow o_2}$ and the subsequence $(r_1 \circ \dots \circ r_n)$ should re-encrypt from o_2 to s_1 . So, either $rk_1 = rk_{o_2 \rightarrow u_2}$ or $rk_1 = rk_{o_2 \rightarrow o_3}$, $u_2 \in \mathbb{S} \cup \mathbb{T}$, $o_3 \in \mathbb{O}$. The first case is not possible by applying Def. 5 and following the previous rationale for $rk_0 = rk_{o_1 \rightarrow u_1}$. Hence, $rk_1 = rk_{o_2 \rightarrow o_3}$ and recursively the sequence would be always re-encrypting from one object to another. So, it is not possible that $\exists (r_0, \dots, r_n), r_i(x) = reencrypt(p, k_i, x), k_i \in$

$\mathbb{K}, (y_i, k_i) \in \mathbb{D}'' \cup \mathbb{E}'' \cup \mathbb{F}'' \cup \mathbb{G}_a''$ such that $m = \text{decrypt}(p, sk_{s_1}, (r_0 \circ \dots \circ r_n)(c_{o_1}))$ and subject $s_1 \in \mathbb{S}$ is not granted to access o_1 . $\square$

5.3.2 Representation and evaluation

The authorization model described above can be represented and implemented in several ways. Following the same approach as in previous works in this PhD thesis, here it is proposed a representation based on an ontology defined in the Ontology Web Language 2 (OWL 2). As commented in previous chapters, this language is based on formal methods and it constitutes a remarkable added value since it provides powerful semantics to define the authorization model and enables the usage of reasoners for automatic computation of that model. Reasoners are able to process complex queries about the authorization model, inferring additional information and performing a formal validation and verification.

The inference capability of reasoners can be used to automatically compute the set of conditions of Corollary 1, which are needed to decide whether a given subject is granted to access an object. This highly simplifies the deployment to be done in a CSP to support data protected with this solution, since there are several standard OWL reasoners available. An additional advantage of using ontologies to represent the authorization model is that it allows the application of advanced techniques such as conflict detection methods as explained in Section 3.3.3. In this case, the OWL 2 EL profile has been selected to represent the authorization model among the three different profiles available for OWL 2 and described in Section 3.1. It supports the required expressiveness (e.g. transitive object properties) and there are high performance reasoning algorithms for this profile. Moreover, since this proposal is mainly intended for data protection following a data-centric security approach instead of protecting services or other kind of resources, the domain representation in CIM used in the previous chapters is not necessary. Thus, an ontology based on the protected authorization model is used instead. Figure 5.1 shows a direct representation in OWL of the protected authorization model described in section Section 5.3.1.

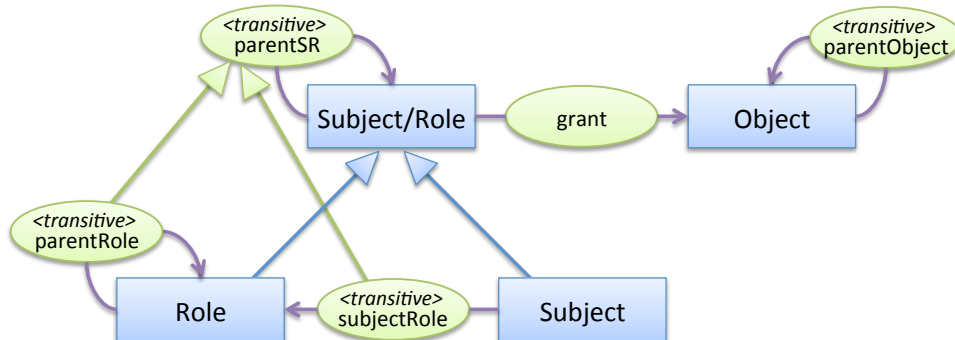


Figure 5.1: Ontology representing the authorization model

The sets representing different entities of the authorization model are represented as classes of the ontology. Namely, the set $\mathbb{O}$ is represented by the class *Object*, $\mathbb{A}$ by *Action*, $\mathbb{S}$ by *Subject*, $\mathbb{T}$ by *Role* and the class *Subject/Role* represents the union of sets $\mathbb{S} \cup \mathbb{T}$. The binary relations of the protected authorization model are directly mapped to object properties in the ontology, whose domains and ranges are the domains and codomains of the binary relations, respectively. Thus, the binary relation G_a is represented by the property *grant*, D by *subjectRole*, E by *parentRole* and F by *parentObject*. OWL provides the axiom *transitive object property*, which enables making a property to be transitive. A transitive property interlinks two elements A and C whenever it interlinks A with B and B with C for some element B. This feature has been used to define the transitive closures of the sets E and F , i.e. E' and F' , respectively. The axiom has been used to define *parentRole* and *parentObject* as transitive properties. The inference capability of reasoners will automatically compute the transitive closures for the corresponding binary relations. Moreover, OWL provides axioms that enable to define inheritance of object properties the same way one can define inheritance of classes. A parent object property can be defined that encompasses both *subjectRole* and *parentRole* object properties in order to simplify the query that will be done to the ontology. This property is called *parentSR* (abbrv. *parentSubjectRole*) meaning that either when a subject A is assigned to a role B or when a role B is parent of another role C, this role B can be considered a parent of either subject A or role C.

The ontology can be queried to check whether a given subject is granted to access a given resource, according to the conditions of Corollary 1. Querying the ontology includes instance recognition and inheritance recognition. The former consists in testing if an element is an instance of a class expression and the latter in testing if a class is a subclass of another class or if a property is a sub-property of another. This enables the formulation of generic queries referring to abstract classes and properties, being the reasoner able to recognize instances belonging to concrete subclasses or sub-properties. Queries are usually defined using SPARQL [88], the query language fostered under the Semantic Web technologies. Listing 5.1 shows the query in SPARQL to obtain the grant specified in Corollary 1 by checking the conditions expressed in (5.17), (5.18) and (5.19).

```

SELECT ?authorizedRole ?authorizedObject
WHERE {
  {
    <subject> <GRANTED> <object>
  } UNION {
    <subject> <GRANTED> ?authorizedObject .
    <object> <PARENT_OBJECT> ?authorizedObject
  } UNION {
    ?authorizedRole <GRANTED> <object> .
    <subject> <PARENT_SUBJECT> ?authorizedRole
  } UNION {
    ?authorizedRole <GRANTED> ?authorizedObject .
    <subject> <PARENT_SUBJECT> ?authorizedRole .
    <object> <PARENT_OBJECT> ?authorizedObject
  }
}

```

<pre> } } </pre>	15
<pre> } </pre>	16

Listing 5.1: SPARQL authorization query

The set of conditions in Corollary 1 presents four different combinations, which are reflected in the four *UNION* statements of the *WHERE* clause of the query. If $u_1 \in \mathbb{S}$, then by (5.17) $u_1 = s_1$ and the access is granted directly to the subject. Then, by (5.19) either $(o_1 = o_2)$ or $(o_1 F' o_2)$. The first case is reflected in line 4 where the grant directly links with s_1 and o_1 . The second case is reflected in lines 6 and 7, where the grant directly links with s_1 and indirectly links with o_1 through o_2 , which is represented by the variable *?authorizedObject*. Otherwise, $u_1 \in \mathbb{T}$ and by (5.18) $\exists r_1 \in \mathbb{T} \mid (s_1 D r_1) \wedge ((r_1 = u_1) \vee (r_1 E' u_1))$. This means that the grant indirectly links with s_1 through a role r_1 , which is represented by the variable *?authorizedRole*. This is reflected both in lines 9 and 10 as well as in lines 12 and 13. Note that this condition of (5.18) can be also specified as $((s_1 D r_1) \wedge (r_1 = u_1)) \vee ((s_1 D r_1) \wedge (r_1 E' u_1))$. Both cases are covered by the *parentSubject* property, which is a super property for both *subjectRole* and *parentRole*. Then, by (5.19) either $(o_1 = o_2)$ or $(o_1 F' o_2)$. The first case is reflected in line 9 where the grant is directly linked with o_1 . The second case is reflected in lines 12 and 14 which indirectly links the grant with o_1 through o_2 represented by the variable *?authorizedObject*.

The output of this query is a set of data structures containing the possible solutions (statements in terms of the ontology) that fulfill the conditions specified in the *WHERE* clause. If no solution is found, then the access is denied since the conditions of Corollary 1 are not met. In this case, data cannot be decrypted since conditions of Theorem 2 are not met and there is no re-encryption chain $(r_0, \dots, r_n)$ that re-encrypts the protected object o_1 from id_{o_1} to id_{s_1} . Otherwise, the possible results should be processed to obtain the adequate re-encryption chain. It should be noted that more than one solution is possible. In this case, the solution resulting in the shorter re-encryption chain would be preferable to reduce the number of re-encryptions (cryptographic operations) that should be done.

Commonly speaking, for the reasoner to find a solution, it should find a path in the ontology between the object o_1 and the subject s_1 . This path represents the set of re-encryptions that should be done to transform some object encrypted under id_{o_1} to that object encrypted under id_{s_1} . The path is the one represented in (5.21), where binary relations in labels directly map with object properties of the ontology. The re-encryption keys are the ones assigned to each element in the sets of the protected authorization model. For instance, the binary relation *D* directly maps to the object property *subjectRole*, which has re-encryption keys assigned by the set of protected subject-role assignments (5.25) as defined in Def. 10.

The concrete path for each solution can be obtained by processing the output of the reasoner. The two variables *authorizedRole* and *authorizedObject* determined by the *SELECT* clause of the query represent the two intermediate elements o_2 and r_2 that are granted when g_1 is not directly linked to s_1 or o_1 . If any of these variables is not assigned

to a value in the query result, this means that g_1 is directly assigned to s_1 , the o_1 or both. In turn, the result structure for each solution contains the set of derivations performed by the reasoner to deduce that solution. This information is used to construct the path and, consequently, the re-encryption chain. Since the ontology is a direct representation of the sets and relations of the protected authorization model, the derivations done by the reasoner are also directly mapped to the original sets and relations. That is, the inference process done by the reasoner has automatically computed the re-encryption chain $(r_0, \dots, r_n)$ specified in Theorem 2.

