



Interoperability and Negotiation of Security Policies

Yanhuang Li

► To cite this version:

Yanhuang Li. Interoperability and Negotiation of Security Policies. Networking and Internet Architecture [cs.NI]. Ecole Nationale Supérieure des Télécommunications de Bretagne - ENSTB, 2016. English. NNT : 2016TELB0414 . tel-01593257

HAL Id: tel-01593257

<https://theses.hal.science/tel-01593257>

Submitted on 26 Sep 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITE BRETAGNE LOIRE

THÈSE / Télécom Bretagne
sous le sceau de l'Université Bretagne Loire
pour obtenir le grade de Docteur de Télécom Bretagne
En accréditation conjointe avec l'Ecole Doctorale Matisse
Mention : Informatique

présentée par

Yanhuang Li

préparée dans le département Logique des usages, sciences sociales et
de l'Information
Laboratoire Labsticc

Interoperability and Negotiation of Security Policies

Thèse soutenue le 24 novembre 2016
Devant le jury composé de :

Mario Südholt
Professeur, École des Mines de Nantes / président

Yves Roudier
Professeur, Université Nice Sophia-Antipolis / rapporteur

Abdelmalek Benzekri
Professeur, IRIT - Université Toulouse III Paul Sabatier / rapporteur

Zonghua Zhang
Maître de conférences (HDR), Télécom Lille / examinateur

Vincent Frey
Resp. programme de recherche, Orange Labs - Cesson-Sévigné / examinateur

Jean-Michel Crom
Architecte de services, Orange Labs - Cesson-Sévigné / examinateur

Frédéric Cuppens
Professeur, Télécom Bretagne / examinateur

Nora Cuppens-Bouahia
Professeur, Télécom Bretagne / directrice de thèse

Sous le sceau de l'Université Bretagne Loire

Télécom Bretagne

En accréditation conjointe avec l'Ecole Doctorale Matisse

Ecole Doctorale – MATISSE

Interoperability and Negotiation of Security Policies

Thèse de Doctorat

Mention : Informatique

Présentée par **Yanhuang LI**

Département : LUSI

Laboratoire : Lab-STICC

Directrice de thèse : Nora Cuppens-Boulahia

Soutenue le 24 Novembre 2016

Jury :

M. Yves ROUDIER, Professeur, Université Nice Sophia Antipolis (Rapporteur)
M. Abdelmalek BENZEKRI, Professeur, Université Toulouse III Paul Sabatier (Rapporteur)
M. Mario SUDHOLT, Professeur, École des Mines de Nantes (Examineur)
M. Zonghua ZHANG, Maître de conférences (HDR), Télécom Lille (Examineur)
M. Vincent FREY, Resp. Programme de recherche, Orange Labs-Cesson (Examineur)
M. Jean-Michel CROM, Architecte de services, Orange Labs-Cesson (Examineur)
M. Frédéric CUPPENS, Professeur, Télécom Bretagne (Examineur)
Mme. Nora CUPPENS-BOULAHIA, Professeur, Télécom Bretagne (Directeur de recherche)

Acknowledgements

First, and foremost, I would like to thank my supervisors Nora Cuppens-Boulahia, Jean-Michel Crom, Frédéric Cuppens and Vincent Frey for their inspiring guide and support during the past years. Their rich experience and extensive expertise in field of research and industry, combining with their passion and rigorous attitude, motivate and encourage me to explore and conquer the world of security. They have not only guided me on technical issues, but also taught me the research paradigms. The cooperation between Télécom Bretagne and Orange is well organized. From the side of Télécom Bretagne, I have learned lots of theories and research methodology. Meanwhile, the support of Orange enables me to participate in some industry projects and training programs.

Besides my supervisors, I would like to thank Dr. Ruan He. Thank you for bringing me into the gates of SDN, NFV and especially Cloud Computing which becomes one of the most beautiful parts in my thesis.

I am particularly grateful for the help I received from Xiaoshu Ji, who made an internship in Orange during my second year of PhD. My contribution on similarity measure for security policies may not have been achieved without her participation and programming. Here I wish that she could achieve more glories in her research career.

I am deeply obliged to my colleagues in Télécom Bretagne and Orange for their support and help during my PhD program. Regarding Télécom Bretagne, my thoughts go to Fabien Autrel, Mariem Graa, Samiha Ayed, Muhammad Sabir Idrees, Anis Bkakria, Tarek Bouyahia, Said Oulmakhzoune, Nada Essaouini, Tarik Moataz, Safaa Hachana, Reda Yaich and Lyes Bayou. I have learnt a great deal of knowledge about security and research methodology from them. From the side of Orange, I would say thanks to the PIA (Projects for Identity Anticipation & Research) team: Joël Evain, Johann Vincent, Nicolas Allery, Julie Bertrand, Boris Pinatel, Sahin Kale, Amine Khalil, Louis Philippe Sondeck, Kevin Corre, Marco Lobe Kome, Simon Becot, Olivier Guillemin, Stephane Labat, Bruno Lezoray, Jean-Michel Magret, Patrick Maroche, Laurent Piffeteau. They supported me during my work in Orange Labs and provided me with a great comfort with their kindness, generosity and humor.

My gratitude also goes to my friends in Rennes: Yue li, Hao Lin, Bihong Huang, Han Yan, Zhe Li, Feng Yan, Ping Yan, Yangyang Chen, Miaoqing Shi, Jialong Duan, Qipeng Song, Wenjing Shuai, Tuo Zhang, Bing Xiao, Zaiqian Wang, Quancheng Zhao. I appreciate the time with you in Rennes and thanks for your kindly help in my daily life.

Finally, I am deeply indebted to my family: my parents always support and advise me in every thing I have chosen to do. Their unflagging care, endless love and faith in me have always been a beacon of confidence and a source of perseverance.

Abstract

Security policy provides a way to define the constraints on behavior of the members belonging to a system, organization or other entities. With the development of IT technology such as Grid Computing and Cloud Computing, more and more applications and platforms exchange their data and services for cooperating. Toward this trend, security becomes an important issue and security policy has to be applied in order to ensure the safety of data and service interaction. In this thesis, we deal with one type of security policy: access control policy. Access control policy protects the privileges of resource's utilization and there exist different policy models for various scenarios. Our goal is to ensure that the service customer well expresses her security requirements and chooses the service providers that fit these requirements.

The first part of this dissertation is dedicated to the service provider selection. In case that the security policies of the service provider are accessible to the service customer, we provide a method for measuring the similarity between security policies. The approach proposed supports different policy models and its correctness is proved by the brute-force based test. Another case is that security policies are not accessible to the service customer or not specified explicitly. Our solution is proposing a policy-based framework which enables the derivation from attribute-based security requirements to concrete security policies. The current framework is used to allocate virtual resource in IaaS Cloud and we have developed an OpenStack-based proof-of-concept.

The second part of the dissertation focuses on the security policy negotiation. We investigate the process of reaching agreement

through bargaining process in which negotiators exchange their offers and counter-offers step by step. The positive result of the negotiation generates a policy contract. Our current approach supports the negotiation between two negotiators with the same policy model. We use specifically, the policy tree as configuration to store and manage security-aware preferences and requirements.

Résumé

Suite au développement des technologies de l'information, et en particulier au déploiement d'infrastructures telles que le Grid Computing et le Cloud Computing, de plus en plus d'applications et plateformes coopèrent en échangeant des données et des services. Cette tendance renforce l'importance de la gestion de la sécurité. Afin d'assurer la sécurité des données et de l'interaction de service une politique de sécurité doit être appliquée. En effet, les politiques de sécurité permettent de définir des contraintes sur le comportement des membres appartenant à un système, une organisation ou d'autres entités. Dans cette thèse, nous nous intéressons aux politiques de contrôle d'accès. Ce type de politique spécifie les privilèges de l'utilisation des ressources et est implémentée par différents modèles selon différents scénarios. Notre objectif ici est d'aider le client du service à bien exprimer ses exigences de sécurité et à choisir les fournisseurs de services qui peuvent la déployer.

La première partie de cette thèse est dédiée à la sélection des fournisseurs de service. Dans le cas où les politiques de sécurité du fournisseur sont accessibles au client, nous proposons une méthode pour mesurer la similarité entre les politiques de sécurité. L'approche proposée prend en charge des modèles de politiques différents et son exactitude est prouvée par un test qui se fonde sur la force brute. Dans le cas où les politiques de sécurité ne sont pas accessibles au client ou ne sont pas explicitement spécifiées, nous proposons un cadre à base de règles permettant la dérivation à partir des exigences de sécurité aux politiques de sécurité concrètes. Ce cadre est utilisé pour allouer des ressources virtuelles dans une infrastructure de type IaaS Cloud et nous y avons développé une preuve de concept en utilisant la brique OpenStack.

La seconde partie de la thèse porte sur la négociation de politiques de sécurité. Nous étudions le processus permettant aux parties en négociation de parvenir à un accord par une série d'échanges d'offres et de contre-offres. Lorsque le résultat de la négociation est positif, un contrat incluant la politique de sécurité acceptée par les parties est généré. L'approche

actuelle prend en charge le mode de négociation entre deux parties utilisant le même modèle de politique. Plus spécifiquement nous utilisons une structure d'arbre de politiques comme configuration locale pour stocker et gérer les préférences et exigences de sécurité.

Contents

Contents	vii
List of Figures	xi
Nomenclature	xii
1 Introduction	1
1.1 Motivation and Challenges	1
1.2 Contributions	3
1.3 Organization	4
I Service Provider Selection	7
2 State of the Art	9
2.1 Introduction	9
2.2 Access Control Model	9
2.2.1 Discretionary Access Control (DAC)	11
2.2.2 Lattice-based Access Control (LBAC or MAC)	12
2.2.3 Role-based Access Control (RBAC)	12
2.2.4 Attribute-based Access Control (ABAC)	14
2.2.5 Organization-Based Access Control (OrBAC)	18
2.3 Service Level Agreement (SLA)	21
2.3.1 WSLA	21
2.3.2 WS-Agreement	22
2.3.3 RBSLA	24

2.3.4	SLang	24
2.3.5	Security related SLA	25
2.4	Policy Similarity Measure (PSM)	27
2.4.1	Problematic	27
2.4.2	Use Cases	29
2.4.3	Existing Approaches	30
2.5	Virtual Resource Allocation	31
2.5.1	Problematic	31
2.5.2	Existing Approaches	32
2.6	Conclusion	33
3	Similarity Measure for Security Policies	35
3.1	Introduction	35
3.2	A Generic Policy Similarity Measure Method	35
3.2.1	Policy Structure	36
3.2.2	Example of Policy Transformation	37
3.2.3	Overview of PSM Algorithm	38
3.2.4	Similarity Score of Rule Elements	39
3.2.4.1	Similarity Score for Categorical Elements.	40
3.2.4.2	Similarity Score for Numerical Elements.	42
3.2.5	Example of Calculation	43
3.3	Experiment Results	44
3.4	Implementation	46
3.4.1	Scenario Description	46
3.4.2	Architecture	48
3.4.3	Performance	50
3.5	Conclusion	52
4	Expression and Enforcement of Security Policy	55
4.1	Introduction	55
4.2	Overview of the User-centric Policy-based Framework	56
4.3	Expression of Security Policy	57
4.3.1	SLA Contract Expression	57
4.3.2	Derivation of Security Policy	60

4.4	Enforcement of Security Policy	62
4.4.1	QoS Filtering	62
4.4.2	Conflict Management	63
4.4.3	Execution of Virtual Resource Allocation	64
4.5	Implementation and Evaluation	65
4.5.1	Experiment 1: contract processing	69
4.5.2	Experiment 2: policy generation	69
4.5.3	Experiment 3: allocation latency	70
4.5.4	Experiment 4: price	70
4.6	Conclusion	72

II Negotiation between Service Customer and Service Provider 73

5 State of the Art 75

5.1	Introduction	75
5.2	Trust Negotiation	75
5.3	Access Negotiation	78
5.4	Access Control Policy Negotiation	80
5.5	Meaning Negotiation	83
5.6	Conclusion	86

6 The Process of Reaching Agreement in Security Policy Negotiation 87

6.1	Introduction	87
6.2	Relation between OrBAC Entities	88
6.3	Relation between OrBAC Rules	90
6.4	Negotiation Configuration	91
6.4.1	Entity Chain	91
6.4.2	Policy Tree	92
6.5	Negotiation Algorithm	94
6.5.1	Algorithm Description	94
6.5.2	Theoretical Results	98

6.6 Application	99
6.6.1 Scenario Description	99
6.6.2 Process of Security Policy Negotiation	101
6.6.3 Prototype	105
6.7 Conclusion	109
7 Conclusion and Perspectives	111
7.1 Main Results	112
7.2 Perspectives	113
Appendix A	119
Appendix B	123
Appendix C	125
List of Publications	131
References	133

List of Figures

2.1 Basic access control model	10
2.2 LBAC model	13
2.3 RBAC model	13
2.4 ABAC model	15
2.5 XACML 3.0 policy language model [1]	16
2.6 Data-flow diagram of XACML 3.0 policy language [1]	17
2.7 OrBAC model	19
2.8 The MotOrBAC tool architecture [2]	20
2.9 Overview of main WSLA concepts [3]	21
2.10 Asymmetric mode of WS-Agreement negotiation [4]	23
2.11 Framework for security mechanisms in Cloud SLAs [5]	26
3.1 The process of similarity score calculation	38
3.2 Abstraction tree for the <i>Role</i> element	40
3.3 Abstraction tree for the <i>Resource</i> element	41
3.4 Inheritance tree for the <i>Role</i> element	41
3.5 Experiment of similarity score (<i>set-4</i>)	45
3.6 Experiment of similarity score (<i>set-8</i>)	45
3.7 Service provider selection for Cloud storage	47
3.8 CloudSim architecture [6]	49
3.9 The sequence diagram of implementation	51
3.10 Execution time of SP ranking (domain number=5)	52
3.11 Execution time of SP ranking (domain number=5~30)	52
4.1 The proposed policy based framework to allocate virtual resources	56

4.2	A DevOps use case of virtual resource allocation	60
4.3	Implementation for virtual resource allocation	66
4.4	Policy generated after Step 1: SLA contract processing	67
4.5	Policy generated after Step 2: QoS filtering	67
4.6	Policy generated after Step 3: Conflict resolution	67
4.7	Final resource allocation solution graph	68
4.8	Deployment of VM on HOST1	68
4.9	Deployment of VMs on HOST2	69
4.10	Time for contract processing	71
4.11	Time for policy generation	71
4.12	Latency for VM allocation	71
4.13	Total price for VM allocation	71
5.1	Typical TN system	76
5.2	Typical access negotiation system	79
5.3	Typical AC policy negotiation system	81
5.4	Typical MN system	84
6.1	An entity chain example for <i>Role</i>	92
6.2	Example of a related policy tree	93
6.3	Example of a distant policy tree	94
6.4	Entity chains of Bob and Chinese ITS station	102
6.5	Bob's negotiation configuration	102
6.6	Chinese ITS station's negotiation configuration	103
6.7	Negotiation between Bob and the Chinese ITS station	104
6.8	Architecture of the prototype	105
6.9	Using the policy tree editor to visualize and edit policy	107
6.10	A message sent during negotiation	108
7.1	A scenario of expression and negotiation service contract for SDN networks	116
2	Experiment of similarity score (<i>set-4</i>).	124

Chapter 1

Introduction

1.1 Motivation and Challenges

Nowadays, data and service exchange across multiple actors becomes an emerging demand to provide dynamic ecosystems. This process involves a large number of actors such as service provider (SP) and service customer (SC). For example, lots of Cloud service providers (CSP) such as Amazon, Microsoft and Orange provide their various services (SaaS, PaaS and IaaS) to Cloud service customers (CSC). Before the implementation of the service, the SC should 1) choose the SP (s) which is compliant with SC's preferences on service terms. 2) reach agreement with SP(s) chosen in order to guarantee the service level and provide fixed service terms for future monitoring.

With respect to SP selection, from SC's point of view, it is always difficult to decide whose service should be chosen so they use brokering technology to rank and select the suitable SPs based on user's requirements. However, most of the current service ranking technologies [7] do not consider the security aspect or they only measure security parameters such as encryption methods [8] and security levels offered by SPs (quantitative and qualitative evaluation) [9, 10]. Among various criteria that need to be considered for the SP selection, security policy is a critical concern and it addresses the constraints on behaviour of the members in a system, organization or other entity. Unlike other measurement criteria, security policies are usually based

on first-order logic which contains predicates and quantification. For example, an access control policy consists of multiple elements and they collectively determine whether a user is allowed to take some actions on certain objects. Thus, existing brokering technologies are difficult to apply on security policies.

Regarding reaching agreement, negotiation is one of the main mechanisms and the output is usually a service contract. Current negotiation technologies cover mainly non-security terms such as QoS and security terms such as trust. In the field of Trust Negotiation (TN), lots of models such as TrustBuilder [11] and XeNA [12] have been developed. These models implement negotiation by disclosing credentials step by step. When it comes to security policy, the solution is limited and restricted. Solutions to negotiate security policies are currently based on syntactic mapping: typically the same attribute must have the identical name. Therefore, negotiation fails when the mapping is not successful. Another difficulty is the implementation and development of this complex integration process for ecosystems. Some related implementations are provided by WS-Security (Web Services Security) [13] and WS-Trust (Web Services Trust) [14] which are in the protocol level. Similarly, Liberty Alliance [15] concerns the overall framework of contract and "metadata" which describes some properties of a SP such as "OrganizationDisplayName", "contactType" and "validUntil". Some researches offer more flexible solutions to negotiate security policies. The work in [16] is a useful starting point but still limited because it supposes that the semantic mapping between different security policies to be negotiated has previously been performed. Moreover, it does not consider situations where different requirements to be negotiated may have different privileges. More related works on security policy negotiation can be found in Section 5.4. However, none of the security policy negotiation solutions provides a complete framework covering policy definition, negotiation configuration, proposal evaluation and negotiation protocol.

1.2 Contributions

The main contributions of this thesis are summarized in three aspects:

Improvement of similarity measure for security policies: We propose a generic and light-weight method [17] to compare and evaluate security policies belonging to different models. Our technique enables a SC to quickly locate SPs with potentially similar policies. The contribution is twofold. On one hand, our method is policy-agnostic and can be applied to various types of policy models. On the other hand, we propose integrating our policy similarity measure algorithm in the SP selection process and the implementation proves that the integration can enrich the services offered.

Enhancing policy expression and enforcement in multi-cloud environments: The work [18] is based on a formal model that applies organization-based access control (OrBAC) [19] policy to IaaS resource allocation. We first integrate the attribute-based security requirements in service level agreement (SLA) contracts. After transformation, the security requirements are expressed as OrBAC rules and these rules are treated together with other non-security demands during the enforcement of resource allocation. We have implemented a prototype for VM scheduling in OpenStack-based multi-cloud environments and evaluated its performance.

Developing a new policy negotiation framework: Based on the meaning negotiation [20] and the bargaining model [21], we propose a framework [22] to negotiate security policy. The model proposed manages from indisputable to flexible preference. In addition, we advance an approach for comparison and evaluation of security policies: negotiator makes a proposal and evaluates the opponent one. Dissimilar results of evaluation lead to different proposals. The great advantage of our method is that it integrates security policy in the negotiation process by developing an exhaustive framework which covers policy evaluation, negotiation configuration, negotiation protocol and negotiation algorithm.

1.3 Organization

The remainder of the dissertation is composed of two parts. The first part (Chapters 2, 3, 4) deals with the SP selection problem which takes security policy into consideration. The second part (Chapters 5, 6) handles the issues of security policy negotiation.

Chapter 2 provides a comprehensive background on security policies, particularly the access control policy. It also gives an overview of the service level agreement (SLA) contract and the related frameworks. Moreover, this chapter introduces the basic techniques for policy similarity measure (PSM). Finally, it reviews virtual resource allocation approaches in Cloud Computing, including those taking security issues into consideration.

Chapter 3 presents a new security policy measure approach for SP selection. The generic PSM method is introduced at first. Then the experiments with our PSM algorithm and some related results are given. Finally, this chapter demonstrates a prototype which executes the SP selection towards resource allocation on Cloud storage.

Chapter 4 deals with a formal approach to express and enforce security policy for virtual resource allocation in IaaS Cloud. Based on the WS-Agreement [23] template, we integrate security requirements in SLA contract then the related security policies can be derived. The deployment solution is also generated from the security policy and non-security constraints. The chapter is ended by our Openstack-based implementation with evaluation.

Chapter 5 provides more comprehensive background on the negotiation paradigm with a focus on trust negotiation (TN), access negotiation, access control policy negotiation and meaning negotiation (MN). Although meaning negotiation does not concern the security aspect, the belief fusion technology handles the process of reaching agreement and this technology is adopted in security policy negotiation presented in the next chapter.

Chapter 6 shows how an agreement can be reached in security policy negotiation by our framework. The core negotiation algorithm is also illustrated with some theoretical results. The chapter ends with a detailed

negotiation scenario between a vehicle and a service station.

Chapter 7 concludes the dissertation and provides our perspectives and future work.

Part I

Service Provider Selection

Chapter 2

State of the Art

2.1 Introduction

Security policy are gaining a prominent place in research and industry domains. Access control policy is one type of security policies and its enforcement guarantees the usage privilege of the system. In an environment where exist SCs and SPs, the first thing to do for a SC before enforcement of security policies is to choose SP(s) which meet SC's requirement and preference. The SLA contract, although be widely used to specify QoS requirements, is also used to carry security-related preference. In this chapter, we are interested in the related work of SP selection. To this end, we firstly introduce different access control policy models. A comprehensive background on SLA contract is then given. Next, we provide a brief overview of relevant work on Policy Similarity Measure (PSM) which is helpful in the process of SP selection. Finally, we close this chapter with an overview on concrete SP selection approaches in the field of Cloud Computing.

2.2 Access Control Model

Access control, more precisely, authorization, is a basic and critical mechanism often used for operating systems. It provides a control solution for some entities (called subjects) to access some other entities (called objects)

through some actions in the system. Usually presented as a software module, access control is a traditional mechanism by means that software applications (originally operating systems) answer the question (request) “can the entity identified as S manipulate the object O via the action A?”. Here the verb “can” should be regarded as privileges but not as capabilities. At the same time, this question can be contextualized with respect to the trust issue as “can I trust S enough to allow him performing the action A on the object O?”. In this section, we present the different access control models and languages that have been proposed to answer such questions. The abstract model of access control mechanism is depicted in Figure 2.1.

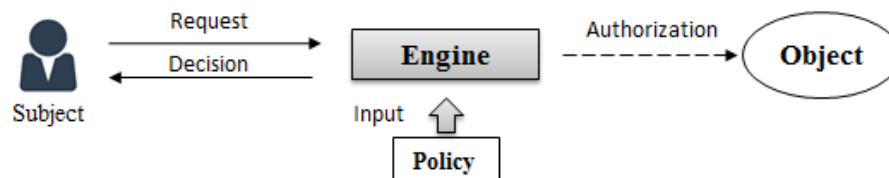


Figure 2.1: Basic access control model

- The *request* represents the type of interaction for which an authorization is requested (e.g. read, use or login).
- The *subject*, is the abstract entity (a human, a program or an artificial agent) requiring authorization.
- The *object* represents the resource that the requester wants to interact with (e.g. a file, a service).
- The *engine* is the decision module that determines if the requester is authorized to perform the requested interaction.
- The *decision* is the reply from the engine regarding the request (e.g. accept, refuse).

2.2.1 Discretionary Access Control (DAC)

Discretionary access control (DAC) is one of the most widespread access control models. It is a decentralized solution. Each object is controlled by its owner and an action enables subjects to have direct access to objects. In DAC, security policy is limited to permissions which specify relations between subjects, objects and actions. The access matrix model provides a framework for describing DAC. Formalized by Harrison, Ruzzo, and Ullmann, the HRU model [24] is such a framework which applies to subjects, objects and actions.

The Access control list (ACL) is an implementation of the HRU model and it is the oldest and most basic form of access control policies. It is commonly deployed in operating systems such as UNIX. In the most general form, a permission is a triple (s, o, a) , stating that a user s is permitted to perform an action a on an object o . Let S be the set of all users of the system, O the set of all objects and A the set of all possible actions. The ACL policies represent a function $f : S \times O \rightarrow A$. Consequently, $f(s, o)$ determines the list of actions that the subject s is permitted to perform over the object o . Table 2.1 illustrates (as a matrix $A = |S| \times |O|$) the access control list of a system where $S = \{s_1, s_2, s_3\}$, $O = \{o_1, o_2, o_3\}$ and $A = \{a_1, a_2, a_3\}$.

subject \ object	o_1	o_2	o_3
s_1	$a_1, a_2, -$	a_2	a_2
s_2	a_2, a_2	$-$	$-$
s_3	a_1, a_1, a_2	a_1, a_2	a_1, a_2

Table 2.1: Example of an ACL policy

Although an ACL model is easy to implement, the approach is not suitable when the number of users largely increases. When future subjects, objects or actions are inserted in the system, the security policies must be updated. As a result, it is difficult to administrate the system and the amount of memory will be largely increased with the insertion of users and resources. Moreover, access control decisions are not related to any characteristic of the resource

and it makes such an approach very vulnerable to attacks such as identity usurpation [25].

2.2.2 Lattice-based Access Control (LBAC or MAC)

Unlike DAC models, the lattice-based access control (LBAC) model, also known as the mandatory access control model (MAC), is deployed when the access to an object depends on its characteristics and those of the subject, and not the wills of the object owner [26]. As illustrated in Figure 2.2, subjects' and objects' characteristics are represented by security labels (or levels) which are assigned to users and resources of the system. The objects' labels reflect the sensibility of a resource and the subject's label classifies the category of objects she is permitted to access. Systems implemented by LBAC models are often called multi-level security systems as the labels used represent a partial order (e.g. Top Secret, Secret, Confidential, Unclassified) which is assumed to form a lattice. In LBAC, the process of access control is reduced to the control of data flow and its objective is to guarantee that data coming from a higher level object never flows to a lower level subject, and that data coming from a lower level subject never flows up to an object of a higher level. For example, a read operation on a resource is represented as a data stream from the object to the subject, while a write access represents a flow of data from the subject to the object. These two security principles are respectively called "no-read-up" and "no-write-down". The Bell-LaPadula [27] is the most famous model implementing LBAC and it has been used in both military and commercial applications. LBAC models are quite efficient and remain relatively manageable in systems with a small number of labels. Nevertheless, its principal limitation is the lack of flexibility and scalability.

2.2.3 Role-based Access Control (RBAC)

The development of the Role-based Access Control (RBAC) was motivated by the fact that in most cases, sensitive resources were generally not owned by

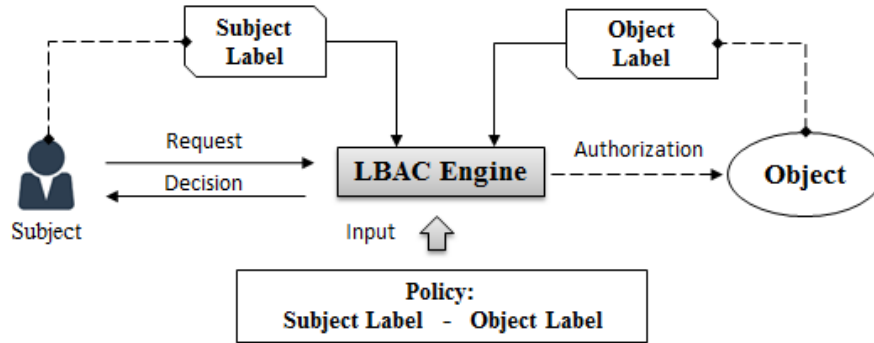


Figure 2.2: LBAC model

users but by the institution where users act in the capacity of a role for a job function. The RBAC policy holds an assignment relation that associates users to roles, and the roles to permissions granted. In this way, a role represents an intermediate layer between subjects and permissions and it brings scalability as the complexity of policy specification and administration is reduced. When a subject joins or leaves the system, only the links between the user and her related roles have to be updated. Therefore, the key components in RBAC are subjects, roles and permissions as illustrated in Figure 2.3.

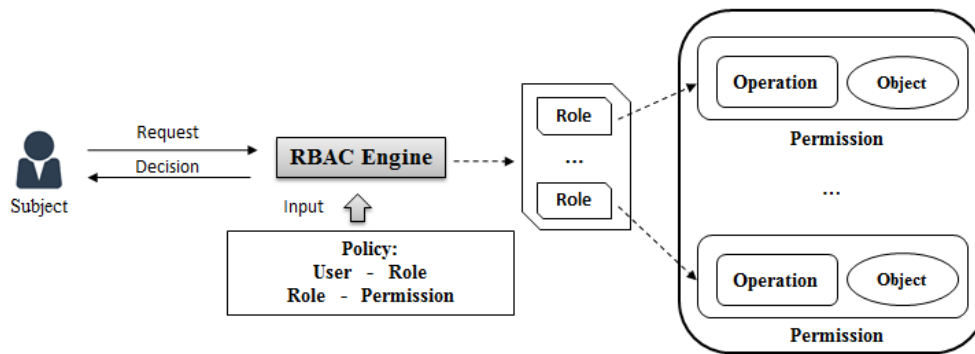


Figure 2.3: RBAC model

RBAC received in the last twenty years considerable attention that conducted to the proposition of a whole family of models [28, 29, 30, 31, 32, 33, 34, 35, 36, 37]. Among those models, $RBAC_0$ is the main and the simplest model. $RBAC_1$ extends $RBAC_0$ with the capability to specify hierar-

chies of roles and permissions' inheritance between roles. $RBAC_2$ extends $RBAC_0$ with constraints to enforce separation of duties, while $RBAC_3$ is a combination of $RBAC_1$ and $RBAC_2$.

Although RBAC is widely used in many commercial and government applications, it can not cover all the different requirements from the real world scenarios. For instance, the role inheritance mechanism proposed in $RBAC_1$ may not be sufficient to model some existing relationships. For example, an assistant may need to be authorized to execute some operations during the absence of her boss, but her role can not inherit all the privileges of the role of her boss. Towards the limitation, different ways of privilege propagation (delegation) should be supported and developed.

2.2.4 Attribute-based Access Control (ABAC)

The main idea of the attribute-based access control (ABAC) model is using policies which combine attributes together instead of identities, roles or clearances for authorizations [38, 39]. Unlike DAC, MAC and RBAC, the decision making of ABAC policies is based on disclosing credentials issued by third party attribute certifiers (e.g. organizations, companies, institutions). Consequently, the privilege of access can be obtained by subjects without being priorly known by the system administrator (or the resource owner). As illustrated in Figure 2.4, there exist four types of attributes.

Subject attributes. Subjects are the entities requesting access to objects.

Each subject can be characterized via an atomic attribute or a set of attributes without explicitly referring to its identity. Almost all information associated with a subject can be considered as an attribute such as name, role, affiliation and address.

Action attributes. Actions are the operations that the user wants to perform. Common action attributes in authorization requests are "read" and "write". In more complex scenarios, the action may be described by a combination of attributes.

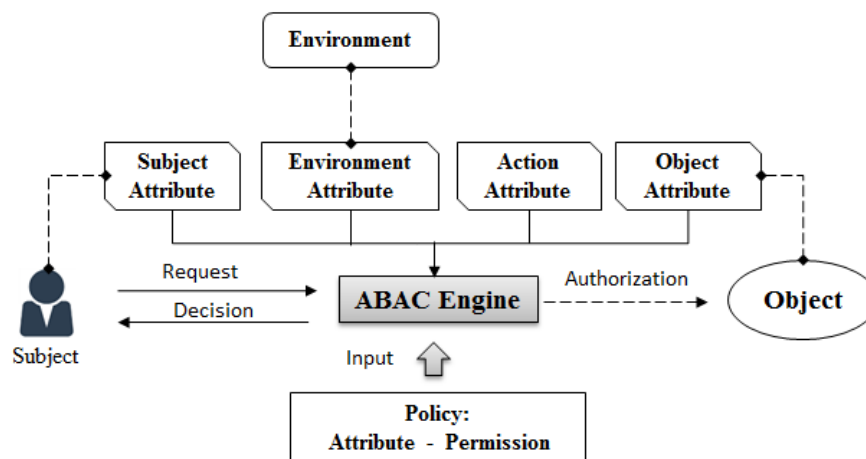


Figure 2.4: ABAC model

Object attributes. Objects are resources that the subject wishes to manipulate. Object attributes can affect the type of the permission granted. They may include the resource's name, type (e.g. text, image). The owner and the information of an object can be extracted automatically from its metadata.

Environment attributes. Unlike DAC, MAC and RBAC, in ABAC, the context (environment) of the interaction affects the access control decision. Context attributes can be time, date, location and so on.

The eXtensible Access Control Markup Language (XACML) is an access control policy language specified by the Organization for the Advancement of Structured Information Standards (OASIS). Based on XML, XACML specifies (i) a common security policy language; (ii) a processing model that describes how to interpret the policies; (iii) a request/response protocol to express access queries and reply to those queries.

The main components of XACML 3.0 policy language [1] are rule, policy and PolicySet. Figure 2.5 presents the language model. A rule is the most elementary unit of policy and it consists of a target, an effect, a condition, obligation expressions and advice expressions. The ABAC model can be implemented in XACML rules by placing subject attributes, resource attributes,

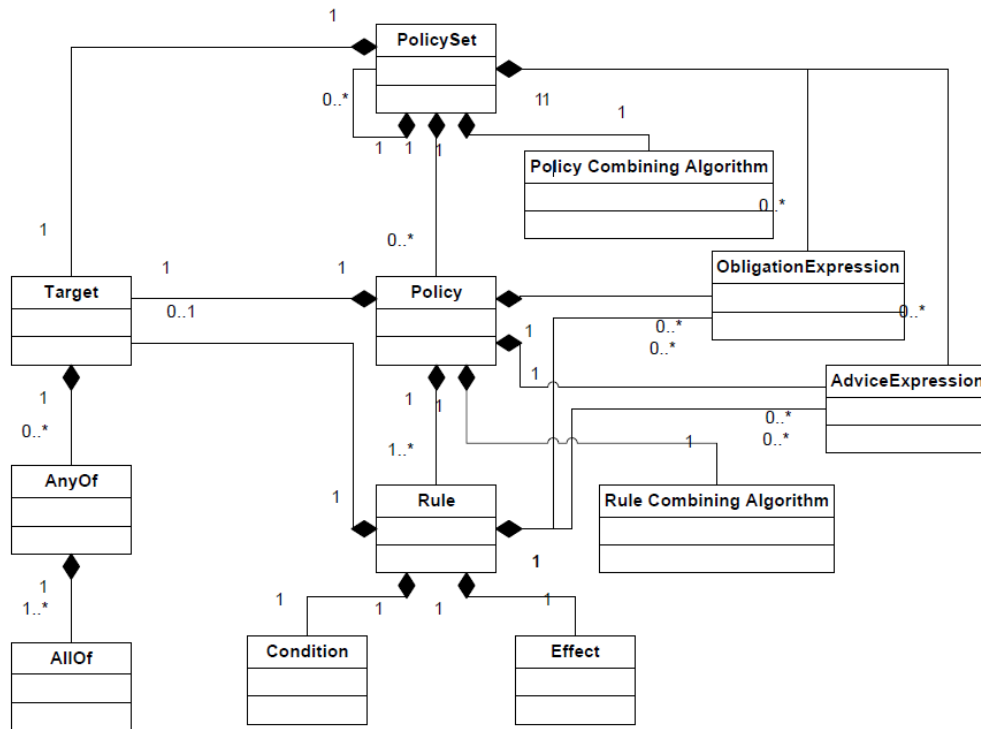


Figure 2.5: XACML 3.0 policy language model [1]

action attributes and environment attributes. PolicySet contains a set of policies. In order to map the relevant policies to a given request, targets can be explicitly specified for rules, policies, and PolicySet. A target defines the set of requests to which the rule is intended to apply in the form of logical expressions by attributes. Moreover, obligation expressions may be added by the rule editor. An obligation is a directive from the policy decision point (PDP) to the policy enforcement point (PEP) (Figure 2.6) on what must be carried out before or after an access is approved. If the PEP is unable to comply with the directive, the approved access may or must not be realized. Figure 2.6 shows the data flow diagram which consists of some major actors:

- **Policy Administration Point (PAP):** manages and defines the policies that will apply.
- **Policy Decision Point (PDP):** evaluates and makes authorization decisions.

- **Policy Enforcement Point (PEP):** intercepts access requests from a user to a resource and enforces the PDP decision.
- **Policy Information Point (PIP):** provides external information to PDP, such as environment and resource attribute information.
- **Context Handler:** converts access requests from the native request format to the XACML format and also converts XACML authorization decisions to the native response format. At the same time, it collects attribute information and resends it to PDP.

