Decentralized Authorization for Inter-domain Collaborations with *i*RBAC Framework

Vom Promotionsausschuss der
Technischen Universität Hamburg-Harburg
zur Erlangung des akademischen Grades

Doktor Ingenieurin (Dr.-Ing.)

genehmigte Dissertation

von

Hannah Kyungsun Baker

aus

Seoul, Republik Korea

2011

Gutachter:

Prof. Dieter Gollmann

Prof. Joachim Posegga

Tag der mündlichen Prüfung:

22.12.2011

Hamburg University of Technology

Security in Distributed Applications

`http://www.sva.tu-harburg.de/`

Harburger Schloßstraße 20

21079 Hamburg

Germany

To my parents

Declaration

I, Hannah Kyungsun Baker, solemnly declare that I have written this Dissertation independently, and that I have not made use of any aid other than those acknowledged in this Dissertation. Neither this Dissertation, nor any other similar work, has been previously submitted to any examination board.

Hamburg, May 7, 2012

Hannah Kyungsun Baker

Abstract

Inter-domain collaborations are composed of a series of tasks, whose run-time environment stretches over heterogeneous systems governed by different sets of policies. Though the collaborators are willing to allow access to their services and resources from outside of their administrative domains in order to reach the common goals of collaborations, they still desire to retain control over deciding under which conditions their resources should be available and which internal information to disclose to their collaborators. For instance, personally identifiable information (PII) is one of the sensitive data collaborators wish to protect. However, more often than not PII is a prerequisite to granting access permissions when the user is from a different administrative domain. Within business-to-business (B2B) collaboration scenarios, however, information on the person's role within the collaborations is actually more relevant than the PIIs.

In order to satisfy autonomous administrations of collaborators including protection of end-users' PIIs, it is essential that authorization policies and mechanisms do not require exchanges of PIIs. Moreover, the administrations of the security policies are to be operated in a decentralized manner. This then opens up requirements of interoperability and scalability of an authorization solution for inter-domain collaborations in terms of policy specifications, enforcement, and administrations.

This thesis proposes an authorization solution that presents not only a *model* that provides a logical structure of security policies but also a *methodology* to guide the construction process of the policies and a set of *modules* that allows decentralized, autonomous administration of the security policies throughout their lifetime. Together these components render an authorization solution to collaborative systems built with standard protocols. In summary, the three components are:

- **Model**
  is elaborated from role-based access control (RBAC) [ANS04] with an additional indirect layer called *interactive* Roles (*i*Roles). This layer abstracts authorized end-users within collaborators' local domains. By introducing the additional indirection, *i*Roles enable autonomous policy administration on user–role and role–permission assignments. Equally importantly, it provides transparent linkage between local end-users in collaborators' domains and functional roles in collaboration definitions.

- **Methodology**
  is a step-by-step guideline that provides a process of building security policies based on this model. The end result of the process is a security policy built in terms of definitions of *i*Roles in the format of an eXtensible Access Control Mark-up Language (XACML) [Mos05] standardized policy, which is one of the most commonly used standards by the reference monitors.

- **Modules**
  is a tool-set, also called the *interactive* RBAC (*i*RBAC), which enables decentralized operations of autonomous administration of the security policies built according to the methodology.

A unique contribution of this work is its full coverage of support for the different stages of the life-cycle of the security policies for inter-domain collaborations. Though a case study has been conducted, a real-time scaled deployment is yet to be applied. Additionally, suitable methods of evaluating this work are still to be established. Nonetheless, this thesis presents the bridging elements of an authorization solution to fill the gap between the works of research communities and the needs of the real world scenarios of inter-domain collaborations not in parts but as a whole composition.

Abstrakt

In Inter-Domain-Kollobarationen werden Aufgaben über Domänengrenzen hinweg bearbeitet. Obwohl die Partner bereit sind, Zugang zu Diensten und Ressourcen von außerhalb ihrer administrativen Domäne zu erlauben, wollen sie die Kontrolle darüber behalten, unter welchen Bedingungen ihre Ressourcen und internen Informationen den Partnern zur Verfügung gestellt werden. Zum Beispiel gehören Persönlich Identifizierbare Informationen (PII) zu den sensitiven Daten, welche die Partner schützen wollen. Doch oft ist PII eine Voraussetzung für die Gewährung der Zugangsberechtigungen für Benutzer aus einer anderen administrativen Domäne. Innerhalb von Business-to-Business (B2B) Kollobarations Szenarien ist Information über die Rolle der Person innerhalb der Kollobaration wichtiger als die PII.

Für die autonome Verwaltungen der Partner einschließlich des Schutzes der Endnutzer PII ist es unerlässlich, dass die Autorisierungsrichtlinien und Mechanismen nicht den Austausch von PII erfordern. Außerdem ist die Verwaltung der Sicherheitsrichtlinien dezentral zu betreiben. Dies führt zu Anforderungen an die Interoperabilität und Skalierbarkeit einer Autorisationslösung für Inter-Domain Kollobarationen im Bereich der Spezifikation, Durchsetzung und Verwaltung von Sicherheitsrichtlinien.

In dieser Arbeit wird eine Lösung zur Autorisation vorgeschlagen, die ein *Modell*, d.h. eine logische Struktur von Sicherheitsrichtlinien, eine *Methodik* für die Erstellung von Sicherheitsrichtlinien, und eine Reihe von *Modulen*, welche eine dezentrale, autonome Verwaltung der Sicherheitsrichtlinien über deren gesamte Lebensdauer ermöglichen, beinhaltet. Zusammen bilden diese Komponenten eine komplette Lösung zur Zugriffskontrolle auf der Grundlage von Standard-Protokollen.

- Modell
  Das Modell basiert auf dem Role-Based Access Control (RBAC) Modell mit einer zusätzlichen Zwischenebene, den sogenannten *interactive Roles*. Diese Schicht abstrahiert den autorisierten Endnutzer in den lokalen Domänen des Partners. Die Einführung von *i*Roles ermöglicht Sicherheitsrichtlinien autonom mittels Benutzer–Rolle und Rolle–Berechtigungszuweisungen zu verwalten. Zudem unterstützt das Framework die transparente Verknüpfung zwischen den lokalen Endbenutzern in ihren Domänen und den funktionalen Rollen in den Definitionen der Kollaborationen.

- Methodik
  Die Methodik ist eine Schritt-für-Schritt-Anleitung, welche einen Prozess zum Aufbau von Sicherheitsrichtlinien basierend auf dem Modell bereitstellt. Das Endergebnis des Prozesses sind die Sicherheitsrichtlinien, geschrieben in der eXtensible Access Control Markup Language (XACML).

- Module
  Die Module führen die autonome, dezentrale Verwaltung der Sicherheitsrichtlinien aus, die nach der Methodik erzeugt wurden.