5.4 Data-centric solution for Cloud data protection

In the protected authorization model specified in Section 5.3.1, it should be observed that data is not encrypted with the data owner identity, but with the object's own identity (e.g. id_{o_1}). This follows a data-centric approach for data protection, in which data is encrypted with its own key under the cryptographic scheme. If a pure PRE scheme is used, the object would be also encrypted using its own key pair. On another hand, a user-centric approach is used for the authorization rules, where a unified access control policy is defined by the data owner for its data. This allows to share common definitions and to greatly simplify access control management, getting the most from role hierarchy and resource hierarchy capabilities.

An architecture is also proposed for the deployment within a CSPs. This architecture takes into consideration the different elements that should be deployed in order to give an overview of how access to protected data is done in this approach. Figure 5.2 depicts the proposed architecture.

When moving data to the cloud, a self-protected package is generated by the data owner. This package contains: the encrypted data objects, the authorization rules and the corresponding re-encryption keys.

Data objects are encrypted before uploading them to the Cloud in order to prevent the CSP to access them. This is done by data owners by using the *encrypt()* function (5.3). According to Def. 8, data should be encrypted using the identity id_{o_1} of the object being uploaded o_1 . A digital envelope approach can be applied to protect data objects instead of direct encryption. This would enhance cryptographic operations like re-encryptions for large data objects. This approach consists in using a symmetric encryption algorithm (e.g. AES) to protect the data object itself. The encryption of data is done with a random symmetric key generated for the purpose of a single encryption. Then, this key is encrypted with the *encrypt()* function. With this procedure, potentially big objects of data (e.g. large documents) are encrypted using symmetric cryptography, whose algorithms are more efficient for data processing. In turn, more costly operations are only applied to the keys used for the symmetric encryption, which are usually small pieces of data of some bytes of length.

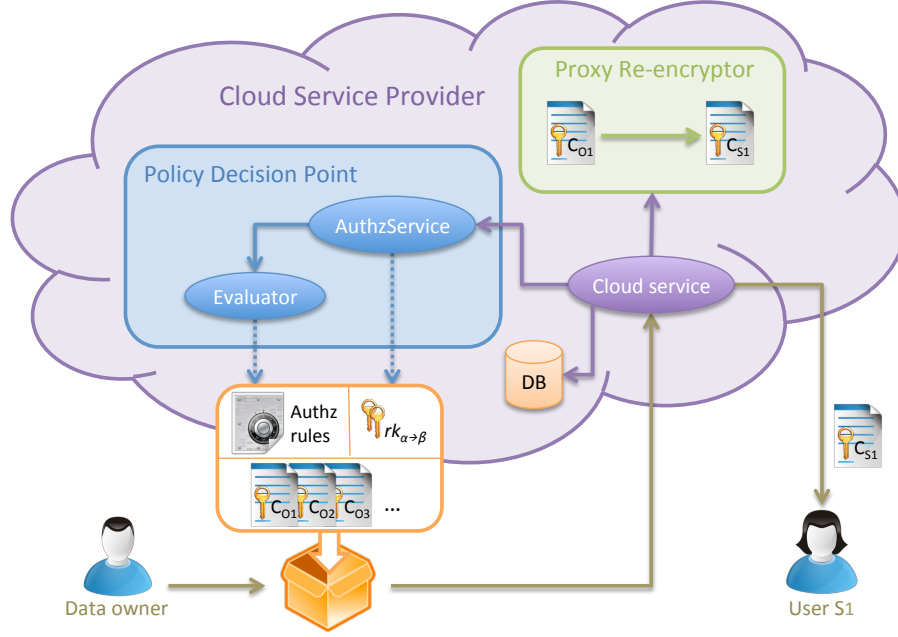


Figure 5.2: Architecture for deployment in a CSP

Authorization rules are defined by the data owner and directly mapped into the authorization model. This is done by including the corresponding elements in the binary relations. For instance, a rule that grants a subject s_1 access to an object o_1 would add an element $g_1 = (s_1, o_1)$ to the binary relation G_a . Depending on the representation of the authorization model, the corresponding rule representation should be generated. Following the same example, in case of an ontology representation, a *granted* object property linking s_1 to o_1 should be included.

Every authorization rule should be accompanied by its corresponding re-encryption key, which is generated by the $rkgen()$ function (5.4). This re-encryption key can only be generated by the data owner using the Master Secret Key and the identity of the involved authorization elements. Note that, although the $rkgen()$ function initially takes the Secret Key of one element, this key can be really computed on demand using the $keygen()$ function, which only needs the Master Secret Key and the identity. That is, by (5.2) and (5.4) $rkgen(p, keygen(p, msk, id_\alpha), id_\alpha, id_\beta) \rightarrow rk_{\alpha \rightarrow \beta}$. This is the main advantage of using IBE instead of a pure PRE scheme. The usage of IBE enables the generation of re-encryption keys with just the MSK and the identities of the authorization elements, avoiding the need to store dedicated public and private key pairs for each authorization element.

The re-encryption keys can be considered as cryptographic tokens provided by data owners in order to enable to perform operations over the authorization model. That is, a CSP will not be able to effectively apply any rule that has not been legitimately defined

by the data owner. If a CSP tries to do so, it would result in a re-encryption chain that would miss some re-encryption keys of the path, being that re-encryption chain useless to access the data.

For privacy purposes, pseudonyms can be used as identities of authorization elements instead of real identifiers. Although the proposed solution avoids the CSP to access the data, it still can infer some knowledge such as the roles or users managed within the data owner organization. Using pseudonyms would avoid the CSP to know the actual identities of such elements.

The application of this data-centric approach results in an self-contained protected object, which can be released to the Cloud and only authorized users could access the data object. This can be useful for Inter-cloud scenarios, where data can travel through different CSPs.

Regarding the CSP, there are two main modules to be deployed: a *Policy Decision Point* (PDP) which manages the authorization model and a *Proxy Re-encryptor* module that performs the cryptographic operations. Cloud services offered by the CSP interact with these two blocks in order to securely manage the data of users. They also make use of a database to store the protected packages uploaded by data owners.

An authorization service (*AuthzService*) acts as entry point to the PDP for Cloud services allowing to query it for authorization decisions. This module takes decisions upon a request from a user s_1 to access to a piece of data o_1 managed by the service. These decisions usually return an access granted or denied statement. For granted accesses, the response also contains the re-encryption chain that should be applied, together with the needed re-encryption keys. This information allows to re-encrypt from c_{o_1} as provided by the data owner to c_{s_1} , which can be decrypted by the requesting user. The service passes this information together with c_{o_1} to the *Proxy Re-encryptor* for this to perform the re-encryption operations. It results in c_{s_1} , which is sent to the requesting user. Making use of its own secret key sk_{s_1} the user can decrypt the data with the *decrypt()* function (5.6). Note that during this process, the CSP is not able to access the data since it only applies a set of *reencrypt()* functions which do not disclose any information about the data being processed.

The PDP considers two different sources of information: authorization rules and re-encryption keys. This information is provided in the protected packages provided by the data owner. As mentioned before, the former contains a representation of the protected authorization model as described in section Section 5.3.1. This can be the ontology explained in Section 5.3.2 or any other valid representation of the model. The latter contains the set $\mathbb{K}$ of re-encryption keys. The *AuthzService* makes use of an evaluator module in charge of making the computations over the authorization model. This module depends on the representation of the model that is being used. In case of an OWL ontology, it corresponds to a reasoner engine able to perform queries over the model. The results of this module are processed by the *AuthzService* to form an authorization decision. In case of a positive decision, it also obtains the corresponding authorization chain and the

re-encryption keys for such a chain. This information conforms the output of the *AuthzService* for a request that is passed to the *Proxy Re-encryptor* module.

5.4.1 Key management

As exposed in section Section 5.1, the IBPRE scheme does not use public and private key pairs in its cryptographic operations. Instead, a Master Secret Key (MSK) is used in combination with identities. This MSK is generated during the setup phase and it should be kept private, avoiding the CSP to access it. On another hand, users accessing the data need their own Secret Key (SK) in order to compute the *decrypt()* function. These secret keys are generated based on the identity of the user and the MSK. There are several approaches for the distribution of these keys to users. In a straightforward approach, SKs can be generated internally by the data owner to keep the MSK protected. However, this will lead to the need of distributing SKs securely to each user.

To this end, IBE schemes -including IBPRE- usually make use of a Private Key Generator (PKG) for the generation and distribution of SKs to users. This entity should be trusted by the data owner because it holds the MSK to generate the SKs. It can be deployed as a service by the data owner in its own premises. This would allow to keep the MSK under control, although it would result in a critical service that should be protected. Another choice would be a third trusted entity that provides the PKG service. In this case, it is of paramount importance to have guaranteed that this entity does not collude with the CSP.

As an alternative, an hybrid proxy re-encryption approach can be applied. This concept was introduced in [100] and it consists in creating a bridge between IBE and Public Key based Encryption (PKE). The IBPRE scheme used in this proposal supports this feature. Thus, it can be used to manage user keys by using well known and standard technologies like a Public Key Infrastructure (PKI). This feature implies the inclusion of two new functions defined as:

$$rkgen_pke(p, sk_\alpha, id_\alpha, pub_\beta) \rightarrow rk_{\alpha \rightarrow \beta} \quad (5.30)$$

$$decrypt_pke(p, priv_\alpha, c_\alpha) \rightarrow m \quad (5.31)$$

The functions are similar to the original ones (5.4) and (5.6), but including some modifications. Details about the modifications that need to be done to these functions can be found in [99]. These functions take public and private keys pub_β and $priv_\alpha$ to apply PKE instead of identity id_β and Secret Key sk_α used for IBE.

Applying these functions makes the re-encryption scheme to lose the *Multi-use* feature, which is required as described in section Section 5.1. That is, once a Re-encryption Key generated by $rkgen_pke()$ is used to re-encrypt, no further re-encryptions can be

done to that encrypted object. However, for the purposes of authorization in this proposal, this kind of re-encryption only needs to be done to re-encrypt the protected object under the requesting user public key. And this is done in the last re-encryption, which is the one that results in the data being encrypted under the user public key. Thus, re-encryption keys generated with the original $rkgen()$ function should still be applied for re-encryptions along the authorization path, except the one affecting the user, which is the last re-encryption.