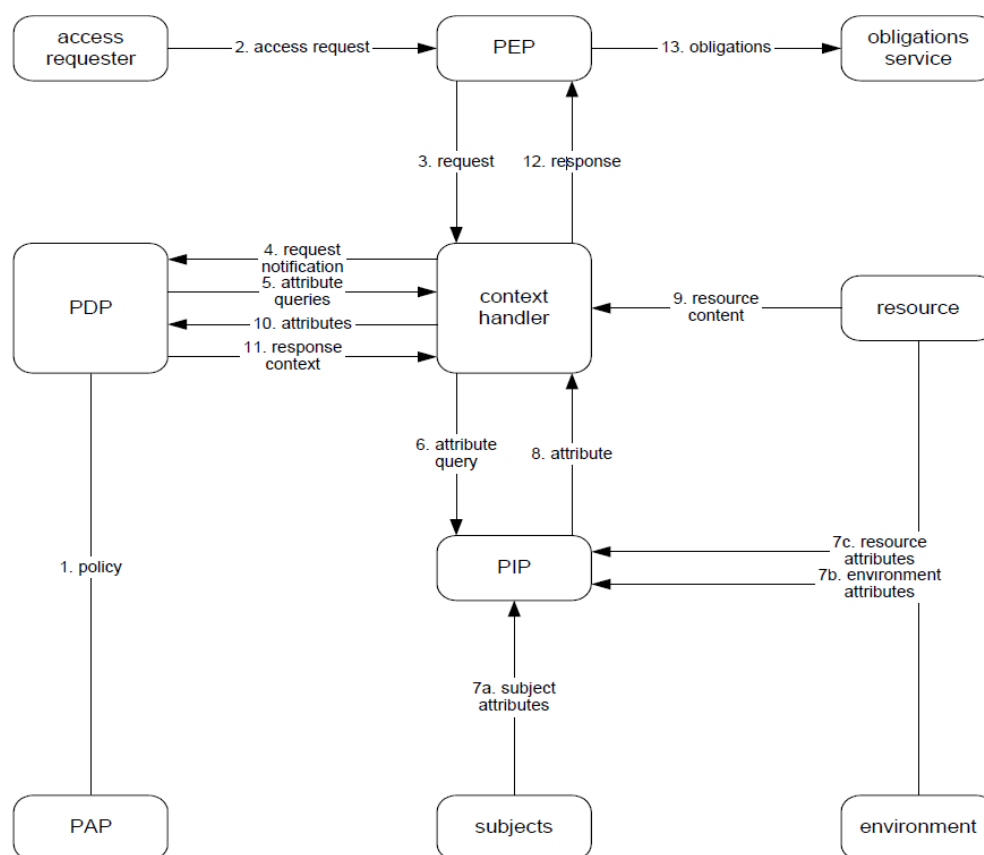


Figure 2.6: Data-flow diagram of XACML 3.0 policy language [1]

In a typical XACML usage scenario, a subject wants to take some actions on a particular target. The access request is firstly submitted to the PEP. Then the context handler forms an XACML request message with attributes of the

subject, action, target, and any other relevant information and sends it to the PDP. After analyzing the request and determining whether the access should be granted or denied according to the XACML policies, the PDP returns the response context (including the authorization decision) to the context handler. Finally, context handler translates the response context to the native response format and sends to the PEP.

Regarding the policy evaluation, the effect indicates the consequence of a rule. Rules may optionally contain a condition, which consists in a Boolean expression that further limits the rule applicability. There exist four values of access control decision: Permit, Deny, NotApplicable and Indeterminate. The latter two values are returned when an error occurs and no decision can be made or when the request can not be answered by the queried service, respectively. In order to decide the final result of composed decisions in PolicySets, various policy combining algorithms are used. For example, the Deny-overrides algorithm gives priority to deny rules. In XACML 3.0 [1], there exist 12 types of policy combining algorithms.

The ABAC brings flexibility and interoperability for policy definition and it can be used in lots of application scenarios such as web service [40] and Cloud Computing. Nevertheless, the flexibility and interoperability make policy administration more difficult: a potentially large number of attributes must be understood and managed, and attributes must be selected by experts. In addition, attributes have no meaning until they are associated with subject, object or environment, thus it is not practical to audit [41]. At the same time, the XACML standard is still not widely adopted by large enterprises by developing their authorization engines and commercial support such as software library is limited.

2.2.5 Organization-Based Access Control (OrBAC)

The OrBAC model [19] is an extension of the RBAC model. By defining a conceptual and industrial framework, it meets the needs of information security and sensitive communication and allows the policy designer to define a security policy independently. The concept of organization is fundamental

in OrBAC. An organization is an active entity that is responsible for managing a security policy. Each security policy is defined for an organization. The model is not limited to permissions, but also includes the possibility to specify prohibitions and obligations. Besides, the security rules do not apply statically but their activation may depend on contextual conditions. Context [42] is defined through logical rules and it can be combined in order to express conjunctive context, disjunctive context and negative context. An OrBAC policy is defined as: **security_rule (organization, role, activity, view, context)** where **security_rule** belongs to {permission, prohibition, obligation}. Once a security policy has been specified at the organizational level, it is possible to instantiate it by assigning concrete entities to abstract entities by the predicates which assign a subject to a role, an action to an activity and an object to a view (Shown in Figure 2.7). Meanwhile, all the operations are related to a specified context:

- *empower(org, subject, role)*: in organization *org*, *subject* is empowered in *role*.
- *consider(org, action, activity)*: in organization *org*, *action* implements *activity*.
- *use(org, object, view)*: in organization *org*, *object* is used in *view*.
- *hold(org, subject, action, object, context)*: in organization *org*, *subject* does *action* on *object* in *context*.

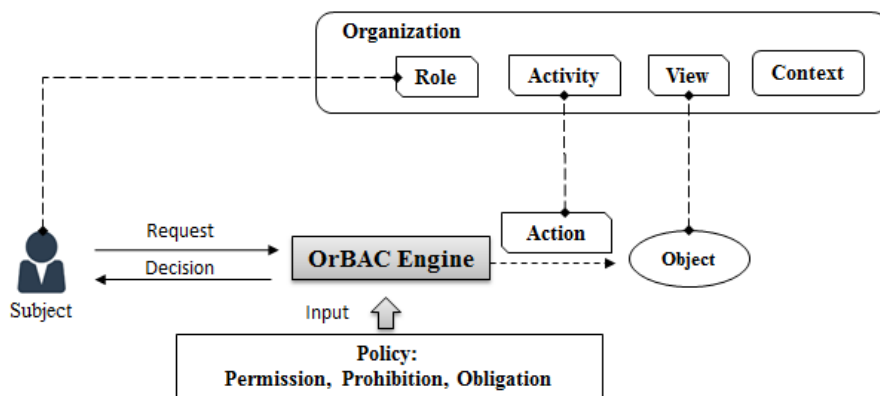


Figure 2.7: OrBAC model

Based on the above definitions, a concrete permission policy can be derived by the following rule ¹:

$$\begin{aligned}
 & permission(org, role, activity, view, context) \\
 & \wedge empower(org, subject, role) \wedge consider(org, action, activity) \\
 & \wedge use(org, object, view) \wedge hold(org, subject, action, object, context) \\
 & \rightarrow is_permitted(subject, action, object)
 \end{aligned}$$

The MotOrBAC tool [43] enables us to visualize OrBAC policies, it implements the OrBAC model and its administration model AdOrBAC [44]. Developed by the SFIIS team of Telecom Bretagne, it provides an user-friendly interface (GUI) to specify and manage OrBAC policies and also AdOrBAC policies. Shown in Figure 2.8, the MotOrBAC tool is composed of two separate modules: MotOrBAC GUI and OrBAC API. The former displays policies with the associated entities and the latter can be used to create and manage security policies by programming. The OrBAC API uses a custom inference engine which uses the join/fork framework from java 7 to provide an efficient derivation process. The policy is saved in an XML document where the abstract and concrete entities are stored [2].

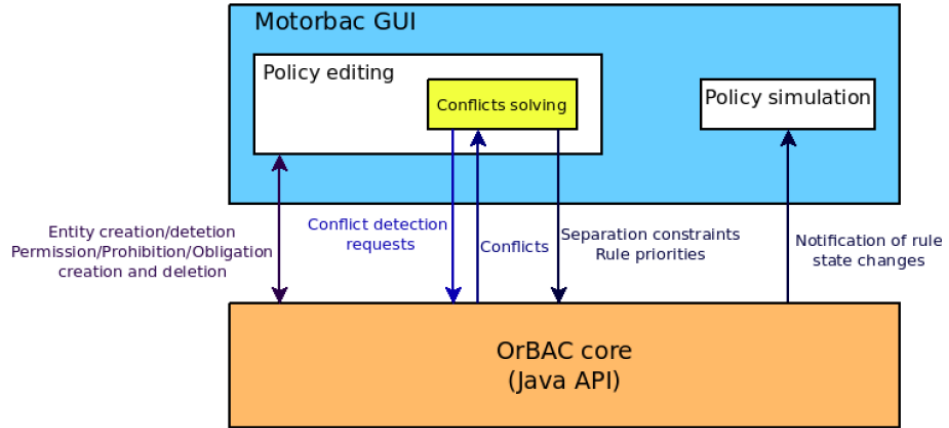


Figure 2.8: The MotOrBAC tool architecture [2]

¹A concrete prohibition policy $is_prohibited(subject, action, object)$ can be derived by the same way from $prohibition(org, role, activity, view, context)$.

2.3 Service Level Agreement (SLA)

With the development of Web Service, QoS between the SC and the SP becomes an important element which needs to be specified, measured and monitored. A SLA is such a contract between human-human, human-service and service-service. Given the diversity of disciplines using SLAs and the numerous interpretations that have been developed in recent years, we propose to start by presenting different SLA languages and frameworks then introduce the security aspect in SLA.

2.3.1 WSLA

Proposed by IBM in 2003, Web Service Level Agreement (WSLA) [3] covers the specification, enforcement and monitoring of SLAs. The WSLA language is based on XML and it allows the creation of machine-readable SLAs in the Web Service environment. Shown in Figure 2.9, a SLA created by WSLA contains typically the following components:

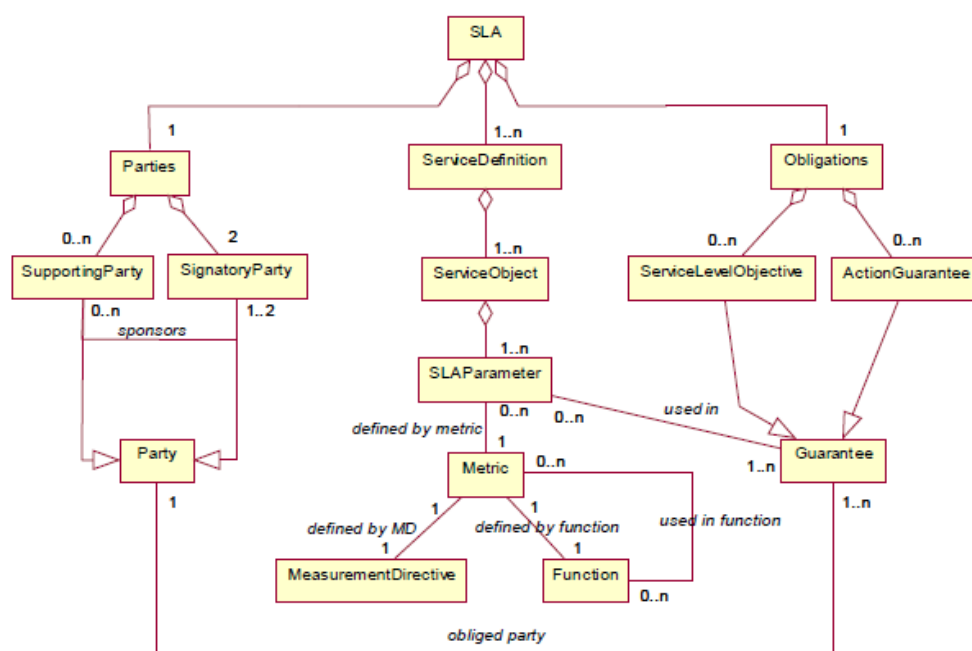


Figure 2.9: Overview of main WSLA concepts [3]

- **Parties:** identify the signing parties (SC and SP) and supporting parties (third parties). Third parties include monitoring providers, condition evaluators and management providers.
- **Service Definition:** specifies the characteristics of the service and its observable parameters.
- **Obligations:** define various guarantees and constraints that may be imposed on SLA parameters.

As an initiative attempt, WSLA proposes a SLA language and a global framework for SLA management. However, related negotiation protocol has not been developed and the specification has not been updated since 2003.

2.3.2 WS-Agreement

WS-Agreement [23] is developed by the Grid Resource Allocation Agreement Protocol (GRAAP) Working Group of the Open Grid Forum. The specification is an XML based language. The structure of WS-Agreement consists of three parts: name, context and terms. Context contains the meta-data for the entire agreement. It specifies the participants in the agreement and the lifetime of this agreement. There exist two term types: service description terms that describe the functionality delivered and guarantee terms outline the assurance on service quality for each piece of functionality. Unlike the WSLA, the WS-Agreement language is extensible by allowing the definition of domain-specific service level objectives. For example, different term description languages such as the Job Submission Description Language (JSDL) [45] could be used to describe service terms and guarantee terms. Such flexibility makes WS-Agreement widely used by lots of research and industrial projects such as BREIN [46], IRMOS [47], and OPTIMIS [48].

Besides the WS-Agreement language, the WS-Agreement negotiation protocol [4] is also proposed. A negotiation may then result in the creation of an agreement using the WS-Agreement specification. During the negotiation, the input is a template which describes service capacity of a SP, the messages

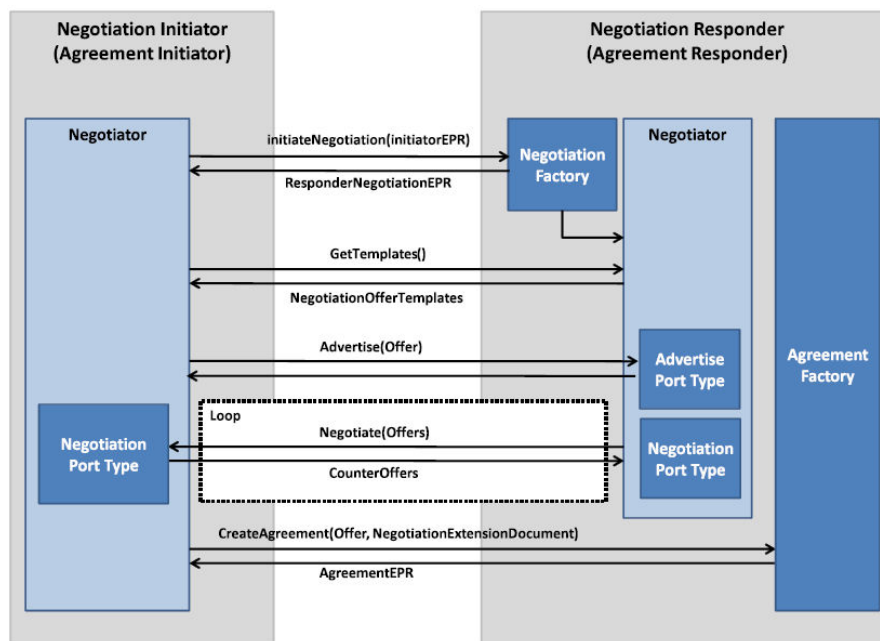


Figure 2.10: Asymmetric mode of WS-Agreement negotiation [4]

exchanged are XML-based and the output is a WS-Agreement contract. The protocol designs two negotiation modes: the asymmetric deployment mode and the symmetric deployment mode. Figure 2.10 presents the asymmetric deployment mode. In this mode, the negotiation process is driven by the SC (negotiation initiator). At first, the SC initiates a new negotiation process by calling the **"initiateNegotiation"** operation. After querying the negotiable templates from the new created negotiation instance, the SC selects the template it wants to negotiate and creates an initial negotiation offer based on the selected template. This offer is then sent to the SP by calling the **"negotiate"** operation. The SP creates one or more counter-offers and sends them back to the SC. The SC chooses the counter-offer which fulfills its requirements best and creates a new agreement with the server by executing the **"createAgreement"** operation. In this scenario, the SP is under a passive role. It does not control the negotiation process and it only reacts to the negotiation requests. In the symmetric deployment mode, both sides implement the **"negotiate"** operation, thus both parties have an active role in the negotiation process.

2.3.3 RBSLA

Rule-Based Service Level Agreement (RBSLA) [49] is a declarative mark-up language that uses knowledge representation concepts for rule-based policy and SLA contract specification. Based on RuleML [50] and logic programming, it provides a set of abstract language constructs to represent, manage, enforce and automatically monitor SLAs at runtime. The rule based SLA approach consists of three layers: (i) knowledge representation layer: a rule engine combining several logical formalisms; (ii) declarative contract logic layer: supports the expression of the RBSLA; (iii) management layer: the contract management tool.

Although RBSLA is designed to be compatible with existing standards, it has not become a standard itself and the high expressiveness makes it difficult to understand and apply by the non-experts. There does not exist any update on theory, case study and tools since 2006.

2.3.4 SLAng

SLAng [51] has been developed in University College London by deriving SLA requirements from real world SLAs. It is a model for inter-organisational service provision for storage, network, middleware and application levels. It focuses on the utilization of SLAs in support of the model-driven development. Based on XML, SLAng is divided into vertical and horizontal SLAs. Horizontal SLAs are contracted between different parties providing the same kind of service and vertical SLAs regulate the support parties getting from their underlying infrastructure. In SLAng, six different SLA types are defined, among them, vertical SLAs are:

- **Hosting:** between service provider and host.
- **Persistence:** between a host and storage service provider.
- **Communication:** between application or host and Internet service providers.

Horizontal SLAs are:

- **Service provision:** between an application or service and service provision.
- **Container:** between container providers.
- **Networking:** between network providers.

For each kind of SLA, a general structure is defined, including responsibilities of the SC, SP and their mutual responsibilities.

The SLAng language proposes a global architecture used to define the SLA contract for inter-organisational service provision. However, it stays in the language level due to the lack of related framework and negotiation protocol.

2.3.5 Security related SLA

Although traditional SLA focuses on the issues of QoS and performance, SLA-based trust and security management have been investigated in recent literature. The concept of security service level agreement is first proposed by Henning [52] as a mechanism to specify the security services required for an effective enterprise. SLA is used to explicitly state the obligation of the providers in terms of implemented security mechanisms, their effectiveness and the implication of possible mismanagement [53]. There have been some initiatives in the field of Cloud Computing that consider security aspects in SLAs. In [5], the authors present a framework (Figure 2.11) for security issues of SLAs in Cloud Computing. The objective of the framework is to help potential Cloud service customer (CSC) to identify the necessary protection mechanisms and facilitate automatic service composition based on a set of predefined security requirements. Chen-Yu et al. describe an ontology [54] for representing security SLAs (SSLA). Based on 13 classes, the proposed ontology can be used to understand the security agreements of a provider, to negotiate the desired security levels, and to audit the compliance of a provider with respect to federal regulations such as HIPAA standards [55].

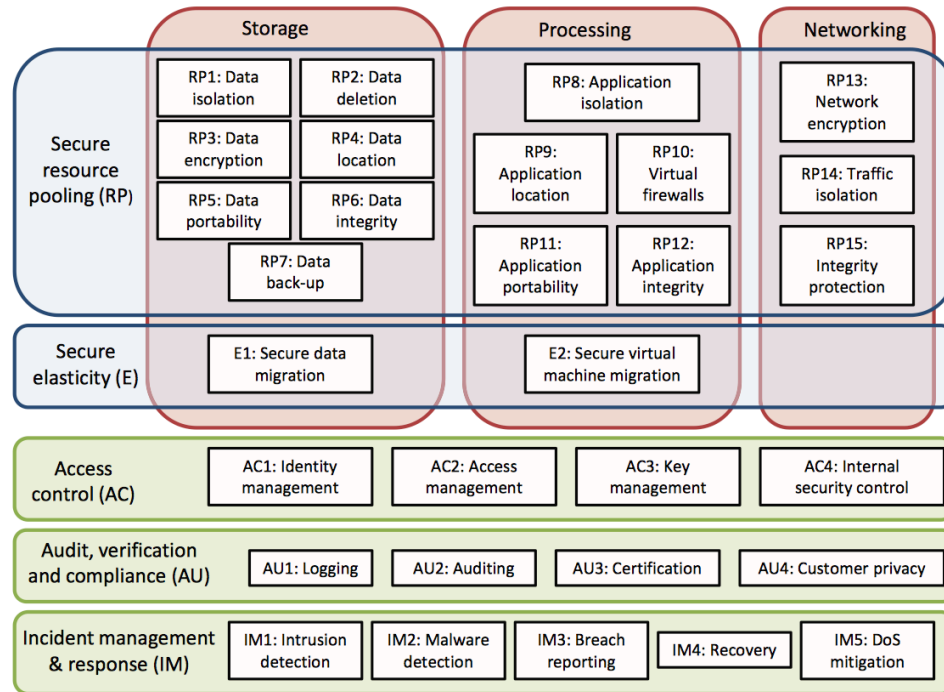


Figure 2.11: Framework for security mechanisms in Cloud SLAs [5]

The Cloud Security Alliance (CSA) [56] is a non profit organization that aims at promoting the use of best practice to increase the security level of Cloud infrastructures. CSA has designed a self-assessment questionnaire framework to define the security information contained in a Security Level Agreement (SecLA): the Consensus Assessments Initiative Questionnaire (CAIQ) [57] which is destined to Cloud service provider (CSP) to document the implemented security measures. Containing more than 200 security relevant questions, CAIQ helps a CSC to understand security coverage and guarantees of Cloud offers. In 2014, the European Commission published standardization guidelines for Cloud Computing SLA [58]. In order to improve the clarity and increase the understanding of SLAs for Cloud services in the market, the guidelines provide general recommendations to CSP and CSC about what they could agree on using SLAs. In terms of security SLA, it covers 8 aspects such as authentication, authorization, cryptography and vulnerability management. Recently, a key Cloud SLA standardization activity

is being carried out by ISO/IEC on “19086 - Information Technology (Cloud Computing) SLA Framework and Terminology”. This prospective standard will address (i) the definition of a standardized framework for Cloud SLAs including both a vocabulary and comprehensive catalogue of commonly used Service Level Objectives (SLOs); (ii) the definition of Cloud SLA-related metrics; (iii) core requirements for implementation; (iv) security and privacy in Cloud SLAs [59].

2.4 Policy Similarity Measure (PSM)

2.4.1 Problematic

Today the use of similarity measure for comparing security policies becomes a crucial technique in a variety of scenarios, such as finding the SP(s) which satisfies SC’s security concerns. Before a collaboration is conducted between different actors, an actor A may need to know if the other actor guarantees a similar level of A ’s security policies. In case that SPs expose their security policies for a SC to evaluate and select, policy comparison is one of the main mechanisms to that end. It consists in measuring the similarity between two security policies and giving an evaluation score.

The first thing to measure the similarity between security policies is to give the mathematical definition of PSM. In [60], Lin et al. propose Equation (2.1) for PSM. In the equation, p_1 and p_2 are two security policies for measure; $Sreq$ denotes the quantity of the access requests with the same decisions from p_1 and p_2 ; Req is the quantity of the access requests applicable to either p_1 or p_2 :

$$S_{policy}(p_1, p_2) = |Sreq|/|Req| \quad (2.1)$$

In an example that we will use in Chapter 3, we consider three XACML policies P_1 , P_2 and P_3 illustrated in [60]. These policies are defined for managing an information system of a research laboratory.

Policy P_1

```

1 PolicyID=P1, Rulecombining=Deny-override
2 <PolicyTarget
3   <Subject GroupName=IBMOpenCollaboration>>
4   <RuleID=R11 Effect=Permit>
5     <Target
6       <Subject Designation is {Professor,PostDoc,Student,TechStaff}>
7       <Resource FileType is {Source,Documentation,Executable}>
8       <Action AccessType is {Read,Write}>>>
9   <RuleID=R12 Effect=Deny>
10    <Target
11      <Subject Designation is {Student,PostDoc,TechStaff}>
12      <Resource FileType is {Source,Documentation,Executable}>
13      <Action AccessType is {Write}>>>
14    <Condition 19:00<=Time<=21:00>

```

Policy P_2

```

1 PolicyID=P2, Rulecombining=Deny-override
2 <PolicyTarget
3   <Subject GroupName=IBMOpenCollaboration,IntelOpenCollaboration>>
4   <RuleID=R21 Effect=Permit>
5     <Target
6       <Subject Designation is {Student,Faculty,TechStaff}>
7       <Action AccessType is {Read, Write}>>>
8     <Condition FileSize <=120MB>
9   <RuleID=R22 Effect=Permit>
10    <Target
11      <Subject Designation=TechStaff>
12      <Action AccessType is {Read,Write}>>>
13    <Condition 19:00<=Time<=22:00>
14  <RuleID=R23 Effect=Deny>
15    <Target
16      <Subject Designation=Student>
17      <Action AccessType=Write>>>
18    <Condition 19:00<=Time<=22:00>
19  <RuleID=R24 Effect=Deny>
20    <Target
21      <Subject Designation is {Student,Faculty,Staff}>
22      <Resource FileType=Media>

```

```
23  <Action AccessType is {Read,Write}>>
```

Policy P_3

```
1  PolicyID=P3, Rulecombining=Deny-override
2  <PolicyTarget
3    <Subject GroupName=Payroll>>
4    <RuleID=R31 Effect=Permit>
5      <Target
6        <Subject Designation=BusinessStaff>
7        <Resource FileType=".xls">
8        <Action AccessType is {Read,Write}>>
9        <Condition 8:00<=Time<=17:00,FileSize<=10MB>
10     <RuleID=R32 Effect=Deny>
11       <Target
12         <Subject Designation=Student>
13         <Action AccessType is {Read,Write}>>
```

From a user's perspective, P_1 is more similar to P_2 than P_3 because most activities described by P_1 for the data owner are allowed by p_2 . Our motivation is to quickly compute similarity scores $S_{policy}(P_1, P_2)$ and $S_{policy}(P_1, P_3)$ with the expectation that the former is higher than the latter. The expected result is to indicate that the similarity between P_1 and P_2 is much higher than the similarity between P_1 and P_3 .

2.4.2 Use Cases

The following three PSM related scenarios are extracted from [61] and [62].

- **Federation:** There are a number of organizations that are currently in a federation with common security policies. A new organization is possible to join the federation by negotiating with the existing members in order to reach certain agreements. One step of the negotiation process is to achieve a common understanding about security policies. To this end, PSM may be helpful to quickly find out the organization whose security policies are relevant to policies owned by the federation and filter the dissimilar ones.

- **Delegation transaction:** In some context, an organization needs to delegate its privileges to others. From the delegator's point of view, it is necessary to know if its security policy is similar to the one of the delegatee. Thus PSM technology is capable to affect the decision of the delegation.
- **Service provisioning in the cloud:** In Cloud Computing, as an user's data is usually processed remotely in unknown machines that she does not own or operate, it is necessary to select a SP whose security policy is close to the one required by the user. With PSM technology, the user is capable to estimate the similarity between two given policies and rank the SPs. After that, policy integration and policy enforcement will be executed.

2.4.3 Existing Approaches

Most existing approaches to evaluate the policy similarity are based on XACML [63] policies. Lin et al. [64] propose an algorithm to evaluate policy similarity by calculating the similarity score between two XACML policies. This is indeed a pioneering work and it effectively distinguishes between categorical predicate and numerical predicate. The second version of the algorithm [60] advances the measure algorithm for numerical predicate and integrates ontology matching. However, the work has two limitations. Firstly, the algorithm only focuses on the literal level (semantic distance calculation) but not logic aspect of security policy. As a result, the similarity score computed may have a large difference with the test value in real cases (presented in Appendix B). Secondly, the algorithm contains 9 weight parameters which need to be configured. Choosing the proper values is not easy. In addition, there are two variants of the former work. Bei et al. [65] investigate the opposite of similarity: dissimilarity. In order to address the rule relationship comparison, they apply fuzzy theory to compute rule dissimilarity. Pham et al. [61] improve the similarity computing approach specified by Lin et al. [64] and also propose a mechanism to calculate a dissimilarity score by identifying related policies which are likely to produce different ac-

cess decisions. The PSM technique is then integrated in various scenarios. Lin et al. [62] present a novel data protection framework in which the policy similarity comparison approach is applied to the policy ranking model. Cho et al. [66] propose a technique that allows similarity evaluation of encrypted policies. Shaikh et al. [67] suggest using similarity measure to select services in a distributed and heterogeneous environment. Bertolino et al. [68] put forward a new approach for access control test prioritization based on similarity.

2.5 Virtual Resource Allocation

2.5.1 Problematic

Today Cloud Computing is essentially provider-centric. An increasing number of fiercely competing CSPs operate multiple heterogeneous Clouds. In terms of infrastructure as a service (IaaS), each provider offers its own, feature-rich solutions for customer virtual machines (VMs). More significantly, in Cloud IaaS, physical hardware is usually shared by multiple virtual resources for maximizing utilization and reducing costs.

In Cloud Computing, a SP's system can be viewed as a large pool of interconnected physical hosts and we use H to present the finite set of hosts from a CSP. V is a VM to be allocated. h_i and v_i represent an unique virtual machine ID and HOST ID separately. With these definitions, virtual resource allocation problem can be summarized as follows:

Definition 1. *Virtual Resource Allocation Problem*

Given a set of HOSTs $H=\{h_1, h_2, \dots, h_m\}$, a set of VMs $V=\{v_1, v_2, \dots, v_n\}$, a set of constraints for HOSTs $C_H=\{C_{h1}, C_{h2}, \dots, C_{hm}\}$, a set of constraints for VMs $C_V=\{C_{v1}, C_{v2}, \dots, C_{vn}\}$. C_H and C_V are logical formulas that define the allowed combinations of deployment conditions for H and V . Find the mapping $p:\{v_1, v_2, \dots, v_m\} \rightarrow \{h_1, h_2, \dots, h_n\}$ where $p(v_i) = h_j$ ($1 \leq i \leq m, 1 \leq j \leq n$).

Unfortunately, virtual resource allocation suffers from a lack of homogeneity: lots of Cloud virtual resources can not be deployed due to deficien-

cies in (1) unified expression; (2) interoperability. Lack of unified expression results in vendor lock-in: services are tightly coupled with the provider and depend on its willingness to deploy them. Lack of interoperability stems from heterogeneous services, and more importantly of service-resource mapping, not compatible across providers. For better interoperability and control, Cloud brokering is nowadays the rising approach towards the user-centric vision. It may be seen as a paradigm in delivering Cloud resources (e.g. compute, storage, network). With the help of brokering technology, user's security needs will be necessarily considered in the cloud and these security requirements can be included in a SLA contract which is a legal document where the service description is formally defined, delivered, and charged.

2.5.2 Existing Approaches

Although virtual resource scheduling problems are NP-complete, it is well-studied by the research community by proposing various heuristic and approximate approaches for addressing different issues. Among three service models (SaaS, PaaS and IaaS) of Cloud Computing, virtual resource allocation in IaaS Cloud has been investigated by some works in the literature. Some of these works [69, 70] focus on the capacity of CSP. In this case, some strategies like immediate, best effort and Nash equilibrium [71] have been applied to allocation algorithms in order to optimize the deployment algorithm with constraints such as QoS and energy [72]. Another effort is SLA-oriented resource management [73]. Wu et al. [74] propose a resource allocation algorithm for SaaS provider to minimize infrastructure cost and SLA violation. In [75], a SLA-aware PaaS Cloud platform that manages the complete resource life cycle is developed. With the WS-Agreement specification, a CSP defines a generic SLA model to deal with high-level metrics, close to end-user perception, and with flexible composition of the requirements from multiple actors in the computational scene. Among lots of CSC's requirements, security is a critical issue to be taken into account. Bernsmed et al. [5] present a security SLA framework for Cloud Computing to help potential CSCs to identify necessary protection mechanisms and facilitate

automatic service composition. Based on some existing frameworks such as ENISA [76] and CAIQ [57] developed in Europe, Cayirci et al. [77] design a Cloud adoption risk assessment model (CARAM) for CSCs to assess the risks that they face by selecting a specific CSP. Berger et al. [78] take isolation constraint and integrity guarantee into consideration and implement controlled access to network storage based on security labels. In [79], different virtual resource orchestration constraints are classified and expressed by attribute-based paradigm. Regarding these constraints, a conflict-free strategy is developed to mitigate risks in IaaS Cloud [80]. Most of the above works have been motivated by security requirements expressed by CSCs. In [81], a CSP specifies its security requirements including forbid constraint which forbids a set of VM instances from being allocated on a specified HOST. However, in multi-cloud environment, as CSCs and CSPs do not have a vision of each other before establishing their contracts, specifying security requirements can be very tricky for both sides. The main focus of these efforts is scheduling VMs either for the purpose of high-performance computing or satisfying security constraints according to the requirements of CSCs.

2.6 Conclusion

We have introduced the main types of security policy models for access control systems. We have seen that different models hold different specifications for abstract and concrete levels. Besides, we have seen that a SLA contract can be used to specify QoS requirements and there exist some efforts to integrate security parameters in it. However, putting security issues in a SLA contract suffers from the lack of integration of the security policy. In terms of security policy, the current PSM approach is not accurate enough and its configuration is complicated. This can affect the result of the SP selection. To end this chapter, approaches for virtual resource allocation have been presented with their advantages and limitations.

Motivated by the limitations of the current PSM method and virtual resource allocation approaches, we present in the following chapters, two

propositions on the SP selection for two use cases. In the first use case, SPs specify directly their security policies in SLA contracts and SC should choose the one(s) compliant with its security requirements. Towards this end, we develop a generic and light-weight method to compare and evaluate security policies belonging to different models. The second use case concerns the SLA contract with security requirements which can be transformed to concrete security policies. In order to fulfill the second use case, a policy-based framework for the CSP selection and virtual resource allocation in Cloud Computing is presented with a related implementation and some statistical evaluations.

Chapter 3

Similarity Measure for Security Policies

3.1 Introduction

A higher score (Formula 2.1) between policies p_1 and p_2 indicates that they are more likely to share an equivalent security level and yield the same decisions. As presented in section 2.4.3, existing approaches cover from semantic to numerical dimensions and the main work focuses mainly on XACML policies. However, few efforts have been made to extend the measure approach to multiple policy models and apply it to concrete scenarios. In this chapter, we propose a new algorithm to calculate the similarity score between two policies. The rest of the chapter is organized as follows. Firstly we introduce the policy similarity measure algorithm with an exhaustive calculation example. Then we illustrate an experiment in which the accuracy of our algorithm is demonstrated. Finally, we give an implementation in which our algorithm works for SPs ranking before the SP selection.

3.2 A Generic Policy Similarity Measure Method

The PSM assigns a similarity score S_{policy} for any two given policies, which approximates the percentage of the rule pairs having the same decision. The

formal definition is given in Equation (3.1), where $Num(sameDecision(r_{1i}, r_{2j}))$ denotes the number of the rule pairs having the same decision for the same access requests and $Num(allDecision(r_{1i}, r_{2j}))$ denotes the amount of the total decision pairs for access requests which are applicable to either policy p_1 or policy p_2 .

$$S_{policy}(p_1, p_2) \approx \frac{Num(sameDecision(r_{1i}, r_{2j}))}{Num(allDecision(r_{1i}, r_{2j}))}, \quad r_{1i} \in p_1, r_{2j} \in p_2 \quad (3.1)$$

The similarity score is a value between 0 and 1. Two equivalent policies are expected to obtain a similarity score equal to 1. We mention that the definition of the policy similarity score in [60] focuses on the percentage of the access requests obtaining the same decisions. Comparing with the former work, our definition of PSM is more fine-grained because the same decision from two policies can be derived from one or multiple rule pairs. Consequently, by considering decisions of rule pairs but not final policy decisions, our PSM is more accurate from both calculation and test aspects. More details are shown in Section 3.3.

3.2.1 Policy Structure

As a generic algorithm, our PSM can be applied on different policy models. This requires a transformation process before calculation. Policies are firstly split into different rules and each rule is expressed in the form of:

$$decision_effect(attr_name_1 : attr_value_1, \dots, attr_name_n : attr_value_n) \quad (3.2)$$

where *decision_effect* is a decision effect such as permit and deny; *attr_name* denotes the name of an attribute and *attr_value* represents an attribute value. We define $(attr_name_i : attr_value_i)$ as a rule element and it can be divided into the following two types:

- **Categorical element:** Attribute value belongs to the *string* data type or is a set of *string*. For example “*Role : admin*” and “*Action : [read, write, create]*” are categorical atomic elements.

- **Numerical element:** Attribute value can be integer, real, date/time data types. The value can be single one or a set or an interval. For example, elements “ $Time : \{3pm, 4pm, 5pm\}$ ”, “ $FileSize : (5, +\infty)$ GB”, “ $Time : [8 : 00, 18 : 00]$ ” are numerical atomic elements.