Der Beitrag dieser Arbeit ist die vollständige Abdeckung der Unterstützung für die verschiedenen Phasen des Lebenszyklus der Sicherheitsrichtlinien für Inter-Domain Kollaborationen. Obwohl eine Fallstudie für die Lösung durchgeführt wurde ist, muss der Lösungsansatz noch in der Praxis umgesetzt und evaluiert werden. Nichtsdestotrotz stellt diese Arbeit einen ersten Schritt dar, um die Lücke zwischen der Forschungsresultaten und der Praxis der Inter-domain Kollaborationen zu schließen.

Contents

1. Introduction

So uncritically do we accept the idea of property in culture that we don't even question when the control of that property removes our ability, as a people, to develop our culture democratically.
– Lawrence Lessig

An ever increasing demand for collaboration across organizational boundaries is prevalent in today's digital world. From e-Business to e-Health, e-Learning, and e-Government sectors, with thriving globalization, cross-organizational interactions between business to business (B2B) and administrator to administrator (A2A) are nowadays rather mandatory than optional. Take the aircraft industry as an example. No aircraft manufacturer produces all of its components in-house but rather works together with a number of different suppliers, sub-contractors, airlines, and authorities in order to build an air plane. In fact, it is not an exceptional situation for such large-scale manufacturers only. Incentives for cost reduction and production efficiency apply to car manufacturers, supermarket chain stores, on-line travel booking firms, hospitals, schools, and public sectors alike through cross-organizational collaborations. According to an UN survey in 2008, the need for connected governance across borders has been highlighted as a major challenge of current e-Government, which has been already emphasized by the OECD (Organisation for Economic Co-operation and Development) in 2007 [un208]:

> "While initially the political and managerial focus was on developing e-services within each public institution, with limited consideration being given to cross-organizational coherence, the focus today has clearly shifted towards **coordinated services** offering one-stop shops to citizens and businesses."

What must take place in order to realize those "coordinated services offering one-stop shops" is the framework enabling B2B and A2A collaborations across organizational boundaries or in other words administrative domains. This is the very subject of this thesis: inter-domain collaborations. Though their purposes may vary from sharing resources to completing a certain process; commonly, all inter-domain collaborations consist of a series of tasks, whose run-time environment stretches over heterogeneous systems governed by different sets of policies and where participating organizations desire to preserve control over their resources. Such collaborations have distinct characteristics in that:

- there is potentially a **large number of collaborators** while
- each of them should be **autonomous** in terms of handling their security policies, and that
- **privacy** of local principals is an important factor to be preserved.

Consequently, one of the major security challenges in modelling those inter-domain collaborations is providing an authorization solution that enables autonomous administrations with privacy enhancement without compromising security principle, i.e. only authorized access is granted. At the core of this challenge lie two problems: if the principal originates from outside of the decision maker's domain, 1) an authorization decision maker does not know who this principal is and 2) which set of privileges this principal owns. In order to answer these questions correctly, it follows logically that not in parts but the entire life-cycle of authorization policy is to be considered. Nonetheless, it is surprisingly difficult to find a solution that addresses the entirety of the life-cycle of security policies within the context of inter-domain collaborations.

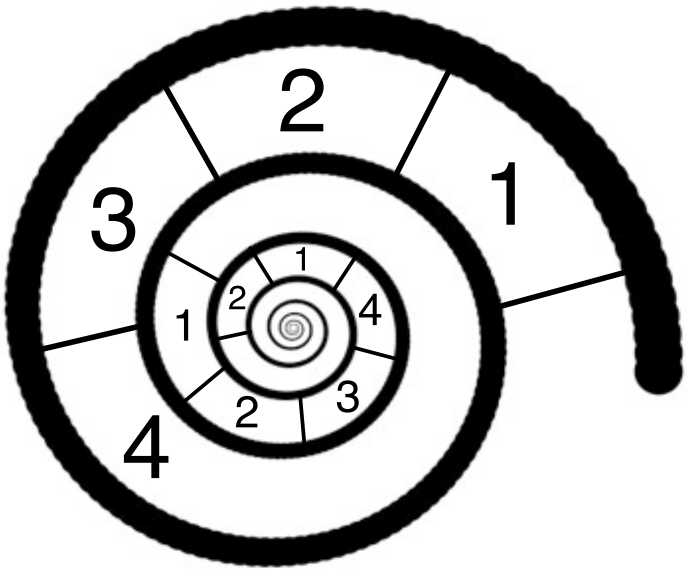


Figure 1.1.: Life-cycle of Security Policies of Inter-domain Collaborations

A general life-cycle of a security policy of inter-domain collaborations is shown in Figure 1.1 where four different phases are repeated. Each phase is explained as below:

1. Creation / Change
   This is the phase when the security policy for inter-domain collaborations is initially specified and stored in a repository to which all the involved collaborators have access. Changes can take place during the lifetime of the policies, and if so, the involved collaborators are notified.

2. Deployment
   Security administrators from the collaborators can access the common repository and deploy parts or the whole security policies to their own local domains. They deploy the related parts of the policies required to make access control decisions during the collaboration run-time.

3. Assignment
   The deployed policies may still be specified further to decide under which conditions they are to be executed. For instance, the authorization conditions to the access permissions may still

need to be assigned to particular groups of local principals or roles. It may also be the case that the policies need to be connected to the local access control policies related to particular sets of resources inquired.

4. Enforcement
   In this phase, the security policies are in use by the collaborative system as collaborations are executed. Collaborators would refer to the policies in order to make authorization decisions.

In this thesis, an authorization model using an abstraction layer called ***interactive roles*** is proposed for the inter-domain collaborations, addressing the two aforementioned factors. This model enables cross-domain authorization based on the given collaboration specifications without compromising scalability as the number of collaborators increases. Secondly, a methodology of extracting privileges and roles from collaboration definitions is introduced so that the resulting set of *i*Roles can enable preservation of collaborators' control over their own local resources including ensuring privacy of their local principals. A framework called ***iRBAC*** is also designed in order to update authorization policies corresponding to the *i*Role model.

1.1. Motivating Scenario

Consider a cross-border scenario of an e-Government collaboration amongst the police departments of EU member states. There is a cooperation agreement amongst European police forces which makes it possible, when a driver is caught with a severe violation of driving rules, to check his or her record in his or her home country. In 2004, 8,000 German drivers were caught in France with major violations; nonetheless, only five were checked against the German database as the procedure was complex and the incentives for authorities non-existent [TA2]. Such a level of control is justified and needs to be made easier.

Now consider what will be required in order for a French police officer to be able to obtain the right set of privileges in order to perform his responsibility. From the French police officer's perspective, he would contact the German police department, prove that he is indeed a valid EU police officer possibly by presenting any information requested, and then would be able to inquire of the record of the driver caught. From the German police department's point of view, it can be trickier. What is the information legitimate to ask to make an authorization decision, and in what form should the criteria and the verdicts be incorporated with the rest of their security policy?

The simplest case would be treating any outside inquiries as exceptions. In this way, the German police department does not have to create any new security policy for an outside access to their database. In that case, based on a proper manual review against the cooperation agreement and the information gathered from the inquiring party, they can either grant or reject on this specific occasion.