In practical terms, using the hybrid approach only implies that re-encryption keys affecting subjects $s_i \in \mathbb{S}$ should be generated with the $rkgen_pke()$ function. That is, when the data owner defines a rule to grant a privilege or assigns a role to a given subject, the corresponding re-encryption key should be generated with (5.30). Otherwise, re-encryption keys should be generated with the original $rkgen()$ function for the rest of the authorization elements.

With this approach, the data owner uses the public key of the user when defining rules in the authorization model. Upon a request, the data object is re-encrypted under the requesting user public key. This user can then decrypt the data by using the corresponding private key. Hence, key management results in managing public and private key pairs of PKE, which can be done by means of commonly used and standard PKI solutions.

It is worth mentioning that the two approaches can be also combined. That is, some users can use PKE while others can still use IBE. Then only thing that needs to be done by the data owner is to use the proper function ($rkgen_pke()$ or $rkgen()$) when generating the corresponding re-encryption keys.

5.4.2 Security considerations

In order to keep the security of the authorization system, some considerations should be taken into account during deployment. The following conditions should hold to securely protect data in the cloud with the proposed solution:

- The CSP should not be able to access the MSK.
- The CSP should not be able to access Secret Keys of authorization elements.
- If a PKG is used, it should be guaranteed that it does not collude with the CSP.

If the CSP gains access to the MSK, it could derive Secret Keys to gain access to the data. It could also generate new re-encryption keys that would allow to apply new authorization rules not defined by the data owner. This would lead to the CSP being able to release data to unauthorized users or third parties.

If the CSP gains access to Secret Keys of authorization elements, it could use these keys to generate new re-encryption keys. As stated before, this would enable the CSP to

release data to unauthorized parties. Moreover, it could also try to find out a re-encryption chain that results in an encrypted object under an identity whose Secret Key is known for the CSP. This would enable the CSP to apply this re-encryption and use the Secret key to access the data.

These two considerations imply that the operations *setup()*, *keygen()* and *rkgen()* should be performed by the data owner. And their output, i.e. keys, should be kept secured by the data owner.

Finally, in case of using a PKG, the collusion of this entity with the CSP may also result in the CSP accessing the data. The CSP is able to search the authorization model for a re-encryption chain that results in an encrypted object under the identity of a user whose secret key is known for the PKG. If the PKG provides this key to the CSP, it would be able to perform the re-encryption and access the data.

5.5 Implementation and performance

A prototypical implementation has been developed to demonstrate the feasibility of the proposal. It has been integrated into Google Cloud services to provide security to documents stored in Google Drive. Since the core of Google Services cannot be modified, integration has been done by developing a Web application running on Google App Engine. This application is registered as a Google Drive application that is integrated into the Google Drive user interface. The Web application contains the modules depicted in Figure 5.2.

The authorization model has been represented as described in Section 5.3.2 using Semantic Web technologies. So, the *Evaluator* in this implementation consists on an ontology reasoner. The Apache Jena framework has been used to manage the ontology and to perform reasoning. The *AuthzService* also makes use of this library to process the output of the reasoner and retrieve the authorization chain.

An implementation of the IBPRE scheme has been developed using elliptic curve cryptography. The Java Pairing-Based Cryptography Library (JPBC) and the Bouncy Castle Crypto APIs have been used for the cryptographic operations. Documents uploaded by the data owner to Google Drive are encrypted by using digital envelopes. They are encrypted using AES-256 with random keys. These keys are then encrypted with the *encrypt()* function. The format of the resulting document has been developed to be compliant with the standard Cryptographic Message Format (CMS) [101].

The implementation supports both PKG and PKI for key management. It also allows the data owner to directly generate and store user keys in case he wants to distribute them by other means. When generating a re-encryption key related to a privilege or role assignment, the application asks the data owner to choose between an IBE or PKE key for that user. In case of IBE, the key will be generated automatically based on the user

identity, otherwise it will ask for the public key of the user.

An analysis has been carried out based on this implementation to test the feasibility of the proposal in terms of performance. Tests have been done with an Intel i5 CPU at 2.7 GHz and 6 GB of RAM. A first set of tests consisted on measuring execution times for the cryptographic functions exposed in Section 5.1.

In order to obtain statistics significant results, operations have been performed in sets of 100 executions, whose average is used as result value. Each execution performs the following steps. First, the *setup()* function (5.1) is executed to initialize the cryptographic scheme. Then a piece of data m is encrypted under a randomly generated identity id_1 with the *encrypt()* function (5.3) to obtain a ciphertext c_1 . The corresponding Secret Key sk_1 is generated with the *keygen()* function (5.2). Then, another random identity id_2 is generated and a re-encryption key $rk_{1 \rightarrow 2}$ is generated with the *rkgen()* function (5.4). This is used to re-encrypt the c_1 with the *reencrypt()* function (5.5). These three last steps are repeated several times, resulting in a ciphertext c_n under identity id_n after n re-encryptions. Finally, the *decrypt()* function (5.6) is used to decrypt c_n and obtain the plain data m . The length of the plain data m , the length of identities id_i and the number of re-encryptions may vary depending on the test. Different tests have been done by changing one of these parameters in order to tested the functions under different circumstances. When a parameter is not changed in a test, default values are 512 bytes for data length, 32 bytes for identity lengths and 100 for the number of re-encryptions.

In a PRE scheme, some operations could be affected by the number of re-encryptions, while others may be independent. A first test has been done by varying the number of re-encryptions from 1 to 100 by incrementing in 10 re-encryptions for each execution set. Figures 5.3 and 5.4 show the results for this test. The *encrypt()* time is not shown because it is the same as the *rkgen()* time and their lines are overlapped in the graphic. Times for the *setup()* and *decrypt()* operations are shown in a separate graphic because they present higher values and showing them together with the rest of functions would distort the Y axis scale. As can be observed, the *setup()*, *keygen()*, *encrypt()*, *rkgen()* and *reencrypt()* functions remain constant independently of the number of re-encryptions. However, *decrypt()* increases with the number of re-encryptions. This is because re-encryptions are applied one over another in the ciphertext and the *decrypt()* has to undo these re-encryptions. So, the more re-encryptions are applied to a ciphertext, the more computations have to be done.

It is worth mentioning that the number of re-encryptions depends on the expressiveness used by the data owner when defining the authorization rules. Re-encryptions for an access request can be observed in (5.21). At least one re-encryption should be done. This is the case when an access grant in the binary relation G_a is directly granting the requesting user access to the requested object. If roles are used, then at least two re-encryptions should be done. The one for the access grant and another one for the subject-role assignment in D . Then, if hierarchical expressiveness is used, several re-encryptions could be needed for the parent-role and parent-object assignments in E and F , respectively. Thus,

the number of re-encryptions would depend on the hierarchical levels that are defined between the role of the requesting user and the granted role plus the levels between the requested and the granted object. It should be noticed that this does not mean the number of roles or the number of objects defined in the authorization model, but levels in their hierarchies. The number of roles or objects do not affect performance of cryptographic operations. In practical terms, role and object hierarchies should present just a few levels. Otherwise, it would result in a complex role and object structure. The test has been done with 100 re-encryptions in order to stress the system to the limit. However, 10 re-encryptions (20 at most) should be enough for a realistic scenario. For this number of re-encryptions, *decrypt()* remains under acceptable execution times.

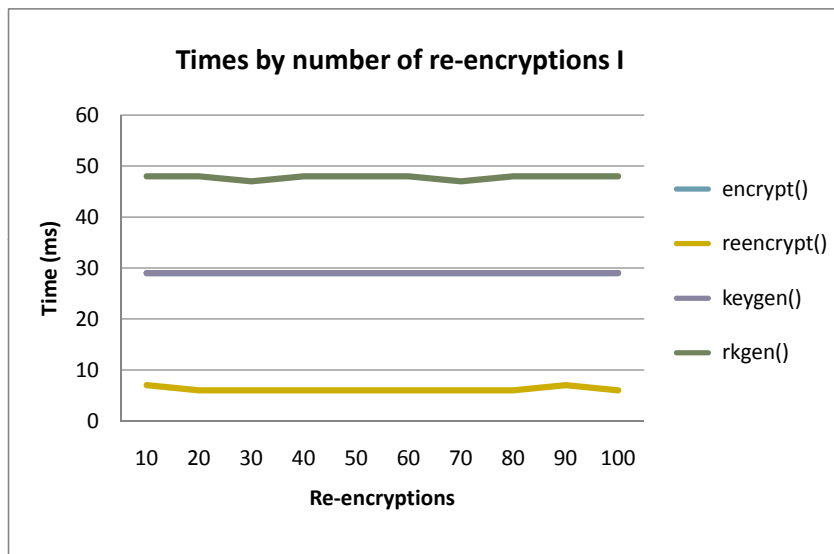


Figure 5.3: Times changing the number of re-encryptions I

In turn, in an IBE scheme, the length of the identities used for the cryptographic operations may also affect the execution times. Another test has been done by varying the length of the identities from 8 to 512 bytes by incrementing in 64 bytes for each execution set. Figures 5.5 and 5.6 show the results for this test. Results do not show any variation for the cryptographic functions. Theoretically it should affect functions dealing with identities, i.e. *keygen()*, *encrypt()* and *rkgen()*. However, the considered lengths for these strings in practical terms are usually small since they are just identifiers. Moreover, having a look to the internal operations, processing done to these strings within the functions is so small that it is also negligible for the execution time.