3.2.2 Example of Policy Transformation

In an example that we use throughout the chapter, we consider three XACML policies mentioned in Section 2.4.1. The policies after transformation to the Form 3.2 are:

policy1 (p_1)

$r_{11} : Permit(Role : \{professor, postDoc, student, techStaff\}, Action : \{read, write\}, Resource : \{source, documentation, executable\}, FileSize : all, Time : [0 : 00, 24 : 00])$

$r_{12} : Deny(Role : \{student, postDoc, techStaff\}, Action : write, Resource : \{source, documentation, executable\}, FileSize : all, Time : [19 : 00, 21 : 00])$

policy2 (p_2)

$r_{21} : Permit(Role : \{student, faculty, techStaff\}, Action : \{read, write\}, Resource : all, FileSize : (-\infty, 120]MB, Time : [0 : 00, 24 : 00])$

$r_{22} : Permit(Role : techStaff, Action : \{read, write\}, Resource : all, FileSize : all, Time : [19 : 00, 22 : 00])$

$r_{23} : Deny(Role : student, Action : write, Resource : all, FileSize : all, Time : [19 : 00, 22 : 00])$

$r_{24} : Deny(Role : \{student, faculty, staff\}, Action : \{read, write\}, Resource : media, FileSize : all, Time : [0 : 00, 24 : 00])$

policy3 (p_3)

$r_{31} : Permit(Role : businessStaff, Action : \{read, write\}, Resource : xls,$

$FileSize : (-\infty, 10]MB, Time : [08 : 00, 17 : 00])$

$r_{32} : Deny(Role : student, Action : \{read, write\}, Resource : all, FileSize : all, Time : [0 : 00, 24 : 00])$

It is worth noting that some numerical elements which have not been explicitly specified but hold their default values should be written explicitly in the rule structures after the transformation. For example, *time* element with the value of $[0 : 00, 24 : 00]$ is inserted in the rule r_{11} after transformation as other rules specified explicitly their *time* elements.

3.2.3 Overview of PSM Algorithm

Illustrated in Figure 3.1, the PSM algorithm takes two policies as the inputs and generates a similarity score as the output. The calculation process can be divided into four steps.

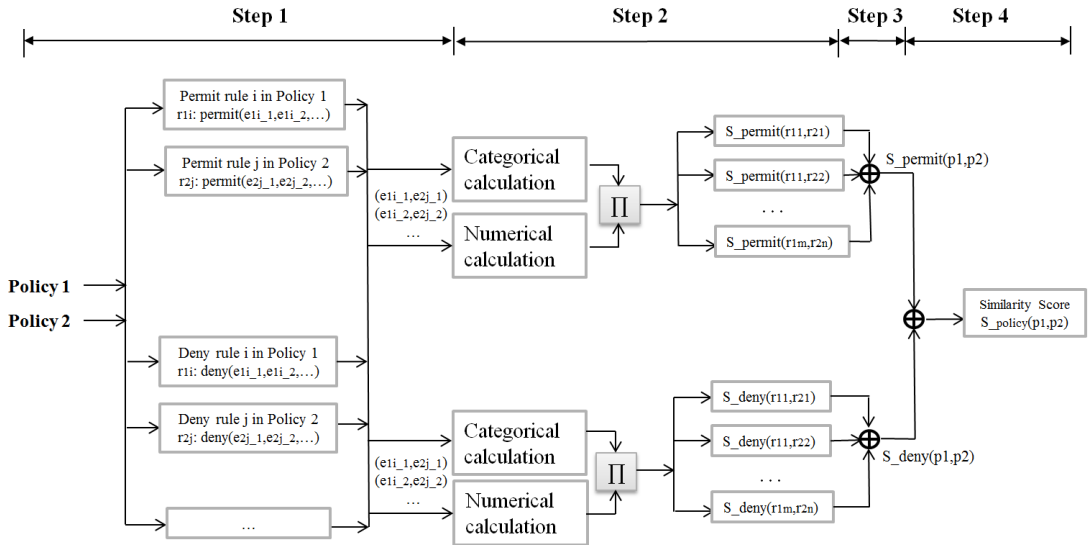


Figure 3.1: The process of similarity score calculation

Step 1: Policy transformation. Illustrated in Section 3.2.2, two policies are split into rules in Form 3.2 which consist of atomic rule elements

$e : (attr_name \oplus attr_value).$

$$\begin{aligned}
 p_1 : & \text{permit}(e_{1i_1}, e_{1i_2}, \dots), \text{permit}(e_{2i_1}, e_{2i_2}, \dots), \dots \\
 & \text{deny}(e_{1i_1}, e_{1i_2}, \dots), \text{deny}(e_{2i_1}, e_{2i_2}, \dots), \dots \\
 p_2 : & \text{permit}(e_{1j_1}, e_{1j_2}, \dots), \text{permit}(e_{2j_1}, e_{2j_2}, \dots), \dots \\
 & \text{deny}(e_{1j_1}, e_{1j_2}, \dots), \text{deny}(e_{2j_1}, e_{2j_2}, \dots), \dots
 \end{aligned}$$

Step 2: Score calculation for the rule pair. Scores of each rule pair belonging to the same decision effect d (permit, deny...) between two policies are calculated. In Equation (3.3), the score for each rule pair is the product of the scores of all the element pairs. Product operation is chosen because any mismatch of element pair may cause different replies from two policies. Details for element pair calculation are shown in Section 3.2.4.

$$S_d(r_{1i}, r_{2j}) = \prod_k S(e_{1i_k}, e_{2j_k}), \quad r_{1i} \in p_1, r_{2j} \in p_2 \quad (3.3)$$

Step 3: Decision effect calculation. Each $S_d(p_1, p_2)$ equals the sum of all the similarity scores of rule pairs in one decision effect (Equation (3.4)).

$$S_d(p_1, p_2) = \sum_i \sum_j S_d(r_{1i}, r_{2j}), \quad r_{1i} \in p_1, r_{2j} \in p_2 \quad (3.4)$$

Step 4: Total score calculation. Shown in Equation (3.5), the total score is based on the scores from different decision effects $S_d(p_1, p_2)$ and the total amount of rule pairs from all the decision effects.

$$S_{policy}(p_1, p_2) = \frac{\sum_d S_d(p_1, p_2)}{\sum_d Num(d)}, \quad d \in (\text{permit}, \text{deny}, \dots) \quad (3.5)$$

3.2.4 Similarity Score of Rule Elements

The score of a rule element pair can be calculated when two elements belong to the same decision effect and share the same attribute name. In Equation (3.3), the score of a rule pair is based on the rule elements having the same attribute name. When an element's attribute name does not appear in

another rule, the access decisions from the two rules are not affected. For this reason, we consider that the score of such element pair is 1. Rule elements can be divided into two types: categorical elements and numerical elements.

3.2.4.1 Similarity Score for Categorical Elements.

For categorical elements, we measure the exact match of two values. A higher score indicates that the two elements share more common attribute values. Equation for similarity score computing between two categorical elements e_1 and e_2 is defined as follows:

$$S_c(e_1, e_2) = \frac{\text{num}(v_1 \cap v_2)}{\text{num}(v_1 \cup v_2 \cup v_3 \dots \cup v_n)} \quad (3.6)$$

$S_c(e_1, e_2)$ presents the exact percentage of the same decision for one element pair extracted from the two rules. $\text{num}(v_1 \cap v_2)$ denotes the quantity of common attribute values between element e_1 and e_2 ; $\text{num}(v_1 \cup v_2 \cup v_3 \dots \cup v_n)$ is the quantity of attribute values among all the elements in two policies and these elements should 1) have the same attribute name 2) belong to the rules of the same decision effect.

Some policy models use abstract elements to represent a set of concrete values. For example, in RBAC, the *Role* element is an abstraction of *Subjects*; in OrBAC, a *Role* is a set of *Subjects*, an *Activity* is a set of *Actions* and a *View* is a set of *Objects*. In this case, the abstract values should be transformed to their related concrete values. For example, abstraction trees for *Role* and *Resource* elements of p_1 , p_2 , p_3 are shown in Figure 3.2 and Figure 3.3.

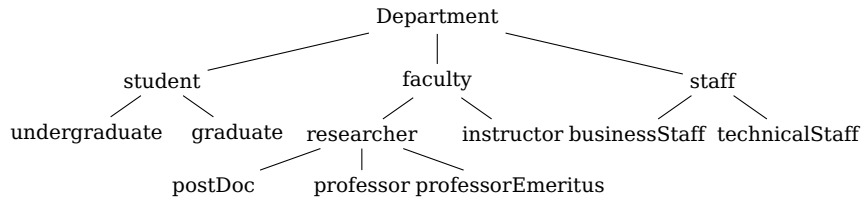


Figure 3.2: Abstraction tree for the *Role* element

To calculate the score of *Role* elements between rules r_{11} and r_{21} specified in

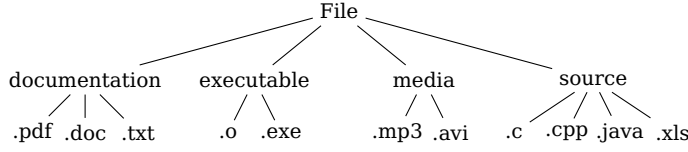


Figure 3.3: Abstraction tree for the *Resource* element

Section 3.2.2, as *student* and *faculty* are two abstract values, they should be translated into concrete values which are leaves: $\{undergraduate, graduate\}$ and $\{postDoc, professor, professor - Emeritus, instructor\}$. After the transformation, we find that the two elements share 5 common attribute values. The disjunction of all the *Role* elements from policy 1 and policy 2 contains 8 attribute values. Applying Equation (3.6), $S_c(e_{r_11(Role)}, e_{r_21(Role)}) = 5/8 = 0.625$.

Another application of the tree architecture is to represent the inheritance relation. The inheritance mechanism is defined in object-oriented programming as an efficient way to design an application. In Java, a class which is derived from another class is called a subclass. A similar mechanism for roles is used in the RBAC [82] and the hierarchy of roles is associated with inheritance of permission. The role inheritance mechanism is extended in the OrBAC model [83]: hierarchies of roles, views and activities are formally defined associated with inheritance relationships. In an inheritance tree, child elements can inherit the privileges of their parent elements. For example, the *Role* elements of a research laboratory may possess an inheritance tree for permission (Figure 3.4). When applying Equation (3.6), all the attribute values having the inheritance relationship in the same inheritance tree should be treated as identical ones.

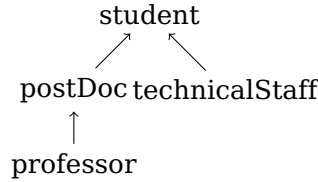


Figure 3.4: Inheritance tree for the *Role* element

3.2.4.2 Similarity Score for Numerical Elements.

The calculation for numerical elements is more complex because numerical attribute values may have different forms such as a single value, a set, a bounded interval and an unbounded interval. Here we propose a unified method defined in Algorithm 1 for computing the similarity score between two numerical elements. The algorithm takes two numerical elements with

Algorithm 1 $S_n(e_1, e_2)$: numerical similarity score calculation

Input: two numerical elements e_1 and e_2

Output: numerical similarity score

```

1: if  $e_1 = e_2$  then
2:   return 1
3: end if
4: if  $e_1 \cap e_2 = \emptyset$  then
5:   return 0
6: else
7:   if both  $e_1$  and  $e_2$  are bounded intervals then
8:     return  $\frac{Len(e_1 \cap e_2)}{Len(e_1 \cup e_2)}$ 
9:   else if both  $e_1$  and  $e_2$  are sets then
10:    return  $\frac{Num(e_1 \cap e_2)}{Num(e_1 \cup e_2)}$ 
11:  else
12:    return 0.5
13:  end if
14: end if

```

the same attribute name as inputs. Firstly, if two elements have the same attribute name, operator(s) and attribute value(s), the score is 1 (lines 1,2). Secondly, the two elements should be checked if their intersection is empty. The algorithm returns 0 as similarity score when the intersection is empty (lines 4,5). Otherwise, there are three cases:

- **Bounded intervals (lines 7,8):** Two elements' values are both bounded intervals. Length of an interval equals the distance between its end-points. To compute the score, we divide the length of the conjunction of two intervals by the length of their disjunction. For example, the score for time elements in r_{12} and r_{23} specified in Section 3.2.2 is:

$$S_n(e_{r_{12}(Time)}, e_{r_{23}(Time)}) = Len(21 - 19) / Len(22 - 19) = 0.67.$$

- **Sets (lines 9,10):** Two elements' values are both sets. To compute the score, we divide the cardinality of the intersection of two sets by the cardinality of their union. For example, $Time_1 = [3 \text{ am}, 4 \text{ am}, 5 \text{ am}]$, $Time_2 = [4 \text{ am}, 5 \text{ am}, 6 \text{ am}]$, $S_n(Time_1, Time_2) = 2/4 = 0.5$.
- **Other cases:** As calculation between two different forms is difficult, we assign a fuzzy value 0.5 as the similarity score. 0.5 is chosen because it is the average value of similarity score.

3.2.5 Example of Calculation

Here we present an exhaustive example to illustrate how the PSM works. Continuing with the three policies P_1 , P_2 , P_3 defined in section 2.4.1 and their abstraction trees introduced in section 3.2.4, we illustrate the four steps of calculation.

1. **Policy transformation:** Shown in Section 3.2.2, the three policies have already been transformed from XACML policies to rules composed of atomic elements.
2. **Rule pair calculation:** Applying Equation (3.3), (3.6) and Algorithm 1, we calculate scores for different rule pairs in each decision effect:

Permit :

$$S_{rule}(r_{11}, r_{21}) = \frac{5}{8} \times 1 \times \frac{9}{11} \times \frac{1}{2} \times 1 = 0.256$$

$$S_{rule}(r_{11}, r_{22}) = \frac{1}{8} \times 1 \times \frac{9}{11} \times 1 \times \frac{3}{24} = 0.013 \quad 1$$

Deny :

$$S_{rule}(r_{12}, r_{23}) = \frac{2}{8} \times 1 \times \frac{9}{11} \times 1 \times \frac{2}{3} = 0.136$$

$$S_{rule}(r_{12}, r_{24}) = \frac{4}{8} \times \frac{1}{2} \times 0 \times 1 \times \frac{2}{24} = 0$$

¹Scores of element pairs between rules r_{11} and r_{21} are: $S_c(e_{r_{11}}(Role), e_{r_{21}}(Role)) = \frac{5}{8}$, $S_c(e_{r_{11}}(Action), e_{r_{21}}(Action)) = 1$, $S_c(e_{r_{11}}(Resource), e_{r_{21}}(Resource)) = \frac{9}{11}$, $S_n(e_{r_{11}}(FileSize), e_{r_{21}}(FileSize)) = \frac{1}{2}$, $S_n(e_{r_{11}}(Time), e_{r_{21}}(Time)) = 1$.

3. **Decision effect calculation:** By Equation (3.4), scores of each decision effect are:

$$S_{permit} = S_{rule}(r_{11}, r_{21}) + S_{rule}(r_{11}, r_{22}) = 0.269$$

$$S_{deny} = S_{rule}(r_{12}, r_{23}) + S_{rule}(r_{12}, r_{24}) = 0.136$$

4. **Total score calculation:** The final similarity score between two policies is calculated by Equation (3.5):

$$S_{policy}(P_1, P_2) = \frac{S_{permit} + S_{deny}}{Num(permit) + Num(deny)} = \frac{0.269 + 0.136}{2 + 2} = 0.101$$

Applying the same process, we can also calculate the similarity score between policies P_1 and P_3 : $S_{policy}(P_1, P_3) = 0.004$. The result meets our expectation expressed in Section 2.4.1: the two scores $S_{policy}(P_1, P_2)$ and $S_{policy}(P_1, P_3)$ shows clearly that policy P_1 is more similar to P_2 than P_3 in terms of the percentage of rule pairs having the same decision. In the next section, an exhaustive experiment will be conducted to prove the correctness of our algorithm.

3.3 Experiment Results

In order to verify if our algorithm is applicable to real cases, we compare the percentage of the same decision pairs with the PSM score. Firstly, we implement a random policy generator which takes rule elements as inputs then generates access control policies in Form 3.2. Secondly, we extract rule elements from four policies with different models and each of them is related to a real scenario: RBAC for project management [84], Net-RBAC for firewall configuration [85], OrBAC for hospital management [43], ABAC for administration of research laboratory [60]. Thirdly, these rule elements are input to the policy generator and each policy pair generated obtains a similarity score by our algorithm. Finally, we input various combinations of elements as access control requests into the four policies and count the percentage of the same decision pair between rules from the outputs. We mention that the

test method which we used is brute-force based: for categorical elements, we take all the combinations of *string* values; for numerical elements, enumerating all the numerical based attribute values in an interval (For example *Time* : [19 : 00, 21 : 00]) is impossible. Without loss of generality, we make equidistant sampling for bounded intervals and bilateral sampling for unbounded intervals. For example, inputs are all the integers from 1 to 24 for *Time* : [0 : 00, 24 : 00]; for *FileSize* : (10, + ∞) MB, inputs are 9 MB and 11 MB.

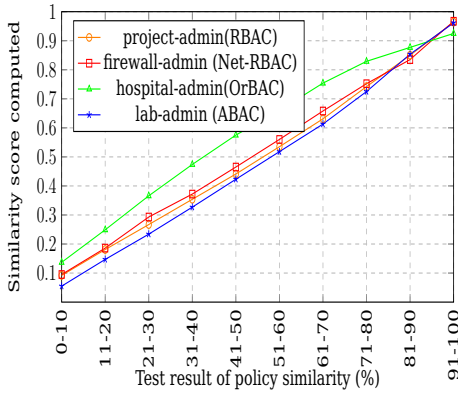


Figure 3.5: Experiment of similarity score (*set-4*)

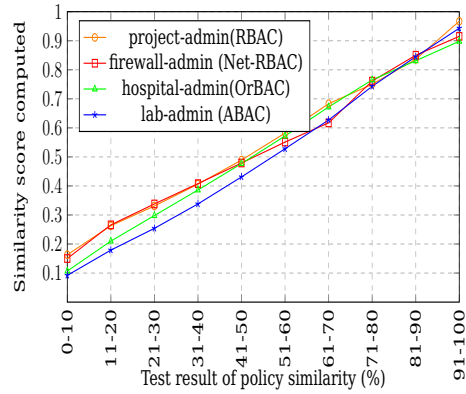


Figure 3.6: Experiment of similarity score (*set-8*)

Table 3.1: Policies tested

Policy	Model	Categorical Element	Numerical Element	Effect
project-admin	RBAC	15	0	permit
firewall-admin	Net-RBAC	4	28	permit
hospital-admin	OrBAC	15	6	permit,deny
lab-admin	ABAC	19	0	permit,deny

Figure 3.5 and Figure 3.6 show the policy similarity score (y-axis) and the same decision percentage for rule pairs (x-axis) in the *set-4* and *set-8* experiments. Each test set contains 1000 pairs of policies. In the *set-4* experiment, each policy has four rules and each policy has eight rules in the *set-8* experiment. The configurations of elements for each policy model are shown in Table 3.1. For example, laboratory administration policies are written in the

ABAC model and these policies contain 19 categorical elements with permit and deny effects. We observe that the score increases when the similarity between two policies increases. At the same time, the experimental values approach to the scores calculated and the quantity of test rules has no impact on the variation of the output curves. The test results enable us to conclude that the PSM score well approximates the similarity between policies.

3.4 Implementation

Our PSM algorithm can be applied to different SPs selection use cases such as network configuration, compute allocation and Cloud storage. This section presents a concrete scenario about Cloud storage.

3.4.1 Scenario Description

SUPERCLOUD [86] is a European project which aims at supporting user-centric deployments across multi-clouds and enabling the composition of innovative trustworthy services. SUPERCLOUD will build a security management architecture and infrastructure to fulfill the vision of user-centric secure and dependable Clouds of Clouds. One use case is to build a middleware layer between Cloud service customers (CSCs) and Cloud providers (CSPs). With this middleware, a CSC could select CSP(s) compliant with CSC's requirements. Here we implement a scenario of Cloud storage. The subjects involved in the scenario are a CSC, a Cloud broker and CSPs. A CSC wants to use the Cloud storage service(s) provided by one or multiple CSPs. At the same time, the CSC wishes that the security policies of CSP meet her requirements. Otherwise, she may launch a negotiation process with CSP(s) whose security policies are most approximate. To this end, the CSC chooses the SUPERCLOUD solution. It is worth noting that discovering CSP(s) whose security level is similar to the level of the CSC is just a pre-selection phase. Other criteria such as price and performance will be taken into consideration in the final negotiation and decision steps.

Figure 3.7 illustrates a scenario of our implementation. In the multi-cloud

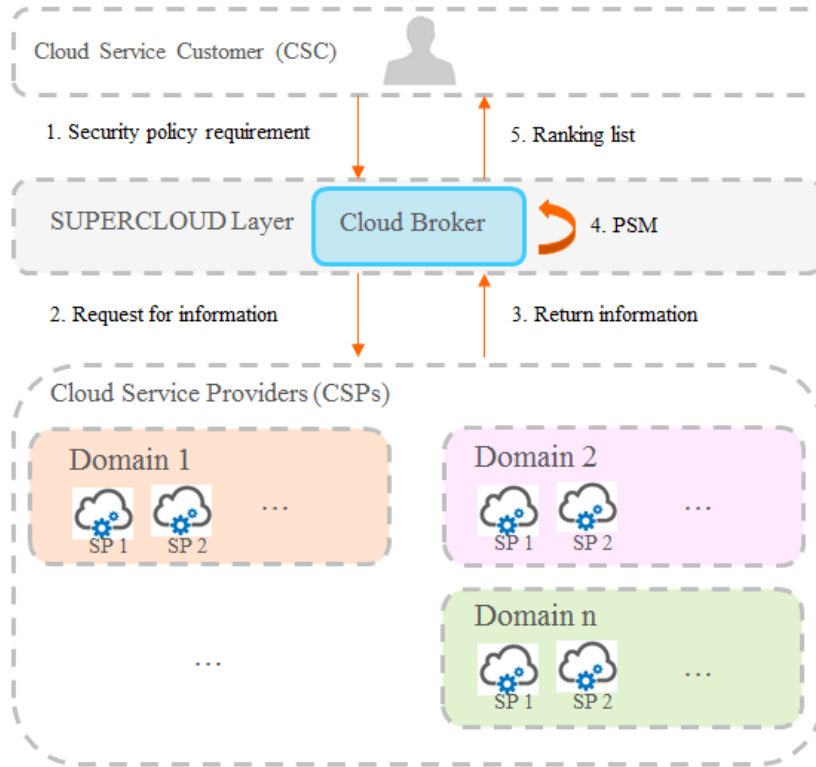


Figure 3.7: Service provider selection for Cloud storage

environments, CSPs are organized by federation and we suppose that CSPs in a Cloud federation share the same domain and two CSPs in the same domain can be composed as a virtual CSP. A virtual CSP provides its service as one CSP by combining the storage volumes of the two sub-CSPs and integrating their security policies. Firstly, a CSC expresses her requirement on the Cloud storage by security policies. For example, the CSC may wish that she could have a space of 100 GB and she is allowed to upload files between 8:00 and 22:00. Then the CSC sends her requirements to the SUPERCLOUD layer where a Cloud broker is deployed. The Cloud broker obtains the information and the security policy templates from the CSPs. Applying our PSM algorithm, the broker proposes a ranking list of the CSPs which meets the client's requirement regarding both the storage space and the security policies. PSM scores between CSC's and each CSP's policy are ranked from high to low. When one CSP's storage space is less than the volume required,

the broker may also propose a composition of two CSPs in the same domain. In this case, two CSPs' security policies should be combined and the policy after composition is also calculated by PSM and ranked. The composition operation depends on concrete use cases. Here we apply the *Conjunction* (&) operation proposed in [87] for our Cloud storage scenario. An example is as follows:

Policies before composition

$CSP_1 : 50\text{ GB}, \text{Permit}(\text{Action} : [\text{upload}, \text{download}], \text{Time} : [8 : 00, 23 : 00])$

$CSP_2 : 50\text{ GB}, \text{Permit}(\text{Action} : [\text{upload}, \text{download}, \text{delete}], \text{Time} : [7 : 00, 22 : 00])$

Policy after composition

$CSP_1 \& CSP_2 : 100\text{ GB}, \text{Permit}(\text{Action} : [\text{upload}, \text{download}], \text{Time} : [8 : 00, 22 : 00])$

Benefiting from the *Conjunction* operation, the storage space after composition is increased by combining the space from each CSP. At the same time, the security policy is stricter by eliminating the action which is not shared by the two sides.

3.4.2 Architecture

The implementation is based on the CloudSim [6] simulation framework. Developed by University of Melbourne, CloudSim is a Java-based toolkit that enables modeling and simulation of Cloud Computing systems and application provisioning environments. It supports both system and behavior modeling of Cloud system components such as data centers, VMs and resource provisioning policies. Since its development in 2009, CloudSim has been widely used in lots of scenarios such as VM allocation, network behavior, Cloud federation, dynamic workloads and power consumption. Figure 3.8 presents the architecture of the CloudSim toolkit. It consists of two layers: the CloudSim layer and the user code layer. The CloudSim layer supports the simulation of virtualized data center's environments which include dedicated

management interfaces for VMs, memory, storage, and bandwidth. The layer handles the fundamental issues such as provisioning of VMs, managing the application execution and monitoring the dynamic system state. At the top position, the user code layer exposes basic entities for hosts (number of machines, their specification, and so on), applications (number of tasks and their requirements), VMs, number of users and their application types, and broker scheduling policies. By extending the basic entities in this layer, developers can perform the following activities: (i) generate a mix of workload request distributions, application configurations; (ii) model Cloud availability scenarios and perform robust tests based on the custom configurations; (iii) implement custom application provisioning techniques for Clouds and their federations [6].

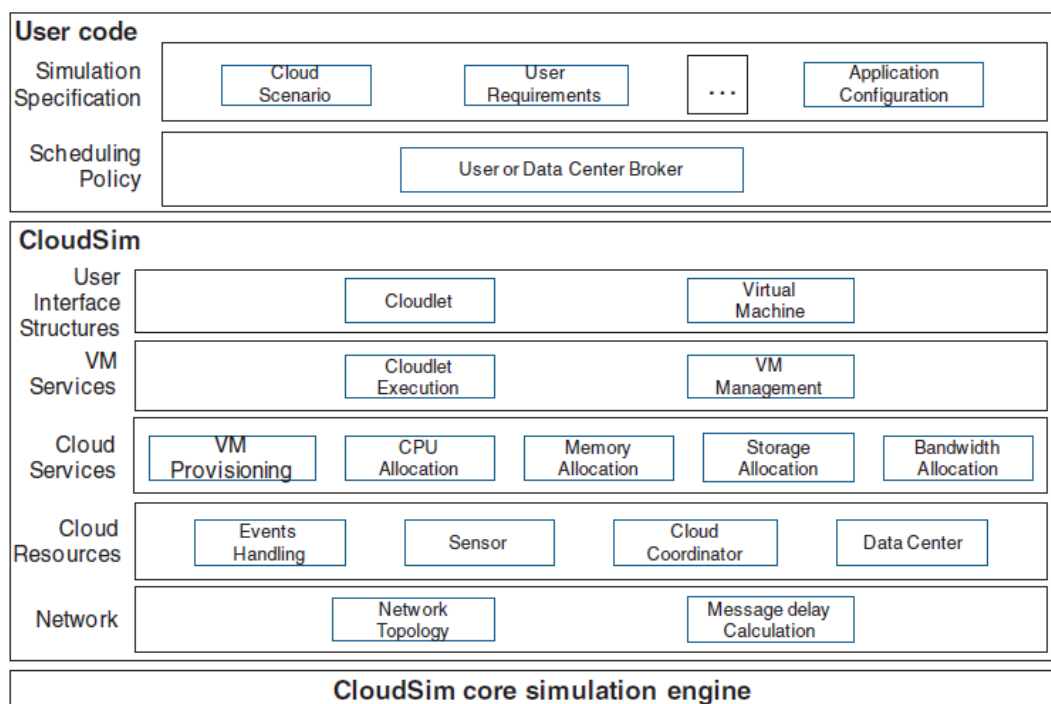


Figure 3.8: CloudSim architecture [6]

There exist mainly four components which relate to the implementation:

- **DataCenter Broker (DB):** it models a broker, which is responsible for mediating negotiations between a CSC and a CSP.

- **Policy Gateway (PG):** an additional policy-based component developed by us. The component delegates some policy related tasks from the DataCenterBroker.
- **Cloud Information Service (CIS):** an entity that registers, indexes and discovers the resources.
- **DataCenter (DC):** it models the core hardware infrastructure offered by a CSP. It encapsulates a set of compute hosts. Here we regard each DataCenter as a CSP.

The messages exchanged between different components are illustrated within the sequence diagram in Figure 3.9. In this sequence of execution, DCs are previously registered in the CIS (Step 1). The exchanged messages at step 2 and step 3 contain the security policy and the storage volume required by a CSC then the CIS returns all the registered DCs (Step 4). In step 5, the PG filters the DCs by storage volume. That is, the DCs whose storage volumes are more than the required volume are chosen. After that, at step 6, the PG makes combination of two DCs in the same domain among DCs which can't fit the volume requirement. The volume after combination is the sum of each volume and the combined policy is the conjunction of each policy. The combination of the DCs simulates the Cloud federation: two combined DCs can be seen as a virtual DC (VDC) and the VDCs which fit the volume requirement are found out. Then the similarity scores between CSC's and each (V)DCs' security policy are computed and ranked in step 7. Receiving the (V)DC list with similarity scores, the DB chooses the (V)DC with the highest score and deploys VMs on the target (V)DC (Steps 8-11).

3.4.3 Performance

The implementation is programmed in Java and it runs on an Intel machine having the configuration: 2.2 GHz with 4 GB of RAM running Windows 8 and JDK 8. We measure the execution time needed until the client receives a SP ranking list. Figure 3.10 shows the execution time with the increase

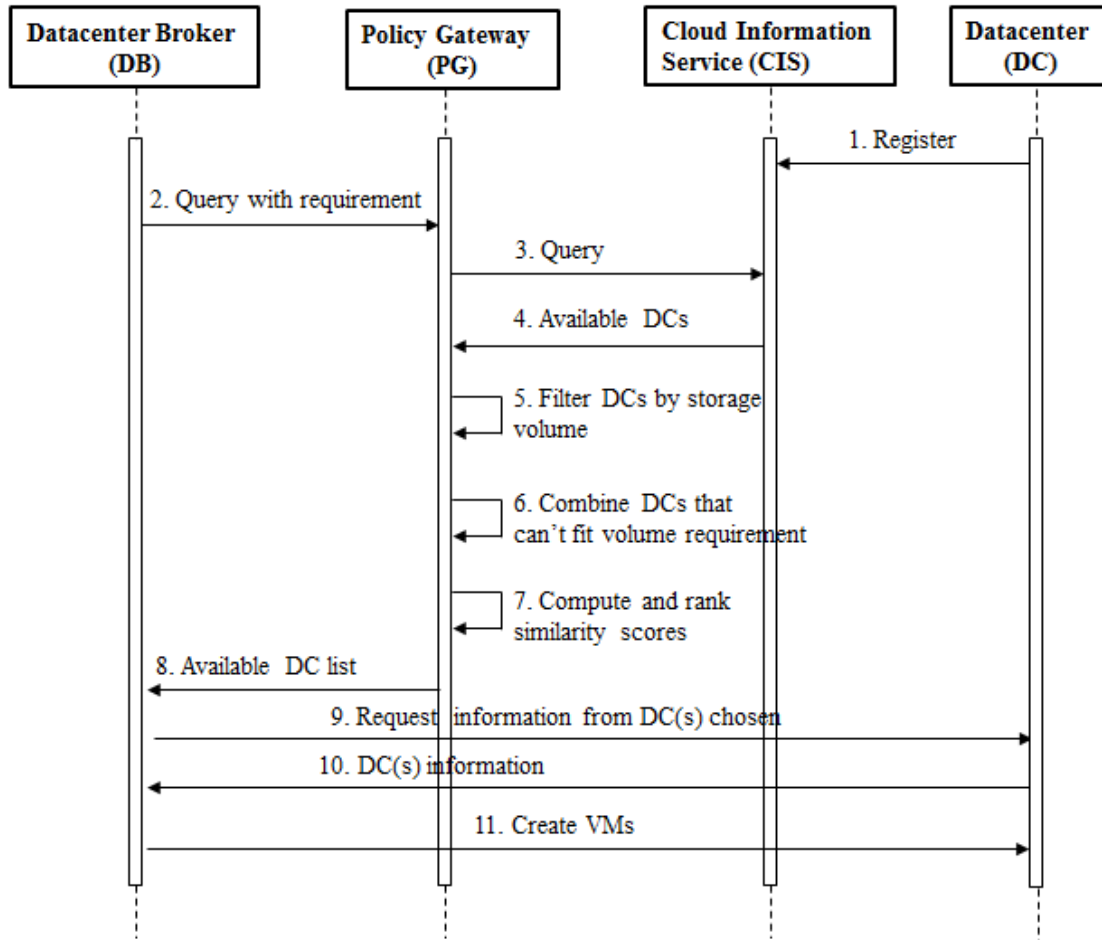


Figure 3.9: The sequence diagram of implementation

of SP number from 0 to 100 in each of the existing five domains. The blue line with triangles presents the execution time with the PSM and the red line with stars shows the execution time without the PSM. For a small scope of domain number, the execution time is not long (execution time < 1.2s). In Figure 3.11, the domain number varies from 5 to 30. The higher surface presents the execution time with the PSM and the lower surface shows the execution time without the PSM. The time increases with bigger scope of SP and domain number. However, on the one hand, the domain and SP number is limited in real case. On the other hand, the maximum value (30s when domain number=30, SP number=100) as the waiting time for a CSC before

selecting CSPs is also acceptable. Thus, from the two figures, we remark that the introduction of the PSM does not cause much of performance loss and it proves that our PSM algorithm is light-weight.

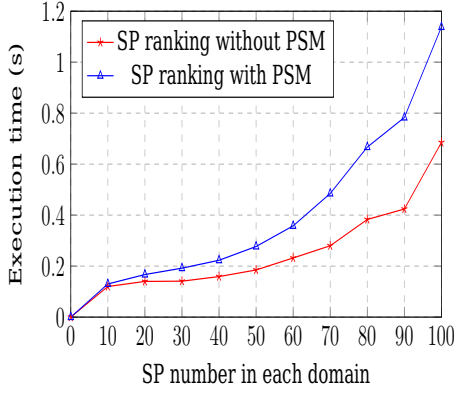


Figure 3.10: Execution time of SP ranking (domain number=5)

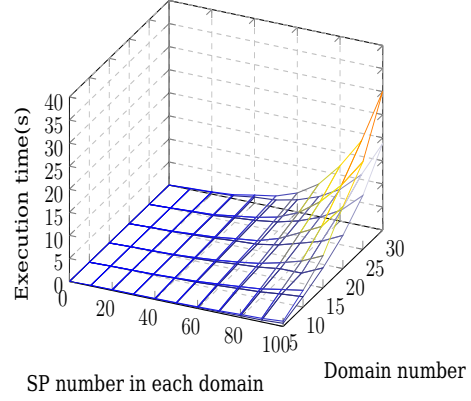


Figure 3.11: Execution time of SP ranking (domain number=5~30)

3.5 Conclusion

The main objective of this chapter is to expose our proposition to show how to measure the similarity between two security policies. The proposition gives mainly a generic and light-weight algorithm with which we can calculate a similarity score between two access control policies. After introducing the categorical measure and numerical measure, we tested our algorithm on four different security policy models in different scenarios and the output of our algorithm approximates to the test result. We demonstrated that our algorithm can be integrated in the SP selection process such as the SP(s) selection for Cloud storage. In the selection process, security policies belonging to different SPs are accessible to a SC so that the SC could make the evaluation and the comparison. At the same time, the implementation proved that this integration can enrich the services offered with efficiency. We decide to work on another use case where SP's security policies are not exposed directly. In this case, both SPs and SCs can express their security

requirements and those requirements could be automatically derived and transformed to concrete security policies. More detail will be found in the next chapter.

Chapter 4

Expression and Enforcement of Security Policy

4.1 Introduction

In this chapter, to overcome the aforementioned issues in section 2.5, we enhance the brokering technology by developing a configuration management process to allocate VMs in IaaS Clouds. Our method is evaluated by setting up a Cloud Computing environment to conduct the virtual resource allocation process. Experimental results show that our approach demands minimal user (CSC and CSP) intervention and enables unskilled Cloud users to have access to complex deployment scenarios. The remainder of this chapter is organized as follows. We first outline the expression of security policies by a CSCs and CSPs with an exhaustive example. Then, we illustrate the enforcement of the security policy for VM allocation. Finally, we describe an implementation integrated with our solution and evaluate four experiments.