Since there is an ongoing cooperation agreement, however, it can very well happen again; in fact, it can be quite frequent. Then, it would be useful to specify a specific policy for the outside police officers from the other EU member states. A quick way of implementing it would be that each police department in the 27 EU member states individually specifies similar access control policies for the police officers from the other 26 EU member states according to their local policy specification methods. Consequently, if a new EU member state joins or if there are any changes in their agreements that result in changes in the way of their collaborations, all of them must modify their local policies accordingly. Obviously, this method does not scale in a sense that an increase in the number of collaborators or collaborations demands modifications of security policies in each participating collaborator's domain. In short, the sum of the modification of the security policies grows polynomially against the number of participating collaborators.

Could there be a more efficient way of specifying policies that will yield more scalable security policies? If so, are there any policy specifying methodologies, security policy models, or implementations available already? Do they sufficiently satisfy the essential requirements to be applied in the context of inter-domain collaborations? These are some of the key questions that have brought this thesis into life.

1.2. Aims of the Thesis

The aim of this thesis is to identify the authorization requirements of large-scale inter-domain collaborations, research on the current state of art in terms of meeting the fore-stated questions, and to propose an authorization solution for heterogeneous enterprise-based inter-domain collaborations satisfying the following properties:

Autonomy enables collaborators to be in charge of their own local resources without revealing unnecessary internal information

Privacy refers to respect of the end-user's personal identification information.

Scalability of methodology of specifying security policies resulting in reduced administrative efforts of the policies in case of the increase of its subjects—number of collaborators and collaborations.

Interoperability enables different collaborators using different IT infrastructures to be able to apply security policies in a consistent manner.

Having looked into the different authorization models, we have come to the point that Role-Based Access Control (RBAC) is a suitable fit as authorization policy model for the real-world inter-domain collaboration scenarios. Consequently, we thus have investigated further in the following areas in order to reach our aim to provide an appropriate authorization solution for inter-domain collaborations:

- Methodology of role formation for inter-domain collaborations
- Model of enabling decentralized, autonomous policy administrations
- Architecture of a framework in which the authorization policies can be developed

Authorization is placed somewhere between the processes of user authentication and access control. User authentication in a collaborative system is often called user provisioning as in single-sign-on systems. Access control is defined differently by different research groups [Gol11], but at the core of it lies a mechanism of making a verdict whether a subject is allowed to perform a certain operation upon a given resource. The authorization policy we look into in this thesis would be the very policy an access control mechanism will rely on in order to make an access control decision. There are closely related areas of study to the authorization challenge of inter-domain collaborations, but they themselves are complex problem domains; and, thus they are out of scope of this thesis work. They are listed as follows:

- User authentication: is to be taken care of either by local security policies or by a collaborative system reliant on the agreement collaborators have prior to the execution of the collaborations.
- Designing collaborations: is out of scope of this thesis. Though the *i*RBAC framework depends on the collaboration definition specifications to derive *i*Role definitions, this work assumes the existence of the collaboration definition as input. However, we foresee that once the *i*Role-based authorization policies become mature enough, collaboration definitions can utilize *iRole* definitions as their actors.
- Collaboration system: refers to a security policy enforcement system. In this case, it refers to a system that enables cross-organizational collaborations. As for the testing and verifying purposes of the usability of the *i*Role-based authorization policies, a web-service based collaborative system has been employed together with the XACML architecture of making the access control decisions. However, we speculate that any collaboration system that can process XACML authorization requests should be able to use the resulting authorization policies derived from the methodology we propose here.

Ideally, a robust authorization system can be plugged in to any collaborative systems and work comparably with their choices of user authentication mechanisms. This we strive to accommodate by usage of standards and interoperable solutions.

1.3. Approach Chosen

For inter-domain collaborations formed amongst long-term affiliations, deploying an RBAC-based authorization policy is a more efficient choice than solely relying on a credential-based solution such as in SPKI[1]. The specification of role privileges in an RBAC system eliminates the repetition of issuing authorization credentials. Moreover, most of the collaboration definitions available nowadays include notions of role or actors such as in Web Service Choreography Description Language[2] and Web Service Business Process Execution Language[3]. In addition to the general administrative benefit RBAC offers,

[1]http://www.ietf.org/rfc/rfc2693.txt
[2]http://www.w3.org/TR/ws-cdl-10/
[3]http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

reference to role instead of local principal provides collaborators with transparency of end-users in the context of inter-domain collaborations. Ferraiolo mentions the following characteristics of large-scale enterprises or organizations as indications of suitable environments for deploying the RBAC model [FCK95]:

- Large number of users and frequent change in job responsibilities and/or functionalities.
- Only a small number of security administrators are provided.
- Large number of protected objects to be shared among users based on their job functions.
- Protected resources are owned by organization, not users. Access control policy is of individual organization and enforced by its security administrators.

Large-scale enterprise inter-domain collaborations certainly fit into the descriptions of the suitable environment of RBAC as described above. Especially, it has a large number of users and frequent changes in job responsibilities and/or functionalities dependent upon which member state is involved in a certain collaboration. Moreover, the protected resources are often owned by organizations and not by the end-users.

The role in the Role-based Access Control (RBAC) model [ANS04] is the core abstraction between users and permissions that reduces the management effort of mapping the two ends. If the mappings between roles and permissions do not change as often as the ones between users and roles, then an RBAC system gains benefits from reusability as well as scalability—reusability from assigning the sets of roles to different users and scalability from deploying the role and permission sets to other environments with similar authorization systems. Nevertheless, all of these potential advantages can be realized only if the roles and the associated assignments to permissions and to users are well-developed.

In spite of the numerous RBAC-based models, we argue that the need still exists to build an RBAC system for inter-domain collaborations such that it enables autonomous administration of collaborators' own resources and provides transparency of local principals. Another unfulfilled need is to provide a method to map roles in organizational context to functional ones in collaboration processes. The survey conducted by Fuchs and Preis [FP08] supports this point of view in saying that the current role models fail to provide the means of taking business requirements and other role perceptions into account. They examined thirteen different RBAC extension models in terms of coping with fifteen different properties of roles in three different perspectives, namely Business, Role, and IT. Their resulting chart shows how the majority of the models define roles only with the exclusive properties of their own interest and that "[they] are not feasible for their usage in IdMIs"[FP08].

Arguing that a similar problem holds for inter-domain collaborations, we propose an RBAC extended model with *interactive roles* which are derived from their functions in the collaborations. The resulting set of roles creates an additional indirect layer between the actors of collaborative definitions and the principals within collaborators' domains so that it can be avoided to directly map local principals and actors of the collaboration definitions.