Finally, it is also interesting to see how times could be affected by the length of the plain data being encrypted. Another test have been carried out by varying this parameter. In this case, four lengths have been considered: 128, 192, 256 and 512 bits. The first three correspond to the common key lengths used for AES encryption. The last one has been also added to test for stronger AES encryption with 512 byte keys as some proposals are

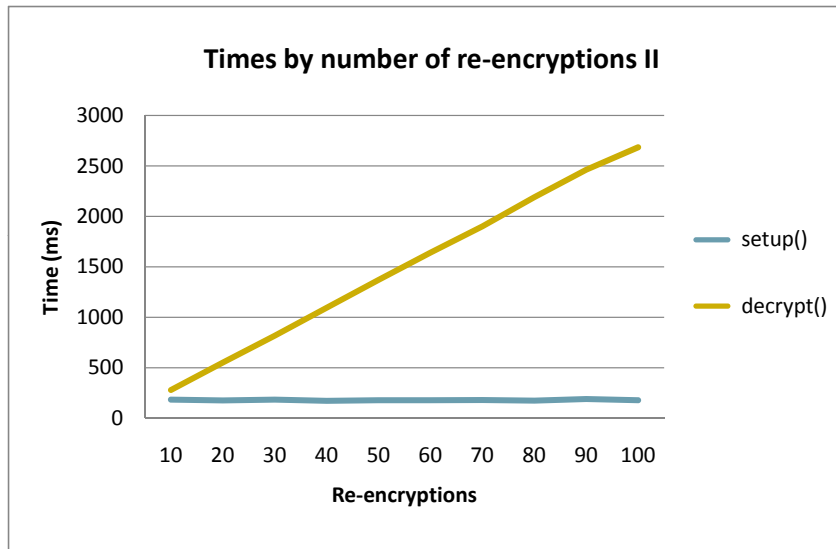


Figure 5.4: Times changing the number of re-encryptions II

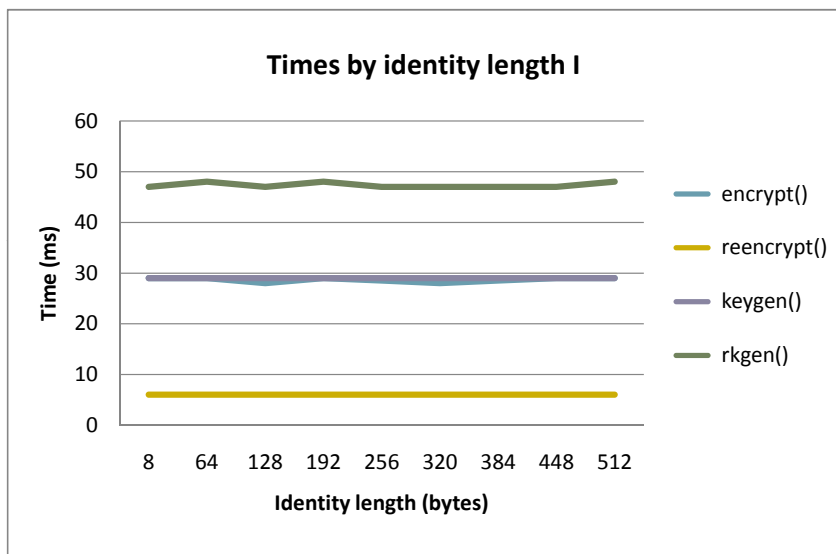


Figure 5.5: Times changing the length of identities I

currently arising [102]. It has been decided to make this test for these lengths because it is the AES key (or any other symmetric encryption algorithm) what would be actually encrypted in practice when using a digital envelope approach, as described in Section 5.4. Figures 5.7 and 5.8 show the results for this test. Again, results show a constant execution time for all functions. Theoretically, *encrypt()*, *reencrypt()* and *decrypt()* could be affected by the length of the data. However, during encryption data is transformed to its

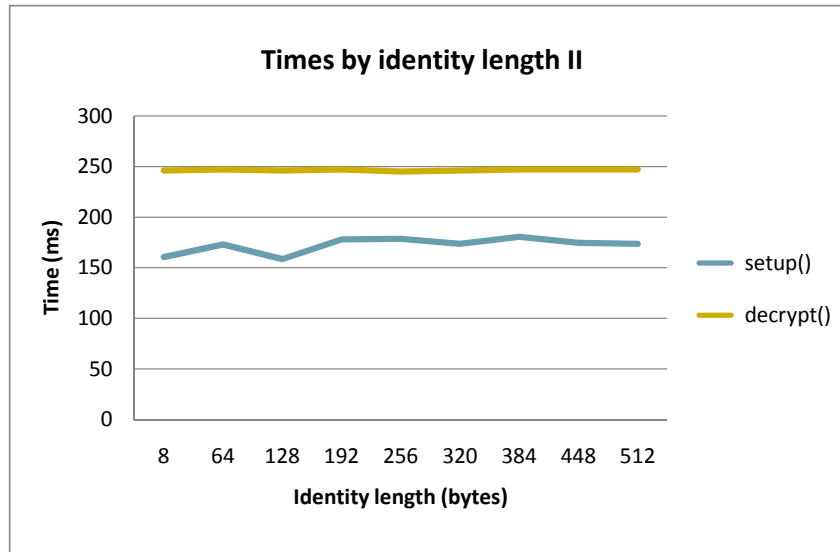


Figure 5.6: Times changing the length of identities II

numeric representation. Then, cryptographic operations are done with this representation, which is not affected by the length of the original data.

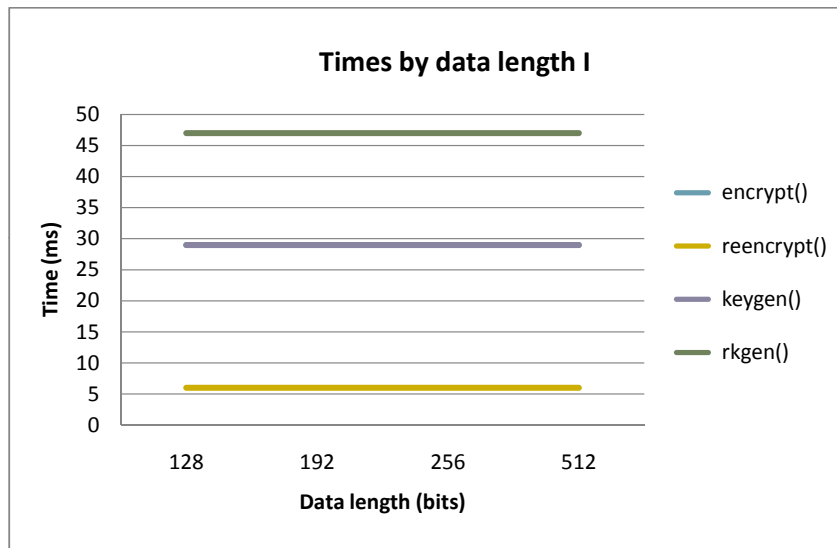


Figure 5.7: Times changing the length of encrypted data I

To sum up, the average execution times obtained during the tests are: 178 ms for *setup()*, 29 ms for *keygen()*, 29 ms for *encrypt()*, 48 ms for *rkgen()*, 6 ms for *reencrypt()* and 247 ms for *decrypt()* with 10 re-encryptions.

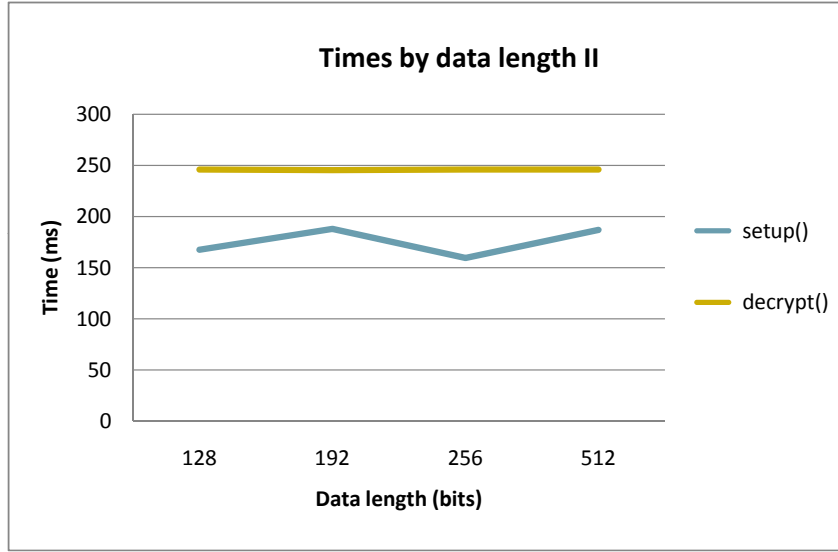


Figure 5.8: Times changing the length of encrypted data II

Another set of tests has been also performed to measure the time needed by the *AuthzService* to take authorization decisions. The authorization model has been represented based on Semantic Web technologies as described in Section 5.3.2. Decision times are related to the number of authorization elements defined within the model, i.e. number of roles, users, grants, objects, etc. These elements are represented in what is called a Knowledge Base (KB), which is used by the reasoner implementing the *Evaluator* to evaluate the authorization query. The elements contained in the KB are called individuals and the set of individuals for a specific execution is referred to as population. In order to study the scalability of the proposal, 6 executions have been done using incremental populations, following a similar approach as in previous chapters of this PhD thesis for performance evaluation. This approach allows getting results with an increasing order of authorization elements. In order to quickly increment the number of individuals, this increment follows an exponential distribution. Table 5.1 shows the number of individuals for each population.

Population	1	2	3	4	5	6
Individuals	500	900	1,400	2,300	3,700	6,100

Table 5.1: Number of individuals by population

Figure 5.9 shows the times obtained during the tests for each population. It shows accumulated times for the three main operations done by the *AuthzService*. It first evaluates the authorization model by calling the *Evaluator*, which executes the SPARQL query in Listing 5.1. In case of a positive answer, the output is processed to compute the autho-

rization chain. Then, the corresponding re-encryption keys for that chain are retrieved. In order to obtain complete times, only executions with a granted decision are taken into account. Access denied decisions would result in shorter times since the authorization chain does not need to be computed.