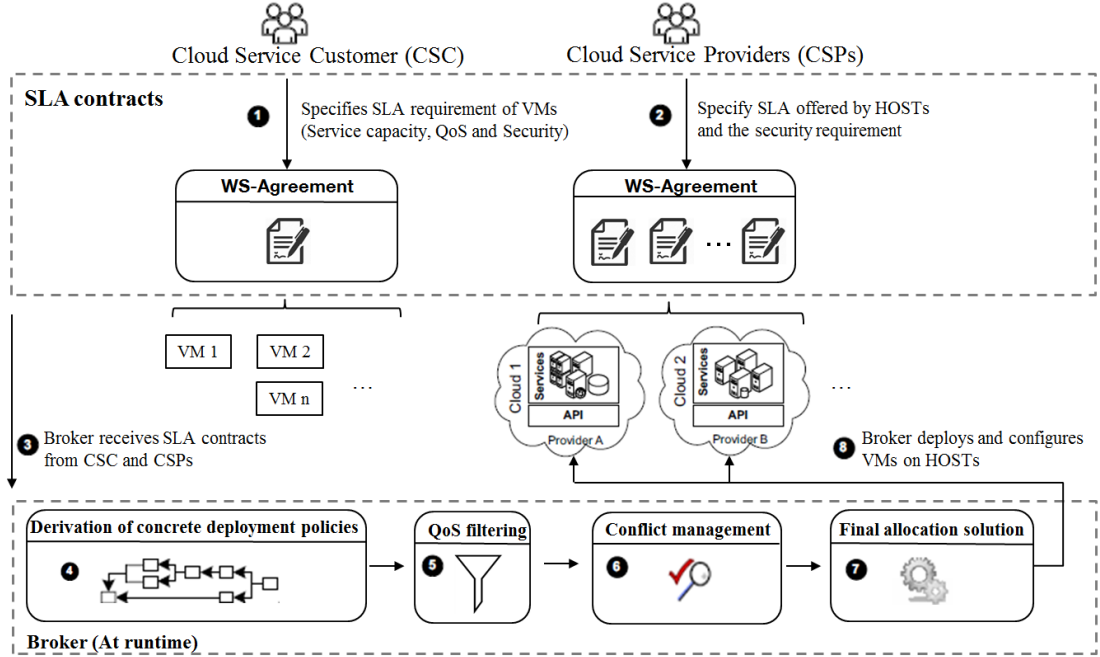


Figure 4.1: The proposed policy based framework to allocate virtual resources

4.2 Overview of the User-centric Policy-based Framework

Shown in Figure 4.1, with WS-Agreement [23] based contracts, both CSC¹ and CSP specify and manage their security requirements related to the infrastructure in order to ensure end-to-end security across different components (Steps 1,2). After receiving the SLA contracts, the broker derives the concrete deployment policies according to security and non-security requirements (Steps 3,4,5). Particularly, the broker is able to arbitrate contradicting demands and make decisions (Step 6). In the end, the broker applies an algorithm to generate the final allocation solution (Step 7) then deploys and configures VMs on HOSTs (Step 8).

¹In the chapter, CSC stands for the end customer of Clouds.

4.3 Expression of Security Policy

4.3.1 SLA Contract Expression

To generate security policies for CSCs and CSPs, we suggest, as a first step, to specify a generic document, which describes the requirements for service capacity, quality of service (QoS) and security constraints. The SLA contract is such a document used in service negotiation and management. Based on a well-formatted template, CSPs and CSCs exchange their proposals until reaching an agreement [22]. Among existing SLA specifications, we choose the WS-Agreement because the format is open so it can integrate various service parameters. Hence a WS-Agreement contract consists of name, context, service terms, guarantee terms and negotiation constraints, CSCs and CSPs can also integrate service capacity, QoS and security requirement in its structure.

As Definition 1, we use $H = \{h_1, h_2, \dots, h_m\}$ to represent a set of HOSTs and $V = \{v_1, v_2, \dots, v_n\}$ to denote a set of VMs. Note that, VM and HOST may have multiple attributes, each with their own values, and these attributes can be assigned either manually by a user or automatically by the system. In terms of security requirements, as CSCs and CSPs do not know the information of each other, they express their security constraints by using attribute-based expressions in Formulas 4.1, 4.2 and 4.3.

$$permission([H_{attr_name} : H_{attr_value}], [V_{attr_name} : V_{attr_value}]) \quad (4.1)$$

$$permission([H_{attr_name} : H_{attr_value}], [v_i]) \quad (4.2)$$

$$separation(v_i, v_j) \quad (4.3)$$

In the three formulas, H_{attr_name} and V_{attr_name} indicate the attribute name for HOST and VM respectively; H_{attr_value} and V_{attr_value} denote separately the attribute value for HOST and VM; each of v_i, v_j represents a unique virtual machine ID (VMID). Formulas 4.1 and 4.2 are used to specify the permission

for VM allocation: HOST(s) with attributes assigned is (are) permitted to deploy VM(s). The difference is that in the first formula, the CSC describes VM property by attribute and in the second formula, VMID is given directly. These two options give the CSCs more flexibility to express their security requirements. In addition, the CSC declares the coexistence constraint by Formula 4.3: v_i and v_j can not be allocated on the same HOST. Formula 4.4 is used by the CSP to express the deployment prohibition. Similar with Formula 4.2, HOST with HOSTID h_i is not permitted to deploy VM(s) assigned with attribute.

$$prohibition([h_i], [V_{attr_name} : V_{attr_value}]) \quad (4.4)$$

In an example that we will use throughout the chapter, we consider a DevOps [88] use case. DevOps is an emerged software development methodology that enhances collaboration between development, quality assurance (QA) and IT operations. Numerous companies are actively practicing DevOps since it aims at helping them to maximize the predictability, efficiency, security, and maintainability of operational processes. Adoption of DevOps is being driven by many factors including using public IaaS. Suppose that a software company has to deploy 3 VMs (v_1, v_2, v_3) in the cloud for a development project. Each VM contains its metadata such as properties, required volume, QoS specification and security constraints. We suppose that each VM runs a project server and there exist three types of VM: production (prod), development (dev), and test. *Prod* server runs live applications supporting the company's daily business and the data is public for e-business customers; *Dev* server consists of the development environment accessible only to developers having the specific access privilege; *Test* server is used to conduct software tests between development and production phase and it is accessible by testers with their private login accounts. At the same time, there exist 2 CSPs (h_1, h_2) and each has its own metadata such as price, location and state indicating if it is certified by security audit organizations. A readable illustration of the VM and HOST configuration is shown in Fig-

ure 4.2. In the scenario, each CSP has one security-related requirement: CSP1 does not want to deploy the VM which will be used for test; CSP2 does not welcome the server for development. At the same time, the software company has four security-related requirements:

- All the VMs should be deployed on certified HOST for the purpose of security.
- As most clients are from Europe, HOST which deploys the virtual machine v_2 should be in Europe in order to reduce the response delay.
- To better protect business assets, VM which is used to test should be deployed in Europe.
- Regarding the backup mechanism, the virtual machines v_1 and v_3 should not be co-located on the same HOST. In case of disaster of HOST, the project server can be quickly recovered from the other HOST.

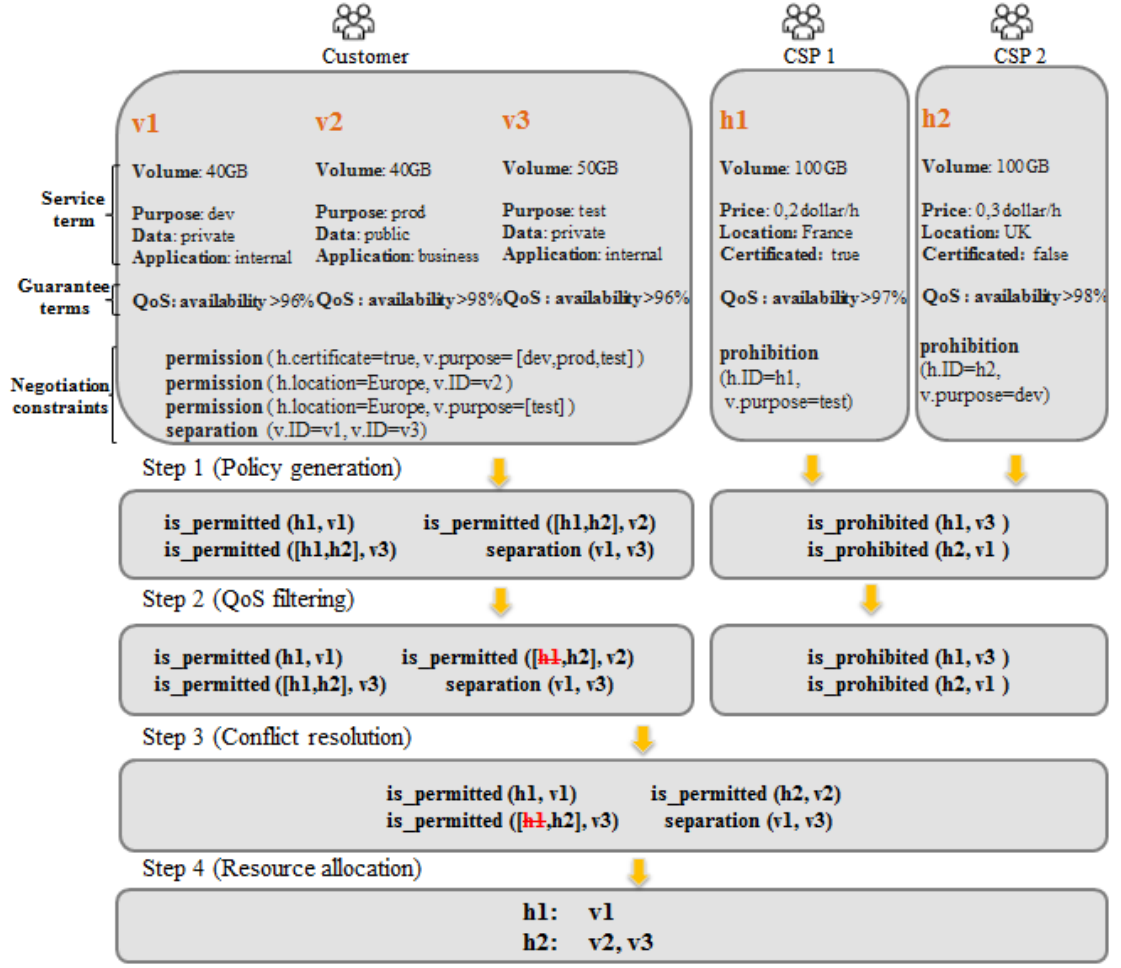


Figure 4.2: A DevOps use case of virtual resource allocation

4.3.2 Derivation of Security Policy

Security constraints need to be transformed to concrete security policies including VMID and HOSTID. Here we suggest using the OrBAC [89, 90] model which supports the expression of permission and prohibition. Derivation of the OrBAC policy from security constraints requires the policy mining technology which parses the configured rules and automatically reaches an instance of high level model corresponding to the deployed policy. Most of the existing RBAC based mining methods [85, 91] generate abstract policy by taking concrete rules as inputs. However, in our scenario, both abstract and concrete rules should be derived from the attribute-based description.

The following is the problem definition.

Definition 2. *Policy Mining Problem.*

Given a set of attributes of Subject S (HOST), a set of attributes of Action A , a set of attributes of Objects O (VM), and SAO_attr an attribute-based subject-action-object assignment relation (Formulas 4.1, 4.2, 4.4), find a set of ROLES, a subject-to-role assignment SR , a set of activity ACTIVITIES, an action-to-activity assignment AA , a set of VIEWS, an object-to-view assignment OV and $RAV \subseteq ROLES \times ACTIVITIES \times VIEWS$, a many-to-many mapping of role-to-activity-to-view assignment relation¹.

Definition 2 formalizes the policy mining problem by taking HOST related attributes and VM related attributes as the input and generating the OrBAC policy as the output. Algorithm 2 is the concrete realization and it explains the generation of the permission policy. First of all, after receiving contracts from a CSC and CSPs, the broker extracts the attribute information of each VM and HOST then generates three kinds of structures as the input: (1) VM list: storing all the attributes of the related VMs; (2) HOST list: storing all the attributes of the related HOSTs; (3) VM security constraint list: storing all the security constraints of the CSC. After initialization of policy p , the concrete action *deploy* is assigned to a new *activity* (lines 2,3). Then the relevant HOSTID list ID_h_list and relevant VMID list VM_v_list are generated from each term in the VM security constraint list c_v (line 4-6). For example, the relevant HOSTID and VMID for the security constraint $permission([“certificate” : “true”], [“purpose” : “dev”])$ are HOST1 and VM1. After finding the relevant VMID(s) and HOSTID(s), an abstract permission with a new role *currentRole* and new view *currentView* is created (line 7-9). Finally, all the HOSTIDs in ID_h_list are assigned to *currentRole* and all the VMIDs in VM_v_list are assigned to *currentView* (line 10-15). The prohibition policy for a CSP is generated in the same way by taking input of the VM list, the HOST list and the HOST security constraint list. Step 1 in Figure 4.2 demonstrates an example of permission and prohibition generation.

¹All the rules share the same action (“deploy”), organization (“superCloud”) and context (“default”). For reasons of simplicity, we do not illustrate organization and context in our algorithm and the derived policy.

Algorithm 2 *permissionGeneration*(l_v, l_h, c_v): permission policy generation

Input: VM list l_v , HOST list l_h , VM security constraint list c_v

Output: OrBAC policy p

```

1: Initiate  $p$ 
2:  $p.activity \leftarrow$  create new activity
3:  $p.consider("deploy", p.activity)$ 
4: for  $c_{vi}$  in  $c_v$  do
5:    $ID\_h\_list \leftarrow$  get relevant HOSTID(s) from  $l_h$ 
6:    $ID\_v\_list \leftarrow$  get relevant VMID(s) from  $l_v$ 
7:    $p.currentRole \leftarrow$  create new role for HOSTs in  $ID\_h\_list$ 
8:    $p.currentView \leftarrow$  create new view for VMs in  $ID\_v\_list$ 
9:    $p_i \leftarrow$  create permission:  $permission(p.currentRole, p.activity, p.currentView)$ 
10:  for  $ID_{hi}$  in  $ID\_h\_list$  do
11:     $p.empower(ID_{hi}, p.currentRole)$ 
12:  end for
13:  for  $ID_{vi}$  in  $ID\_v\_list$  do
14:     $p.use(ID_{vi}, p.currentView)$ 
15:  end for
16: end for
17: return  $p$ 

```

4.4 Enforcement of Security Policy

4.4.1 QoS Filtering

The process of policy generation in Algorithm 2 does not consider QoS constraints. In the next step, permissions which are not compliant with the QoS requirements should be eliminated during the policy enforcement phase. Shown in Step 2 of Figure 4.2, this process aims to disable the permission which does not satisfy the QoS constraints. To this end, an evaluation between the VM's performance requirements and the HOST's capacity will be conducted. For example, in our scenario, QoS requirements contain the term of availability and the deployment permission between VM2 and HOST1 is disabled.

4.4.2 Conflict Management

After generating OrBAC policies from security constraints and executing the QoS filtering, the broker aggregates permission rules for the CSC and prohibition rules for CSPs like:

$$is_permitted(\{h_i\}, v_k) \quad (4.5)$$

$$is_prohibited(h_j, \{v_l\}) \quad (4.6)$$

In Formula 4.5, each VM v_k has a set of hosts $\{h_i\}$ which allows it to be deployed and in Formula 4.6, a set of VM $\{v_l\}$ are not permitted to be deployed on HOST h_j . The rewriting of rules is used to detect conflicts between permissions and prohibitions. A conflict corresponds to the situation where a subject *HOST* is permitted and prohibited simultaneously to perform a given action *deploy* on a given object *VM*. We divide conflicts into the following two types and for each type an allocation solution is proposed.

Type I: conflict with concession space. Defined in Formula 4.7, HOST h_j is permitted and prohibited simultaneously to deploy VM v_k . In fact, except for h_j , VM v_k has other allocation solutions. In this case, we disable h_j from the allocation permissions of v_k (Formula 4.8). For example, in step 3 of Figure 4.2, $is_permitted(\{h_1, h_2\}, v_3)$ and $is_prohibited(h_1, v_3)$ belong to this type and the solution is disabling $is_permitted(h_1, v_3)$.

$$\begin{aligned} conflict_TypeI(h_j, v_k) \leftarrow & is_permitted(\{h_i\}, v_k) \wedge is_prohibited(h_j, \{v_l\}) \\ & \wedge h_j \in \{h_i\} \wedge v_k \in \{v_l\} \wedge (\{h_i\} \setminus h_j) \neq \phi \end{aligned} \quad (4.7)$$

$$disable(is_permitted(h_j, v_k)) \leftarrow conflict_TypeI(h_j, v_k) \quad (4.8)$$

Type II: conflict without concession space. Shown in Formula 4.9, compared with the conflict of type I, the difference is that in Type II, except for

h_i , VM v_k has no other deployment solution. In this case, we adopt a priority based approach proposed in [92] and introduce two labels $p(v)$ and $p(h)$ as priorities of VM and HOST. $p_1 \prec p_2$ means that p_2 has higher priority than p_1 . As the virtual resource allocation is related to different factors such as risk and trust, the priorities could be predefined by users or determined by the broker. For example, some of the CSPs' prohibitions can be disabled by the broker in case that the CSC has a lower risk score. Making decisions on the priority is beyond the scope of this chapter. A possible priority judgement method can be based on the maturity level which defines how well are the security issues treated within an organisation and evaluates the experience that the security administrators have [93]. Here we suppose that the CSPs obtain a higher priority to fulfill all their security requirements. Thus, in Formula 4.10, the current conflict resolution is disabling the permission of h_i . For example, the solution for the conflict between $is_permitted(h_3, v_1)$ and $is_prohibited(h_3, v_1)$ is disabling the former rule.

$$\begin{aligned} conflict_TypeII(h_i, v_k) \leftarrow is_permitted(h_i, v_k) \wedge is_prohibited(h_j, \{v_l\}) \\ \wedge (h_i = h_j) \wedge v_k \in \{v_l\} \end{aligned} \quad (4.9)$$

$$disable(is_permitted(h_i, v_k)) \leftarrow conflict_TypeII(h_i, v_k) \wedge p(v_k) \prec p(h_i) \quad (4.10)$$

4.4.3 Execution of Virtual Resource Allocation

The aim of the previous steps is to generate the final VM allocation solution. Without loss of generality, we demonstrate the generation of the allocation solution from a security policy by considering the CSC's preference on price. Algorithm 3 shows the resource allocation process. It takes permission policy p , VM list l_v , HOST list l_h and separation constraint c (Formula 4.3) as input and generates the deployment solution which maps VMs to HOSTs. In each permission rule, VMID and a list of its possible target HOSTs are extracted (line 1-4). To satisfy the price preference of the CSC, the target HOSTs are ranked from lower price to higher price (line 5) thus the one with

the lower price will be chosen preferentially. The final deployment solution depends on mainly two factors (line 9): (1) if the VM has a coexistence conflict with the VMs which have been already deployed on the HOST. (2) if the HOST has enough volume to deploy the VM. Step 4 in Figure 4.2 shows an example of the resource allocation.

Algorithm 3 *resourceAllocation*(p, l_v, l_h, c): virtual machine allocation

Input: OrBAC permission p , VM list l_v , HOST list l_h , separation constraint c

Output: deployment solution

```

1: for each concrete rule  $r_i$  in  $p$  do
2:   if  $r_i$  is active then
3:      $ID_{vi} \leftarrow$  get object in  $r_i$ 
4:      $ID\_h\_list \leftarrow$  get all the HOSTIDs permitted for  $ID_{vi}$  in  $r_i$ 
5:     Rank  $ID\_h\_list$  from lower price to higher price
6:     for  $ID_{hj}$  in  $ID\_h\_list$  do
7:        $v_i \leftarrow$  get VM from  $l_v$  by  $ID_{vi}$ 
8:        $h_j \leftarrow$  get HOST from  $l_h$  by  $ID_{hj}$ 
9:       if  $ID_{vi}$  not in separation constraint  $c$ 
         and  $h_j$  has enough volume for  $v_i$ 
         and  $v_i$  has not been allocated then
10:        add ( $v_i$  attaches host  $h_j$ ) to solution
11:       end if
12:     end for
13:   end if
14: end for
15: return solution

```

4.5 Implementation and Evaluation

In the SUPERCLOUD [86] project, one use case is to develop a middleware layer between CSCs and CSPs and this middleware could allocate virtual resources on physical infrastructures. In this context, there is a need to consider multi-cloud environments with security constraints. For example, virtual resources should not be mapped to physical resources that do not comply with their security requirements; physical resources should not deploy virtual resources that are potentially harmful to their operation; or virtual resources should not coexist on the same physical resource as another

potentially malicious virtual resource [94].

In order to implement and evaluate our virtual resource allocation framework, we setup an IaaS Cloud environment on a physical machine (Intel(R) Core(TM) i7-4600U 2.7 GHz with 16 GB of RAM running Windows 7). Then different VMs (2 cores and 2 GB of RAM) are created on a VirtualBox platform with a Ubuntu system. We now install a DevStack [95] based Cloud framework, a quick installation of OpenStack [96] ideal for experimentation. Each VM is regarded as a physical HOST for the purpose of experimentation. At the same time, a Java based program runs as the Cloud broker and connects to the VirtualBox platform by SSH protocol. The OrBAC policy is generated and managed by the Java-based OrBAC API [43]. Figure 4.3 illustrates our experimental architecture.

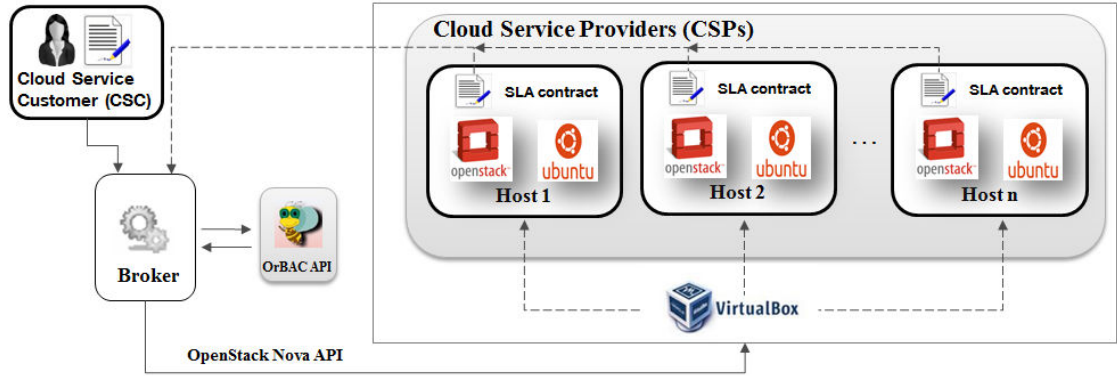


Figure 4.3: Implementation for virtual resource allocation

The scenario taken by the implementation is based on Figure 4.2. By using the MotOrBAC tool introduced in Section 2.2.5, we illustrate OrBAC policies generated in different steps in the scenario by Figure 4.4, Figure 4.5 and Figure 4.6. In the GUI interface of MotOrBAC, green rows and red rows represent separately permission rules and prohibition rules. The “preempted” status with an orange icon indicates that the rule is disabled. Figure 4.4 shows the rules after the WS-Agreement processing. Permission and prohibition rules are derived respectively from the security requirements of the CSC and the CSPs. Shown in Figure 4.5, after QoS filtering, the deployment permission concerning VM2 and HOST 1 is disabled. After Step 3, the con-

flict is resolved by disabling permission rules related to HOST1 and VM3 (Figure 4.6).

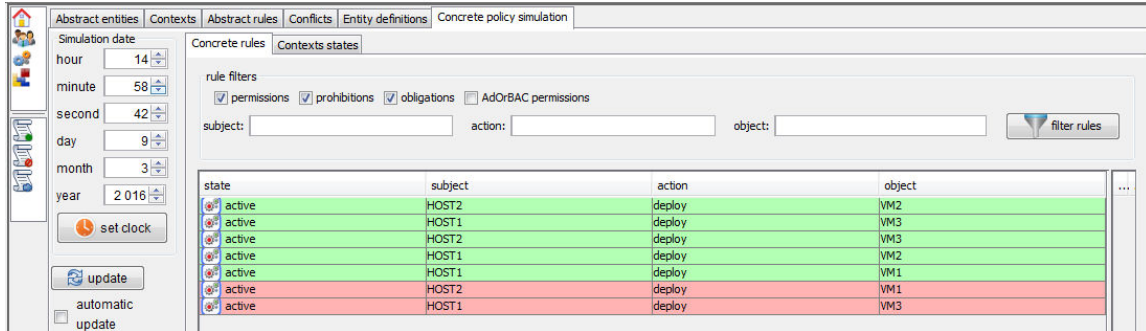


Figure 4.4: Policy generated after Step 1: SLA contract processing

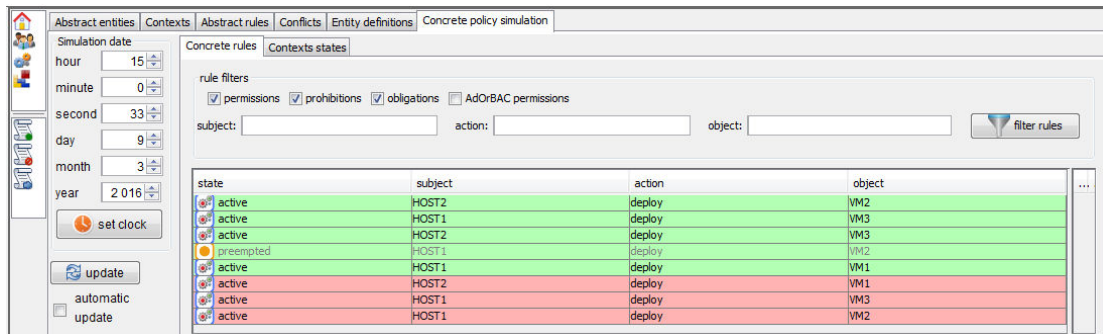


Figure 4.5: Policy generated after Step 2: QoS filtering

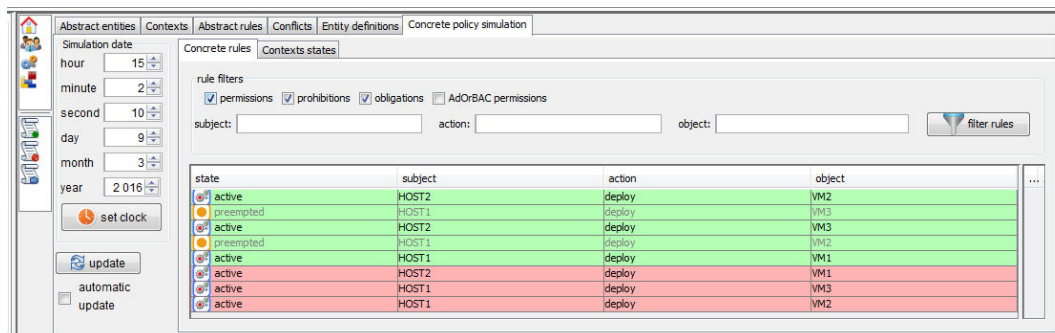


Figure 4.6: Policy generated after Step 3: Conflict resolution

The final resource allocation solution generated by Algorithm 3 after the conflict resolution is visualized in Figure 4.7. The graphs are generated by

using GraphStream [97], a Java library for the modeling and analysis of dynamic graphs. The window on the left shows the presence of VMs and HOSTs before allocation and the right one presents the final solution. Connecting by a black line, we can see that VM1 should be attached to HOST1; VM2 and VM3 are to be deployed on HOST2.

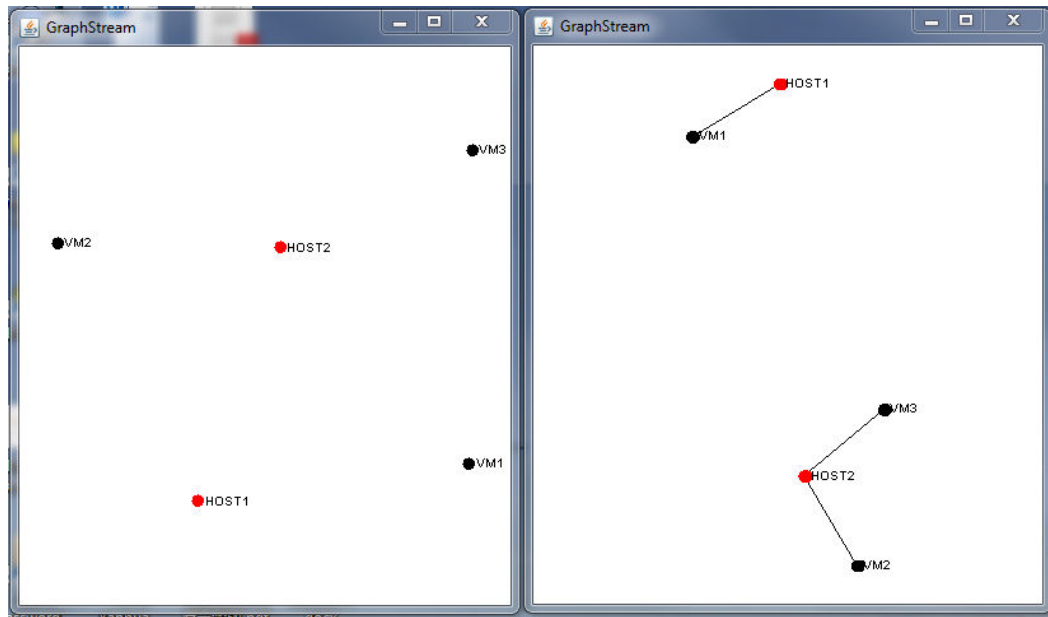


Figure 4.7: Final resource allocation solution graph

Once the final deployment solution is generated, the broker calls the Nova API by command-line [98] then creates the instances on the target HOSTs (Figure 4.8, Figure 4.9).



Figure 4.8: Deployment of VM on HOST1



Figure 4.9: Deployment of VMs on HOST2

4.5.1 Experiment 1: contract processing

This experiment measures the duration for contract processing which is the runtime required by the broker to process the JSON [99] based WS-Agreement file (see Appendix C) and generates VM and HOST lists. Since there does not exist a great difference between the SLA contracts of VM and HOST, here we measure the contract processing time for VMs. We vary the VM number from 0 to 125 and for each number we randomly generate service attributes in different quantities from 5 to 20. Figure 4.10 shows the result. For a small scope of VM and attribute number, the runtime is very low (30ms). The time increases with a bigger scope of VM and attribute number. The maximum duration of the experiment is less than 100ms which indicates that the runtime is acceptable.

4.5.2 Experiment 2: policy generation

In the second experiment, we analyze the required time for the OrBAC policy generation (Algorithm 2 for permission and similar algorithm for prohibition generation) once contracts are processed by the broker. In Figure 4.11, we study the amount of time the broker takes to generate security policies with an increasing number of VMs and HOSTs. For example, 60 as the value in the x-axis and y-axis indicates that there exist 60 VMs and HOSTs and the corresponding value in z-axis (400ms) shows the short time needed to generate the OrBAC policies.

4.5.3 Experiment 3: allocation latency

Our third experiment investigates the impact of VM number and HOST number on the execution time of Algorithm 3. In Figure 4.12, VM and HOST number vary from 10 to 60. Given 60 as VM and HOST number, the allocation latency takes about only 1 second. In the real case, as the HOST number is limited, the estimation of the allocation latency is acceptable and it confirms the efficiency of our resource allocation algorithm.

4.5.4 Experiment 4: price

The experiment measures the cost for a CSC after the VMs allocation. We generate the VMs randomly from 10 to 60 and configure 8 HOSTs. For simplicity, each HOST is supposed to provide only one type of IaaS solution with a fixed price from 0.02 dollars/hour to 0.08 dollars/hour¹. Then we compare the total price between two allocation solutions. The first solution is illustrated in Algorithm 3 which concerns CSC's price preference by ranking the HOSTs from lower price to higher price (Algorithm 3: line 4); The second solution is also based on the Algorithm 3 without considering the price preference, thus VMs are randomly allocated on HOSTs. As a result, Algorithm 3 shows a great advantage in reducing the deployment cost.

¹The prices are inspired from the current IaaS Cloud solution of Amazon EC2 and Microsoft Azure. For example, in Amazon EC2, the price for the instance of m4.xlarge (4 cores, 16G RAM) is 0.239\$/h and it costs 0.308\$/h (4 cores, 7G RAM) for the instance of A3 in Microsoft Azure.

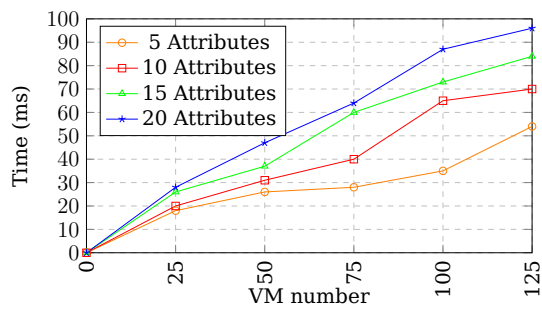


Figure 4.10: Time for contract processing

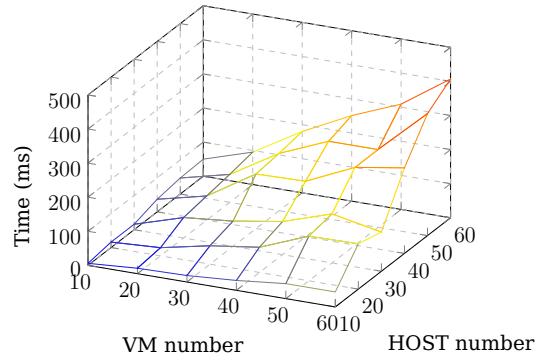


Figure 4.11: Time for policy generation

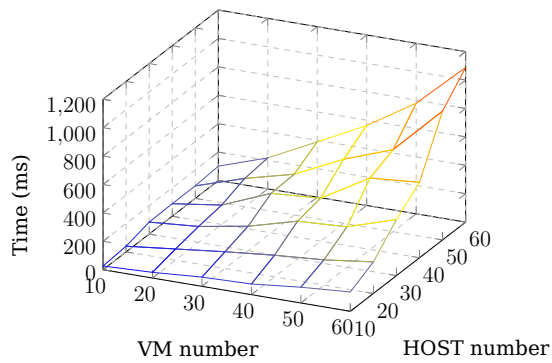


Figure 4.12: Latency for VM allocation

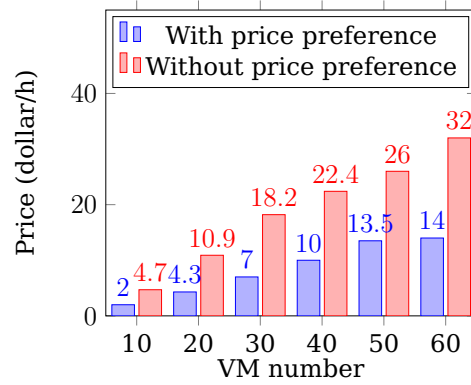


Figure 4.13: Total price for VM allocation

4.6 Conclusion

In this chapter, we have presented, formalized and enforced security requirements for virtual resource allocation. Our approach is to capture security and non-security requirements from both CSC and CSP, and apply a formal policy model to drive virtual resource allocation. We first presented the SLA contracts for CSCs and CSPs which contain service capacity, QoS and security constraints. We then transformed the attribute-based SLA contracts to concrete OrBAC policies. Finally, we allocated virtual resources after resolving conflicts in policies and demonstrated the efficiency and reliability of our solution by an OpenStack-based implementation. In particular, our solution tackles the lack of application of existing policy models that can support security-related expression when dealing with multiple Clouds.

Part II

Negotiation between Service Customer and Service Provider

Chapter 5

State of the Art

5.1 Introduction

We have seen in the first part of this dissertation that some security related issues must be treated during the SPs selection phase. In this second part, we will focus on the process of reaching agreement toward security related issues such as the usage control and service options. In fact, negotiators at this moment have already established their trust relationship and shared a common vocabulary. Since the usage control and service options are related directly to access control policies, the agreement on these policies should be reached and guaranteed from both sides. The following sections of this chapter outline the main preliminaries and related work needed to present our contribution. We begin with trust negotiation and the existing systems. Then, we present the notion of access negotiation and its related negotiation systems. After that, we outline the classification of access control policy negotiations and the development of negotiation paradigms. At the end, we introduce the meaning negotiation with a focus on the belief fusion paradigm.