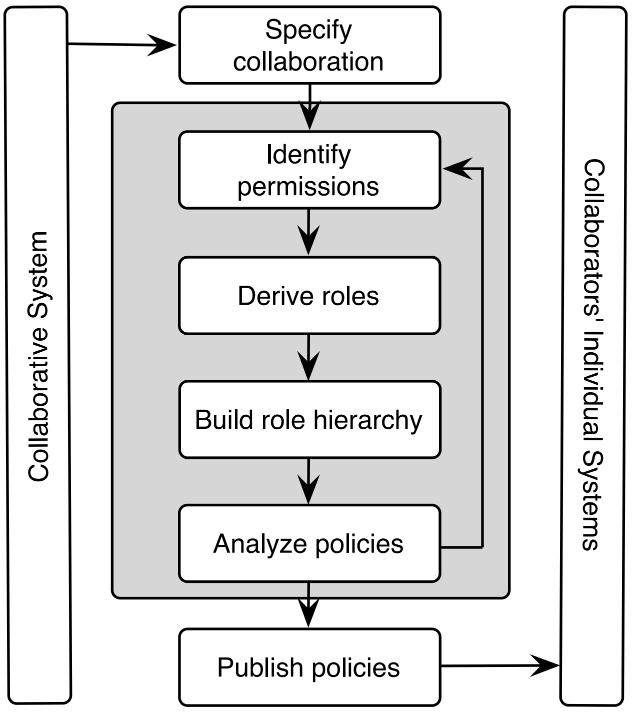


Figure 1.2.: Overview of the *i*Role-based Policy Development Methodology

The creation of *i*Roles follows a methodology of building such security policies in a decentralized autonomous manner. Figure 1.2 shows the overview of the methodology. The steps for role creation are boxed in the middle within the grey-colored rectangle. It takes the collaboration specifications as an input and produces the policies that can be viewed by collaborators. The resulting policies represent roles with their permission sets. Given the definitions, assignments of user–role can be done independently within each collaborator's system.

The framework called *i*RBAC, which builds an RBAC system with the *i*Roles, is also designed. It offers necessary functionalities of policy managements such as decentralized autonomous user–role assignments and querying of *i*Roles by collaborative administrators. The *i*Role layer makes it possible even for collaborators without RBAC-based security policies to endorse the collaborative security policies without introducing an RBAC system into their own domains. The framework also discusses the generic problems of building an RBAC system over the heterogeneous systems such as preserving the consistency property.

1.4. Structure of the Thesis

This thesis is structured following the OM-AM (Object, Model–Architecture, Mechanism) framework which has originated from the long-standing objective-mechanism distinction in the security literature [San00]. Aiming at providing assurance between different components of a system, the framework consists of four layers called *Objective*, *Model*, *Architecture*, and *Mechanism* from top to bottom, as depicted in Figure 1.3. The first two layers on the top answer a question of "what" while the two layers at the bottom explain that of "how." The answer to "what" is dealt with in Chapter 3 while the

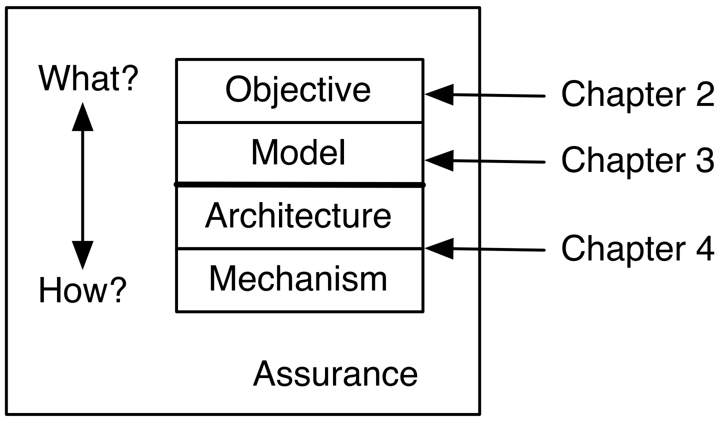


Figure 1.3.: OM-AM Framework

objectives are drawn from the survey of the current state of the art of security solutions for inter-domain collaborations in Chapter 2. In Chapter 4 the concrete architecture of modules to implement the model stated in Chapter 3 is explained. Afterwards, a case study of using the model and the tool-set called *i*RBAC is presented in Chapter 5. Chapter 6 compares others' works with our solution; and, finally, concluding remarks are provided in Chapter 7. Below is the chapter by chapter summary:

Chapter 2 scrutinizes inter-domain collaborations in a three-folded way. First of all, it shows the overview of how such collaborations work. By doing so, the scope and the nature of inter-domain collaborations have been clarified; their distinct characteristics are pinpointed. Secondly, the state of art of inter-domain collaborations is illustrated from a business perspective. This has been done by introducing five different e-Government use cases provided by the EU project, R4eGov[4]. Lastly, the state of art from the technical perspective has been introduced. Surveying the existing technologies that enable inter-domain collaborations as well as the security solutions currently available, the paradigms chosen and the technical trends are discussed. It then closes with the remaining security challenges for inter-domain collaborations to thrive.

Chapter 3 introduces the *interactive Role* model as well as how it is described in the format of eXtensible Access Control Markup Language (XACML) and its RBAC profile including role hierarchy. It also discusses a possible security violation in the case where *i*Roles are used as local roles in the context of preserving the consistency property.

Chapter 4 explains the *i*RBAC framework which consists of a component to build an RBAC system by role engineering schemes and another component dedicated to the required administrative operations and communications amongst collaborators' administrators. It further demonstrates an implementation of the framework suitable for web-service based inter-domain collaboration systems.

Chapter 5 demonstrates a case study of building an RBAC system with the *i*RBAC framework in Europol and Eurojust collaborations. It begins to explain the backgrounds of their cooperation and the existing agreements amongst the involved collaborators. Then, the basic units of their scenario compositions are introduced. Afterwards, the intermediate results of following the methodology of

[4]EU Commission contract number: IST-2004-026650, Towards e-Administration in the Large

building *i*Role-based policies are presented. After presenting the summary of the case study, the chapter ends with discussions on the lessons learned. Issues and open questions confronted during the case study undertaken are being explained such as how to judge what are “good roles” and what are the advantages and disadvantages of using the standard, eXtensible Access Control Mark-up Language (XACML).

Chapter 6 discusses two major fields of work related to this thesis: the first one is authorization models used for distributed systems while the second one is regarding role engineering. In the latter area, different approaches of role engineering methodologies are compared to the approach we have chosen.

Chapter 7 summarizes the thesis contribution, discusses the limitations, and closes with the direction of further work.

2. Preliminaries

No man is an island.
– John Donne

In this chapter, inter-domain collaborations are surveyed in a three-folded way. First of all, the scope and the nature of inter-domain collaborations are clarified; by doing so, their distinct characteristics are pinpointed. Secondly, the state of the art of inter-domain collaborations is illustrated from a business perspective. This has been done by introducing five different e-Government use cases provided by the EU project, R4eGov[1]. Lastly, the state of the art from a technical perspective is presented. Illustrating the existing technologies that enable inter-domain collaborations as well as the security solutions currently available, their technical approaches are explained. The chapter then closes with the remaining security challenges for inter-domain collaborations to thrive.