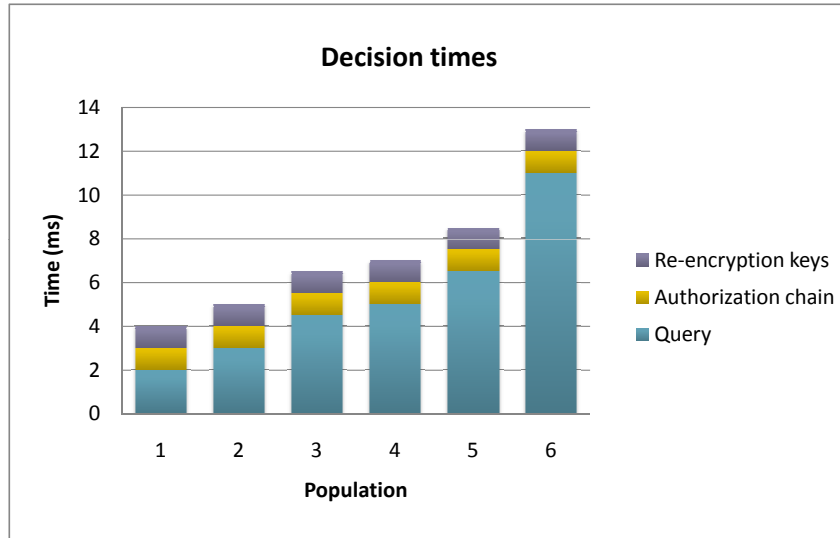


Figure 5.9: Authorization decision times

As can be observed, the time to perform the query is incremented as the population grows, while times for building the authorization chain and retrieving re-encryption keys remain constant. This is because the query need to analyze the ontology in order to check the conditions. However, computing the authorization chain from a given answer is independent of the ontology. So is the retrieval of re-encryption keys. Times for authorization decisions remain below 13 ms for populations of more than 6000 authorization elements.

Chapter 6

Conclusion and future work

This chapter depicts some conclusions about the research work carried out during the realization of this PhD thesis and describes future lines of research and statements of direction.

The main contributions of this research work are summarized, providing a final overview of the achieved results. Then, some ideas and open points that could not be covered by this work are introduced as guidelines for further research.

6.1 Conclusions and main achievements

The management of authorization for the resources and services of a Cloud system results in a complex task which should be faced by administrators. Policy-based approaches for authorization provide an intuitive and scalable way to keep large information systems under control. Policies are specified by means of rules which are based on domain concepts which are also modelled by the authorization solution. Policy and domain languages with high expressiveness, which allow the representation of the resources to be protected and their semantics, would ease the authorization management task. Moreover, the protection of policy rules with cryptographic techniques enables the data owner to keep full control over data moved to the cloud, assuring a reliable and trustworthy application of these rules by any CSP.

This PhD thesis has presented a semantic-based approach enabling authorization by using Semantic Web technologies. High expressiveness for policy and domain definitions is provided, reducing the gap between abstractions and reality to aid administrators in authorization management. The adoption of an RBAC-based approach is also closer to current authorization solutions, resulting more natural for administrators to adopt the proposed model. A semantically enriched authorization model has been proposed, overcoming some of the limitations in expressiveness available in its predecessors and combining the main features present in previous proposals as analyzed in Section 2.2. It provides

support for advanced authorization with enhanced RBAC expressiveness such as hierarchical RBAC, conditional RBAC and hierarchical objects.

Semantic Web technologies have been demonstrated as useful for describing authorization models. The usage of Semantic Web technologies and languages are able to endow an authorization system with automated reasoning capabilities, making it able to infer additional information not explicitly specified and to perform a formal validation of the constraints specified in the ontology. These reasoning capabilities enable advanced features like concept alignment and policy conflict detection. The usage of ontological models for expressing both domain information and authorization policies avoid any mismatch between the semantics of both representations, which is a potential problem which usually appears in most current authorization proposals for Cloud computing. Ontologies are used to provide a representation of the underlying information system and the resources to be protected. This allows the inclusion of semantics, providing administrators with language constructors to define object, property and cardinality restrictions, among other advanced features for domain definition. The usage of the Common Information Model to represent and define domain information and to express policy definitions supposes an added value for these domain definitions. It constitutes an information model accepted internationally that additionally provides basic requirements for a modeling language such as well defined semantics, flexibility, extensibility, and independent of administrative domain. Moreover, being a widely used OASIS standard, it can be also combined with already existing management mechanisms like WBEM, which would ease the management of domain information or even provide a link to the real system. Alternatively, other ontological models can be also used for domain representations. The usage of ontologies enables support for multiple authorization domain representations, dealing with heterogeneity of distributed systems and enabling organizations to use their own domain concepts without having knowledge about the rest of participant organizations. An automatic concept alignment is performed to use all this heterogeneous information when taking authorization decisions.

The proposed approach for authorization also represents one step towards the automatic management of security services, considering not only authorization services, but also providing additional reasoning mechanisms to cope with issues such as detection of conflicts between different policies. The usage of ontologies that provide semantically enriched information about both the domain and the authorization policies enables the automatic detection of semantic conflicts in the authorization model. Semantic conflicts usually result more difficult to detect than modal policy conflicts, since they not only depend on the definition of the policies, but also on the semantics related to the specific domain to which they are applied. Ontologies allow to capture such a semantic information and define a set of meta-rules specifying conflicting behaviours and definitions. The automatic reasoning capabilities provided by reasoners that evaluate the policies are able to detect the conflicts based on the analysis of the meta-rules and taking into consideration the semantics of the domain.

An authorization architecture specifically designed to tackle the multi-tenancy and

heterogeneity properties of Cloud computing has been also proposed. The architecture is based on Web services and it has been integrated and tested in both Grid and Cloud computing scenarios. The architecture provides the components needed to support the authorization model to manage and control resource sharing in order to establish who is allowed to perform what actions. Advanced federation capabilities among tenants have been also successfully supported. A fine-grained trust model for managing federations in Cloud computing environments has been also provided. This trust model enables the management of possible business alliances (coalitions and federation) that could arise among the Cloud tenants.

The authorization scheme is enhanced with a data-centric access control solution for self-protected data that can run in untrusted CSPs and provides extended Role-Based Access Control expressiveness. The proposed data-centric solution follows a rule-based approach, where rule specifications are also based on Semantic Web technologies that enable enriched rule definitions and advanced policy management features. Access control computations are delegated to the CSP, being this not only unable to access the data, but also unable to release it to unauthorized parties. Advanced cryptographic techniques have been applied to protect the authorization model. A re-encryption key complement each authorization rule as cryptographic token to protect data against CSP misbehavior. The IBPRE scheme has been used in order to provide a comprehensive and feasible solution. The application of Semantic Web technologies for the representation and evaluation of the authorization model supposes a novel approach in data-centric solutions, which usually lack of advanced management features such as policy reasoning or conflict detection capabilities. Guidelines for deployment in a Cloud Service Provider have been also given for this data-centric solution, including an hybrid approach compatible with Public Key Cryptography that enables the usage of standard Public Key Infrastructure (PKI) for key management and distribution.

The data-centric model presented in this PhD thesis supposes a novel approach for a data-centric RBAC-based model for access control in which data is encrypted and self-protected, offering an alternative to the ABAC model. An RBAC approach would be closer to current access control methods, resulting more natural to apply for access control enforcement than ABE-based mechanisms. In terms of expressiveness, it is said that ABAC supersedes RBAC since roles can be represented as attributes. However, when it comes to data-centric approaches in which data is encrypted, ABAC solutions are constrained by the expressiveness of ABE schemes. The cryptographic operations used in ABE usually restrict the level of expressiveness for access control rules. For instance, role hierarchy and object hierarchy capabilities cannot be achieved by current ABE schemes. Moreover, they usually lack some combination with a user-centric approach for the access control policy, where common authorization-related elements like definition of users or role assignments could be shared by different pieces of data from the same data owner.

The work carried out during the realization of this PhD thesis has been structured in three major researches that correspond to the three main chapters of this document. Namely, chapters 3, 4 and 5. These researches conform complete research works by

themselves, including proof of concept implementations with performance and statistical results in order to test the feasibility of the proposal at the end of each research work. This has permitted to evaluate the evolution of the proposal through the three researches, assuring a suitable solution as the complete PhD research progressed.

An initial proof of concept implementation has been tested in Globus against several system scenarios, from more dynamical to more statical ones. Then, the refined model and architecture for Cloud computing has been validated by means of a prototype implementation for the OpenStack platform using both Java and Python technologies. Finally, a proof of concept implementation of the data-centric proposal has been also developed and a working prototypical deployment of the proposal has been integrated within Google services.

A similar evaluation approach has been followed for the three research works, where different work loads have been employed in the testbed, evaluating the solution by using populations with an increasing number of individuals present in the knowledge base. The provided statistics show the behaviour of the reasoning and query processes under different situations, achieving satisfactory authorization times and good performance results in all cases.

The availability of advanced authorization capabilities can be a differentiating feature for Cloud Service Providers, which demand the design of suitable authorization models to enhance the access control to the Cloud resources. From the point of view of a CSP, the benefits of the approach presented in this PhD thesis may result of particular interest. The authorization model presented herein enables the deployment of services for customers with quality of service guarantees over the security and privacy, which would increase the trustworthiness of customers and encourage them for the adoption of Cloud computing as new IT paradigm. Users would also benefit of enhanced security in Clouds for their resources and the data-centric approach would enable them to securely move data to the Cloud being sure that it will not be accessed or released by the CSP unless authorized.

The research results achieved during this PhD thesis have been published in a set of international scientific publications, including international journals, conferences and book chapters. It is also an indicator of the quality and impact of the work done that the main core of this research, exposed in chapters 3 to 5, has been published or is being reviewed in three international journals with JRC Q1 index.

6.2 Future work and research guidelines

As statement of direction, the proposed authorization model and architecture can be extended to cover additional requirements for future Cloud deployments, such as the movement towards the Intercloud, where federation and interconnection between Cloud Service Providers take place. As stated in [7], “recent trends in cloud computing go towards the development of new paradigms (heterogeneous, federated, distributed clouds) as opposed

to the current centralised model, with tight interactions between the computing and networking infrastructures”. This poses new challenges on security and privacy issues and opens a new field of research and future work. The data-centric solution presented in this PhD thesis addresses some of the challenges of these scenario such as the loss of control over the data from the data owner point of view when data is managed in potentially untrusted CSPs. However, there is still room for improvement of the current authorization model and the proposed architecture. They can be extended to support CSP interconnection and interaction. In this regard, the usage of ontologies providing semantically enriched information and definitions can favorably contribute to achieve interoperability among the different domains of different CSPs.