5.2 Trust Negotiation

Over the past decade, trust negotiation (TN) as proposed by [100, 101] has been acknowledged as an effective mechanism for two entities to establish

a bilateral trust relationship by exchanging digital credentials. The established relationship helps SPs to make access decision about whether its sensitive resource can be accessed by an unknown service requester. As in Figure 5.1, a typical TN system contains four parts:

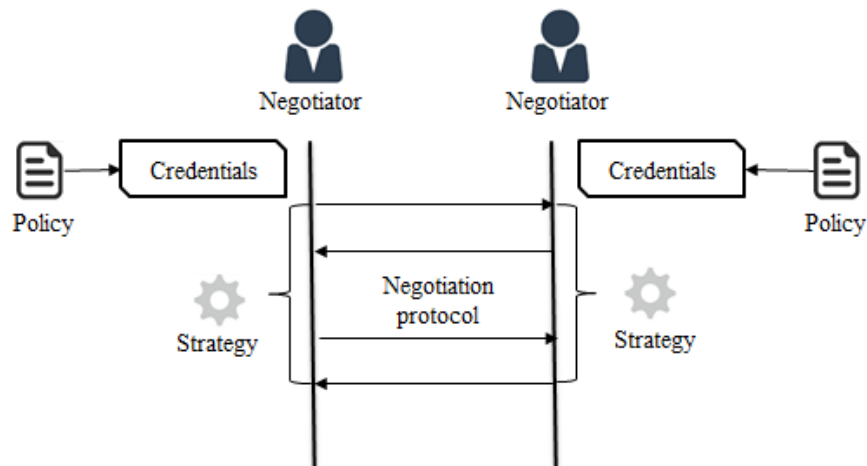


Figure 5.1: Typical TN system

- **Credential:** unlike the paper assertion we use in the real world (e.g. passport, driving licence, student card), it represents digital documents or messages that are certified (signed) by a credential issuer. Typically, a credential contains attribute information such as identity number, age or anything else owned by a person or an organization not directly related to the identity.
- **Policy:** a statement that specifies under which conditions an entity (human or artificial) can be trusted for a specific issue (e.g. resource action, task delegation).
- **Negotiation protocol:** defines rules managing the negotiation interactions. Applying the negotiation protocol, negotiators exchange their messages in an orderly way.
- **Strategy:** implemented by an algorithm, it determines how the local resource should be disclosed. There exist mainly two kinds of strategies: non-policy-exchange strategy and policy-exchange strategy [102].

The former strategy allows two participating entities to exchange as many credentials as possible. Sensitive credentials unlocked can be disclosed by the credentials sent from the counterpart [100]. In a policy-exchange strategy, entities disclose explicitly the policies protecting the relevant local sensitive credentials. Disclosure of local sensitive credentials is only available when the credentials sent from the counterpart fulfil the relevant local policies.

TrustBuilder [11] is the first implemented TN system which can be used in open distributed systems. Based on the ABAC [38, 39] model, the access control policies for resources are written as a declarative specification of the attributes needed in order to gain access to these resources. The system contains three modules: the credential verification module, the negotiation strategy module and the policy compliance checker. The core element of the architecture is the negotiation strategy module which enforces negotiation strategies to minimize credentials disclosure. Two different compliance checkers and two communication protocols have been implemented in TrustBuilder. Based on the TrustBuilder system, an extension called TrustBuilder2 [103, 104] is proposed. Compared with the previous TrustBuilder system, it adds many improvements: support for arbitrary policy languages, arbitrary credential formats, interchangeable negotiation strategies, flexible policy and credentials store.

Trust- \mathcal{X} [105, 106] has been developed as an XML-based framework for trust negotiation, specially conceived for peer-to-peer environment. In such environment, both the negotiating parties are equally responsible for negotiation management and can drive the negotiation process by selecting the appropriate strategy. The system implements an XML-based language, named \mathcal{X} -TNL, to specify certificates and policies. A novel aspect of \mathcal{X} -TNL is the support for trust ticket which is used to certify that the two parties have already successfully negotiated a resource so the subsequent negotiations can be simplified. Once TN is successful, each entity will generate an issued trust ticket and send it to the counterpart to avoid repeating authorization over a certain period. The negotiation process consists of four phases: the

introductory phase, the sequence generation phase, the certificate exchange phase and the caching of trust sequences phase. The main strategy used in Trust- \mathcal{X} consists in releasing policies to minimize the disclosure of credentials. As a result, only credentials necessary for the success of a negotiation are effectively disclosed [107].

PROTUNE [108] is a rule-based trust negotiation system. PROTUNE's language is based on normal logic program rules: $A \leftarrow L_1, \dots, L_n$ where A is the head of the rule and L_1, \dots, L_n is the body of the rule. In addition, the standard function-free logic programming language can be adopted as the internal format of the PROTUNE's language. PROTUNE rules are used to define access control and release policies. Before the negotiation, PROTUNE agents need to share a few built-in predicates and rule semantics. Policy authors are free to define and use high-level abstraction. During the negotiation process, agents exchange their requirements by disclosing selected parts of their policies in the form of logic programming rules. At the same time, according to the release policies, credentials are disclosed step by step. Different strategies can be adopted by negotiators. Current PROTUNE provides a cooperative strategy: at each step, all the releasable information which appears to be relevant to the success negotiation is disclosed. Another feature of PROTUNE is its facility in supporting the automated creation of high-quality documentation: "how-to", "why/why-not" and "what-if" queries can be answered by contextualized explanations.

5.3 Access Negotiation

So far, negotiation has been mainly used for trust establishment which can be served as a precondition for access. The next step may concern the concrete access permissions of service terms. In access negotiation, a requester negotiates the access attributes with resource holder and a successful negotiation generates an access authorization (Figure 5.2). There exist some related systems as follows.

XeNA [12] is an XACML-based negotiation system which brings trust ne-

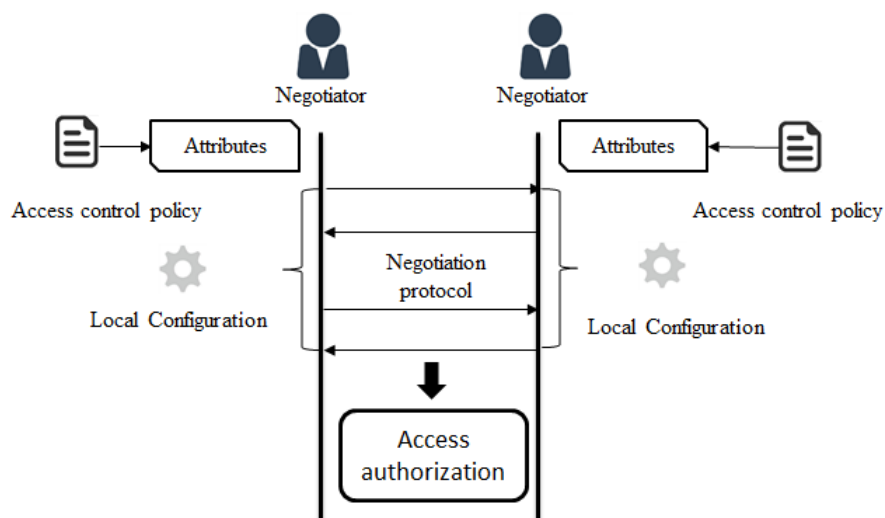


Figure 5.2: Typical access negotiation system

gotiation and access control management together within the same architecture. Extended from the TrustBuilder [11] system, the XeNA trust engine proposes a full support of XACML access control and negotiation policies. A main character is that the system uses a resource classification methodology which concerns three classes of resources: resources with a direct access, resources with a direct negotiated access and resources with an indirect negotiated access. Thus, for different types of resources, different negotiation processes take place before the access control management. In XeNA, the negotiation module is in charge of collecting resources required to establish a level of trust and to ensure a successful evaluation of access. The access control management is based on an extended RBAC profile of XACML [109]. This extended profile responds to advanced access control requirements and allows the expression of several access control models within XACML. In [110], authors propose putting security negotiation into practice by applying the XeNA framework to Intelligent Transportation Systems (ITS). For a vehicular communication system, negotiable resources can be security policies, digital credentials and privacy preferences. Those resources can be specified by OrBAC [19] permissions as a negotiation policy. After classifying different types of services, the vehicle service contract can be negotiated

by the XeNA framework.

WS-AC [111] is a fine grained access control system for web services. The system allows users to express, validate and enforce ABAC policies. Consists of service parameters, negotiation triggers and attribute conditions, an access control policy can be used to evaluate if an access request is granted, refused or negotiable. A request is compliant with a policy if all the conditions over the attributes specified in the policy are evaluated to true after comparing with the attributes taken by the request. An access request is rejected if it does not comply with any of the existing policies for the request service; an access will be granted if the parameters papering in the access request are all and only the parameters specified in the policy and their values are compatible with the values admitted by the policy; an access request is not fully acceptable by a policy and may be negotiated if (1) the access request and the policy are specified using a different set of parameters or (2) one or more clusters appearing in the policy do not have a corresponding tuple of parameter values in the access request. Meanwhile, if a negotiation trigger is defined in the policy, the negotiation is carried out in sending a negotiated access proposal (NAP) including a set of acceptable parameters. The negotiation process may take multiple rounds until (1) the request is acceptable thus the access is granted or (2) the request is not acceptable without other possible proposals then the access is denied. In addition, the system encapsulates WS-AC policies in WS-Policy [112].

5.4 Access Control Policy Negotiation

In trust negotiation and access negotiation, access control policies do not change. Nevertheless, they are changeable during access control policies negotiation in which different parties negotiate in order to commonly share resources (Figure 5.3). Gligor et al. [113] define the problem as the common access state negotiation. The common access state is the state where different parties achieve a common objective by sharing some resources. The shared resource means that access privilege of the owner is granted. There

exist three types of common state negotiation:

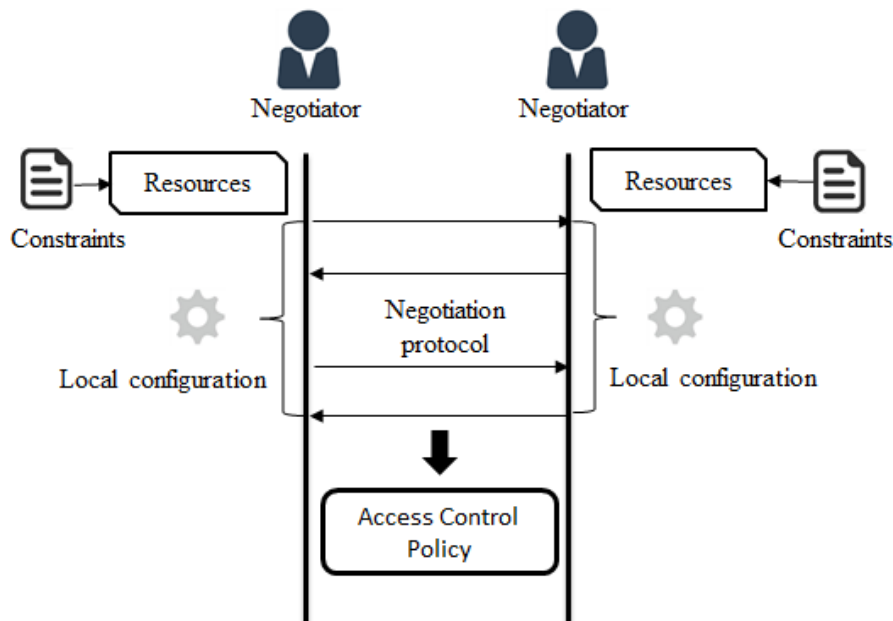


Figure 5.3: Typical AC policy negotiation system

- **Negotiation with no constraints:** all the negotiators share the common objectives. Typically, all the parties have a single common objective.
- **Negotiation with global constraint:** although all the negotiators have a complete knowledge of each other's objectives, some of their objectives may not coincide with each other's.
- **Negotiation with local constraints:** different from the second type, negotiators may not have a complete knowledge from each other.

Towards automated negotiation of access control policies, the work in [114] examines the problem of negotiating a shared access state, assuming all negotiators use the RBAC policy model. Based on a mathematical framework, negotiation is modelled as a Semiring-based Constraint Satisfaction Problem (SCSP) [115]. SCSP is an extension of the Constraint Satisfaction

Problem (CSP). A CSP problem consists of a set of variables, a domain of possible values for each variable, and a set of constraints specifying acceptable combinations of values for one or more variables. A solution for CSP is to find out an assignment of values to the variables that satisfy all the constraints of the problem. SCSP extends CSP by considering that constraints are not Boolean but belong to an appropriate semiring. With semiring-based constraint logic, the framework is expressive enough to represent a large class of policies such as RCL2000 [116] and RBAC [82]. Khurana et al. [117] then propose a negotiation agent which implements a round robin negotiation protocol: a coalition state will be reached if all other negotiators agree on it, otherwise the other negotiators make counter-proposals. The negotiation agent consists of a constraint compiler, a constraint evaluator and an optimizer. In the constraint compiler, the negotiator's access control constraints are expressed in the form of SCSP. In [118], authors argue that the guidance provided by constraints is not enough to bring practical solutions to automatic negotiation. Thus, they define an access control policy language which is based on Datalog¹ with constraints and the language can be used to define the formal semantics of XACML [1]. Then they use the language to specify the access control policies in real cases such as remote and hot grid service deployment: a SC deploys services on remote grid nodes after negotiating access control policies. A negotiation procedure and three types of meta-policies are designed for the creation of proposals, the conflict resolution and the policy validation during the negotiation. Meta-policies are used to select different combining algorithms and validate queries according to both side's requirements. Towards the need for human consent in organizational settings, Mehregan et al. [119] develop an extension of the Relationship-Based Access Control (ReBAC) model [120] to support multiple ownership, in which a policy negotiation protocol is in place for co-owners to come up with and give consent to an access control policy. Such multiple ownership is modelled by a social network graph in which vertexes represent users, edges represent interpersonal relationships and edge labels denote

¹Datalog is a declarative logic programming language and a subset of Prolog and it is often used as a query language for deductive databases.

the type of relationships that the edges signify (e.g., friend, parent, etc). The spirit of ReBAC is that the requested access graph shall satisfy some graph theoretic properties imposed by the access control policy. During negotiation, the draft policy is assessed by formally defined availability criteria: policy satisfiability, resiliency and feasibility.

5.5 Meaning Negotiation

Meaning negotiation (MN) is a negotiation process in which negotiators propose definitions and properties about a set of terms then accept or reject the definitions. MN has received significant attention in the Artificial Intelligence community. One of the paradigms to model MN is called belief fusion. Its scope is to construct a commonly accepted knowledge as the process of merging information from different sources. Belief is information held by human or artificial agents about the world that can be false, uncertain, have an elementary nature or involve a complex logical structure. Contrary to belief, knowledge is usually defined as an unquestionable piece of information about the world [121]. The basic problem of belief fusion is that how should an agent change her beliefs and how to bridge the gap of reaching consistency [122]. To this end, belief negotiation process is needed and Figure 5.4 illustrates a typical MN negotiation system. Booth et al. [123] propose dividing the negotiation process into stages. The first stage is weakening the individual pieces of information into a form in which they can be consistently added together; in the second stage, the information obtained is added together.

As MN may involve a different number of negotiators, various models in the Game Theory literature have been investigated, for example Bargaining [21], Pleading [124] and English Auction [125]. In the Bargaining Game, two agents discuss how to share one dollar. They make simultaneously proposals and send other proposals if their initial proposals are not compatible. The Pleadings Game is a normative formalization and computational model of civil pleading, founded in theory of legal argumentation and conflicts be-

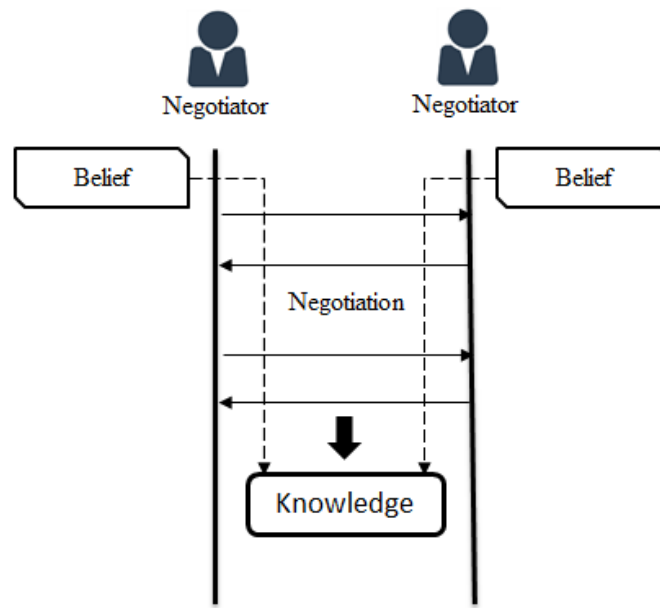


Figure 5.4: Typical MN system

tween arguments can be resolved by arguing about the validity and priority of rules. The English Auction model involves more than two negotiators. The auction begins by an initial proposal price. Then agents alternately make proposals in which the price is more than the current highest bid. Finally, the agent which proposes the highest bid wins the auction.

Burato et al. [20, 126] apply the Bargaining and the English Auction models to MN negotiation and propose a general negotiation framework. The agreement outcome is reached by testing the compatibility relation between the proposal by agents and the outcome can be an agreement or a disagreement. The framework contains mainly three components as follows:

- **Negotiation configuration:** based on the theory tree which presents agent's preferences about terms.
- **Proposal comparison system:** compares the proposal received with the current local one and concludes one of the four relations: equivalence, restriction, compatibility and inconsistency.
- **Negotiation algorithm:** specifies the sequence of negotiation mes-

sages including counter-offers and decisions. The counter-offer is made according to the relation concluded by the proposal comparison system. The algorithm can be used for two kinds of negotiation scenarios: Bargaining and English Auction.

The Bargaining negotiation consists of three phases:

1. **Initialization:** the system keeps the initial view-points of the two agents as their current local definition.
2. **Demand stage:** each agent sends its term proposition and receives the others'. Messages exchanged are evaluated by each side to find if an agreement is reached. In case of an agreement, the negotiation ends with a common term. Otherwise, the negotiation goes into the third stage.
3. **The war of attrition stage:** conducted by negotiation strategies, two agents exchange their offers by choosing nodes in their theory trees. The node can be reached by weakening, changing action or by renewing the previous proposal. This stage ends when an agreement is reached or each agent holds the node which is non-negotiable then the negotiation fails.

The English Auction negotiation involves n ($n \geq 3$) agents and the target is to obtain a viewpoint shared by α ($1 < \alpha \ll n$) agents. α is called the degree of sharing which is fixed by the auctioneer. The auctioneer is the first bidding agent which controls and decides the process development. In the negotiation process, only the auctioneer receives proposals from the other players after broadcasting its proposal. Negotiation between auctioneer and other agents is still conducted by the Bargaining negotiation mode. The auctioneer evaluates the counter-proposals by deciding if the α degree is satisfied. If it is, the agreement on a common term is reached, otherwise, the auctioneer will check if the negotiation should continue.

5.6 Conclusion

We have seen the essential preliminaries concerning the negotiation approaches with a focus on trust, access, access control (AC) policy and meaning negotiation. A comparison of different negotiation types is shown in Table 5.1. It is worth noting that “N/A” in the row of trust negotiation means that the knowledge of others’ constraints depends on the negotiation strategy. A negotiator knows the constraints of others’ credentials if the constraints are disclosed at the same time with credentials [100, 101]. However, none of the solutions presented for the AC policy negotiation provides a complete framework covering policy definition, negotiation configuration, proposal evaluation and negotiation protocol. Meanwhile, in most of the solutions, the so-called negotiation is just a one-round process without possibility to exchange messages in multi-rounds. To overcome these limitations, we will present, in the next chapter, the process of negotiation related to the access control policy and propose a framework and an algorithm based on MN negotiation.

Table 5.1: Comparison of different negotiation types

Negotiation types	Message	Objective	knowledge of other’s constraints
Trust negotiation	Credential, Policy	Trust relationship	N/A
Access negotiation	Credential, Policy	Access privilege	No
AC policy negotiation	Policy proposal	AC policy	No
Meaning negotiation	Term proposal	Term definition	No

Chapter 6

The Process of Reaching Agreement in Security Policy Negotiation

6.1 Introduction

In the part I, we presented approaches for SPs selection by considering the security policy and the security requirement. After selecting the SP(s) which meet SC's preferences, the SC and the SP may need to negotiate some more fine-grained security policies. In case that SC has no other SPs to choose or it has been already assigned to a SP, the SC may also need to reach an agreement in security policy. In this chapter, we propose a framework and an algorithm to negotiate a security policy such as an OrBAC policy. The negotiation mechanism is based on a policy evaluation approach. For that, we put forward the whole architecture that we consider to negotiate an OrBAC policy. Our whole framework is based on the bargaining model which manages indisputable and flexible preference. We advance an approach for comparison and evaluation of security policies: a negotiator makes a proposition and evaluates the opponent one. Dissimilar results of an evaluation lead to different reactions. The chapter is structured as follows. We firstly review the relation between policy entities. Based on the entity relationship,

we specify an approach to be used for rule comparison and evaluation. Then we propose the policy negotiation framework and give examples of its configuration. Finally we explain the negotiation algorithm and show a concrete scenario and the related prototype.

6.2 Relation between OrBAC Entities

In OrBAC, it is possible to consider the inheritance relation of roles and also of activities, views and organizations. We present the inheritance relation by using the predicates *sub_role*, *sub_activity*, *sub_view* and *sub_organization* introduced in [83]. Besides, we also define the *sub_context* predicates for the context entity. The five predicates belong to two types of relations:

- **Hierarchy relation:** *sub_role* presents the hierarchy relation. For example, *sub_role(org, r₁, r₂)* indicates that in organization *org*, role *r₁* is a *sub* role of *r₂*. Suppose that in a company, *r₁* is a staff member and she guides a trainee *r₂*. Then *r₁* could inherit all the permissions of *r₂*.
- **Specialization relation:** *sub_activity*, *sub_view* and *sub_context* belong to the specialization relation. For example, concerning the document management, activity *manage* may be specialized into three activities: *create*, *consult* and *update*. Thus, *update* is a *sub_activity* of *manage*. For instance, in a hospital, a physician is permitted to manage medical records of her patients and we can derive that she is also permitted to update the medical records. Similarly, a *medical_record* can be a *sub_view* of *hospital_file* and *context₁ = certificate* can be a *sub_context* of *context₂ = (certificate ∧ IDCard)*.

It is possible that predicate *sub_organization* belongs to either hierarchy or specialization relations. For example, *departement1* may be hierarchically higher than *departement2*. Thus, *role* in *departement1* can inherit all the privileges in *departement2*. Another example regarding the specialization relation is that *market_departement* can be one of the *sub_organization* of the *business_departement*, thus *role* in *market_departement* can inherit all the

privileges in *business_departement*. An exhaustive relationship for each OrBAC entity is shown in TABLE 6.1¹.

Table 6.1: Relationship between OrBAC entities

Predicate	Definition
$\text{sub_role}(\text{org}, r_1, r_2)$	$\forall \text{org}, \forall r_1, \forall r_2, \forall a, \forall v, \forall c, \text{Permission}(\text{org}, r_2, a, v, c) \wedge \text{sub_role}(\text{org}, r_1, r_2) \rightarrow \text{Permission}(\text{org}, r_1, a, v, c)$
$\text{sub_activity}(\text{org}, a_1, a_2)$	$\forall \text{org}, \forall r, \forall a_1, \forall a_2, \forall v, \forall c, \text{Permission}(\text{org}, r, a_2, v, c) \wedge \text{sub_activity}(\text{org}, a_1, a_2) \rightarrow \text{Permission}(\text{org}, r, a_1, v, c)$
$\text{sub_view}(\text{org}, v_1, v_2)$	$\forall \text{org}, \forall r, \forall a, \forall v_1, \forall v_2, \forall c, \text{Permission}(\text{org}, r, a, v_2, c) \wedge \text{sub_view}(\text{org}, v_1, v_2) \rightarrow \text{Permission}(\text{org}, r, a, v_1, c)$
$\text{sub_context}(\text{org}, c_1, c_2)$	$\forall \text{org}, \forall r, \forall a, \forall v, \forall c_1, \forall c_2, \text{Permission}(\text{org}, r, a, v, c_2) \wedge \text{sub_context}(\text{org}, c_1, c_2) \rightarrow \text{Permission}(\text{org}, r, a, v, c_1)$
$\text{sub_organization}(\text{org}_1, \text{org}_2)$	$\forall \text{org}_1, \forall \text{org}_2, \forall r, \forall a, \forall v, \forall c, \text{Permission}(\text{org}_2, r, a, v, c) \wedge \text{sub_organization}(\text{org}_1, \text{org}_2) \wedge \text{relevant_role}(\text{org}_1, r) \wedge \text{relevant_activity}(\text{org}_1, a) \wedge \text{relevant_view}(\text{org}_1, v) \rightarrow \text{Permission}(\text{org}_1, r, a, v, c)$

We denote *sub* relation by the symbol: " $<$ ". $e_1 < e_2$ indicates that e_1 is a *sub* entity of e_2 . Based on the *sub* relation, we propose three other relations: equivalent, relevant and inconsistent. Along with the *sub* relation, the four relationships will be used to define relations between OrBAC rules in the next section. Let e_1 and e_2 be two related entities² of OrBAC policy, the other three relations between e_1 and e_2 are:

Equivalent: if e_1 is semantically equal to e_2 , they have the equivalent relation denoted with $e_1 = e_2$;

Relevant: $e_1 < e_2$ or $e_2 < e_1$ or $e_1 = e_2$, in this case we say that e_1 and e_2 have a relevant relation and denote it with $e_1 \sim e_2$. We note that both the equivalent relation and the *sub* relation are the subcases of relevant relation;

¹ $\text{relevant_a}(\text{org}, c)$ means that entity c which belongs to abstract entity a is defined in organization org .

²We remark that two entities are related if they belong to the same type of abstract entity. For example, *teacher* and *administrator* are related entities because they belong to the same abstract entity: *role*.

Inconsistent: if e_1 has not a relevant relation with e_2 , the two entities have an inconsistent relation and we denote this with $e_1 \sim e_2$.

6.3 Relation between OrBAC Rules

In the negotiation process, the comparison and the evaluation of a received proposition with the current local offer are necessary for a negotiator to make its decision. In [127], Coma et al. define four relation patterns between the contract grantor and the contract grantee in a contract compatibility session. Those relations are used to generate an interoperability contract which contains a set of policies. Based on the entity relations we previously defined, we apply the relation patterns to the comparison between two OrBAC permission rules. Let r_i and r_j be two OrBAC rules, e_{ik} , e_{jk} rule entities belong to r_i and r_j respectively, five relations between rules are:

Restriction: $r_i \prec r_j \Leftarrow \forall e_{ik} \forall e_{jk} (((e_{ik} < e_{jk}) \vee (e_{ik} = e_{jk})) \wedge \exists e_{ik} \exists e_{jk} (e_{ik} < e_{jk}))$. If at least one entity of r_i is a *sub* entity of r_j and other related entities have an equivalent relation, then r_i is a restriction of r_j denoted with $r_i \prec r_j$. At the same time, we say that r_j is a generalization of r_i . Example:

$r_i \prec r_j \Leftarrow \text{staff} < \text{trainee}, r_i : \text{permission}(\text{company_A}, \text{staff}, \text{read}, \text{document}, \text{default}),$
 $r_j : \text{permission}(\text{company_A}, \text{trainee}, \text{read}, \text{document}, \text{default})$

Total compatibility ($T_compatibility$): $r_i = r_j \Leftarrow \forall e_{ik} \forall e_{jk} (e_{ik} = e_{jk})$. If all the related entities in r_i and r_j are equivalent, then r_i and r_j have a total compatibility relation denoted with $r_i = r_j$.

Symmetric compatibility ($S_compatibility$): $r_i \simeq r_j \Leftarrow \forall e_{ik} \forall e_{jk} ((e_{ik} \sim e_{jk}) \wedge \exists e_{ik} \exists e_{jk} (e_{ik} \prec e_{jk}) \wedge \exists e_{ik} \exists e_{jk} (e_{jk} \prec e_{ik}))$. All the related entities are relevant, at least one entity in r_i is a *sub* entity of r_j and at least one entity in r_j is a *sub* entity of r_i . This relation is denoted with $r_i \simeq r_j$. Example:

$r_i \simeq r_j \Leftarrow \text{staff} < \text{trainee}, \text{update} < \text{read}, r_i : \text{permission}(\text{company_A}, \text{staff}, \text{read}, \text{document}, \text{default}), r_j : \text{permission}(\text{company_A}, \text{trainee}, \text{update}, \text{document}, \text{default})$

Partial compatibility ($P_compatibility$): $r_i \bowtie r_j \Leftarrow \forall e_{ik} \forall e_{jk} (\exists e_{ik} \exists e_{jk} (e_{ik} \sim e_{jk}) \wedge \exists e_{ik} \exists e_{jk} (e_{ik} \approx e_{jk}))$. At least one pair of related entities is relevant and there exists at least one pair of entities having an inconsistent relation. In this case, the rules are partially compatible but not comparable. We denote it with $r_i \bowtie r_j$. Example:

$r_i \bowtie r_j \Leftarrow \text{financial_file} \approx \text{technical_file}, r_i : \text{permission}(\text{company_A}, \text{staff}, \text{read}, \text{financial_file}, \text{default}), r_j : \text{permission}(\text{company_A}, \text{staff}, \text{read}, \text{technical_file}, \text{default})$

No compatibility ($No_compatibility$): $r_i \nmid r_j \Leftarrow \forall e_{ik} \forall e_{jk} (e_{ik} \approx e_{jk})$. If all the related entities have inconsistent relations, the two rules are not comparable and they have a no compatibility relation denoted with $r_i \nmid r_j$.

6.4 Negotiation Configuration

6.4.1 Entity Chain

Before negotiation, two participants should have already shared their vocabulary and held their entity chains which register the *sub* relation between entities. Five types of entities: organization, role, activity, view and context, may possess their entity chains.

Definition 3. Entity Chain

Given different entities e_i, e_j ($i, j = [0, n]$) which belong to the same type of entity (organization, role, activity, view, context) and $e_i < e_j$ ($i < j$), an entity chain is a chain $EntityChain = \langle V, E \rangle$ (V denotes a vertex and E denotes an edge) where:

- i) e_0 is the head;
- ii) $V \subseteq \{e_i\}$;

iii) $E \subseteq \{(e_i, e_j)\}$, where both e_i and e_j are in V and $e_i < e_j$.

Example 1. An entity chain of *Role* entity is shown in Fig. 6.1. In a company, an *employee* has more privileges than a *trainee* but less authority than a *boss*. Hence *employee* is a *sub role* of *trainee* and *boss* is a *sub role* of *employee*.



Figure 6.1: An entity chain example for *Role*

6.4.2 Policy Tree

Before negotiation, different participants should have their own configuration indicating their preferences. In [20], the negotiation configuration is based on the theory tree which presents the preference on term definition. However, a theory tree contains only comparable terms but not non-comparable ones. Two comparable terms share some common elements and one term is a restriction of the other. For example, terms $T_1 = p$ and $T_2 = p \vee q$ are two comparable terms and T_1 is a restriction of T_2 . Conversely, $T_3 = s$ and $T_4 = h \wedge f$ are two non-comparable terms. Indeed, in some negotiation cases, the presentation of preferences should not only focus on comparable terms but also on non-comparable terms. In our model, we distinguish policy tree into the related policy tree for comparable policies and the distant policy tree for non-comparable policies.

Definition 4. Related Policy Tree

For rule r_0 which holds some option rules r_i, r_j ($i, j = [1, n], i < j$), its related policy tree is a finite graph $PolicyTree = \langle V, E \rangle$ (V denotes a vertex and E denotes an edge) where:

- i) The initial rule r_0 is the root which is the most preferred ;
- ii) $V \subseteq \{r_i\}$ where for every r_i , $r_0 \prec r_i$ or $r_0 \simeq r_i$;
- iii) $E \subseteq \{(r_i, r_j)\}$, where both r_i and r_j are in V and $r_i \prec r_j$ or $r_i \simeq r_j$;

- iv) all the leaves are stubborn rules which are unquestionable and the least preferred.

By definition, the configuration is characterized by the degree of preference. We assume that each related policy tree has at least an initiate rule and a stubborn rule. An exceptional case is that the negotiation configuration for a related policy tree has only a head rule which indicates that this rule is absolutely stubborn and it can not give up itself by redirecting to another proposition.

Example 2. A possible configuration for a related tree of an OrBAC rule r_0 is shown in Fig. 6.2. r_{11} has $S_compatibility$ relation with r_0 which indicates that if r_0 can not be agreed by the opponent, r_{11} could be proposed by weakening some entities and strengthening other ones. r_{12} and r_{21} are two stubborn rules. r_{21} is a generalization of r_{11} ; r_{12} has $S_compatibility$ relation with r_0 . In the example, the tree has two directions: $\{r_0, r_{11}, r_{21}\}$ and $\{r_0, r_{12}\}$. Each direction is designed with the aim of weakening one or some entities.

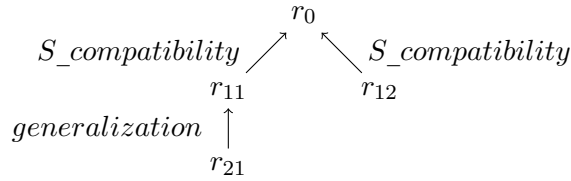


Figure 6.2: Example of a related policy tree

Definition 5. *Distant Policy Tree*

A distant policy tree of r_0 which contains rules r_i, r_j ($i, j = [0, n], i < j$) is a finite graph $PolicyTree = \langle V, E \rangle$ (V denotes a vertex and E denotes an edge) where:

- i) r_0 is the root which prefers to be replaced by other rules;
- ii) $V \subseteq \{r_i\}$ where for every r_i , $r_i \bowtie r_0$ or $r_i \dashv\vdash r_0$;
- iii) $E \subseteq \{(r_i, r_j)\}$, where both r_i and r_j are in V and $r_i \bowtie r_j$ or $r_i \dashv\vdash r_j$.

A distant policy tree could be treated as a trigger because rules of lower depth could be proposed as counter-offers of the root rule. The condition to activate or deactivate a distant tree depends on the negotiation strategy which gives more flexibility to the negotiation configuration. The distant policy tree will be useful when multiple rules are negotiated at the same time since a counter-offer may be comparable with other proposed rules. An example of this case is shown in Section 6.6.2. In case that only one rule is negotiated in a policy, the distant policy tree does not need to be configured.

Example 3. The distant tree of r_0 is presented in Figure 6.3. r_0 and r_1 have *No_compatibility* relation, r_0 and r_2 have *P_compatibility* relation. In the example, r_1 and r_2 are more preferred than r_0 and it indicates that if r_0 is proposed, r_1 or r_2 could be proposed as counter-offers.

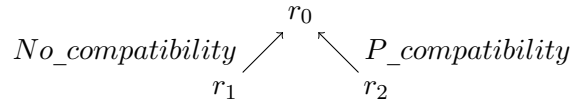


Figure 6.3: Example of a distant policy tree

6.5 Negotiation Algorithm

6.5.1 Algorithm Description

This section describes the core negotiation algorithm for our framework. We suppose that negotiation participants prefer to reach an agreement which is in their configuration space. Besides, in order to ensure the security level during the transaction, negotiators do not usually refuse a restriction of its current rule. We show how a service provider and a requester negotiate an OrBAC policy contract. Compatible with the WS-Agreement negotiation protocol [4], our algorithm contains five actions: propose, receive, create agreement, agreement, refuse and also four stages: information stage, demand stage, bargaining stage, contract establishment stage.

Algorithm 4 Negotiate Security Policy

```

1:  $i, j \leftarrow 1$ ;
2:  $r_j, r_{loc}^{cur}, r_{rec}^{cur} \leftarrow null$ ;
3:  $r_{loc}^{stub}, r_{rec}^{stub} \leftarrow \mathbf{false}$ ;
4: Establishes ontological mapping with opponent;
5:  $r_i \leftarrow$  initial proposal;
6:  $r_{loc}^{cur} \leftarrow r_i$ ;
7: Send  $r_i$  to opponent;
8:  $r_{rec}^{cur} \leftarrow receive(r_j)$ ;
9: if ( $r_{rec}^{cur} = r_{loc}^{cur}$ ) or ( $r_{rec}^{cur} \prec r_{loc}^{cur}$ ) then
10:   return CreateAgreement;
11: else
12:   while not ( $(r_{loc}^{stub} = \mathbf{true})$  and ( $r_{rec}^{stub} = \mathbf{true}$ )) do
13:     if opponent is in stubbornness set then
14:        $r_{rec}^{stub} \leftarrow \mathbf{true}$ ;
15:     end if
16:     if  $r_{rec}^{cur} = createAgreement$  then
17:       Go to contract establishment stage
18:     else if condition to trigger distant tree by  $r_{rec}^{cur}$  is satisfied then
19:        $searchDistantTree(i, r_i, r_{loc}^{cur}, r_{rec}^{cur})$ ;
20:     else if ( $r_{rec}^{cur} = r_{loc}^{cur}$ ) or ( $r_{rec}^{cur} \prec r_{loc}^{cur}$ ) then
21:       return CreateAgreement;
22:     else if ( $r_{loc}^{cur} \simeq r_{rec}^{cur}$ ) or ( $r_{loc}^{cur} \prec r_{rec}^{cur}$ ) then
23:        $searchRelatedTree(i, r_i, r_{loc}^{cur}, r_{rec}^{cur})$ ;
24:     else
25:       return Refuse;
26:     end if
27:      $j \leftarrow j + 1$ ;
28:      $r_{rec}^{cur} \leftarrow receive(r_j)$ ;
29:   end while
30: end if
31: if ( $r_{rec}^{cur} = r_{loc}^{cur}$ ) or ( $r_{rec}^{cur} \prec r_{loc}^{cur}$ ) then
32:   return CreateAgreement;
33: else
34:   return Refuse;
35: end if

```

Information stage: (Algorithm 4, lines 1-4) let r_i be the proposition rule sent and r_j the proposition rule received. r_{loc}^{cur} is the current local offer and r_{rec}^{cur} is the proposal received. r_{loc}^{stub} and r_{rec}^{stub} are used to indicate if a stubborn rule is reached on each side. Before negotiation starts, negotiators

Algorithm 5 searchDistantTree($i, r_i, r_{loc}^{cur}, r_{rec}^{cur}$)

```

1:  $i \leftarrow i + 1$ ;
2:  $r_i \leftarrow nextMove(r_{loc}^{cur}, r_{rec}^{cur}, T_p^{distant})$ ;
3:  $r_{loc}^{cur} \leftarrow r_i$ ;
4: Send  $r_i$  to opponent;

```

Algorithm 6 searchRelatedTree($i, r_i, r_{loc}^{cur}, r_{rec}^{cur}$)

```

1: if  $r_{loc}^{cur} \subset T_p^{related}$  then
2:    $i \leftarrow i + 1$ ;
3:   if  $r_{loc}^{stub} = \text{true}$  then
4:      $r_i \leftarrow r_{loc}^{cur}$ ;
5:   else
6:      $r_i \leftarrow nextMove(r_{loc}^{cur}, r_{rec}^{cur}, T_p^{related})$ ;
7:      $r_{loc}^{cur} \leftarrow r_i$ ;
8:   end if
9:   if  $(r_{rec}^{cur} = r_{loc}^{cur})$  or  $(r_{rec}^{cur} \prec r_{loc}^{cur})$  then
10:    return CreateAgreement;
11:  end if
12:  Send  $r_i$  to opponent;
13: else
14:  return Refuse;
15: end if

```

exchange some information in order to establish their ontological mapping. Consequently, the two sides have a common knowledge of entities.