2.1. Overview of Inter-domain Collaboration

According to Clark and Jones' Organizational Interoperability Maturity Model, C2 [CJ99], inter-domain collaborations can be categorized into five different levels from zero collaborations to a complete unification. They are denoted as follows: *Independent (0)*, *Ad-hoc* (1), *Collaborative* (2), *Integrated* (3), *Unified* (4) [CJ99]. While the lowest level represents an isolated environment without interoperability and the highest level a complete universal interoperation, the three levels in the middle show different maturity levels of interoperability. The *Ad-hoc* level is characterized with peer-to-peer connections with limited organizational framework with liaison officers as the main means of information exchange whereas the *Collaborative* level is characterized by a recognized framework to support interoperability together with allocated roles and responsibilities to reach a common goals. However, it still represents a collection of functional coalition operations and that the components are still in a distributed manner [CJ99]. The framework of the *Integrated* level is more mature than that of the *Collaborative* level with "shared value systems and shared goals, a common understanding and a preparedness to interoperate" [CJ99]. The *Unified* level demonstrates similar characteristics as the *Independent (0)* level except it consists of a heterogeneous environment complying to a uniform set of regulations and infrastructures.

The maturity level of the e-Government collaborations addressed in this thesis can be best described as inter-domain collaborations at the *Collaborative* level of Clark and Jones' model. Level 0 and Level 4 are out of question since their environments are monolithic or artificially homogeneous; moreover,

[1]http://www.r4egov.eu/

collaborators may not be willing to take this level of uniformity for non-technical reasons. For instance, it is rather impossible to impose an organization to use a certain application or appliance in order to participate in an inter-domain collaboration.

Level 1 type collaborations are beneficial for a short-term dynamic coalition relying on a mutually trusted broker model. Nonetheless, this type of collaborations is not efficient in the long-term due to the repetitious operations of establishing initial trust and service agreements. Level 3, on the other hand, presumes shared value systems and a single line of reporting, which would be beyond the scope of e-Governmental collaborations. This maturity level of interoperability can be found in the interactions within a company whose branches are in different locations perhaps even with different IT infrastructures. The type of collaborations reflected by Level 2 demand significant amount of communication and sharing of "knowledge about the situation or context" [CJ99] amongst collaborators; and, their interactions are dynamic and flexible. Thus, such collaborations often rely on an over-arching framework with support of interoperability and allocation of roles and responsibilities [CJ99].

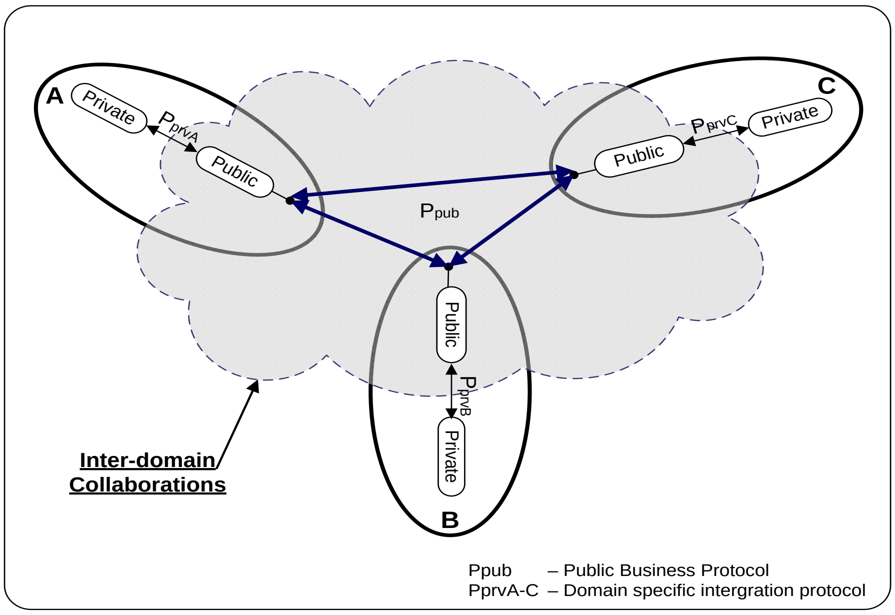

A
Private
PprvA
Public
C
PprvC
Private
Public
Ppub
Public
PprvB
Private
B
Inter-domain Collaborations
Ppub – Public Business Protocol
PprvA-C – Domain specific intergration protocol

Figure 2.1.: Overview of Inter-domain Collaborations

Collaborations of such type are in fact widespread, e.g. e-Business, e-Health and e-Science, which would also fit in to e-Governmental collaborations. Figure 2.1 shows a bird's eye view of the level 2 collaboration. There are three collaborative partners, "A", "B", and "C"; each domain may or may not contain "private" and "public" views of their IT infrastructures dependent upon the local policies. Each domain also may or may not have its own domain specific protocol to integrate its public and private parts of business infrastructures dependent on how their infrastructures are organized. However, they must have agreed public business protocols to communicate amongst the collaborative partners. Moreover, there must exist some IT infrastructure facilitating the collaboration such as repository of collaboration process specifications, resources shared, and policies governing the collaborations. The gray colored cloud represents the realm of the kind of the inter-domain collaborations addressed.

Collaborations of such type face new challenges that do not exist in collaborations within a single domain. Those challenges sprout from the following distinct characteristics of those inter-domain collaborations. In general, they pertain the following attributes:

- **Preservation of autonomy and privacy**: Each participating organization of an inter-domain collaboration remains independent in terms of administrating their resources, infrastructures, and policies related to them or how to operate them.
- **Consistency of security policies**: It should be assumed that collaborative participating organizations run heterogeneous systems with potentially conflicting security policies. The policy for collaborations may face conflicts with local policies of collaborators. Solutions dealing with such cases must be provided.
- **Need of coordination**: Coordinations are inevitable in order to facilitate secure, interoperable collaborations. Mainly those coordinations are between the systems in the collaborative realm and those in collaborators' domains. In addition, cross-organizational monitoring and administrative communications are required not only for the correct execution and completion of the process, but they are essential in order to preserve consistency of the collaborative security policy and yet allowing all the entitled actions by users from other collaborative domains. Monitoring and logging are essential for reviewing and auditing processes.

The first two characteristics are major attributes to which the rest of the work in this thesis comes back for check; on the other hand, the last attribute has been regarded only partially. Although they are important, coordination for monitoring and logging have not been dealt with in this thesis, for they themselves are their own topics of other theses [vRR09]. As far as authorization related coordination is concerned, Chapter 4 thoroughly deals with it.

2.2. State of the Art: Business Perspective

In this section, a handful of current e-Governmental use cases are described as examples of inter-domain collaborations, and security requirements from business perspective are being discussed as a result of looking into the use cases.