Moreover, it would be also an interesting research step the analysis and comparison of further advanced trust models and their suitability for Cloud computing, including support for Intercloud scenarios. This would imply to also take into account possible trust relationships between tenants in different CSPs which are federated or interrelated. It would be also interesting to research how the relationships between the CSPs would affect the trust relationships established between tenants in different CSPs.

Another point of improvement is on advanced conflict detection and resolution techniques. A possible line of research would be the design of a distributed architecture in which semantic conflicts can be detected as well the analysis and definition of conflict resolution processes and techniques for enabling the system to (semi-) automatically return to a normal state. The reasoning capabilities provided by the proposed authorization model could be exploited to provide a semi-automated answer when a conflict appears. Likewise, the design and development of an expressiveness extension for policy languages in order to get constructors and operators in these languages to define, model, and establish conflict situations and relationships between conflicts and causes would be another interesting research. Finally, a detailed study on how conflict processes could be applied to scenarios modelled by means of stochastic approaches, in which some part of information is assumed and it is not explicit on the scenario representation would be also an interesting line of research.

On another front, XACML [8] is currently the most widely used standard for access control policy representation in distributed systems as exposed in Section 2.3. Thus, it would be worth to extend this work to link the proposed model with XACML representations, as well as adapting the architecture to be compatible with the protocols and management framework defined by this standard. This would also imply research on how to include domain definitions in this language or how to link it with other standardized domain definition models like CIM. Advantage could be taken from the ontological approach provided in the proposed model to perform concept alignment between policy and domain definitions. Integration with the DMTF Web-based Enterprise Management (WBEM) [55] would be also interesting in order to establish an automated discovery of the domain definitions in CIM.

The proof of concept developments and implementations developed during this work

can be also extended and improved to achieve a suitable and exploitable solution. Current developments can be used as starting point for more complete and tested software solutions that can be applied in production environments. In this regard, it would be desirable to extend the integration with OpenStack [95] in order to be more complete, providing interceptors not only for Nova but also for other components of OpenStack like Glance and Swift. It would be also interesting to work on the integration of the authorization model with the OpenStack identity manager component KeyStone [103]. The idea would be to achieve a fully functional authorization framework for OpenStack, since it is one of the major reference open implementations for Cloud IaaS.

Regarding the proposed RBAC-based data-centric solution, future lines of research include the analysis of novel cryptographic techniques that could enable the secure modification and deletion of data in the Cloud. This would allow to extend the privileges of the authorization model with more actions like *modify* and *delete*. Another interesting point is the obfuscation of the authorization model for privacy reasons. Although the usage of pseudonyms is proposed in Chapter 5, but more advanced obfuscation techniques can be researched to achieve a higher level of privacy. Additionally, as introduced in Section 1.4, both RBAC and ABAC approaches have their own advantages and disadvantages. Following current proposals like [104], which tries to incorporate attributes in the RBAC scheme, further research could be done to try to achieve a data-centric solution which merges both approaches trying to find an improved data-centric authorization model for self-protected data that could combine some of the benefits and features of both schemes.

Bibliography

- [1] Lawrence Lamers. DSP0004 - Common Information Model (CIM) infrastructure v2.8.0. DMTF standard, Distributed Management Task Force (DMTF), August 2014.
- [2] Liyang Yu. *A Developer's Guide to the Semantic Web, Second Edition*. Springer, 2014.
- [3] Peter Mell and Timothy Grance. The nist definition of cloud computing. Technical report, National Institute of Standards and Technology (NIST), September 2011.
- [4] Diogo A. B. Fernandes, Liliana F. B. Soares, Joao V. Gomes, Mário M. Freire, and Pedro R. M. Inácio. Security issues in cloud environments: a survey. *International Journal of Information Security*, 13(2):113–170, 2014. doi:10.1007/s10207-013-0208-7.
- [5] European Commission. Horizon 2020 - work programme 2014 - 2015 - 5.i. information and communication technologies - revised. Technical report, EU, April 2015.
- [6] Lee Badger, David Bernstein, Robert Bohn, Frederic de Vault, Mike Hogan, Michaela Iorga, Jian Mao, John Messina, Kevin Mills, Eric Simmon, Annie Sokol, Jin Tong, Fred Whiteside, and Dawn Leaf. US government Cloud computing technology roadmap volume I - high-priority requirements to further USG agency Cloud computing adoption. Technical report, National Institute of Standards and Technology (NIST), November 2014.
- [7] European Commission. Horizon 2020 - work programme 2016 - 2017 - 5.i. information and communication technologies. Technical report, EU, October 2015.
- [8] Erik Rissanen. eXtensible Access Control Markup Language (XACML) version 3.0. Oasis standard, OASIS XACML TC, January 2013.
- [9] Cloud Security Alliance. Security guidance for critical areas of focus in cloud computing v3.0. Technical report, CSA, 2003.

- [10] Yu Zhang, Jing Chen, Ruiying Du, Lan Deng, Yang Xiang, and Qing Zhou. Feacs: A flexible and efficient access control scheme for cloud computing. In *Trust, Security and Privacy in Computing and Communications, 2014 IEEE 13th International Conference on*, pp. 310–319, Sept 2014. doi:10.1109/TrustCom.2014.42.
- [11] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *Public Key Cryptography - PKC 2011*, volume 6571 of *Lecture Notes in Computer Science*, pp. 53–70. Springer Berlin Heidelberg, 2011. doi:10.1007/978-3-642-19379-8_4.
- [12] Balamurugan B and Venkata P. Extensive survey on usage of attribute based encryption in cloud. *Journal of Emerging Technologies in Web Intelligence*, 6(3), 2014.
- [13] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS '06*, pp. 89–98, New York, NY, USA, 2006. ACM. doi:10.1145/1180405.1180418.
- [14] Ed Coyne and Timothy R. Weil. ABAC and RBAC: Scalable, flexible, and auditable access management. *IT Professional*, 15(3):14–16, 2013. doi:10.1109/MITP.2013.37.
- [15] Empower ID. Best practices in enterprise authorization: The RBAC/ABAC hybrid approach. White paper, Empower ID, 2013.
- [16] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosz, and M. Dean. SWRL: A Semantic Web Rule Language combining OWL and RULEML. Technical report, World Wide Web Consortium (W3C), 2004.
- [17] Juan M. Marín Pérez, Gregorio Martínez Pérez, and Antonio F. Skarmeta Gómez. SecRBAC: Secure data in the Clouds. *IEEE Transactions on Services Computing*, 2015 (*Submitted for publication*). Special Issue on Security and Dependability of Cloud Systems and Services.
- [18] Jorge Bernal Bernabé, Juan M. Marín Pérez, José M. Alcaraz Calero, Félix J. García Clemente, Gregorio Martínez Pérez, and Antonio F. Gómez Skarmeta. Semantic-aware multi-tenancy authorization system for Cloud architectures. *Future Generation Computer Systems*, 32:154–167, 2014. doi:10.1016/j.future.2012.05.011. Special Issue on Exploiting Semantic Technologies with Particularization on Linked Data over Grid and Cloud Architectures.
- [19] Juan M. Marín Pérez, Jorge Bernal Bernabé, José M. Alcaraz Calero, Félix J. García Clemente, Gregorio Martínez Pérez, and Antonio F. Gómez Skarmeta. Taxonomy of trust relationships in authorization domains for Cloud computing.

- The Journal of Supercomputing*, 70(3):1075–1099, 2014. doi:10.1007/s11227-014-1117-x.
- [20] María A. Martínez Carreras, Juan M. Marín Pérez, Jorge Bernal Bernabé, José M. Alcaraz Calero, Gregorio Martínez Pérez, and Antonio F. Gómez Skarmeta. Towards a semantic-aware collaborative working environment. *Computing and Informatics*, 30(1):7–30, 2013.
- [21] Félix J. García Clemente, José M. Alcaraz Calero, Jorge Bernal Bernabé, Juan M. Marín Pérez, Gregorio Martínez Pérez, and Antonio F. Gómez Skarmeta. Semantic web-based management of routing configurations. *Journal of Network and Systems Management*, 19(2):209–229, 2011. doi:10.1007/s10922-010-9169-6.
- [22] Juan M. Marín Pérez, Jorge Bernal Bernabé, José M. Alcaraz Calero, Félix J. García Clemente, Gregorio Martínez Pérez, and Antonio F. Gómez Skarmeta. Semantic-based authorization architecture for Grid. *Future Generation Computer Systems*, 27(1):40–55, 2011. doi:10.1016/j.future.2010.07.008.
- [23] José M. Alcaraz Calero, Juan M. Marín Pérez, Jorge Bernal Bernabé, Félix J. García Clemente, Gregorio Martínez Pérez, and Antonio F. Gómez Skarmeta. Detection of semantic conflicts in ontology and rule-based information systems. *Data & Knowledge Engineering*, 69(11):1117–1137, 2010. doi:10.1016/j.datak.2010.07.004. Special issue on contribution of ontologies in designing advanced information systems.
- [24] Manuel Gil Pérez, Jorge Bernal Bernabé, Juan M. Marín Pérez, Daniel J. Martínez Manzano, and Antonio F. Gómez Skarmeta. A policy-based dependability management framework for critical services. *International Journal on Advances in Internet Technology*, 2(4):289–307, 2009.
- [25] Juan M. Marín Pérez, Antonio Moragón Juan, Jaime Arrazola Pérez, Javier Monge Rabadán, and Antonio F. Skarmeta Gómez. Security and privacy in vehicular communications with inter-trust. In *Cybersecurity & Privacy Innovation Forum 2015. CSP FORUM 2015*, Brussels, Belgium, Apr 2015 *In press*. Springer.
- [26] Juan M. Marín Pérez, Jorge Bernal Bernabé, Gregorio Martínez Pérez, and Antonio F. Skarmeta Gómez. Managing identity-related information across borders. In *Policies & Research in Identity Management (Future of Identity Workshop). IFIP IDMAN 2013. 3rd IFIP WG 11.6 Working Conference on*, London, United Kingdom, Apr 2013.
- [27] Jorge Bernal Bernabé, Juan M. Marín Pérez, José M. Alcaraz Calero, Félix J. García Clemente, Gregorio Martínez Pérez, and Antonio F. Gómez Skarmeta. Towards an authorization system for cloud infrastructure providers. In *Security and Cryptography. SECRIPT 2011. Proceedings of the International Conference on*, pp. 333–338, Seville, Spain, July 2011. IEEE Computer Society Press.