Demand stage: (Algorithm 4, lines 5-10) at the beginning of the negotiation process, each participant sends its initial rule proposition r_i to its opponent. When both participants receive the opponent's initial offer r_j , the relation between current local rule r_{loc}^{cur} and received rules r_{rec}^{cur} is evaluated. If two rules have a $T_compatibility$ relation or if a received rule is a restriction of the current local rule, a *CreateAgreement* message is sent, then the negotiation will enter the contract establishment stage. Otherwise, participants begin the bargaining stage.

Bargaining stage: (Algorithm 4, lines 11-35) at the beginning of the bargaining stage, the stubbornness conditions r_{loc}^{stub} , r_{rec}^{stub} of both sides are tested.

The loop will be executed until one side makes a decision or both sides reach their stubborn rules. After receiving a current proposal rule r_{rec}^{cur} , one of the five possible reactions will be made:

1. The received message r_{rec}^{cur} is *CreateAgreement* (Algorithm 4, line 16): the message indicates that the opponent agrees on the current local rule r_{loc}^{cur} , the negotiation goes to contract establishment stage.
2. r_{rec}^{cur} activates the condition to trigger distant tree $T_p^{distant}$ (Algorithm 4, line 18): a new proposition r_i will be proposed by calling the proposition method $nextMove(r_{loc}^{cur}, r_{rec}^{cur}, T_p^{distant})$ (Algorithm 5). In fact, different negotiation strategies could contain different conditions to trigger the distant tree.
3. Rule received r_{rec}^{cur} has a $T_compatibility$ relation with the current local rule r_{loc}^{cur} or is a restriction of it (Algorithm 4, line 20): r_{rec}^{cur} is accepted by sending the message *CreateAgreement*.
4. The received rule r_{rec}^{cur} has a $S_compatibility$ relation with the current local rule r_{loc}^{cur} or r_{loc}^{cur} is a restriction of r_{rec}^{cur} (Algorithm 4, line 22): the related tree will be searched if such a tree exists (Algorithm 6). If r_{loc}^{cur} achieves a stubborn rule, the current proposition will be maintained. Otherwise, a new proposition r_i will be proposed by calling the proposition method $nextMove(r_{loc}^{cur}, r_{rec}^{cur}, T_p^{related})$: r_i is generated by weakening some entities (rule of lower depth is proposed) or changing branch. r_{rec}^{cur} will be accepted if it is a restriction of r_i or has a $T_compatibility$ relation with it. If the related tree does not exist, the received proposition will be refused.
5. For other cases (Algorithm 4, line 24): message *Refuse* is sent.

Contract establishment stage: the *CreateAgreement* message indicates that the rule is agreed by the sender. At the same time, a policy contract based on an agreement is generated by the agreement maker and sent to the opponent. Here a policy contract is composed of OrBAC rules. Upon

receiving the *CreateAgreement* message and generating the policy contract, the opponent evaluates it and replies by the message *Agreement* or *Refuse*.

6.5.2 Theoretical Results

This part deals with some results obtained by our formalization. We say that the algorithm is complete if it could reach an agreement when there is a positive outcome; The algorithm is correct when the positive outcome is a shared policy between negotiators.

Theorem 1. The negotiation algorithm is correct and complete.

Suppose that there does not exist a possible final agreement. In such a case, a negotiator, after visiting all nodes of its policy tree (the related tree and the distant tree), reaches one leaf node that constitutes its current local proposition. The same happens for its opponent. Each negotiator can not propose another rule because the current local rule is stubborn. By executing the line 34 in Algorithm 4, the negotiator will send the message *Refuse* then the negotiation fails. Suppose that the final agreement exists and the process starts with the proposal of a negotiator, it continues to compare the propositions received with the local rule which is assumed as the current one (lines 9, 16, 18, 20, 22, 31 in Algorithm 4 and line 9 in Algorithm 6). It makes a new proposal (line 4 in Algorithm 5 and line 12 in Algorithm 6) until a proposition received from the opponent is a restriction or has a $T_compatibility$ relation with its current one. Then the negotiation is successful with the negotiators sharing an agreement about a rule.

Theorem 2. In case that only one rule is negotiated, the negotiation algorithm solves the negotiation problem in $\mathcal{O}(n \times c)$ where n is the maximum number of nodes among negotiator's policy trees and c is the number of trees.

Consider the case in which there is no possible agreement. Since the stubborn rules of the two negotiators can not make an agreement, we shall visit at least all the nodes of one direction of a policy tree and at most all the trees. In the latter case, stubborn rules of these trees will not be

achieved except for the last tree visited. The searching strategy depends on the *nextMove* method which is not specified in this dissertation (line 2 in Algorithm 5 and line 6 in Algorithm 6). Moreover, if such an agreement can be found, negotiation necessarily terminates before. Therefore the case in which both sides reach their stubborn rules holds the highest complexity.

6.6 Application

The framework and algorithm above could be applied to different negotiation cases from agent-agent to agent-human. The difference between the two cases is that for the agent-agent case, each side should configure its negotiation framework and for the latter case, only the agent side should configure it. In this section, we illustrate a concrete scenario between two agents.

6.6.1 Scenario Description

INTER-TRUST [128] is an European project which aims at developing a framework to support trustworthy applications in heterogeneous networks and devices based on the enforcement of interoperable and changing security policies. One use case of this project is negotiating a security policy between a vehicle and the infrastructure for Intelligent Transport Systems (ITS). Here we adopt the scenario defined in [129]. The subjects involved in the example are: Bob (car's owner and car's local security policy manager), a French ITS station (French ITS service provider) and a Chinese ITS (Chinese ITS service provider). In our scenario, Bob has a set of security rules, which are defined when he subscribed as a premium user for his first service contract with the French ITS:

S1 : *permission(Operator_A, premium_user, access, DRP_service, France)*

S2 : *permission(Operator_A, premium_user, access, Safety_service, France)*

S3 : *permission(Operator_A, premium_user, access, CSA_service, France)*

This service contract gives him a set of privileges to gain access to different services. Bob travels to another country (China for example) and wants to use the same services with the same privileges offered by the French ITS. On the one hand, the service contract provided by the Chinese ITS is not the same as the one provided by the French ITS. On the other hand, there is an agreement between the French ITS and the Chinese ITS, which allows clients of both service providers to use services while traveling in another region. Table 6.2 shows the service mapping in this agreement which combines the services proposed in France with ones meaning the same in China.

Table 6.2: Services Mapping Table

Service ID	French ITS	Chinese ITS
S1	DRP_service	Cooperative-navigation
S2	Safety_service	Driving assistance-Road Hazard Warning
S3	CSA_service	Speed management
S4	-	Driving assistance-Cooperative awareness
S5	-	Location-based-services
S6	-	ITS station life cycle management
S7	-	Communities_services

Consequently, the result of this mapping is a redefinition of the policy contract by the Chinese ITS as follows:

S1 : *permission(Operator_B, custom_user, access, DRP_service, Bob_in_China)*

S2 : *permission(Operator_B, custom_user, access, Safety_service, Bob_in_China)*

S3 : *permission(Operator_B, business_user, access, CSA_service, Bob_in_China)*

S4 : *permission(Operator_B, custom_user, access, DrivingAssistance_*
CooperativeAwareness, Bob_in_China)

S5 : *permission(Operator_B, custom_user, access, Location_based_services,*
Bob_in_China)

S6 : *permission(Operator_B, business_user, access, ITSStationLifeCycleManagement,*
Bob_in_China)

S7 : *permission(Operator_B, business_user, access, Communities_services,*
Bob_in_China)

By default, a *custom_user* is a free user who can benefit from only basic services and a *business_user* should pay extra amount for the additional specified services. A *business_user* could also have access to basic services at the same time. Thus, the role *business_user* can be considered as a *sub* role of *custom_user*. Bob is a *custom_user* but he does not want to lose the CSA_service (S3) and he also does not want to pay an extra amount to have that service. Hence, he will try to negotiate the operator's policy based on his initial proposition. Our negotiation framework will be applied to the negotiation scenario. In the scenario, Bob negotiates a contract which contains seven OrBAC rules and those rules will be negotiated at the same time instead of one by one. For the purpose of synchronization, we assume that the agreement will be reached when all the rules are agreed simultaneously by the message *CreateAgreement*. Otherwise the negotiation continues.

6.6.2 Process of Security Policy Negotiation

As service terms are based on a common service vocabulary and the vocabulary is accessible for users, we suppose that Bob and the Chinese ITS station hold the same entity chains shown in Fig. 6.4. The negotiation configurations of Bob and the Chinese ITS station contain 1) initial proposition, 2) related policy tree, 3) distant policy tree. Shown in Fig. 6.5 and Fig. 6.6, Bob's empty related policy tree indicates that he does not want to weaken any entity of the policy in the process of negotiation. However, the ITS has its related policy tree which shows the possibilities to weaken the roles of $S3$ by $S3_1$, $S3_3$ and to weaken its view by $S3_2$. In the current negotiation algorithm for the vehicle and ITS station (Algorithm 4, line 18), a counter-offer from the distant tree will be triggered when 1) r_{rec}^{cur} has not been previously received and has a $T_compatibility$ relation with the current local rule r_{loc}^{cur} ; 2) r_{loc}^{cur} belongs to a distant tree. In fact, different negotiation strategies could have different conditions to activate or deactivate the distant tree.

At the demand stage, the ITS station and Bob exchange their initial propositions. However, the initial rules could not be agreed on. As a consequence, the bargaining stage takes place. Upon receiving $S3'$, $S4'$ and $S5'$, the ITS

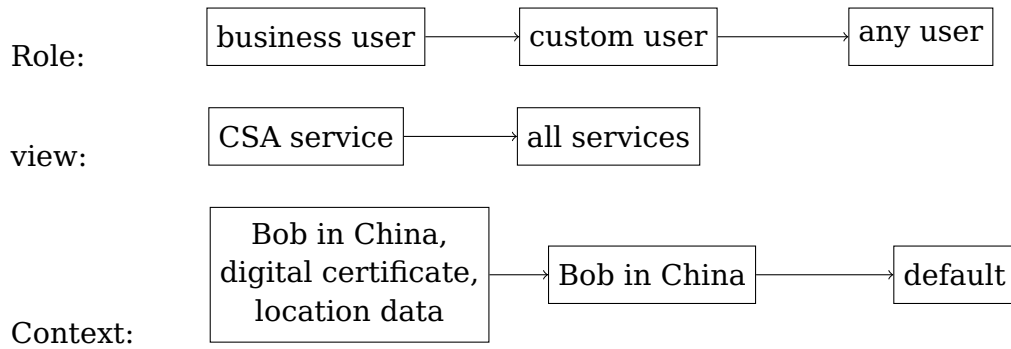


Figure 6.4: Entity chains of Bob and Chinese ITS station

Initial proposition	<p>S1, S2, S3', S4', S5', S6, S7</p> <p>S3': permission(Operator_B, custom_user, access, CSA_service, Bob_in_China)</p> <p>S4': permission(Operator_B, business_user, access, DrivingAssistance_CooperativeAwareness, Bob_in_China)</p> <p>S5': permission(Operator_B, business_user, access, Location_based_services, Bob_in_China)</p>
Related policy tree	-
Distant policy tree	<pre> graph TD S5'((S5')) --> S4'((S4')) </pre>

Figure 6.5: Bob's negotiation configuration

station makes an evaluation: $S3'$ is a generalization of $S3$; $S4'$ and $S5'$ are restrictions of $S4$ and $S5$. According to the negotiation algorithm, $S5'$ could be accepted. After visiting the related policy tree of $S3$ and the distant policy tree of $S4'$, new propositions $S3_1$ and $S2'$ with an initial proposition $S5'$ are sent to Bob. For Bob, $S3_1$ and $S2'$ are considered as acceptable rules because they are restrictions of $S3'$ and $S2$ which are his current local rules. Besides, $S5'$ has its distant policy tree which contains $S4'$. As a result, the second proposition of Bob is a combination of $S3_1$, $S2'$ and $S4'$. From the point of

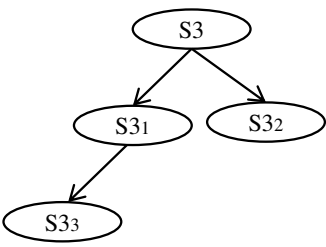
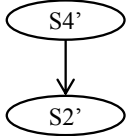
Initial proposition	S1, S2, S3, S4, S5, S6, S7
Related policy tree	 <pre> graph TD S3((S3)) --> S31((S31)) S3 --> S32((S32)) S31 --> S33((S33)) </pre> <p>S31: permission(Operator_B, custom_user, access, CSA_service, Bob_in_China & digitalCertificate & locationData)</p> <p>S32: permission(Operator_B, business_user, access, all_services, urgent)</p> <p>S33: permission(Operator_B, any_user, access, CSA_service, urgent)</p>
Distant policy tree	 <pre> graph TD S4p((S4')) --> S2p((S2')) </pre> <p>S2': permission(Operator B, business user, access, Safety_service, Bob_in_China)</p> <p>S4': permission(Operator_B, business_user, access, DrivingAssistance_CooperativeAwareness, Bob_in_China)</p>

Figure 6.6: Chinese ITS station's negotiation configuration

view of the ITS station, S_{31} and $S_{2'}$ have a total compatibility relation with its current local rules and $S_{4'}$ is a restriction of S_4 . According to the ITS's condition to trigger the distant tree search, the distant tree of $S_{4'}$ will not be searched again because it is the second time that the ITS station receives the proposal $S_{4'}$. Consequently, the ITS station accepts the second proposition from Bob by sending the message *CreateAgreement*. The ultimate policy contract is established after that Bob replies *Agreement*. All the negotiation process is presented in Fig. 6.7 and the final policy contract is a combination

of OrBAC rules:

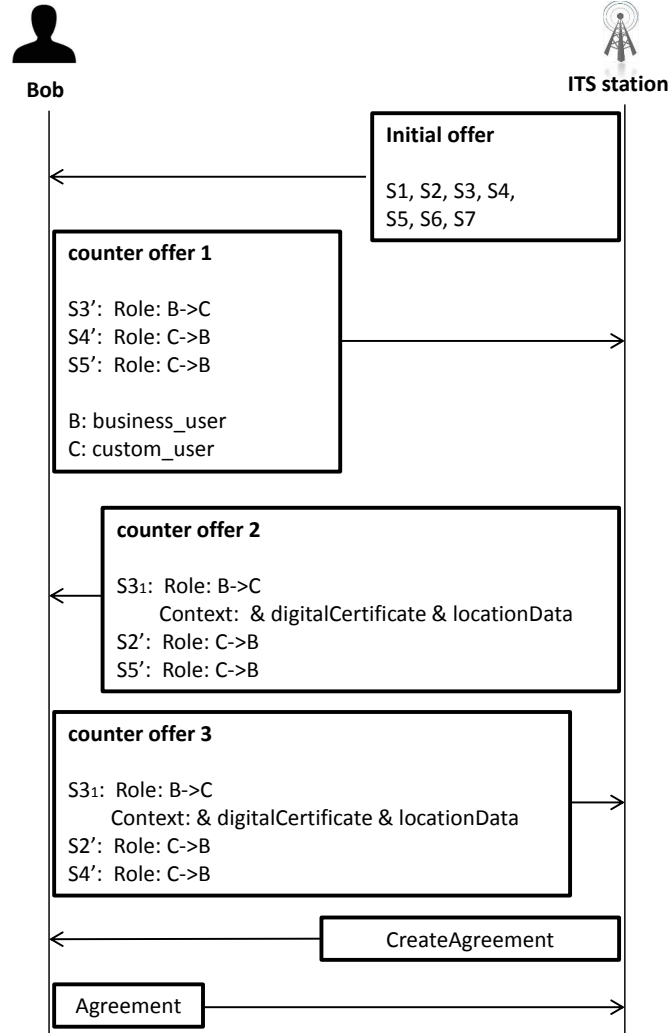


Figure 6.7: Negotiation between Bob and the Chinese ITS station

S1 : *permission*(Operator_B, custom_user, access, DRP_service, Bob_in_China)
S2 : *permission*(Operator_B, **business_user**, access, Safety_service, Bob_in_China)
S3 : *permission*(Operator_B, **custom_user**, access, CSA_service, Bob_in_China
 & **digitalCertificate** & **locationData**)
S4 : *permission*(Operator_B, **business_user**, access, DrivingAssistance_
 CooperativeAwareness, Bob_in_China)

S5 : *permission(Operator_B, custom_user, access, Location_based_services, Bob_in_China)*

S6 : *permission(Operator_B, business_user, access, ITSSStationlifeCycleManagement, Bob_in_China)*

S7 : *permission(Operator_B, business_user, access, Communities_services, Bob_in_China)*

6.6.3 Prototype

We have developed a Java-based prototype to demonstrate the scenario. Figure 6.8 presents its architecture. The prototype contains two major components: the negotiation module and the local configuration component.

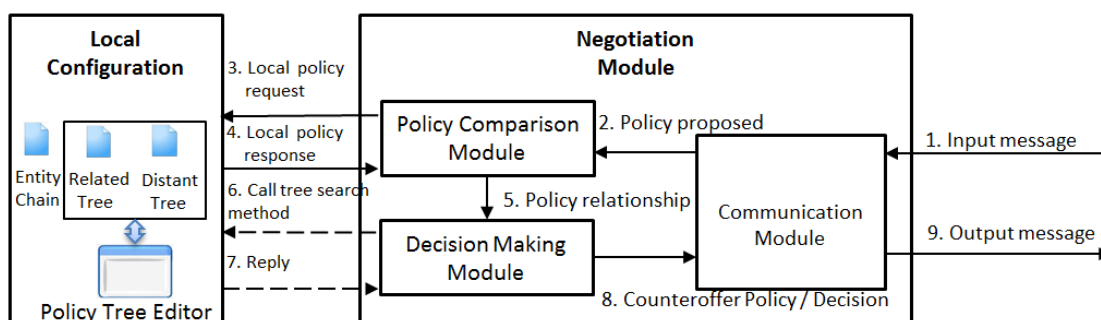


Figure 6.8: Architecture of the prototype

The function of each sub component is presented as follows:

- **Communication Module:** sends and receives messages, each message contains an OrBAC policy or decision.
- **Decision Making Module:** executes the negotiation algorithm, generates a decision or a policy as a counter-offer.
- **Policy Comparison module:** compares the received OrBAC policy with the current local one and returns their relation.

- **Policy Tree Editor:** an independent GUI tool which could be used to edit related tree and distant tree and save the related files.

Shown in Figure 6.8, the *Communication Module* takes a message as an input. If the message is a decision, the *Communication Module* will execute negotiation algorithm directly by replying with a message or stopping the negotiation. In case that the message is a OrBAC policy, the received policy will be sent to the *Policy Comparison Module* in step 2. Then the *Policy Comparison Module* makes a comparison between the policy received and the local policy by requesting the current local policy from *Local Configuration* component (Steps 3,4). After the comparison, the *Policy Comparison Module* sends the rule relationships (Step 5) to the *Decision Making Module* which executes the negotiation algorithm. Steps 6, 7 will be executed when "searchDistantTree (Algorithm 5)" or "searchRelatedTree (Algorithm 6)" function is called. Finally, a counter-offer policy or decision will be sent to the *Communication Module* (Step 8) and forwarded to the opponent negotiator in the communication channel (Step 9).

From the negotiator's point of view, the negotiator needs to configure the (1) initial contract, (2) entity chain, (3) preference on security policy. Initial contract contains different OrBAC rules which can be written in the form of XML. An initial contract including the rule of S1 is showed as follows.

Example of initial contract

```

1 <rule1>
2   <right>permission</right>
3     <organization>Operator_B</organization>
4     <role>customer_user</role>
5     <activity>access</activity>
6     <view>DRP_service</view>
7     <context>BobInChina</context>
8 </rule1>
9 <rule2>
10  ...
11 </rule2>

```

The entity chain can be also written in the form of XML. Following is an example of an entity chain for *role* and it indicates that in a *role* entity chain, *business_user* is a *sub* entity of *custom_user* and *custom_user* is a *sub* entity of *any_user*.

Example of an entity chain for the *Role* entity

```

1 <role>
2   <chainRole>
3     <entityRole>business_user</entityRole>
4     <entityRole>custom_user</entityRole>
5     <entityRole>any_user</entityRole>
6   </chainRole>
7 </role>

```

We need the *Policy Tree Editor* to configure the related tree and the distant tree. The policy is saved in a local file (.tr) which can be opened and visualized by the *Policy Tree Editor*. Figure 6.9 shows the GUI interface of the *Policy Editor* with which we can load, save and edit a policy.

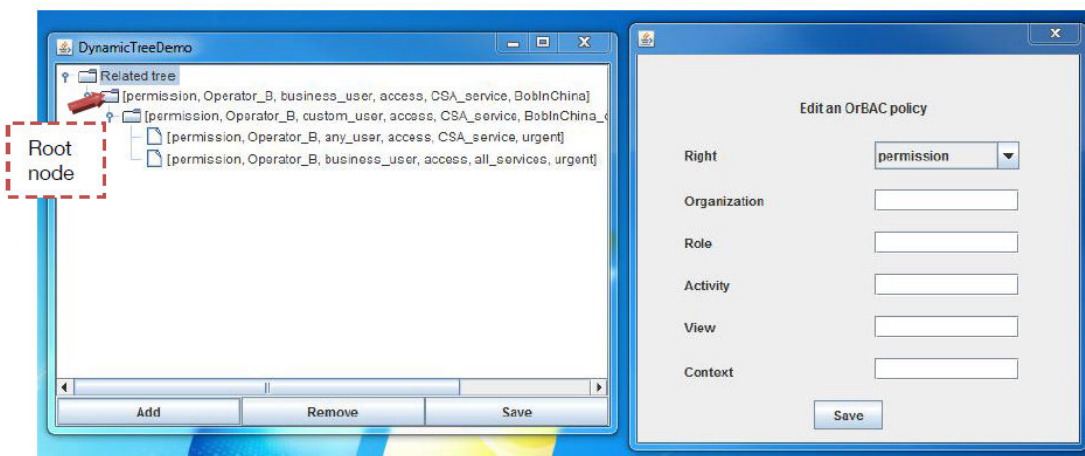


Figure 6.9: Using the policy tree editor to visualize and edit policy

After the local configuration is ready, the negotiator can start the negotiation by running the negotiation module then the message windows will be shown. Figure 6.10 is an example of a message sent during the negotiation.

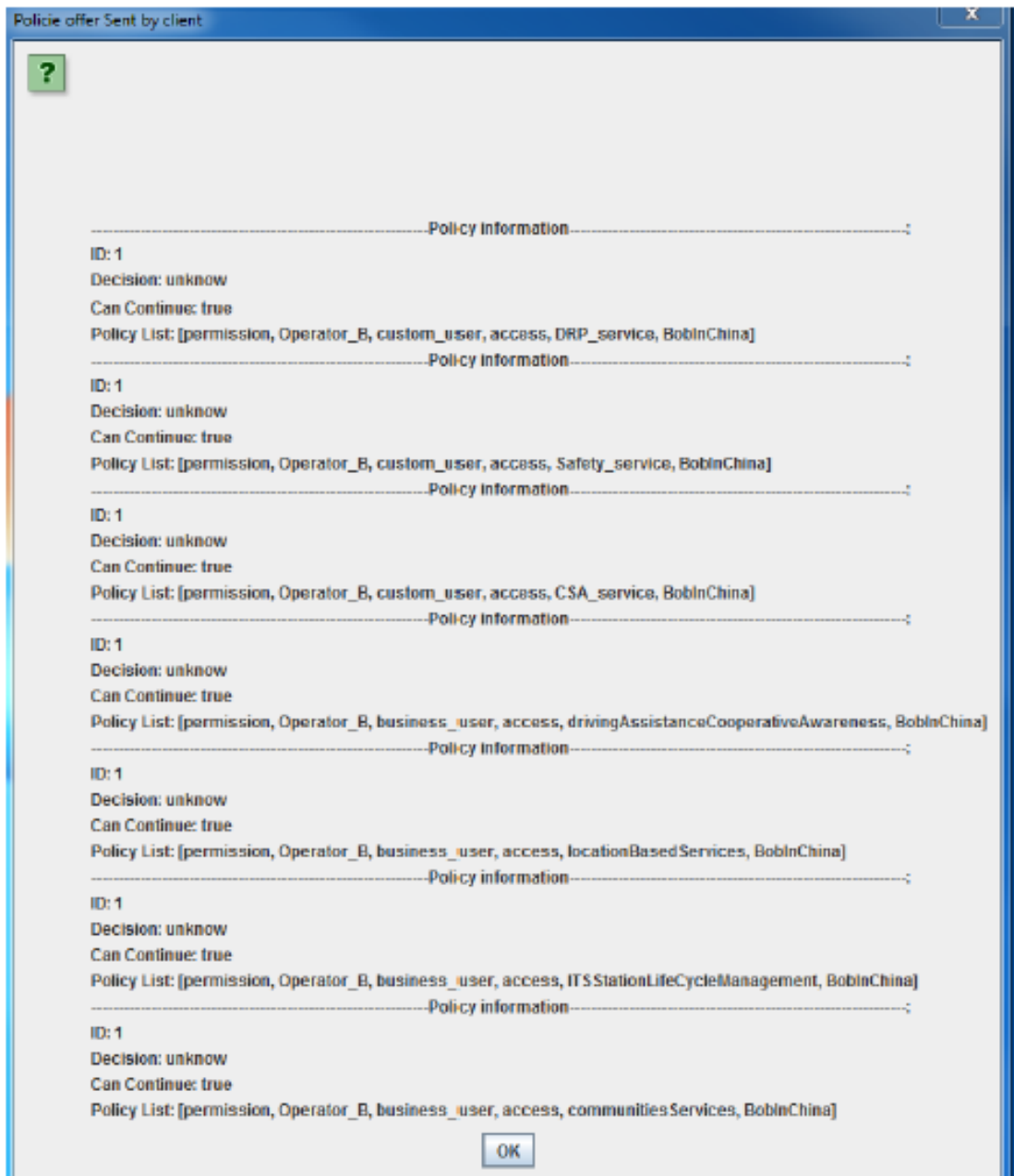


Figure 6.10: A message sent during negotiation

6.7 Conclusion

The main objective of this chapter is to expose our proposition to show how to negotiate security policies. The proposition gives mainly a framework with which we can configure negotiable policies by presenting preferences. The preferences related to security policies are configured in the form of tree architecture. Introducing five relationships, we showed how the security rules are compared and evaluated. An algorithm was also given to the negotiation process. In addition, we integrated OrBAC policies in our negotiation model. The proposed algorithm also deals with the tree search module which makes the next proposition. Detail of this module is a part of future work which concerns the negotiation strategy and the Game Theory may be applied.

Chapter 7

Conclusion and Perspectives

The emerging service mode with multiple SPs brings more flexibility and efficiency to SCs regarding choosing services. In the process of data and service exchange, the security policy plays a fundamental role in the privilege management. With security policy, actors are able to edit their own privileges and specify the one which restricts permissions for visitors. For example, it can be applied to define service contents for a vehicle station, or to express preferences from CSPs and CSCs in virtual resource allocation.

Although the security policy offers several benefits, its evaluation and negotiation still present a variety of challenges, especially among different security policy models and between negotiators with their own preferences. In this sensitive context, the first objective of this thesis is to provide a general method for security policy evaluation. We hope that our approach will be helpful in the SP selection. By similarity score, the similarity level between two security policies is quantified. We have shown that the scores produced by the PSM method are related to the similarity rates from our test. A particular prototype is made for cloud-storage-based SPs selection. However, the assumption that both parties disclose their security policies for evaluation does not suffice in all the scenarios. In case that the security policies are not expressed explicitly, we proposed a framework that derives security requirements to security policies. Meanwhile, the framework is used in a scenario where VMs are allocated in an IaaS infrastructure.

The second objective of this thesis is to introduce guidelines for the pro-

cess of the security policy negotiation between two actors, typically negotiators aiming to reach agreement on access control policy. We have integrated the bargaining game and the meaning negotiation in our proposition.

7.1 Main Results

The main results of this dissertation are stated as follows:

A new similarity measure method for security policies: A generic and light-weight method [17] is proposed to compare and evaluate security policies belonging to different models. With the method, a SC is able to quickly locate SPs with potential similar policies. At the same time, our method shows more accuracy through the brute-force based tests. We propose integrating the policy similarity measure algorithm in the SP selection process and a prototype has been developed to execute the algorithm in the Cloud storage selection process.

A policy-based framework for the expression and enforcement of security policies in multi-cloud environments: The framework [18] applies OrBAC [19] policy to IaaS resource allocation. The attribute-based security requirements in a SLA contract can be derived to concrete OrBAC rules then these rules are considered together with other non-security demands during the enforcement of resource allocation. The contribution meets key-functional requirements for user-centric as (i) it addresses the SLA configuration options at the IaaS layer from service capacity to security constraint. (ii) it considers multiple requirements of security and applies the OrBAC model to translate attribute-based security constraints to concrete policies. (iii) it provides a conflict management mechanism to detect and handle the contradictory requirements from a CSC and CSPs, with the possibility to judge the policy priority by evaluating users' profiles. (iv) it proposes a resource allocation algorithm which takes account of resource capacity, QoS and security policy. A prototype for VM scheduling in an OpenStack-based

multi-cloud environments is developed.

A model for security policy negotiation: Based on the bargaining game [21] and the meaning negotiation, a framework and an algorithm [22] are developed to negotiate a common security policy. A prototype is also developed to simulate vehicle negotiation process. With policy tree based configurations, two negotiators are possible to reach their agreement step by step. Compared with other works, our method is indeed the pioneering one which integrates the security policy in the negotiation process with a complete support of configuration, protocol, algorithm and strategy.

7.2 Perspectives

We give a set of future research directions that could be investigated as a continuation of the results presented in this thesis.

Integrating the PSM technology in security policy negotiation: As a PSM score presents the similarity level between two policies, it may be useful in the security policy negotiation process. After introducing the PSM score, relationships between security policies are not only classified but also quantized. Consequently, more strategies can be executed according to the PSM score. Besides, the decision making can be also based on PSM score which brings more fine-grained control to the negotiation process.

Introducing contextual based policy in virtual resource allocation: Our current solution for policy-based virtual resource allocation is based on the OrBAC policy which holds “default” for the context. A context is viewed as an extra condition that must be satisfied to activate a given security rule. The capacity to express context conditions enables users to integrate context-based requirements in their WS-Agreement template then those requirements can be derived to context-based OrBAC policies. A context-aware security policy also offers the possibility for the users to specify their security

preferences that will be enforced during the service discovery process [130]. As mentioned in [42], context requirements are possible to cover different aspects such as time, space and history. By this way, a user's deployment requirement is enriched and more diverse.

Applying AI technologies and strategy to policy negotiation: The proposed process of reaching agreement in security policy negotiation is based on the bargaining game and the meaning negotiation. A general framework and a tree search strategy are specified and illustrated. However, the current negotiation model is only suitable for one-one negotiation. In order to negotiate security policies among a SC and multiple SPs, our model can be extended by using other models in the Game Theory such as the English auction [125]. Another direction is to diversify negotiation strategies: different policy tree search strategies can be applied by negotiators for different scenarios.

Improving interoperability between different policy models during policy negotiation: As different access control policy models have their advantages and limitations, users may take different models to specify the privileges. When different policies belonging to different models are needed to be negotiated, interoperability becomes an important issue to be overcome. Although some work such as [131] has proposed using the ABAC model to unify the DAC, MAC and RBAC models, there exist lots of investigations to do regarding (i) unifying more AC policy models; (ii) developing an interoperable policy engine and integrating it in policy negotiation.

Extending policy-based resource allocation framework to more scenarios: Presented in Chapter 4, our current allocation framework can be used for the VM deployment scenario which belongs to the "compute" aspect of Cloud Computing. In addition, our solution can be used in more related aspects such as "storage" and "network". Regarding the "storage" aspect, the policy-based framework enables users to express their security requirements for their data; at the same time, SPs which offer their stor-

age space also specify their preferences on the characteristics of data. The result of execution of our framework is that a user's data is stored by multiple distributed SPs. Another possible application in the "network" aspect, is that the user expresses their traffic routing requirements and sends them to the SDN controller. Then the controller selects a routing path among switches which hold security related preference of a SP. The final solution of the routing path is the one which satisfies the user's and SPs' requirements simultaneously.

Our current work concentrates on the resource allocation and the security configuration in SDN networks. The context is the same as the one defined in Chapter 4. With the evolution of hardware, network services and data, Cloud Computing becomes one of the key technologies that satisfies the growing demands of software and hardware resources with its availability and efficiency for the requested resources. At the same time, SDN is becoming the backbone of the cloud infrastructure. It offers many advantages such as programmability, agility, abstraction, centralization, visualization and flexibility. As a result, many cloud providers select SDN as a cloud network service and offer it to customers. However, due to the rising number of network cloud providers and their offers, network cloud customers must find the provider which best satisfies their requirements. In this context, based on network security policy, we propose a negotiation and an enforcement framework for SDN service provider selection. Our solution is a pioneering attempt to tackle this issue, specifically in terms of security policy. We integrate it in an existing SDN security environment. Our solution transforms the customer's security requirements into SDN firewall rules and deploys them as OpenFlow rules in the SDN infrastructure. Figure 7.1 shows a negotiation scenario of our proposal. SDN orchestrator works as a broker between client and SPs. The scenario consists of steps as follows. Firstly both the NSC and the NSPs specify their security requirements related to the infrastructure in order to ensure end-to-end security across different components (Steps 1,2). After receiving security requirement expressions from the NSC, the SDN Orchestrator assesses the expressions by comparing them with service templates of NSP then starts a negotiation process with

NSC when necessary (Step 3). A successful negotiation generates an agreement about security expression (Step 4) which will be derived to high-level security policies of the infrastructure (Step 5). Particularly, the high-level policies are translated to OpenFlow rules when the NSP adopts the OpenFlow protocol [132] (Step 6). In the end, the SDN orchestrator deploys the generated OpenFlow rules on the chosen NSP (Step 7).

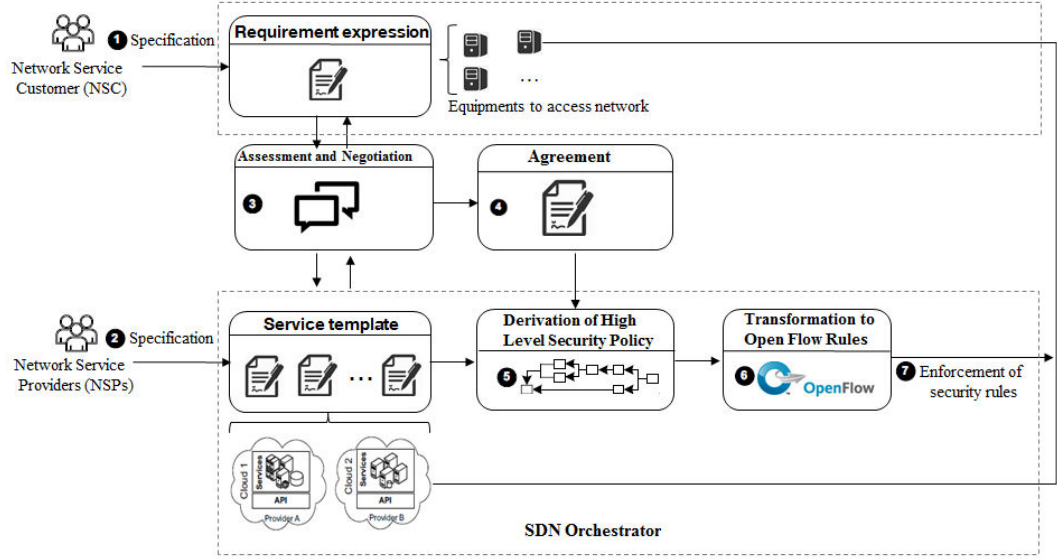


Figure 7.1: A scenario of expression and negotiation service contract for SDN networks

The scenario above brings some technical challenges. Firstly, we are planning to use a structure which consists of atomic elements to specify a security rule.

$$r : \{e_1, e_2, \dots, e_n\} \quad (7.1)$$

where e_i is an atomic rule element and each rule element has five properties:

- **Type:** $e.type \in \{subject, action, object, context\}$. Each network rule should contain elements belonging to *subject*, *action* and *object* type. The *context* type is an option.