2.2.1. Use Cases from R4eGov

Five different e-Governmental use cases conducted within an EU project, R4eGov[2], demonstrate the current state of the art of the European e-Government readiness. Compiled by the corresponding organizations themselves, they describe the collaborations from the business perspectives. The list below summarizes the key observations of the collaborations captured from each of the use cases along with their common characteristics:

- **German Supreme Court (Bundesgerichtshof – BGH)**

  The BGH is a court of appeal aiming to safeguard legal conformity through the clarification

[2]http://www.r4egov.eu/

Table 2.1.: Readiness of e-Government Collaborations[LL07]

Use Cases	Readiness	Number of Collaborators	Number of Potential Users	Maturity Level of Interoperability
BGH	X	1	$\approx$ 440	Independent
EP/EJ	V+	29	$\approx$ 6000	Ad-hoc or Collaborative
eVisa	X	World	$\approx$ 3 million	Ad-hoc
BKA	V+	5 (EU)	4-8 million	Collaborative
eProcurement	V	27	136,000	Ad-hoc or Collaborative

and the development of law. Due to the highly specialized tasks that the BGH deals with, the number of actors involved in their collaborations are restricted to specific judges, lawyers and judicial clerks. Currently, their collaborations are mainly paper-based and within the organization [ABN07].

- **Europol/Eurojust**
  Europol, the European Police Office (EP) and Eurojust, the European Judicial Cooperation (EJ), are European agencies that have been set up to facilitate the 27 EU member states in their fight against organized crime across national borders. To accomplish their primary mission, both Europol and Eurojust carry out distinct tasks in the context of joint efforts amongst the police, customs, immigration services, and justice departments of the EU member states [ABN07]. Acquiring an European arrest warrant and requesting mutual legal assistance for witness protection during court proceedings are a couple of exemplary tasks of their collaborations. Their collaborations can be categorized as between Ad-hoc and Collaborative levels as the interactions are often done through liaison officers and in spontaneous connections although there is a potential of providing for an overarching framework for more systematic interactions. In the collaborations of Europol and Eurojust with the EU member states, a number of globally visible roles have been detected (The specific list of external roles identified as well as the corresponding number of users to those roles are stated in Section 5.1.1). The total number of users involved in the Europol and Eurojust collaborations is approximately 6000 [ABN07].

- **eVisa**
  The eVISA use case focuses on the possible collaborations between the educational institutes and the mobility facilitating agents (e.g. local consulates and border management posts) in order to deal with visa related issues of international students in the EU member states. Numerous collaborative scenarios can be dynamically composed not only between public administrators for the purposes of border control but also between administrators and end-users for offering services to involved citizens. The eVisa use case also has the potential to include a larger number

of countries due to the wide-spread student exchange programs among the universities within the EU member states as well as non-European country nationals studying in Europe [ABN07].

- **Austrian Federal Chancellery (Bundeskanzleramt – BKA)**
  Aiming to render the entire life cycle of legal texts in completely electronic format, the Austrian Federal Government has launched the e-Recht (e-Law) project in 2001. This use case illustrates various activities that can happen to the digitized legal text such as authoring, querying, reviewing, and transferring. Its main concern is between administration to administration collaborations. Due to the lack of compatibility, only five other EU countries can interact with the current content repository addressed in this use case, but theoretically, its information system can be of use at the EU level once the interoperability issues are consolidated [ABN07].

- **eProcurement**
  The eProcurement use case deals with a cross-border call for tender process in the EU member states. A company which wishes to bid at a cross-border call for tender in the EU is required to provide proof of its legal existence and conform to its national legal, fiscal, and social obligations. On the other hand, a public authority receiving the tender must be able to check the validity of the submitted documents from the bidders. This verification task is handled by a third party authority called Trade Registers [ABN07]. This use case has examined the particular scenarios related to the Trade Register of the Paris Business Court (GTCP). Due to the differences among the EU national laws and regulations related to the public tender procedures, call for tender procedures may differ from country to country even within the European Union.

Excluding the BGH use case, all of the scenarios depicted in the case studies contain a large number of potential collaboration participants. The number of collaborators (*# of Collaborators*) and the number of potential user (*# of Potential Users*) columns of Table 2.1 show numeric estimations of the size of the potential multi-domain collaborations. The *Readiness* column states informal ratings of deployment readiness for electronic multi-domain collaborations. The BGH and eVisa case studies are rated as *not ready* (marked with ‘X’) due to the lack of necessary technical and legal frameworks. The eProcurement use case is rated as *partially ready* (marked with ‘V’) since it is still bound by the limitations from national regulations. The EP/EJ and BKA case studies are rated as *ready* (marked with ‘V+’), for they have foundational IT infrastructures and the legal endorsements for their collaborations.

The column, *Maturity Level of Interoperability*, classifies the types of collaborations of the use case scenarios according to Clark and Jones’ classification [CJ99], which has been discussed in the beginning of this chapter. The BGH scenarios operate within a rather monolithic environment as independent interactions. The EP/EJ and eProcument scenarios is best described as a mixture of Ad-hoc and Collaborative interactions since the course of their collaborations is dynamically decided whether to allow actors to join or leave during the collaborations. For instance, the eProcurement scenarios start up with open Ad-hoc based interactions for bidders to join; but then, once the registration period is over, the collaboration is within a closed group of bidders and a Trade Register. Meanwhile, the BKA scenarios are entirely within their content management system environment, which is a closed environment with fixed collaborators up front whereas the eVisa case represents Ad-hoc type of interactions without showing a developed organized framework of interactions but rather relying on peer-to-peer exchange of information. However, it is arguable that the distinction between *Ad-hoc* and *Collaborative* collaborations is rather unclear.

2.2.2. Security Requirements

The major security requirements captured for the e-Government use cases are found as follows [Lee07]:

Locality of control Local authorities must be able to determine authorization decisions based on their own security policies autonomously.

Assured management of decentralization Having a mechanism of decentralized policy management is inevitable for inter-domain collaborations; nevertheless, the decentralized management must display the merits of trustworthiness [TNI87].

Compatibility with standards Use of standard-based modules must be encouraged for interoperability within heterogeneous communities.

Flexibility in choices of enforcement mechanisms Policy specification and enforcement should be separated so that various mix-and-match combinations of them can be allowed.

Scalable deployment The authorization system must be easily deployable to dynamically joining collaborators as well as to a new collaboration setting.

Preservation of end-user privacy Collaborative partners must be able to protect their internal end-users from unnecessary disclosure of information.

Though these criteria are selected specifically for the European e-Govenment scenarios, these are a sufficiently generic set of requirements that they can be applied to the majority of other inter-domain collaborations.

2.3. State of the Art: Technical Perspective

This section takes a technical perspective to assess the state of the art of current inter-domain collaborations. The existing technologies utilized to realize inter-domain collaborations are first introduced; then, security solutions that are being integrated with the fore-mentioned technologies of composing inter-domain collaborations are illustrated.

2.3.1. Enabling Technologies

Currently, there are two major trends in terms of forming inter-domain collaborations. One of them runs the collaboration in an ad-hoc fashion in a loosely-coupled multi-domain environment whereas the other method is by creating an artificial environment that is similar to that of a monolithic one where centralized policy management and authorization services are applicable. In order to form a collaborative environment, it is currently often required to employ similar infrastructures, middleware, and proprietary specifications amongst collaborative partners. Some of the examples of this type of collaborations are Virtual Organization (VO) and workflow management systems. In the following, one example of each of the aforementioned major trends are described: Web-service based collaboration as an example of ad-hoc fashioned multi-domain collaboration and VO, of an artificial environment created by collaborators.