- [28] Jorge Bernal Bernabé, Juan M. Marín Pérez, Daniel J. Martínez Manzano, Manuel Gil Pérez, and Antonio F. Gómez Skarmeta. Towards a policy-driven framework for managing service dependability. In *Dependability. DEPEND 2009. Second International Conference on*, pp. 66–72, Athens, Greece, June 2009. IEEE Computer Society Press. doi:10.1109/DEPEND.2009.16. Best Paper Award.
- [29] Juan M. Marín Pérez, Jorge Bernal Bernabé, Daniel J. Martínez Manzano, Gregorio Martínez Pérez, and Antonio F. Gómez Skarmeta. Towards the definition of a web service based management framework. In *Emerging Security Information, Systems and Technologies. SECURWARE 2008. Second International Conference on*, pp. 362–367, Cap Esterel, France, Aug 2008. IEEE Computer Society Press. doi:10.1109/SECURWARE.2008.19.
- [30] Daniel J. Martínez Manzano, Juan M. Marín Pérez, Jorge Bernal Bernabé, Manuel Gil Pérez, and Antonio F. Gómez Skarmeta. Design-time learning for operational planning improvement. In *Emerging Security Information, Systems and Technologies. SECURWARE 2008. Second International Conference on*, pp. 326–331, Cap Esterel, France, Aug 2008. IEEE Computer Society Press. doi:10.1109/SECURWARE.2008.57.
- [31] Juan M. Marín Pérez, Jorge Bernal Bernabé, Jesús D. Jiménez Re, Gregorio Martínez Pérez, and Antonio F. Gómez Skarmeta. A proposal for translating from high-level security objectives to low-level configurations. In *Systems and Virtualization Management: Standards and New Technologies. SVM 2007. 1st International DMTF Academic Alliance Workshop on*, Toulouse, France, Oct 2007.
- [32] Juan M. Marín Pérez, Jorge Bernal Bernabé, José M. Alcaraz Calero, Jesús D. Jiménez Re, Félix J. García Clemente, Gregorio Martínez Pérez, and Antonio F. Gómez Skarmeta. *Rule-Driven Architecture for Managing Information Systems*, pp. 417–435. CRC Press, 2013.
- [33] Jorge Bernal Bernabé, Juan M. Marín Pérez, José M. Alcaraz Calero, Jesús D. Jiménez Re, Félix J. García Clemente, Gregorio Martínez Pérez, and Antonio F. Gómez Skarmeta. *Security Policy Specification*, pp. 66–93. IGI Global, 2013.
- [34] David W Chadwick, Gansen Zhao, Sassa Otenko, Romain Laborde, Linying Su, and Tuan Anh Nguyen. PERMIS: a modular authorization infrastructure. *Concurrency and Computation: Practice and Experience*, 20(11):1341–1357, August 2008.
- [35] Valery Tschopp. Argus, the emi authorization service. In *The 1st EMI Technical Conference*, Vilnius, Lithuania, April 2011.
- [36] C. Aiftimiei, A. Aimar, A. Ceccanti, M. Cecchi, A. Di Meglio, F. Estrella, P. Fuhrmam, E. Giorgio, B. Konya, L. Field, J.K. Nilsen, M. Riedel, and J. White. Towards next generations of software for distributed infrastructures: The european

- middleware initiative. In *E-Science (e-Science), 2012 IEEE 8th International Conference on*, pp. 1–10, October 2012. doi:10.1109/eScience.2012.6404415.
- [37] Roberto Alfieri, Roberto Cecchini, Vincenzo Ciaschini, Luca dell’Agnello, Ákos Frohner, Alberto Gianoli, Károly Lörentey, and Fabio Spataro. Voms: an authorization system for virtual organizations. In *European Across Grids Conference*, pp. 33 – 40, 2003.
- [38] Mine Altunay, Paul Avery, Kent Blackburn, Brian Bockelman, Michael Ernst, Dan Fraser, Robert Quick, Robert Gardner, Sebastien Goasguen, Tanya Levshina, Miron Livny, John McGee, Doug Olson, Ruth Pordes, Maxim Potekhin, Abhishek Rana, Alain Roy, Chander Sehgal, Igor Sfiligoi, and Frank Wuerthwein. A science driven production cyberinfrastructure - the open science grid. *Journal of Grid Computing*, 9(2):201–218, 2011. doi:10.1007/s10723-010-9176-6.
- [39] Andrzej Uszok, Jeffrey M. Bradshaw, and Renia Jeffers. KAoS: A policy and domain services framework for grid computing and semantic web services. In *Trust Management*, volume 2995 of *Lecture Notes in Computer Science*, pp. 16 – 26. Springer, 2004.
- [40] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott. KAoS policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. In *Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY’03)*, pp. 93 – 96. IEEE Computer Society, 2003.
- [41] L. Kagal, T. Finin, and A. Joshi. A policy language for a pervasive computing environment. In *Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, pp. 63–74, 2003.
- [42] Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The ponder policy specification language. In *Proceedings of the International Workshop on Policies for Distributed Systems and Networks (POLICY 2001)*, pp. 18 – 38, London, UK, 2001. Springer-Verlag.
- [43] XIAO-YONG LI, YONG SHI, and WEI MA YU-GUO. Multi-tenancy based access control in cloud. In IEEE, editor, *2010 International Conference on Computational Intelligence and Software Engineering*, volume 1, pp. 1–4. IEEE, Dec 2010.
- [44] Dancheng Li, Cheng Liu, Qiang Wei, Zhiliang Liu, and Binsheng Liu. Rbac-based access control for saas systems. In IEEE, editor, *2010 2nd International Conference on Information Engineering and Computer Science*, pp. 1–4. IEEE, Dec 2010.

- [45] Avvari Sirisha and G. Geetha Kumari. Api access control in cloud using the role based access control model. In IEEE, editor, *Trendz in Information Sciences & Computing*, pp. 135–137. IEEE, IEEE, 2010.
- [46] Wei-Tek Tsai and Qihong Shao. Role-based access-control using reference ontology in clouds. In IEEE, editor, *Tenth International Symposium on Autonomous Decentralized*, pp. 121–128. IEEE, IEEE, 2011.
- [47] Anil L. Pereira. Rbac for high performance computing systems integration in grid computing and cloud computing. In IEEE, editor, *IEEE International Parallel & Distributed Processing Symposium*, pp. 914–921. IEEE, IEEE, May 2011.
- [48] Jing Xu, Tang Jinglei, He Dongjian, Zan Linsen, Chen Lin, and Niu Fang. Research and implementation on access control of management-type saas. In IEEE, editor, *The 2nd IEEE International Conference on Information Management and Engineering*, pp. 388–392. IEEE, IEEE, April 2010.
- [49] Chen Danwei, Huang Xiuli, and Ren Xunyi. Access control of cloud service based on ucon. *LNCS Cloud Computing*, 5931:559–564, 2009.
- [50] Jaehong Park and Ravi Sandhu. The uconabc usage control model. *ACM Transactions on Information and System Security*, 7(1):128–174, 2004.
- [51] Doudou Fall, Gregory Blanc, Takeshi Okuda, Youki Kadobayashi, and Suguru Yamaguchi. Toward quantified risk-adaptive access control for multi-tenant cloud computing. In *The 6th Joint Workshop on Information Security*, pp. 1–14, 2011.
- [52] Jose M. Alcaraz-Calero, Nigel Edwards, Johannes Kirschnick, Lawrence Wilcock, and Mike Wray. Towards a multi-tenancy authorization system for cloud services. *IEEE Security and Privacy*, 8(6):48–55, Nov 2010.
- [53] Organization for the Advancement of Structured Information Standards (OASIS). <https://www.oasis-open.org>.
- [54] Distributed Management Task Force, Inc (DMTF). <http://www.dmtf.org>.
- [55] M. Mishra and S.S. Bedi. *WBEM for Heterogeneous Network Management: Web Based Enterprise Management (WBEM)*. VDM Publishing, 2014.
- [56] K. Twidle, N. Dulay, E. Lupu, and M. Sloman. Ponder2: A policy system for autonomous pervasive environments. In *Autonomic and Autonomous Systems, 2009. ICAS '09. Fifth International Conference on*, pp. 330–335, April 2009. doi:10.1109/ICAS.2009.42.
- [57] Alexander Lawall, Dominik Reichelt, and Thomas Schaller. Resource management and authorization for cloud services. In *Proceedings of the 7th International Conference on Subject-Oriented Business Process Management, S-BPM ONE '15*, pp. 18:1–18:8, New York, NY, USA, 2015. ACM. doi:10.1145/2723839.2723864.