- **Domain:** $e.domain \in \{protocol, time...\}$. Domain restricts the unit of an element.
- **Value:** $e.value$. There are two types of values: *variable* to be assigned and *non-variable* which are already assigned. We used x_i to present a *variable*. Both *variable* and *non-variable* can be assigned by three kinds of data types:
 - **constant:** numeric value or semantic value. For example $e.value : TCP$.
 - **interval:** numeric interval. For example, $e.value : [8 : 00, 20 : 00]$.
 - **set:** numeric or semantic set. For example $e.value : \{15 : 00, 16 : 00, 17 : 00\}$, $e.value : \{UDP, TCP, ICMP\}$
- **Public preference** (pub_{pre}): A variable can possess its public preference which is accessible as public information. Interval (numeric) and set (numeric, semantic) can be used for preference specification.
- **Private preference** (pri_{pre}): A variable can possess its private preference which is the local configuration for negotiation and can not be disclosed to others. The expression is similar to the one for pub_{pre} .

For the rule without *context* type element, we add a context element with $e.domain = \top$ and $e.value = \top$. " \top " indicates that all the propositions are acceptable. The intersection between any value and " \top " is the value itself. In terms of preference, coexistence of pub_{pre} and pri_{pre} introduces the possibility of lying which makes negotiation more complicated. For simplicity, in our current proposal, a value should not hold pub_{pre} and pri_{pre} at the same time.

We intend to specify different types of network security policies classified in [133]: *consume/produce* policies for end-system, *propagate* policies for communication, *transform* policies for protocol and *filter* policies for firewall. In the rule expression, variables can be expressed in two status: assigned and not assigned. The objective of negotiation is to instantiate the variables not assigned and reach an agreement on the variables which are already assigned.

Appendix A

List of Acronyms

ABAC: Attribute-Based Access Control

AC: Access Control

ACL: Access Control List

AI: Artificial Intelligence

CAIQ: Consensus Assessments Initiative Questionnaire

CSC: Cloud Service Customer

CSP: Cloud Service Provider

DAC: Discretionary Access Control

IaaS: Infrastructure as a Service

ITS: Intelligent Transport System

LBAC: Lattice-based Access Control

MAC: Mandatory Access Control

MN: Meaning Negotiation

OrBAC: Organization-Based Access Control

PaaS: Platform as a Service

PSM: Policy Similarity Measure

QoS: Quality of Service

RBAC: Role-Based Access Control

RBSLA: Rule-Based Service Level Agreement

SC: Service Customer

SaaS: Software as a Service

SDN: Software Defined Networking

SecLA: Security Level Agreement

SLA: Service Level Agreement

SSLA: Security Service Level Agreement

SP: Service Provider

TN: Trust Negotiation

VM: Virtual Machine

WS-Agreement: Web Services Agreement

WSLA: Web Service Level Agreement

XACML: extensible Access Control Markup Language

Appendix B

Brute-force based test for existing work

Figure 2 shows the brute-force test result of policy similarity score by using the same test environment illustrated in Section 3.3. The y-axis represents the PSM score computed by the algorithm proposed in [60]; the x-axis shows the test result of policy similarity defined by Equation (2.1). We remark that the similarity score computed does not approximate to the test result. The main reason is that, firstly, as a brute-force based test method, our input requests are more exhaustive than the ones generated by other test tools such as MTBDD [134]. Secondly, the PSM algorithm defined in [60] focuses only on the literal level but not logic aspect of security policy. As a result, two security rules sharing the majority of common elements are considered to hold a higher similarity score. However, the rest of elements may cause totally different decisions which indicates that the two rules are not similar in terms of output.

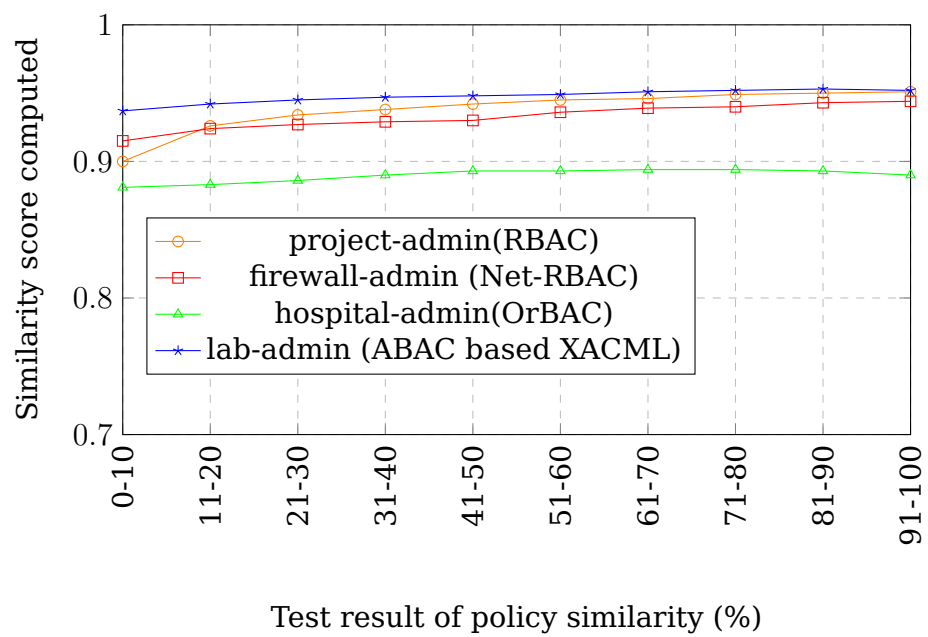


Figure 2: Experiment of similarity score (*set-4*).

Appendix C

JSON-based WS-Agreement contracts

WS-Agreement contract specified by CSC

```
1  {"name":"clientTemplate",
2   "context":"VM-deployment",
3
4   "serviceRequirement":
5     {
6       "VM1_volume":"40_GB"
7       "VM2_volume":"40_GB",
8       "VM3_volume":"50_GB",
9     },
10
11  "serviceDescription":
12    {
13      "VM1_purpose":"dev",
14      "VM1_data":"private",
15      "VM1_application":"internal",
```

```
16
17     "VM2_purpose": "prod",
18     "VM2_data": "public",
19     "VM2_application": "business"
20
21     "VM3_purpose": "test",
22     "VM3_data": "private",
23     "VM3_application": "internal",
24 },
25
26 "guaranteeTerm":
27 {
28     "VM1_availability": "more_96_percentage",
29     "VM2_availability": "more_98_percentage"
30     "VM3_availability": "more_96_percentage",
31 }
32 "creationConstraint":
33 [
34     ["permission", {"certificate": "true"}, {"purpose": "dev"}],
35
36     ["permission", {"certificate": "true"}, {"purpose": "prod"}],
37
38     ["permission", {"certificate": "true"}, {"purpose": "test"}],
39
```

```
40  ["permission",{"location":"Europe"},{"ID":"VM2"}],  
41  
42  ["permission",{"location":"Europe"},{"Purpose":"test"}],  
43  
44  ["separation",{"ID":"VM1"},{"ID":"VM3"}]  
45 ],  
46 }
```

WS-Agreement contract specified by CSP1

```
1  {  
2  "name":"HOST1",  
3  "context":"VM-deployment",  
4  
5  "serviceDescription":  
6    {  
7      "volume":"100_GB",  
8      "price":"0.2_dollar",  
9      "location":"France"  
10     "certificate":"true",  
11     },  
12  
13   "guaranteeTerm":  
14     {  
15       "availability":"more_97_percentage"  
16     }
```

```
17
18 "creationConstraint":
19 [
20 ["prohibition",{"ID":"HOST1"},{"purpose":"test"}]
21 ],
22
23 }
```

WS-Agreement contract specified by CSP2

```
1 {
2   "name":"HOST2",
3   "context":"VM-deployment",
4
5   "serviceDescription":
6     {
7       "volume":"100_GB",
8       "price":"0.3_dollar",
9       "location":"UK"
10      "certificate":"false",
11    },
12
13   "guaranteeTerm":
14     {
15       "availability":"more_98_percentage"
16     }
```



```
17
18 "creationConstraint":
19 [
20 ["prohibition",{"ID":"HOST2"},{"purpose":"dev"}]
21 ],
22
23 }
```


List of Publications

International Conferences

- LI. Yanhuang, N. Cuppens-Boulahia, C. Jean-Michel, F. Cuppens and F. Vincent. Expression and Enforcement of Security Policy for Virtual Resource Allocation in IaaS Cloud. In IFIP International Information Security and Privacy Conference. Springer International Publishing, Proceedings, pages 105-118, 2016.
- LI. Yanhuang, N. Cuppens-Boulahia, C. Jean-Michel, F. Cuppens, F. Vincent and J. Xiaoshu. Similarity Measure for Security Policies in Service Provider Selection. In : International Conference on Information Systems Security. Springer International Publishing, Proceedings, pages 227-242, 2015.
- LI. Yanhuang, N. Cuppens-Boulahia, C. Jean-Michel, F. Cuppens and F. Vincent. Reaching Agreement in Security Policy Negotiation. In : 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications. IEEE, Proceedings, pages 98-105, 2014.
- Cuppens-Boulahia Nora, LI Yanhuang, Zerkane Salaheddine, Espes David, Cuppens Frédéric, Crom Jean-michel, Negotiation and Enforcement of Network Security Policies for SDN Providers. In ACM Asia Conference on Computer and Communications Security (ASIACCS), ACM, 2017 (Submit).

Research Projects

- Alex Palesandro, Aurélien Wailly, Ruan He, Yvan Rafflé, Jean-Philippe Wary, Yanhuang Li, Soren Bleikertz, Alysson Bessani, Reda Yaich, Sabir Idrees, Nora Cuppens, Frédéric Cuppens, Ferdinand Brasser, Jialin Huang, Majid Sobhani, Krzysztof Oborzynski, Gitesh Vernekar, Meilof Veeningen, Paulo Sousa. D2.1: Architecture for Secure Computation Infrastructure and Self-Management of VM Security, SuperCloud project. Technical Report, November, 2015.
- Samiha Ayed, Muhammad Sabir Idrees, Nora Cuppens, Frédéric Cuppens, Yanhuang Li, Khalifa Toumi, Mohamed Aouadi, Ana Cavalli, Jorge Bernal Bernabé, Juan M. Marin Pérez, Fernando Pereniguez, Jose L. Hernandez, Antonio F. Skarmeta Gomez, Wissam Mallouli, Edgardo Montes de Oca, Bachar Wehbi, Crisan de los Santos. Description of models for the specification of secure interoperability policies - final version, Inter-Trust project. Technical Report, March, 2015.

Posters

- LI. Yanhuang, N. Cuppens-Boulahia, C. Jean-Michel, F. Cuppens and F. Vincent. Interoperability and Negotiation of Security Policies, journée doctorant, Orange Labs, Paris, 2015.
- LI. Yanhuang, N. Cuppens-Boulahia, C. Jean-Michel, F. Cuppens and F. Vincent. Interoperability and Negotiation of Security Policies, journée thématique, Télécom ParisTech, 2014.

References

- [1] Erik Rissanen et al. extensible access control markup language (XACML) version 3.0, 2013. [xi](#), [15](#), [16](#), [17](#), [18](#), [82](#)
- [2] Fabien Autrel. *MotOrBAC 2 user manual v2.5*. [xi](#), [20](#)
- [3] Heiko Ludwig, Alexander Keller, Asit Dan, Richard P King, and Richard Franck. Web service level agreement (WSLA) language specification. *IBM Corporation*, pages 815–824, 2003. [xi](#), [21](#)
- [4] Oliver Waeldrich, Dominic Battré, Francis Brazier, Kassidy Clark, Michel Oey, Alexander Papaspyrou, Philipp Wieder, and Wolfgang Ziegler. WS-Agreement negotiation version 1.0. In *Open Grid Forum*, volume 35, page 41, 2011. [xi](#), [22](#), [23](#), [94](#)
- [5] Karin Bernsmed, Martin Gilje Jaatun, and Astrid Undheim. Security in service level agreements for cloud computing. In Frank Leymann, Ivan Ivanov, Marten van Sinderen, and Boris Shishkov, editors, *CLOSER*, pages 636–642. SciTePress, 2011. ISBN 978-989-8425-52-2. URL <http://dblp.uni-trier.de/db/conf/closer/closer2011.html>. [xi](#), [25](#), [26](#), [32](#)
- [6] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011. [xi](#), [48](#), [49](#)

- [7] Ang Li, Xiaowei Yang, Srikanth Kandula, and Ming Zhang. CloudCmp: comparing public cloud providers. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 1–14. ACM, 2010. [1](#)
- [8] Stephen S. Yau and Yin Yin. Qos-based service ranking and selection for service-based systems. In *Services Computing (SCC), 2011 IEEE International Conference on*, pages 56–63. IEEE, 2011. [1](#)
- [9] Jesus Luna, Hamza Ghani, Daniel Germanus, and Neeraj Suri. A security metrics framework for the cloud. In *Security and Cryptography (SECRYPT), 2011 Proceedings of the International Conference on*, pages 245–250. IEEE, 2011. [1](#)
- [10] Ahmed Taha, Ruben Trapero, Jesus Luna, and Neeraj Suri. AHP-based quantitative approach for assessing and comparing cloud security. In *Trust, Security and Privacy in Computing and Communications (Trust-Com), 2014 IEEE 13th International Conference on*, pages 284–291. IEEE, 2014. [1](#)
- [11] Ting Yu, Marianne Winslett, and Kent E Seamons. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Transactions on Information and System Security (TISSEC)*, 6(1):1–42, 2003. [2](#), [77](#), [79](#)
- [12] Diala Abi Haidar, Nora Cuppens-Boulahia, Frédéric Cuppens, and Hervé Debar. XeNA: an access negotiation framework using XACML. *annals of telecommunications-Annales des télécommunications*, 64(1-2):155–169, 2009. [2](#), [78](#)
- [13] Anthony Nadalin, Gene Thurston AmberPoint, Peter Dapkus BEA, Hal Lockhart BEA, Symon Chang CommerceOne, Thomas DeMartini ContentGuard, Guillermo Lao ContentGuard, TJ Pannu ContentGuard, Shawn Sharp Cyclone Commerce, Ganesh Vaideeswaran Documentum, et al. Web services security. *SOAP Message Security. Version*, 1, 2002. [2](#)

-
- [14] Web services trust language (WS-Trust). 2002. [2](#)
 - [15] Paul Madsen, Jeff Hodges, and Bronislav Kavsan. Liberty metadata description and discovery specification. *Liberty Alliance Project, Version*, 1:1–33, 2003. [2](#)
 - [16] Bernhard Hollunder. Domain-specific processing of policies or: WS-Policy intersection revisited. In *Web Services, 2009. ICWS 2009. IEEE International Conference on*, pages 246–253. IEEE, 2009. [2](#)
 - [17] Yanhuang Li, Nora Cuppens-Boulahia, Jean-Michel Crom, Frédéric Cuppens, Vincent Frey, and Xiaoshu Ji. Similarity measure for security policies in service provider selection. In *Information Systems Security*, pages 227–242. Springer, 2015. [3](#), [112](#)
 - [18] Yanhuang Li, Nora Cuppens-Boulahia, Jean-Michel Crom, Frédéric Cuppens, and Vincent Frey. Expression and enforcement of security policy for virtual resource allocation in IaaS cloud. In *IFIP International Information Security and Privacy Conference*, pages 105–118. Springer, 2016. [3](#), [112](#)
 - [19] Anas Abou El Kalam, Salem Benferhat, Alexandre Miège, Rania El Baida, Frédéric Cuppens, Claire Saurel, Philippe Balbiani, Yves Deswarte, and Gilles Trouessin. Organization Based Access Control. In *4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, 2003. [3](#), [18](#), [79](#), [112](#)
 - [20] Elisa Burato and Matteo Cristani. The process of reaching agreement in meaning negotiation. In *Transactions on Computational Collective Intelligence VII*, pages 1–42. Springer, 2012. [3](#), [84](#), [92](#)
 - [21] Shinsuke Kambe. Bargaining with imperfect commitment. *Games and Economic Behavior*, 28(2):217–237, 1999. [3](#), [83](#), [113](#)
 - [22] Yanhuang Li, Nora Cuppens-Boulahia, Jean-Michel Crom, Frederic Cuppens, and Vincent Frey. Reaching agreement in security policy

- negotiation. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on*, pages 98–105. IEEE, 2014. 3, 57, 113
- [23] Alain Andrieux, Karl Czajkowski, Asit Dan, Kate Keahey, Heiko Ludwig, Toshiyuki Nakata, Jim Pruyne, John Rofrano, Steve Tuecke, and Ming Xu. Web services agreement specification (WS-Agreement). In *Open Grid Forum*, volume 128, page 216, 2007. 4, 22, 56
- [24] Michael A Harrison, Walter L Ruzzo, and Jeffrey D Ullman. Protection in operating systems. *Communications of the ACM*, 19(8):461–471, 1976. 11
- [25] Liang Chen. *Analyzing and Developing Role-Based Access Control Models*. PhD thesis, Royal Holloway, University of London, 2011. URL <http://digirep.rhul.ac.uk/file/817519d1-0731-c09f-1522-e36433db3d2c/1/liangcheng.pdf>. 12
- [26] Ravi S. Sandhu. Lattice-based access control models. *Computer*, 26(11):9–19, November 1993. ISSN 0018-9162. doi: 10.1109/2.241422. URL <http://dx.doi.org/10.1109/2.241422>. 12
- [27] D Elliott Bell and Leonard J La Padula. Secure computer system: Unified exposition and multics interpretation. Technical report, DTIC Document, 1976. 12
- [28] D.F. Ferraiolo and D.R. Kuhn. Role-based access controls. *arXiv preprint arXiv:0903.2171*, pages 554 – 563, 2009. URL <http://arxiv.org/abs/0903.2171>. 13
- [29] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-Based Access Control Models. *Computer*, 29(2):38–47, 1996. doi: 10.1109/2.485845. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=485845. 13
- [30] Matunda Nyanchama and Sylvia Osborn. The role graph model and conflict of interest. *ACM Transactions on Information and System*

-
- Security*, 2(1):3–33, February 1999. ISSN 10949224. doi: 10.1145/300830.300832. URL <http://portal.acm.org/citation.cfm?doid=300830.300832>. 13
- [31] Ravi Sandhu, David Ferraiolo, and Richard Kuhn. The NIST model for role-based access control: towards a unified standard. In *Proceedings of the fifth ACM workshop on Role-based access control*, RBAC '00, pages 47–63, New York, NY, USA, 2000. ACM. ISBN 1-58113-259-X. doi: 10.1145/344287.344301. URL <http://doi.acm.org/10.1145/344287.344301>. 13
- [32] Nathan Dimmock, András Belokosztolszki, David Eysers, Jean Bacon, and Ken Moody. Using trust and risk in role-based access control policies. In *Proceedings of the ninth ACM symposium on Access control models and technologies*, SACMAT '04, pages 156–162, New York, NY, USA, 2004. ACM. ISBN 1-58113-872-5. doi: 10.1145/990036.990062. URL <http://doi.acm.org/10.1145/990036.990062>. 13
- [33] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security*, 4(3):224–274, August 2001. ISSN 10949224. doi: 10.1145/501978.501980. URL <http://portal.acm.org/citation.cfm?doid=501978.501980>. 13
- [34] Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a role-based trust-management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, SP '02, pages 114–, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1543-6. URL <http://dl.acm.org/citation.cfm?id=829514.830539>. 13
- [35] Guido Boella and Leendert van der Torre. Role-based rights in artificial social systems. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, IAT '05, pages

- 516–519, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2416-8. doi: 10.1109/IAT.2005.123. URL <http://dx.doi.org/10.1109/IAT.2005.123>. 13
- [36] Yan Wang and Vijay Varadharajan. Role-based recommendation and trust evaluation. In *The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007)*, pages 278–288. IEEE, July 2007. ISBN 0-7695-2913-5. doi: 10.1109/CEC-EEE.2007.83. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4285225>. 13
- [37] T. Finin, A. Joshi, L. Kagal, J. Niu, R. Sandhu, W. Winsborough, and B. Thuraisingham. Rowlbac: representing role based access control in owl. In *Proceedings of the 13th ACM symposium on Access control models and technologies, SACMAT '08*, pages 73–82, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-129-3. doi: 10.1145/1377836.1377849. URL <http://doi.acm.org/10.1145/1377836.1377849>. 13
- [38] Eric Yuan and Jin Tong. Attributed based access control (ABAC) for Web services. In *IEEE International Conference on Web Services (ICWS'05)*. IEEE, 2005. ISBN 0-7695-2409-5. doi: 10.1109/ICWS.2005.25. 14, 77
- [39] Adam J. Lee. *Towards Practical and Secure Decentralized Attribute-Based Authorisation Systems*. PhD thesis, University of Illinois, 2008. 14, 77
- [40] Eric Yuan and Jin Tong. Attributed based access control (ABAC) for web services. In *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*. IEEE, 2005. 18
- [41] Ed Coyne and Timothy R Weil. ABAC and RBAC: scalable, flexible, and auditable access management. *IT Professional*, 15(3):0014–16, 2013. 18

-
- [42] Céline Coma, Nora Cuppens-Boulahia, Frédéric Cuppens, and Ana R Cavalli. Context ontology for secure interoperability. In *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, pages 821–827. IEEE, 2008. 19, 114
- [43] Fabien Autrel, Frédéric Cuppens, N Cuppens-Boulahia, and Celine Coma. MotOrBAC 2: a security policy tool. In *3rd Conference on Security in Network Architectures and Information Systems (SAR-SSI 2008), Loctudy, France*, pages 273–288, 2008. 20, 44, 66
- [44] Frédéric Cuppens and Alexandre Mieke. AdOrbac: an administration model for OrBAC. *International Journal of Computer Systems Science & Engineering*, 19(3):151–162, 2004. 20
- [45] Ali Anjomshoaa, Fred Brisard, Michel Drescher, Donal Fellows, An Ly, Stephen McGough, Darren Pulsipher, and Andreas Savva. Job submission description language (jsdl) specification, version 1.0. In *Open Grid Forum, GFD*, volume 56, 2005. 22
- [46] Henar Muñoz, Ioannis Kotsiopoulos, András Micsik, Bastian Koller, and Juan Mora. Flexible sla negotiation using semantic annotations. In *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops*, pages 165–175. Springer, 2010. 22
- [47] Irmos project. URL <http://www.irmosproject.eu/>. 22
- [48] Wolfgang Ziegler, Ming Jiang, and Kleopatra Konstanteli. OPTIMIS SLA framework and term languages for SLAs in cloud environment. *OPTIMIS Project Deliverable D, 2*, 2011. 22
- [49] Adrian Paschke. RBSLA a declarative rule-based service level agreement language based on RuleML. In *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, volume 2, pages 308–314, Nov 2005. doi: 10.1109/CIMCA.2005.1631486. 24

- [50] Harold Boley, Adrian Paschke, and Omair Shafiq. RuleML 1.0: the overarching specification of web rules. *Lecture Notes in Computer Science*, 6403(4):162–178, 2010. [24](#)
- [51] J. Skene, D. Davide Lamanna, and W. Emmerich. Precise service level agreements. In *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on*, pages 179–188, May 2004. doi: 10.1109/ICSE.2004.1317440. [24](#)
- [52] Ronda R Henning. Security service level agreements: quantifiable security for the enterprise? In *Proceedings of the 1999 workshop on New security paradigms*, pages 54–60. ACM, 1999. [25](#)
- [53] K. Bernsmed, M.G. Jaatun, P.H. Meland, and A. Undheim. Security SLAs for federated cloud services. In *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on*, pages 202–209, Aug 2011. doi: 10.1109/ARES.2011.34. [25](#)
- [54] Chen-Yu Lee, K.M. Kavi, R.A. Paul, and M. Gomathisankaran. Ontology of secure service level agreement. In *High Assurance Systems Engineering (HASE), 2015 IEEE 16th International Symposium on*, pages 166–172, Jan 2015. doi: 10.1109/HASE.2015.33. [25](#)
- [55] Sheldon Borkin. The HIPAA final security standards and ISO/IEC 17799. *Collect. Information Security Reading Room*, 2003. [25](#)
- [56] Cloud Security Alliance. <http://cloudsecurityalliance.org>, 2011. URL <https://cloudsecurityalliance.org>. [26](#)
- [57] CSA. Consensus assessment initiative questionnaire (CAIQ), 2014. URL <https://cloudsecurityalliance.org/research/cai>. [26](#), [33](#)
- [58] European Commission. The cloud service level agreement standardisation guidelines, 2014. [26](#)
- [59] Standardizing cloud security SLAs - SPECS project. Technical report, 2015. URL http://www.specs-project.eu/wp-content/uploads/2015/04/SPECS_std_one-pager_final_v2.pdf. [27](#)

-
- [60] Dan Lin, Prathima Rao, Rodolfo Ferrini, Elisa Bertino, and Jorge Lobo. A similarity measure for comparing XACML policies. *Knowledge and Data Engineering, IEEE Transactions on*, 25(9):1946–1959, 2013. [27](#), [30](#), [36](#), [44](#), [123](#)
- [61] Quan Pham, Jason Reid, and Ed Dawson. Policy filtering with XACML. 2011. [29](#), [30](#)
- [62] Dan Lin and Anna Squicciarini. Data protection models for service provisioning in the cloud. In *Proceedings of the 15th ACM symposium on Access control models and technologies*, pages 183–192. ACM, 2010. [29](#), [31](#)
- [63] OASIS Standard. extensible access control markup language (XACML) version 2.0, 2005. [30](#)
- [64] Dan Lin, Prathima Rao, Elisa Bertino, and Jorge Lobo. An approach to evaluate policy similarity. In *Proceedings of the 12th ACM symposium on Access control models and technologies*, pages 1–10. ACM, 2007. [30](#)
- [65] Wu Bei, Chen Xing-yuan, and Zhang Yong-fu. A policy rule dissimilarity evaluation approach based on fuzzy theory. In *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, pages 1–6. IEEE, 2009. [30](#)
- [66] Eun Cho, Gabriel Ghinita, and Elisa Bertino. Privacy-preserving similarity measurement for access control policies. In *Proceedings of the 6th ACM workshop on Digital identity management*, pages 3–12. ACM, 2010. [31](#)
- [67] Rizwana AR Shaikh and M Sasikumar. Dynamic parameter for selecting a cloud service. In *Computation of Power, Energy, Information and Communication (ICCPEIC), 2014 International Conference on*, pages 32–35. IEEE, 2014. [31](#)

- [68] Antonia Bertolino, Said Daoudagh, Donia El Kateb, Christopher Henard, Yves Le Traon, Francesca Lonetti, Eda Marchetti, Tejeddine Mouelhi, and Mike Papadakis. Similarity testing for access control. *Information and Software Technology*, 58:355–372, 2015. [31](#)
- [69] Alessandro Ferreira Leite, Vander Alves, Genaina Nunes Rodrigues, Claude Tadonki, Christine Eisenbeis, and Alba Cristina Magalhaes Alves de Melo. Automating resource selection and configuration in inter-clouds through a software product line method. In *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*, pages 726–733. IEEE, 2015. [32](#)
- [70] Amit Nathani, Sanjay Chaudhary, and Gaurav Somani. Policy based resource allocation in IaaS cloud. *Future Generation Computer Systems*, 28(1):94–103, 2012. [32](#)
- [71] Guiyi Wei, Athanasios V Vasilakos, Yao Zheng, and Naixue Xiong. A game-theoretic method of fair resource allocation for cloud computing services. *The journal of supercomputing*, 54(2):252–269, 2010. [32](#)
- [72] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems*, 28(5): 755–768, 2012. [32](#)
- [73] Rajkumar Buyya, Saurabh Kumar Garg, and Rodrigo N Calheiros. SLA-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. In *Cloud and Service Computing (CSC), 2011 International Conference on*, pages 1–10. IEEE, 2011. [32](#)
- [74] Linlin Wu, Saurabh Kumar Garg, and Rajkumar Buyya. SLA-based resource allocation for software as a service provider (saas) in cloud computing environments. In *Cluster, Cloud and Grid Computing (CC-Grid), 2011 11th IEEE/ACM International Symposium on*, pages 195–204. IEEE, 2011. [32](#)

-
- [75] Andrés García García, Ignacio Blanquer Espert, and Vicente Hernández García. SLA-driven dynamic cloud resource management. *Future Generation Computer Systems*, 31:1–11, 2014. [32](#)
- [76] European Network and Information Security Agency. *Cloud Computing: Benefits, risks and recommendations for information security*. ENISA, 2009. [33](#)
- [77] Erdal Cayirci, Alexandr Garaga, Anderson Santana, and Yves Roudier. A cloud adoption risk assessment model. In *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pages 908–913. IEEE Computer Society, 2014. [33](#)
- [78] Stefan Berger, Ramón Cáceres, Ken Goldman, Dimitrios Pendarakis, Ronald Perez, Josyula R Rao, Eran Rom, Reiner Sailer, Wayne Schildhauer, Deepa Srinivasan, et al. Security for the cloud infrastructure: Trusted virtual data center implementation. *IBM Journal of Research and Development*, 53(4):6–1, 2009. [33](#)
- [79] Khalid Bijon, Ram Krishnan, and Ravi Sandhu. Virtual resource orchestration constraints in cloud infrastructure as a service. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, pages 183–194. ACM, 2015. [33](#)
- [80] Khalid Bijon, Ram Krishnan, and Ravi Sandhu. Mitigating multi-tenancy risks in IaaS cloud through constraints-driven virtual resource scheduling. In *Proceedings of the 20th ACM Symposium on Access Control Models and Technologies*, pages 63–74. ACM, 2015. [33](#)
- [81] Ravi Jhawar, Vincenzo Piuri, and Pierangela Samarati. Supporting security requirements for resource management in cloud computing. In *Computational Science and Engineering (CSE), 2012 IEEE 15th International Conference on*, pages 170–177. IEEE, 2012. [33](#)
- [82] Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996. [41](#), [82](#)

- [83] Frédéric Cuppens, Nora Cuppens-Boulahia, and Alexandre Miège. Inheritance hierarchies in the OrBAC model and application in a network environment. *Proc. Foundations of Computer Security (FCS04)*, pages 41–60, 2004. 41, 88
- [84] Configuring keystone. <http://docs.openstack.org/developer/keystone/configuration.html>. 44
- [85] Safaà Hachana, Nora Cuppens-Boulahia, and Frédéric Cuppens. Mining a high level access control policy in a network with multiple firewalls. *Journal of Information Security and Applications*, 20:61–73, 2015. 44, 60
- [86] Supercloud project: User-centric management of security and dependability in clouds of clouds. <http://www.supercloud-project.eu/>. 46, 65
- [87] Piero Bonatti, Sabrina De Capitani di Vimercati, and Pierangela Samarati. An algebra for composing access control policies. *ACM Transactions on Information and System Security (TISSEC)*, 5(1):1–35, 2002. 48
- [88] Devops. <http://en.wikipedia.org/wiki/DevOps>, . 58
- [89] Anas Abou El Kalam, RE Baida, Philippe Balbiani, Salem Benferhat, Frédéric Cuppens, Yves Deswarte, Alexandre Mieke, Claire Saurel, and Gilles Trouessin. Organization based access control. In *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, pages 120–131. IEEE, 2003. 60
- [90] Frédéric Cuppens and Nora Cuppens-Boulahia. Modeling contextual security policies. *International Journal of Information Security*, 7(4): 285–305, 2008. 60
- [91] Jaideep Vaidya, Vijayalakshmi Atluri, and Qi Guo. The role mining problem: finding a minimal descriptive set of roles. In *Proceedings of*

-
- the 12th ACM symposium on Access control models and technologies*, pages 175–184. ACM, 2007. 60
- [92] Frédéric Cuppens, Nora Cuppens-Boulahia, and Meriam Ben Ghorbel. High level conflict management strategies in advanced access control models. *Electronic Notes in Theoretical Computer Science*, 186:3–26, 2007. 64
- [93] Michel Kamel, Romain Laborde, François Barrère, and Abdelmalek Benzekri. A trust-based virtual collaborative environment. *JDIM*, 6(5): 405–413, 2008. 64
- [94] Nuno Neves Fernando M. V. Ramos. Preliminary architecture of the multi-cloud network virtualization infrastructure. Technical report, Faculdade de Ciencias da Universidade de Lisboa, 2015. 66
- [95] Devstack. <http://docs.openstack.org/developer/devstack/>, . 66
- [96] Openstack open source cloud computing software. <https://www.openstack.org/>. 66
- [97] Graphstream: A dynamic graph library. <http://graphstream-project.org/>. 68
- [98] Compute service command-line client. <http://docs.openstack.org/cli-reference/nova.html/>. 68
- [99] Json. <http://en.wikipedia.org/wiki/JSON>. 69
- [100] William H Winsborough, Kent E Seamons, et al. Negotiating disclosure of sensitive credentials. 1999. 75, 77, 86
- [101] William H Winsborough, Kent E Seamons, and Vicki E Jones. Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, volume 1, pages 88–102. IEEE, 2000. 75, 86

- [102] Yunxi Zhang and Darren Mundy. Remembrance of local information status for enforcing robustness of policy-exchanged strategies for trust negotiation. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on*, pages 106–113. IEEE, 2014. 76
- [103] Adam J Lee, Marianne Winslett, and Kenneth J Perano. Trustbuilder2: A reconfigurable framework for trust negotiation. In *Trust Management III*, pages 176–195. Springer, 2009. 77
- [104] Adam J Lee. Trustbuilder2 user manual version 0.1. Technical report, Technical report, May, 2007. 77
- [105] E. Bertino, E. Ferrari, and A. Squicciarini. X-tnl: An XML-based language for trust negotiations. In *Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks, POLICY '03*, pages 81–, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1933-4. URL <http://dl.acm.org/citation.cfm?id=826036.826848>. 77
- [106] E. Bertino, E. Ferrari, and A. C. Squicciarini. Trust-x: A peer-to-peer framework for trust establishment. *IEEE Trans. on Knowl. and Data Eng.*, 16(7):827–842, July 2004. ISSN 1041-4347. doi: 10.1109/TKDE.2004.1318565. URL <http://dx.doi.org/10.1109/TKDE.2004.1318565>. 77
- [107] A. Squicciarini, E. Bertino, Elena Ferrari, F. Paci, and B. Thuraisingham. PP-trust-X: A system for privacy preserving trust negotiations. *ACM Trans. Inf. Syst. Secur.*, 10(3), July 2007. ISSN 1094-9224. doi: 10.1145/1266977.1266981. URL <http://doi.acm.org/10.1145/1266977.1266981>. 78
- [108] Piero Bonatti, Juri Luca De Coi, Daniel Olmedilla, and Luigi Sauro. A rule-based trust negotiation system. *IEEE Transactions on Knowledge and Data Engineering*, 22(11):1507–1520, 2010. 78

-
- [109] Diala Abi Haidar, Nora Cuppens-Boulahia, Frederic Cuppens, and Herve Debar. An extended RBAC profile of XACML. In *Proceedings of the 3rd ACM workshop on Secure web services*, pages 13–22. ACM, 2006. [79](#)
- [110] Muhammad Sabir Idrees, Samiha Ayed, Nora Cuppens-Boulahia, and Frederic Cuppens. Car2x communication-putting security negotiation into practice. In *2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*, pages 1–5. IEEE, 2014. [79](#)
- [111] Elisa Bertino, Anna C Squicciarini, Ivan Paloscia, and Lorenzo Martino. WS-AC: A fine grained access control system for web services. *World Wide Web*, 9(2):143–171, 2006. [80](#)
- [112] Siddharth Bajaj, Don Box, Dave Chappell, Francisco Curbera, Glen Daniels, Phillip Hallam-Baker, Maryann Hondo, Chris Kaler, Dave Langworthy, Anthony Nadalin, et al. Web services policy 1.2-framework (ws-policy). *W3C Member Submission*, 25:12, 2006. [80](#)
- [113] Virgil D Gligor, Himanshu Khurana, Radostina K Koleva, Vijay G Bharadwaj, and John S Baras. On the negotiation of access control policies. In *Security Protocols*, pages 188–201. Springer, 2001. [80](#)
- [114] Vijay G Bharadwaj and John S Baras. Towards automated negotiation of access control policies. In *null*, page 111. IEEE, 2003. [81](#)
- [115] Stefano Bistarelli, Ugo Montanari, Francesca Rossi, Thomas Schiex, Gérard Verfaillie, and H  lene Fargier. Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison. *Constraints*, 4(3):199–240, 1999. [81](#)
- [116] Gail-Joon Ahn and Ravi Sandhu. Role-based authorization constraints specification. *ACM Transactions on Information and System Security (TISSEC)*, 3(4):207–226, 2000. [82](#)
- [117] Himanshu Khurana and Virgil D Gligor. A model for access negotiations in dynamic coalitions. In *Enabling Technologies: Infrastructure*

- for Collaborative Enterprises, 2004. WET ICE 2004. 13th IEEE International Workshops on*, pages 205–210. IEEE, 2004. [82](#)
- [118] Wei Xue, Jinpeng Huai, and Yunhao Liu. Access control policy negotiation for remote hot-deployed grid services. In *e-Science and Grid Computing, 2005. First International Conference on*, pages 9–pp. IEEE, 2005. [82](#)
- [119] Pooya Mehregan and Philip WL Fong. Policy negotiation for co-owned resources in relationship-based access control. In *Proceedings of the 21st ACM on Symposium on Access Control Models and Technologies*, pages 125–136. ACM, 2016. [82](#)
- [120] Philip WL Fong. Relationship-based access control: protection model and policy language. In *Proceedings of the first ACM conference on Data and application security and privacy*, pages 191–202. ACM, 2011. [82](#)
- [121] Eric Grégoire and Sébastien Konieczny. Logic-based approaches to information fusion. *Information Fusion*, 7(1):4–18, 2006. [83](#)
- [122] Richard Booth. A negotiation-style framework for non-prioritised revision. In *Proceedings of the 8th conference on Theoretical aspects of rationality and knowledge*, pages 137–150. Morgan Kaufmann Publishers Inc., 2001. [83](#)
- [123] Richard Booth. Social contraction and belief negotiation. *Information Fusion*, 7(1):19–34, 2006. [83](#)
- [124] Thomas F Gordon. The pleadings game. *Artificial Intelligence and Law*, 2(4):239–292, 1993. [83](#)
- [125] Peter R Wurman, Michael P Wellman, and William E Walsh. A parametrization of the auction design space. *Games and economic behavior*, 35(1):304–338, 2001. [83](#), [114](#)

-
- [126] Elisa Burato and Matteo Cristani. Contract clause negotiation by game theory. In *Proceedings of the 11th international conference on Artificial intelligence and law*, pages 71–80. ACM, 2007. 84
- [127] Céline Coma, Nora Cuppens-Boulahia, Frédéric Cuppens, and Ana R Cavalli. Interoperability of context based system policies using o2o contract. In *Signal Image Technology and Internet Based Systems, 2008. SITIS'08. IEEE International Conference on*, pages 137–144. IEEE, 2008. 90
- [128] Inter-trust project. <http://www.inter-trust.eu/>. 99
- [129] Scenario about one use case ITS_S services access control and negotiation management. Technical report, 2014. URL inter-trust.lcc.uma.es. 99
- [130] Slim Trabelsi, Laurent Gomez, and Yves Roudier. Context-aware security policy for the service discovery. In *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, volume 1, pages 477–482. IEEE, 2007. 114
- [131] Xin Jin, Ram Krishnan, and Ravi S Sandhu. A unified attribute-based access control model covering DAC, MAC and RBAC. *DBSec*, 12:41–55, 2012. 114
- [132] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008. 116
- [133] Romain Laborde, Bassem Nasser, Frédéric Grasset, François Barrere, and Abdelmalek Benzekri. A formal approach for the evaluation of network security mechanisms based on RBAC policies. *Electronic Notes in Theoretical Computer Science*, 121:117–142, 2005. 117
- [134] Dan Lin, Prathima Rao, Elisa Bertino, Ninghui Li, and Jorge Lobo. Exam: a comprehensive environment for the analysis of access control

policies. *International Journal of Information Security*, 9(4):253–273, 2010. [123](#)

1. Introduction

Aujourd'hui sur Internet, de nombreuses entreprises s'associent pour établir des écosystèmes larges et dynamiques. Lors de transactions commerciales, des acteurs jouent le rôle de fournisseurs ou de consommateurs de services. Par exemple, Amazon, Microsoft, Google et Orange proposent leurs plateformes de Cloud à leurs clients. Comme différents fournisseurs de services peuvent définir et exprimer différentes exigences de politique de sécurité, il faut les mettre en correspondance et en négocier l'interopérabilité. Pour ce faire, un contrat de service portera la politique de sécurité avec d'autres éléments comme le prix et la qualité de service. Pour automatiser l'établissement de ce contrat, différentes technologies comme la définition et l'expression de sécurité, la comparaison et l'évaluation des politiques, la négociation et la composition de service doivent être développées (Figure 1).