Web-Service based Collaborations

Service-oriented Architecture (SOA) is a paradigm in application development in order to implement loosely-coupled, distributed environments. Within these environments, participants can overcome heterogeneity by communicating through standardized, implementation-independent interfaces. Such implementation-independent interfaces are the key property of web-services. Another concrete definition of web-services is found in *Web services: Concepts, Architectures and Applications* by G. Alonso as "self-contained, modular business applications that have open, Internet-oriented, standards-based interfaces [ACKM04]".

Following the SOA philosophy, application logic is not embedded in monolithic programs, instead, it is distributed among the participants of processes. Each web-service defines its application logic in a set of published XML artifacts so that another web-service can discover the others according to its needs. In its basic level, web-services are built out of three W3C standards: 1) SOAP [GHMM07] is used to transfer data, 2) Web Service Description Language (WSDL) [CMRW07], for describing the services, and 3) Universal Description, Discovery and Integration (UDDI) [CHvRR04], for listing what services are available. Nowadays, web-services are more often than not chosen to realize B2B processes across organizational boundaries, aiming to make the integration necessary to construct B2B processes as automated as possible.

Inter-domain collaborations, being a type of B2B process, can be composed out of a web-service implementation. In fact, there are a number of choices of combination of so called "web-service building blocks" in order to compose inter-domain collaborations. Dependent on the business model and how the process should be managed, various ways of composing web-services are available. Among so many, a couple of well-known standards to implement any e-* communities is WS-BPEL (Web-Services Business Process Execution Language) [AAAB07] and WS-CDL (Web-Service Choreography Description Language) [KBR$^+$05].

WS-BPEL[3] is a language that can support the specification of both composition schemas as well as coordination protocols. By following composition schemas, one can comprehend the business logic of the set of web-services composed together as a coherent whole. In a nutshell, a WS-BPEL specification represents the process by the order of invocation of web-services involved, which is also called "orchestration." It is composed of structured activities such as Sequence, Switch, While, and Flow where they have similar meanings as in C/C++ programming languages. The not so obvious one, "Flow,"

[3]BPEL4WS is the acronym actually used in the WS-BPEL specification document.

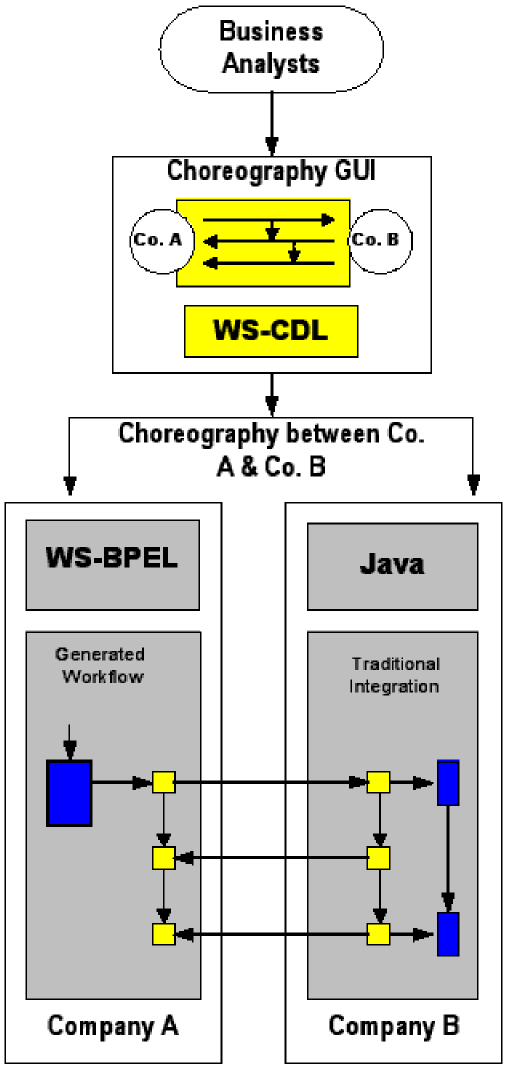


Figure 2.2.: Integration of Web-service Applications using WS-CDL [KBR+05]

allows activities to be started in parallel. In addition, the WS-BPEL specification includes definition of roles that take part in the message exchanges with the process as well as correlation information, telling how messages are to be routed to the correct composition instances.

Similar to WS-BPEL, WS-CDL is also a language for the purpose of coordinating the interactions of web-services. Unlike BPEL, however, it defines collaboration from a global viewpoint rather than taking that of one of the collaborators. This means that the viewpoint of WS-CDL understands all the external interactions and interfaces amongst collaborators whereas WS-BPEL takes the perception of one collaborator and perceives all the external interactions the collaborator is a part of as well as all the internal activities that must be fulfilled once triggered by one of its own web-services. The focus of WS-CDL is to provide an interoperable choreography of the externally visible interactions amongst collaborators. Its language allows to describe roles of participants, information that is being exchanged, constraints as so called "workunit", and choreography that include activities and ordering structures. It also allows to specify the collaborations with reasonably well synchronized time boundaries.

As depicted in Figure 2.2, it is possible that WS-CDL and WS-BPEL coexist within the same inter-domain collaboration. According to the figure, Company A and Company B integrate their web-services based applications. The respective business analysts at both companies agree upon the web-services participating in their collaborations, the involved activities, interactions, the ordering of the process, and the constraints under which the activities should occur. These can then be ensured as an interoperable representation of the WS-CDL-based choreography. As far as how each of the Web-services is going to be implemented, both Company A and B have full control. Furthermore, Company A utilizes WS-BPEL to implement its business logic while Company B relies on the J2EE solution to incorporate Java in order to implement its own part of the choreography.

Another possibility is to compose web-service blocks across organizational borders to implement workflow type of collaboration including execution of coordinated transactions. Some of the components necessary then to utilize are WS Coordination [RNFJ09], WS-CAF (Web Services Composite Application Framework), WS Atomic Transaction [RNLW09], and WS Business Activities [RNFL07]. WS Coordination offers an extensible framework which allows an application to create a coordination context. This context is needed for propagating activities to other web services. WS Coordination was introduced by Microsoft, IBM and BEA and provides a coordination service which consists of activation, a registration service, and a set of coordination types. The activation service is responsible for generating the activity context. In order to join a created activity and its related context, the registration service handles the registration of participants. The offered coordination types are WS Atomic Transaction and WS Business Activities, which are both providing a set of protocols. This framework is built on top of the basic web service standards, SOAP and WSDL.