-
- [58] David Yu Chang, Messaoud Benantar, John Yow-chun Chang, and Vishwanath Venkataramappa. Authentication and authorization methods for cloud computing platform security, January 1 2015, US Patent 20,150,007,274.
- [59] Rakesh Bobba, Himanshu Khurana, and Manoj Prabhakaran. Attribute-sets: A practically motivated enhancement to attribute-based encryption. In Michael Backes and Peng Ning, editors, *Computer Security - ESORICS 2009*, volume 5789 of *Lecture Notes in Computer Science*, pp. 587–604. Springer Berlin Heidelberg, 2009. doi:10.1007/978-3-642-04444-1_36.
- [60] Guojun Wang, Qin Liu, and Jie Wu. Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS '10*, pp. 735–737, New York, NY, USA, 2010. ACM. doi:10.1145/1866307.1866414.
- [61] Junae Liu, Zhiguo Wan, and Ming Gu. Hierarchical attribute-set based encryption for scalable, flexible and fine-grained access control in cloud computing. In Feng Bao and Jian Weng, editors, *Information Security Practice and Experience*, volume 6672 of *Lecture Notes in Computer Science*, pp. 98–107. Springer Berlin Heidelberg, 2011. doi:10.1007/978-3-642-21031-0_8.
- [62] W3C OWL Working Group. OWL 2 Web Ontology Language: Document overview (second edition). W3C recommendation, World Wide Web Consortium (W3C), December 2012.
- [63] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. OWL 2 Web Ontology Language: Profiles (second edition). W3C recommendation, World Wide Web Consortium (W3C), December 2012.
- [64] Marta Majewska, Bartosz Kryza, and Jacek Kitowski. Translation of Common Information Model to Web Ontology Language. *LNCS Computational Science - ICCS 2007*, 4487:414–417, 2007.
- [65] Harry Chen, Filip Perich, Tim Finin, and Anupam Joshi. Soupa: Standard ontology for ubiquitous and pervasive applications. In *International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pp. 258–267, 2004.
- [66] York Sure, Stephan Bloehdorn, Peter Haase, Jens Hartmann, and Daniel Oberle. The SWRC ontology - semantic web for research communities. In *Progress in Artificial Intelligence, 12th Portuguese Conference on Artificial Intelligence, EPIA 2005*, *Lecture Notes in Computer Science*, pp. 218 – 231. Springer, 2005.
- [67] Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for OWL-DL with rules. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 3(1):41–60, 2005.

- [68] Praveen Kumar Paladugu. Wbem based management in linux. Technical white paper, Dell, February 2011.
- [69] Ravikanth Chaganti. Introducing windows remote management and CIM. In *Windows PowerShell Desired State Configuration Revealed*, pp. 43–74. Apress, 2014. doi:10.1007/978-1-4842-0016-2_2.
- [70] George Ericson and Wojtek Kozaczynski. DSP0221 - Managed Object Format (MOF) v3.0.1. DMTF standard, Distributed Management Task Force (DMTF), April 2015.
- [71] Stephen Quirolgico, Pedro Assis, Andrea Westerinen, Michael Baskey, and Ellen Stokes. Toward a formal common information model ontology. *LNCIS Web Information Systems*, 3307:11–21, 2004.
- [72] Felix J. Garcia, Gregorio Martinez, Andres Munoz, Juan A. Botia, and Antonio F. Gomez. Towards semantic web-based management of security services. *Springer Annals of Telecommunications*, 63(3-4):183–194, 2008.
- [73] Ian T. Foster. Globus toolkit version 4: Software for service-oriented systems. In Hai Jin, Daniel A. Reed, and Wenbin Jiang, editors, *IFIP International Conference on Network and Parallel Computing*, Springer-Verlag LNCIS 3779, volume 3779 of *Lecture Notes in Computer Science*, pp. 2–13. Springer, 2005.
- [74] Bo Lang, Ian Foster, Frank Siebenlist, Rachana Ananthakrishnan, and Tim Freeman. A multipolicy authorization framework for grid security. *IEEE International Symposium on Network Computing and Applications*, pp. 269–272, 2006.
- [75] Scott Cantor, John Kemp, Rob Philpott, and Eve Maler. Assertions and protocols for the OASIS Security Assertion Markup Language (SAML) v2.0. Oasis standard, OASIS Security Services (SAML) TC, March 2005.
- [76] Anne Anderson. SAML 2.0 profile of XACML version 2. Oasis xacml tc working draft, OASIS XACML TC, July 2007.
- [77] Von Welch, Frank Siebenlist, Ian Foster, John Bresnahan, Karl Czajkowski, Jarek Gawor, Carl Kesselman, Sam Meder, Laura Pearlman, and Steven Tuecke. Security for grid services. In *HPDC '03: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*, page 48, Washington, DC, USA, 2003. IEEE Computer Society.
- [78] Karl Czajkowski, Donald F. Ferguson, Ian Foster, Jeffrey Frey, Steve Graham, Igor Sedukhin, David Snelling, Steve Tuecke, and William Vambenepe. The ws-resource framework. Technical report, Computer Associates International, Inc., Fujitsu Limited, Hewlett-Packard Development Company, International Business Machines Corporation and The University of Chicago, 2004.

-
- [79] Fabien Gandon and Guus Schreiber. RDF 1.1 XML syntax. W3C recommendation, World Wide Web Consortium (W3C), February 2014.
- [80] Mark A. Musen. The protég   project: A look back and a look forward. *AI Matters*, 1(4):4–12, June 2015. doi:10.1145/2757001.2757003.
- [81] Alan L Rector, Nick Drummond, Matthew Horridge, Jeremy Rogers, Holger Knublauch, Robert Stevens, Hai Wang, and Chris Wroe. Designing user interfaces to minimise common errors in ontology development: The CO-ODE and HyOn-tUse projects. In *Proceedings of the UK e-Science All Hands Meeting 2004*, pp. 493 – 499, September 2004.
- [82] Rui G. Pereira and Mario M. Freire. SWedt: A semantic web editor integrating ontologies and semantic annotations with resource description framework. In *Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/I-CIW 2006)*, page 200, Washington, DC, USA, 2006. IEEE Computer Society.
- [83] Andres Munoz, Antonia Vera, Juan A. Botia, and Antonio F. Gomez Skarmeta. Defining basic behaviours in ambient intelligence environments by means of rule-based programming with visual tools. In J. C. Augusto, editor, *1st Workshop of Artificial Intelligence Techniques for Ambient Intelligence. ECAI*, 2006.
- [84] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):10, 2007.
- [85] Emil C. Lupu and Morris Sloman. Conflicts in policy-based distributed systems management. *IEEE Transactions on Software Engineering*, 25(6):852–869, November 1999.
- [86] Carlos Ribeiro, Andr   Z  quete, Paulo Ferreira, and Paulo Guedes. SPL: An access control language for security policies with complex constraints. In *Proceedings of the Network and Distributed System Security Symposium (NDSS 2001)*, pp. 89–107, February 2001.
- [87] Shannon Hastings, Scott Oster, Stephen Langella, David Ervin, Tahsin Kurc, and Joel Saltz. An open source toolkit for rapid development of strongly typed grid services. *Journal of Grid Computing*, 5(4):407 – 427, 2007.
- [88] Steve Harris and Andy Seaborne. SPARQL 1.1 query language. W3C recommendation, World Wide Web Consortium (W3C), March 2013.
- [89] Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. Jena: Implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference*, pp. 74–83. ACM Press, 2004.

- [90] Erol Bozsak, Marc Ehrig, Siegfried Handschuh, Andreas Hotho, Alexander Maedche, Boris Motik, Daniel Oberle, Christoph Schmitz, Steffen Staab, Ljiljana Stojanovic, Nenad Stojanovic, Rudi Studer, Gerd Stumme, York Sure, Julien Tane, Raphael Volz, and Valentin Zacharias. Kaon - towards a large scale semantic web. In *EC-Web*, pp. 304–313, 2002.
- [91] Dmitry Tsarkov and Ian Horrocks. *Automated Reasoning*, volume 4130 of *Springer Lecture Notes in Computer Science*, chapter FaCT++ Description Logic Reasoner: System Description, pp. 292–297. Springer Berlin / Heidelberg, 2006.
- [92] Thomas Vetterli, Anca Vaduva, and Martin Staudt. Metadata standards for data warehousing: Open information model vs. common warehouse metadata. *ACM SIGMOD Record*, 29(3):68 – 75, 2000.
- [93] Markus Debusmann and Alexander Keller. SLA-driven management of distributed systems using the common information model. In *Proceeding of the 8th IFIP/IEEE International Symposium on Integrated Network Management*, pp. 1–14, 2003.
- [94] Hongyan Mao, Linpeng Huang, and Minglu Li. Web resource monitoring based on common information model. In *IEEE Asia-Pacific Conference on Services Computing*, pp. 520–525, 2006.
- [95] Omar Khedher. *Mastering OpenStack*. Packt Publishing, July 2015.
- [96] Jitendra Kotamraju. The Java API for XML-based Web Services (JAX-WS) 2.2 Rev a. Java specification request, Oracle America, Inc., May 2011.
- [97] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved Proxy Re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security*, 9(1):1–30, February 2006. doi:10.1145/1127345.1127346.
- [98] Fenghe Wang, ZhenHua Liu, and Chunxiao Wang. Full secure identity-based encryption scheme with short public key size over lattices in the standard model. *International Journal of Computer Mathematics*, pp. 1–10, 2015. doi:10.1080/00207160.2015.1029464.
- [99] Matthew Green and Giuseppe Ateniese. Identity-based proxy re-encryption. In *Proceedings of the 5th International Conference on Applied Cryptography and Network Security*, ACNS '07, pp. 288–306, Berlin, Heidelberg, 2007. Springer-Verlag. doi:10.1007/978-3-540-72738-5_19.
- [100] Eu-Jin Goh Dan Boneh and Toshihiko Matsuo. Proposal for p1363.3 Proxy Re-encryption, August 2006.
- [101] R. Housley. Cryptographic message syntax (CMS). RFC 5652, Internet Engineering Task Force (IETF), September 2009.

- [102] Omer K. Jasim Mohammad, Safia Abbas, El-Sayed M. El-Horbaty, and Abdel-Badeeh M. Salem. Innovative method for enhancing key generation and management in the aes-algorithm. *CoRR*, abs/1504.03406, 2015.
- [103] Baojiang Cui and Tao Xi. Security analysis of openstack keystone. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2015 9th International Conference on*, pp. 283–288, July 2015. doi:10.1109/IMIS.2015.44.
- [104] QasimMahmood Rajpoot, ChristianDamsgaard Jensen, and Ram Krishnan. Integrating attributes into role-based access control. In Pierangela Samarati, editor, *Data and Applications Security and Privacy XXIX*, volume 9149 of *Lecture Notes in Computer Science*, pp. 242–249. Springer International Publishing, 2015. doi:10.1007/978-3-319-20810-7_17.