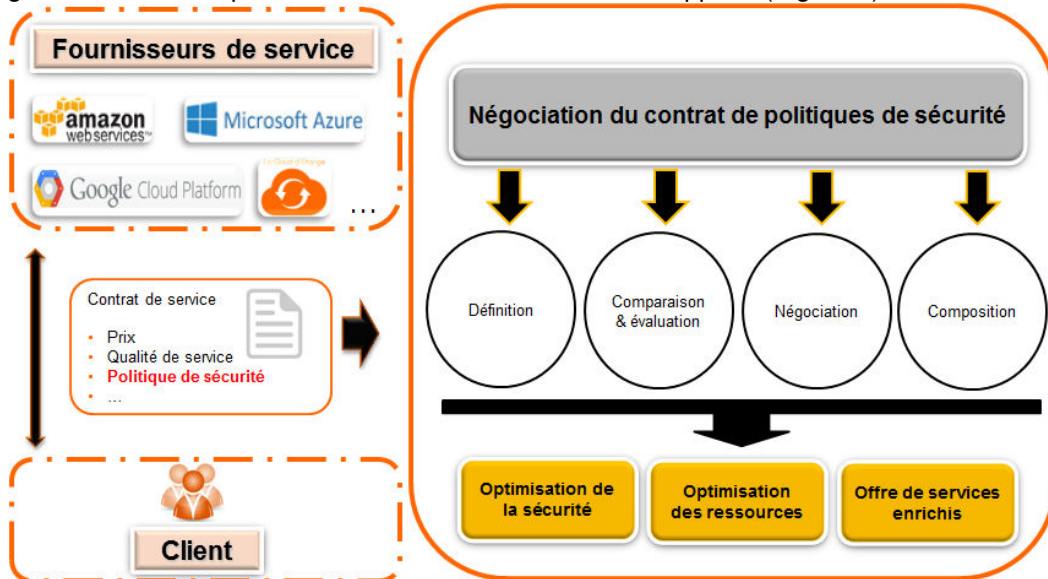


Figure 1 : Contrat entre fournisseur et consommateur de services

2. Sélection des fournisseurs de service

2.1 Mesurer la similarité entre les politiques de sécurité (PSM)

2.1.1 Introduction

Un score plus élevé entre les politiques p_1 et p_2 indique qu'elles sont plus susceptibles de partager un niveau de sécurité équivalent, et donner les mêmes décisions. Les approches existantes couvrent des dimensions numériques et typologiques, et se concentrent principalement sur des politiques XACML. Cependant, peu d'efforts ont été faits pour étendre l'approche de mesure aux multiples modèles de politique et les appliquer à des scénarios concrets. Dans ce chapitre, nous proposons un nouvel algorithme pour calculer le score de similarité entre deux politiques.

Le PSM¹ attribue un score de similarité S_{policy} pour deux politiques, qui se rapproche du pourcentage des paires de règles aboutissant à la même décision. La définition formelle est donnée dans l'équation 1, où $Num(sameDecision(r_{1i}, r_{2j}))$ désigne le nombre de paires de règles aboutissant à la même décision et $Num(allDecision(r_{1i}, r_{2j}))$ désigne le

¹ Policy Similarity Measure = Mesure de Similarité de Politiques

nombre total de paires. Le score de similarité est une valeur comprise entre 0 et 1. Deux politiques de sécurité sont dites équivalentes si leur score de similarité est égal à 1.

$$S_{policy}(p_1, p_2) \approx \frac{Num(sameDecision(r_{1i}, r_{2j}))}{Num(allDecision(r_{1i}, r_{2j}))}, \quad r_{1i} \in p_1, r_{2j} \in p_2$$

Equation 1 : score de similarité

2.1.2 Processus de calcul

Comme le montre dans la figure 2, l'algorithme de PSM prend deux politiques en entrée et génère un score de similarité en sortie. Le processus de calcul peut être divisé en quatre étapes.

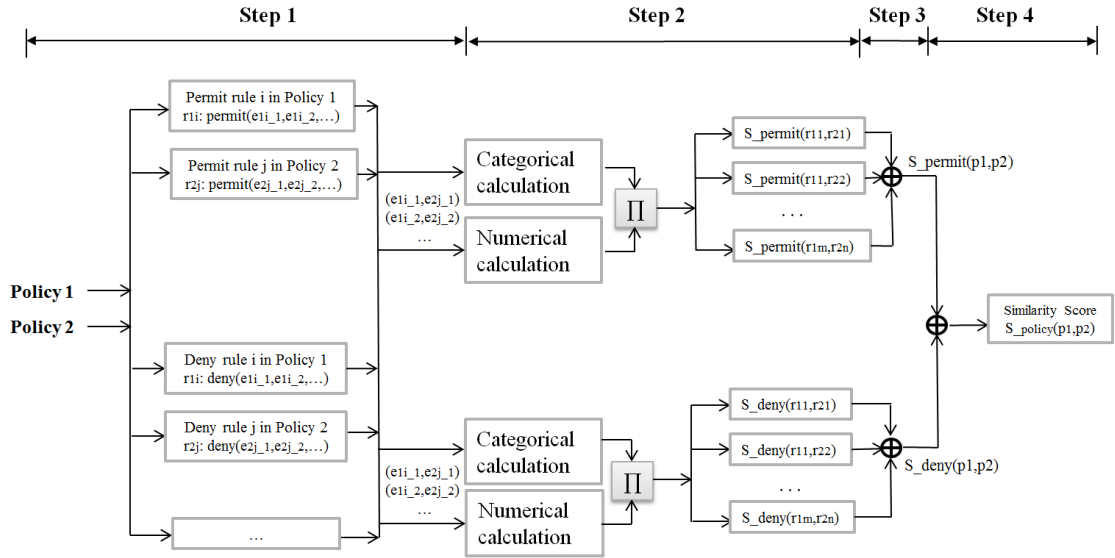


Figure 2 : Le processus de calcul du score de similarité

Etape 1 : transformation de la politique. Les politiques sont divisées en règles atomiques.

$$p_1 : permit(e_{1i_1}, e_{1i_2}, \dots), deny(e_{2i_1}, e_{2i_2}, \dots), \dots$$

$$p_2 : permit(e_{2j_1}, e_{2j_2}, \dots), deny(e_{2j_1}, e_{2j_2}, \dots), \dots$$

Etape 2 : calcul du score de la paire de règles. Les scores de chaque paire de règles appartenant au même effet de décision (d) entre les deux politiques sont calculés. Dans l'équation (3.3), le score pour chaque paire de règles est le produit des scores de toutes les paires d'éléments.

$$S_d(r_{1i}, r_{2j}) = \prod_k S(e_{1i_k}, e_{2j_k}), \quad r_{1i} \in p_1, r_{2j} \in p_2 \quad (3.3)$$

Etape 3 : calcul de l'effet de la décision. Chaque $S_d(p_1, p_2)$ équivaut à la somme de tous les scores de similarité des paires de règles d'un seul effet de décision (équation 3.4).

$$S_d(p_1, p_2) = \sum_i \sum_j S_d(r_{1i}, r_{2j}), \quad r_{1i} \in p_1, r_{2j} \in p_2 \quad (3.4)$$

Etape 4 : calcul du score total. Comme montré dans l'équation (3.5), le score total est calculé à partir des scores des différents effets de décision $S_d(p_1, p_2)$ et du nombre total de paires de règles.

$$S_{policy}(p_1, p_2) = \frac{\sum_d S_d(p_1, p_2)}{\sum_d Num(d)}, \quad d \in (permit, deny, \dots) \quad (3.5)$$

2.1.3 Résultats de l'expérience

Afin de vérifier si notre algorithme est applicable à des cas réels, nous comparons le pourcentage des mêmes paires de décision avec le score de PSM. Tout d'abord, nous mettons en œuvre un générateur de politiques qui prend des éléments de règle en entrée et génère des politiques. Deuxièmement, nous avons extrait des éléments de règle de quatre différents modèles de politique, chacun est lié à un scénario réel : RBAC (*Role-Based Access Control*) pour la gestion de projet, Net-RBAC pour la configuration de pare-feu, OrBAC (*Organization-Based Access Control*) pour la gestion de l'hôpital, ABAC (*Attribute-Based Access Control*) pour l'administration de laboratoire de recherche. Troisièmement, ces éléments de règle sont entrés dans le générateur de la politique et chaque paire de politique générée obtient un score de similarité par notre algorithme. Enfin, nous entrons diverses combinaisons d'éléments comme des requêtes de contrôle d'accès et comptons le pourcentage de la même paire de décision entre les décisions de sortie.

Les figures 3 et 4 montrent le score de similarité (axe Y) et pourcentage de même décision pour les paires de règles (axe X) dans set-4 et set-8. Dans set-4, chaque politique contient quatre règles et dans set-8 chaque politique a huit règles. Chaque ensemble de test contient 1000 paires de politiques. Nous observons que le score augmente lorsque la similarité entre deux politiques augmente. En même temps, les valeurs expérimentales approchent les scores calculés et la quantité de règles n'a aucun impact sur la variation des courbes de sortie. Le résultat du test nous permet de conclure que le score de PSM se rapproche bien de la similarité entre les politiques de sécurité.

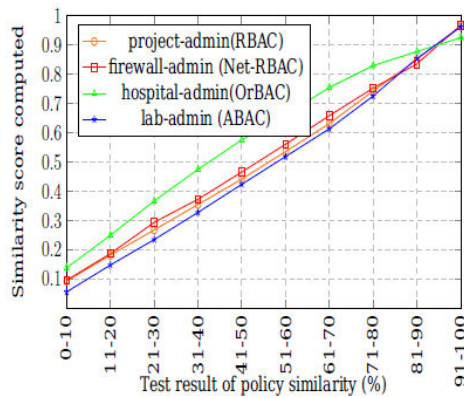


Figure 3 : expérience du score de similarité (set-4)

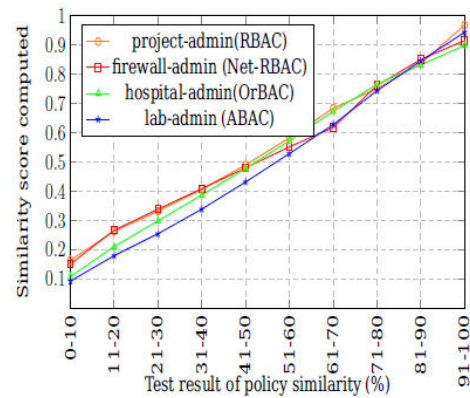


Figure 4 : expérience du score de similarité (set-8)

2.2 Expression et application de la politique de sécurité pour l'allocation de ressources virtuelles dans une infrastructure de type IaaS Cloud

2.2.1 Introduction

Aujourd'hui, le Cloud Computing est essentiellement fournisseur-centrique. Un nombre croissant des fournisseurs de services de Cloud proposent plusieurs Clouds hétérogènes. En termes de IaaS (infrastructure as a service), chaque fournisseur propose ses propres solutions pour les machines virtuelles (VM) des clients. De manière plus significative, dans le Cloud IaaS, le matériel physique est généralement partagé par plusieurs ressources virtuelles pour maximiser l'utilisation et la réduction des coûts.

Malheureusement, l'allocation des ressources virtuelles souffre d'un manque d'homogénéité : de nombreux ressources virtuelles de Cloud ne peuvent pas être déployées faute (1) d'expression unifiée et (2) d'interopérabilité. Le manque d'expression unifiée entraîne un « *vendor lock-in* » : les services sont étroitement couplés avec le fournisseur et dépendent de sa volonté de les déployer. Du manque d'interopérabilité découlent des services hétérogènes et, effet plus important, des ressources qui ne sont pas compatibles entre fournisseurs. Pour une meilleure interopérabilité et un meilleur contrôle, le courtage de Clouds tente aujourd'hui de mettre en œuvre une approche centrée sur l'utilisateur, qui peut être considérée comme un paradigme dans la prestation de ressources de Cloud (par exemple calcul, stockage, réseau). Avec l'aide de ce courtage, les besoins de sécurité de l'utilisateur seront nécessairement pris en compte dans le Cloud et ces exigences de sécurité peuvent être incluses dans le contrat SLA², est un document juridique où la description du service est formellement définie, livrée et facturée.

² Service Level Agreement

2.2.2 Framework pour allouer des ressources virtuelles

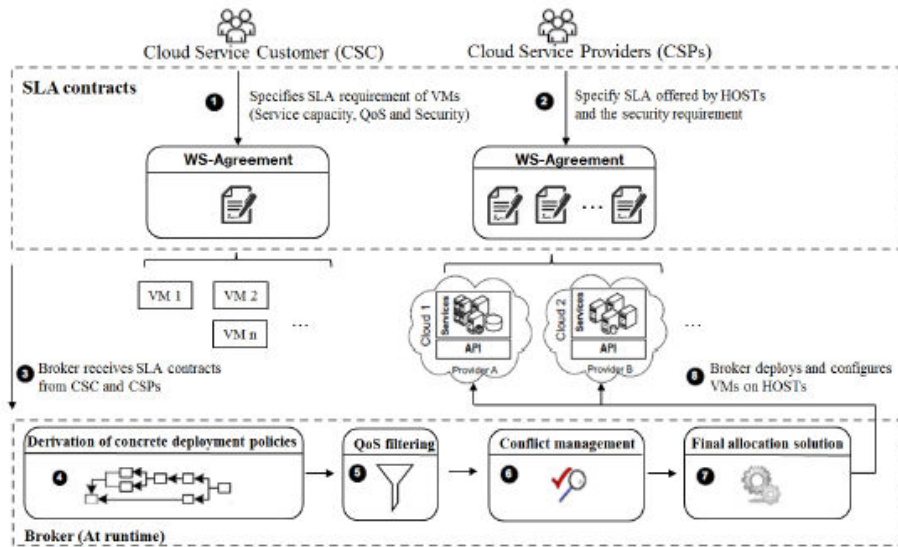


Figure 5 : Le Framework proposé pour allouer des ressources virtuelles

Comme montré dans la figure 5, au moyen de ces contrats à base de WS-Agreement, CSC et CSP précisent et gèrent leurs exigences de sécurité liées à l'infrastructure afin d'assurer la sécurité de bout en bout entre les différents composants (étapes 1, 2). Après avoir reçu les contrats de SLA, le Broker dérive les politiques de déploiement concrets selon les exigences de sécurité et autres (étapes 3, 4, 5). En particulier, le Broker est capable d'arbitrer les revendications contradictoires et de prendre des décisions (étape 6). A la fin, le Broker applique un algorithme pour générer la solution d'allocation finale (étape 7), puis déploie et configure les VMs sur les HOSTs (étape 8).

2.2.3 Implémentation

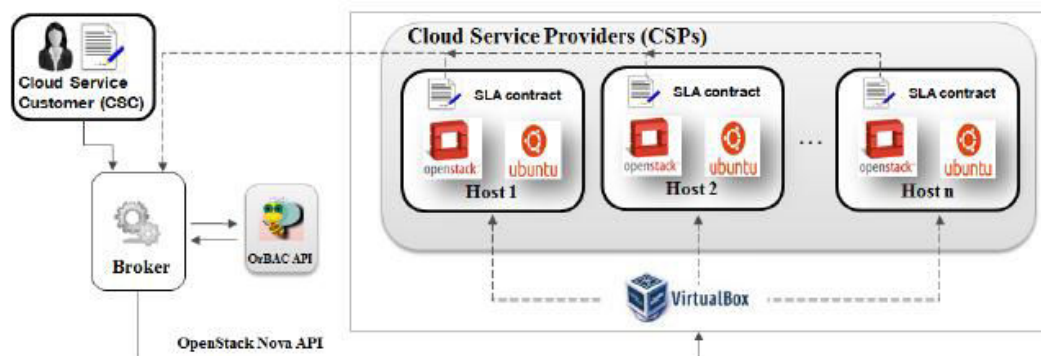


Figure 6 : implémentation pour l'allocation des ressources virtuelles

Afin de mettre en œuvre et d'évaluer notre Framework d'allocation des ressources virtuelles, nous configurons un environnement de IaaS Cloud sur une machine

physique (Intel (R) Core (TM) i7-4600U 2,7 GHz avec 16 Go de RAM sous Windows 7). Ensuite, différentes machines virtuelles (2 cœurs et 2 Go de RAM) sont créés sur la plate-forme VirtualBox avec le système Ubuntu. Nous installons DevStack Framework, une version d'OpenStack pour l'expérimentation. Chaque VM est considérée comme un hôte physique. En même temps, un programme Java fonctionne comme Cloud Broker et il se connecte à la plate-forme VirtualBox par le protocole SSH. La politique OrBAC est générée et gérée par l'API OrBAC basée sur Java. La figure 6 illustre notre architecture expérimentale.

3. Négociation de politiques de sécurité

3.1 Introduction

Dans la partie précédente, nous avons présenté des approches de sélection des fournisseurs de services en considérant la politique et l'exigence de sécurité. Après avoir sélectionné le fournisseur de services, client et fournisseur de services peuvent avoir besoin de négocier des politiques de sécurité concrètes. Dans le cas où le client n'a pas d'autres fournisseurs de services, il peut aussi avoir besoin de parvenir à un accord sur la politique de sécurité. Dans ce chapitre, nous proposons un Framework et un algorithme visant à négocier la politique de sécurité exprimée par exemple dans le modèle OrBAC. Le mécanisme de négociation se fonde sur une approche d'évaluation des politiques. Notre Framework fait appel à modèle de négociation qui part de la préférence indiscutable vers la préférence flexible. Nous ne donnons pas de définition de la stratégie de négociation mais nous supposons que la façon de choisir le prochain mouvement dans la configuration de négociation est prédéfinie. D'autre part, nous adoptons une approche pour la comparaison et l'évaluation des politiques de sécurité : le négociateur fait la proposition et évalue celle du partenaire. Différentes relations entre règles conduisent à des réactions différentes.

OrBAC (*Organization-based access control*) a été présenté pour la première fois en 2003. Dans OrBAC (Figure 7), l'expression d'une politique d'autorisation est centrée sur le concept d'organisation. Le modèle OrBAC reprend les concepts de rôle, d'activité, de vue et d'organisation. Chaque organisation définit ainsi les rôles, les activités et les vues dont elle souhaite réglementer l'accès en appliquant une politique d'autorisation. Les modèles de contrôle d'accès reposent habituellement sur les trois entités : sujet, action, objet. Pour contrôler l'accès, on spécifie si un sujet a la permission de réaliser une action sur un objet. Le modèle de contrôle d'accès OrBAC n'est pas restreint aux permissions. Il inclut aussi la possibilité de spécifier des interdictions et des obligations. OrBAC possède la notion de contexte, ainsi ses politiques de sécurité peuvent être exprimées dynamiquement. De plus, OrBAC

possède des concepts tels que la hiérarchie (organisation, rôle, activité, vue, contexte), la structuration d'entités et les contraintes de séparation.



Figure 7. Modèle de politique OrBAC

3.2 Comparaison des politiques OrBAC

Dans OrBAC, il est possible de considérer les relations d'héritage des rôles et aussi des activités, des vues et des organisations. Nous présentons relation d'héritage en utilisant les prédicats « sub_role », « sub_activity », « sub_view » et « sub_organization ». De plus, nous définissons aussi prédicats sub_context pour l'entité de contexte. Par exemple, sub_role(org, r_1 , r_2) indique que dans l'organisation *org*, rôle r_1 est une sous-entité de r_2 . Supposons que dans une entreprise, r_1 est un staff et il guide un stagiaire (r_2). Alors r_1 pourrait hériter toutes les autorisations de r_2 .

3.2.1 Relations entre les entités OrBAC

Nous disons que deux entités concrètes appartient à la même entité abstraite sont des « *related entities* ». Nous avons dérivé trois autres relations entre « *related entities* » de OrBAC e_1 et e_2 :

Equivalent : si e_1 est sémantiquement égal à e_2 , ils sont « *equivalent* ».

Relevant : si un élément est une sous entité de l'autre élément ou s'ils sont « *equivalent* », les deux éléments sont « *relevant* ».

Inconsistent : si e_1 et e_2 ne sont pas « *relevant* », ils sont « *inconsistent* ».

3.2.2 Relations entre les règles OrBAC

Les relations entre les entités OrBAC dérivent cinq relations entre les règles OrBAC r_1 et r_2 :

Restriction : si au moins une entité de r_1 est une sous-entité de r_2 et les autres « *related entities* » sont « équivalent », alors r_1 est une restriction de r_2 .

Total compatibility : si toutes les entités liées à r_1 et r_2 sont «équivalent », alors r_1 et r_2 ont une relation de « total compatibility ».

Symmetric compatibility : si toutes les « *related entities* » sont « relevant », et au moins une entité de r_1 est une sous entité de r_2 et au moins une entité de r_2 est une sous entité de r_1 .

Partial compatibility : si au moins une paire de « *related entities* » est « relevant » et il existe au moins une paire de « *related entities* » ayant une relation « inconsistent ». Dans ce cas, les règles ont relation de « partial compatibility » et elles ne sont pas comparables.

No compatibility : si toutes les « *related entities* » ont des relations « inconsistent », les deux règles ne sont pas comparables et ont relation de « no compatibility ».

3.3 Framework de négociation

Le Framework de négociation (Figure 8) comporte trois parties : protocole de négociation, configuration de négociation et module de négociation.

3.3.1 Protocole de négociation

Le protocole de négociation est compatible avec WS-Agreement : au début, le service demandeur demande l'offre de fournisseur de services. Après avoir reçu l'offre, le demandeur échange des propositions avec le fournisseur de services jusqu'à une décision (accepter ou refuser).

3.3.2 Configuration de négociation

Le module de configuration possède une architecture d'arbre. Deux types d'arbres sont introduits : « *related tree* » et « *distant tree* ». « *Related tree* » contient les règles avec trois types de relations: « *restriciton* », « *total compatibility* » et « *symmetric compatibility* », les règles de niveau supérieur sont préférées à celles de plus bas niveau. La proposition sur une règle reçue dépend de la stratégie de recherche. Par exemple, une règle reçue qui est plus stricte que celle locale sera acceptée ; en revanche pour une règle reçue qui est moins stricte, une autre règle de niveau inférieur sera envoyée en tant que contre-offre. Le deuxième type d'arbre est « *distant tree* » qui contient des règles avec deux types de relations : « *partial compatibility* » et

« *no compatibility* ». « *Distant tree* » peut être utilisé dans le scénario où plusieurs règles sont négociées en même temps. La proposition d'une règle peut déclencher la proposition d'une autre règle dans son « *Distant tree* ».

3.3.3 Module de négociation

Le module de négociation prend en charge l'application du protocole de négociation. Il reçoit une proposition de règle et évalue la relation entre la règle reçue avec celle de locale. Après l'exécution de l'algorithme de négociation, une contre-offre ou une décision finale sera prise et envoyée.

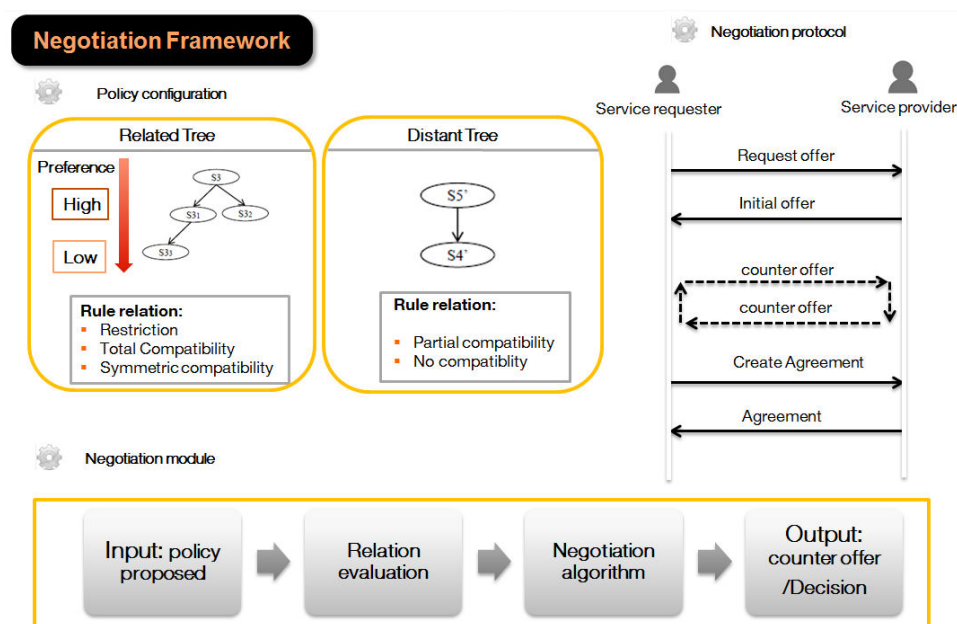


Figure 8 : Framework de négociation

4. Conclusion et perspectives

4.1 Conclusion

Le mode de service émergent avec de multiples fournisseurs de services apporte plus de flexibilité et d'efficacité pour les clients dans leur choix de services. Dans le processus d'échange de données et de services, la politique de sécurité joue un rôle fondamental dans la gestion des privilèges. Avec la politique de sécurité, les acteurs sont capables d'exprimer leurs propres privilèges et spécifier les restrictions d'autorisations pour des visiteurs. Par exemple, il peut être appliqué pour définir le contenu de service pour le véhicule communicant, ou d'exprimer des préférences de client et de fournisseur de services dans l'allocation des ressources virtuelles. Bien que la politique de sécurité offre plusieurs avantages, son évaluation et sa négociation présentent encore de nombreux défis, en particulier parmi les différents modèles de politique de sécurité et entre les négociateurs avec leurs propres préférences. Dans ce

contexte sensible, le premier objectif de cette thèse est de fournir une méthode générale pour la mesure et l'évaluation des politiques de sécurité, utile dans la sélection des fournisseurs de services. Le score de similarité permet de quantifier le niveau de similarité entre deux politiques de sécurité, comme nous l'avons montré dans nos travaux de recherche. Cependant, les fournisseurs de services ne divulguent pas toujours leurs politiques de sécurité, ce qui nuit à leur évaluation. Dans le cas où les politiques de sécurité ne sont pas exprimées explicitement, nous avons proposé un Framework qui dérive les politiques de sécurité à partir des exigences de sécurité. Le Framework proposé est utilisé dans un scénario où des machines virtuelles sont attribuées dans une infrastructure IaaS. Le deuxième objectif de cette thèse est d'introduire le processus de négociation de la politique de sécurité entre deux acteurs visant à parvenir à un accord sur la politique de contrôle d'accès.

4.2 Perspectives

Nous donnons un ensemble de futures directions de recherche qui pourraient être étudiés comme suite aux résultats présentés dans cette thèse.

4.2.1 L'intégration de la technologie de PSM dans la négociation de la politique de sécurité

Comme le score de PSM présente le niveau de similarité entre deux politiques, il peut être utile dans le processus de négociation de la politique de sécurité. Après l'introduction de ce score de PSM, les relations entre les politiques de sécurité ne sont pas seulement classées mais aussi quantifiées. Dès lors, des stratégies peuvent être exécutées en fonction du score de PSM. Par ailleurs, la prise de décision peut également faire appel au score de PSM et affiner ainsi le contrôle du processus de négociation.

4.2.2 L'introduction de la politique contextuelle dans l'allocation des ressources virtuelles:

Notre solution actuelle d'allocation des ressources virtuelle est appliquée à une politique OrBAC possédant «*default*» comme contexte. Un contexte est considéré comme une condition supplémentaire qui doit être satisfaite pour activer une règle de sécurité. La capacité d'exprimer la condition de contexte permet d'intégrer des exigences (temps, espace, histoire...) contextuelles dans les modèles de WS-Agreement puis ces exigences dont peuvent être dérivées des politiques OrBAC .

4.2.3 Améliorer l'interopérabilité entre les modèles politiques différents au cours de la négociation politique.

Comme les différents modèles de politiques de contrôle d'accès ont leurs avantages et leurs limitations, les utilisateurs peuvent utiliser différents modèles pour spécifier les privilèges. Lorsque les politiques appartiennent à différents modèles, l'interopérabilité devient une difficulté à surmonter. Bien que certains travaux ont proposé d'utiliser le modèle ABAC pour unifier les modèles DAC, MAC et RBAC, des travaux de recherche doivent être menés pour (i) l'unification de plusieurs modèles de politiques; (ii) le développement de moteurs interopérables de politique et leur intégration dans la négociation de politique.

4.2.4 L'extension du Framework d'allocation des ressources aux plusieurs scénarios.

Notre Framework d'allocation à base de politique peut être utilisé pour le scénario de déploiement de VMs dans le domaine du Cloud Computing. En outre, notre solution peut être utilisée dans d'autres contextes tels que le stockage et le réseau. En ce qui concerne l'aspect stockage, le Framework permet aux utilisateurs d'exprimer leurs exigences de sécurité pour leurs données; en même temps, les fournisseurs de services qui offrent leur espace de stockage peuvent également spécifier leurs préférences sur les caractéristiques des données. Avec notre Framework, les données de l'utilisateur sont stockées par des fournisseurs de services distribués. Une autre application possible dans le réseau est la possibilité donnée à l'utilisateur d'exprimer ses besoins de routage de trafic auprès du contrôleur SDN (software defined networking). Le contrôleur sélectionne alors le chemin de routage entre les commutateurs en considérant la préférence de sécurité du fournisseur de services.

Résumé

Suite au développement des technologies de l'information, et en particulier au déploiement d'infrastructures telles que le Cloud Computing, de plus en plus d'applications et plateformes coopèrent en échangeant des données et des services. Cette tendance renforce l'importance de la gestion de la sécurité. Afin d'assurer la sécurité des données et de l'interaction de service une politique de sécurité doit être appliquée. Dans cette thèse, nous nous intéressons aux politiques de contrôle d'accès. Ce type de politique spécifie les privilèges de l'utilisation des ressources et est implémentée par différents modèles selon différents scénarios. Notre objectif ici est d'aider le client du service à bien exprimer ses exigences de sécurité et à choisir les fournisseurs de services qui peuvent la déployer.

La première partie de cette thèse est dédiée à la sélection des fournisseurs de service. Dans le cas où les politiques de sécurité du fournisseur sont accessibles au client, nous proposons une méthode pour mesurer la similarité entre les politiques de sécurité. Dans le cas où les politiques de sécurité ne sont pas accessibles au client ou ne sont pas explicitement spécifiées, nous proposons un cadre à base de règles permettant la dérivation à partir des exigences de sécurité aux politiques de sécurité concrètes.

La seconde partie de la thèse porte sur la négociation de politiques de sécurité. Nous étudions le processus permettant aux parties en négociation de parvenir à un accord par une série d'échanges d'offres et de contre-offres. Lorsque le résultat de la négociation est positif, un contrat incluant la politique de sécurité acceptée par les parties est généré.

Mots-clés : Politique de sécurité, Evaluation, Informatique en nuage, Négociation, Interopérabilité, Contrôle d'accès, Fournisseur de services, Courtage

Abstract

Security policy provides a way to define the constraints on behavior of the members belonging to a system, organization or other entities. With the development of IT technology such as Grid Computing and Cloud Computing, more and more applications and platforms exchange their data and services for cooperating. Toward this trend, security becomes an important issue and security policy has to be applied in order to ensure the safety of data and service interaction. In this thesis, we deal with one type of security policy: access control policy. Access control policy protects the privileges of resource's utilization and there exist different policy models for various scenarios. Our goal is to ensure that the service customer well expresses her security requirements and chooses the service providers that fit these requirements.

The first part of this dissertation is dedicated to service provider selection. In case that the security policies of the service provider are accessible to the service customer, we provide a method for measuring the similarity between security policies. Another case is that security policies are not accessible to the service customer or not specified explicitly. Our solution is proposing a policy-based framework which enables the derivation from attribute-based security requirements to concrete security policies.

The second part of the dissertation focuses on the security policy negotiation. We investigate the process of reaching agreement through bargaining process in which negotiators exchange their offers and counter offers step by step. The positive result of the negotiation generates a policy contract.

Keywords : Security policy, Evaluation, Cloud computing, Negotiation, Interoperability, Access control, Service provider, Broker