Due to the modular architecture of web-services, when they are assembled for a business purpose, it is often the case that there must exist an additional interface to the Internet that serves as a gateway. Inside such a gateway, there exists a set of components that perform distinct tasks before a message is being routed to a web-service. For example, it will need to harmonize public choreographed processes with private services within the organization or to validate incoming requests. This can also apply to outgoing messages. Thus, the gateway infrastructure handles pre-processing and post-processing of the input or output to or from the internal modules in order to make them standard-compliant. In general, the effort for integrating different web-service building blocks with a “gateway” infrastructure is not trivial.

Virtual Organization (VO)

A virtual organization (VO) is a distributed network of independent organizations that provides an artificial environment of closed context for the participants to share resources or services to complete their common objectives. VO has been adopted in many applications where geographically separated organizations with different IT infrastructures need to work together such as dynamic enterprise, on-demand service provider, business-to-business collaboration, etc. In fact, VO is the key outcome of what grid computing or so called “cloud computing” communities have developed. According to the Open Grid Services Architecture (OGSA), VO is defined with few distinct characteristics such

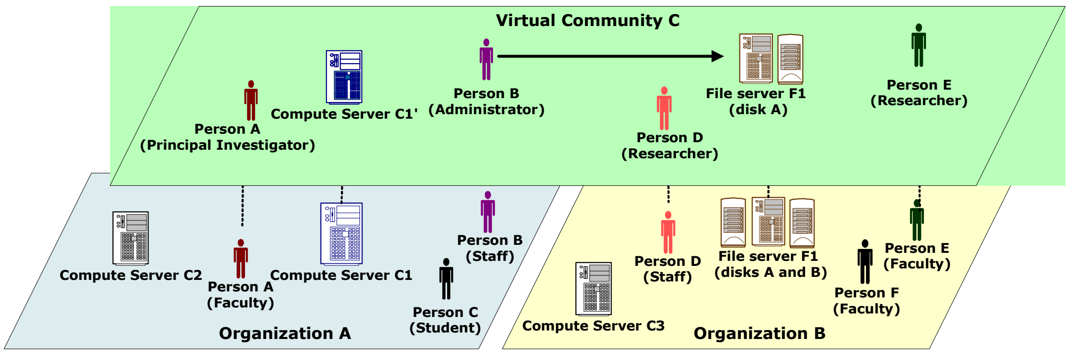


Figure 2.3.: Concept of Virtual Organization [FKSB05]

as 1) having an independent ownership over resources and information systems each participating organization possesses, 2) pursuing a common objectives and 3) being formed based on a trust relation specified within a closed context [FKSB05].

Figure 2.3 illustrates the concept of VO. In it are two organizations and an overlaid VO that is governed by its own policy. Notice that some entities from the participating organizations are available to the VO community; nevertheless, those entities have their own specific attributes and properties within the context of the VO. For instance, person A is a 'Faculty' within Organization A, but the person is known as 'Principal Investigator' in the VO community. Not only entities are defined by a set of attributes, but capabilities and jobs are also defined by sets of attributes and properties. Altogether they compose the closed context.

As far as the implementation of VO is concerned, the open-source software GT4 is the latest version of Globus Toolkit developed by Globus Alliance[4] which leads the Global Grid Forum (GGF). As for the security service components, there are a number of developments. VOMS (Virtual organization membership service) developed by EU DataGrid and DataTAG projects is a credential push system in which the VOMS server digitally signs X. 509 attribute certificates for VO users so that they can be used for access to the resources. CAS (Community Authorization Service) [PWF$^+$02] is another one that issues attribute assertions to VO users. To incorporate with CAS, resource providers need to delegate a part of their authority to a VO Server. VO community then composes resource policy for its VO. There are more integrable security solutions such as Shibboleth and PERMIS; these are explained in detail in 2.3.2.

2.3.2. Existing Security Solutions

In this section, the core security services essential for inter-domain collaborations to take a place are explained. They are authentication or authorization solutions that can be plugged into a collaboration systems such as VO. The first solution is Shibboleth which provides user provision while the others are authorization solutions.

[4]http://www.globus.org/

Shibboleth

Shibboleth[5] is a standard-based open-source middleware which provides a single sign-on (SSO) across or within organizational boundaries through a web interface. In order to perform authentication, Shibboleth relies on two entities: Identity Provider (IdP) and Service Provider (SP). Once a user is authenticated against its IdP, the IdP informs the SP by sending it a proof of the user authentication and its accreditations using the Security Assertion Markup Language (SAML) standard protocol [CKPM05]. According to the Shibboleth architecture, the users' accounts are managed uniquely by their home organizations.

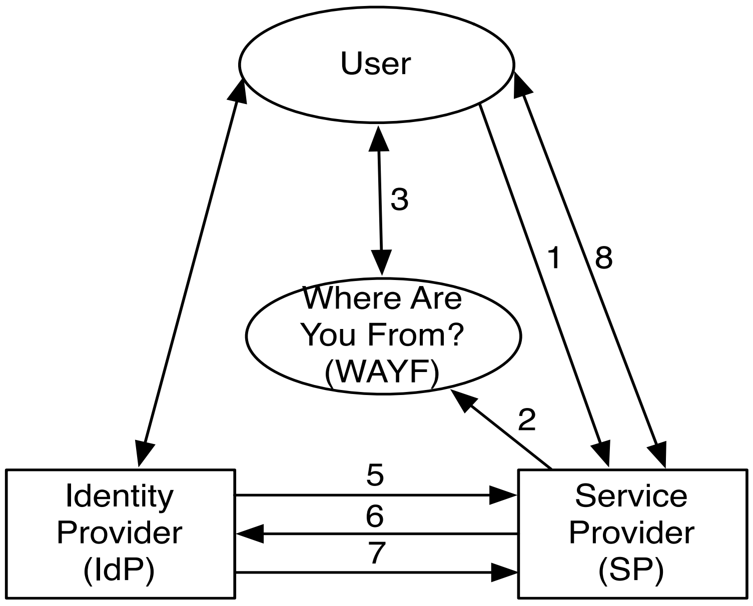


Figure 2.4.: Information Flow in Shibboleth [JNT]

A typical information flow using Shibboleth is depicted by Figure 2.4; there are many other variations simpler than this, but this flow gives well-rounded explanations of the data flow between IdP and SP. This process can be utilized both for authentication as well as for access control. The process in the figure is explained step-by-step as following:

1. The User attempts to access a Shibboleth-protected resource on the Service Provider site.
2. The User is redirected to the federation WAYF (Where Are You From?).
3. The User select his or her home institution Identity Provider[6] from the list presented by the WAYF.
4. The Identity Provider, by whatever means it deems appropriate, ensures that the User is authenticated.

[5]http://shibboleth.internet2.edu

[6]Although the User's home institution is taken in the above summary to be equivalent to the Identity Provider, in fact an institution may choose to outsource the Identity Provider function to another organisation. However, this does not affect the principle of operation.

5. After successful authentication, a one-time Handle (session identifier) is generated for this User session and is sent to the Service Provider.

6. The Service Provider uses the Handle to request attribute information from the Identity Provider for this User.

7. The Identity Provider, on the basis of its Attribute Release Policy, allows or denies attribute information to be made available to this Service Provider.

8. Based on the attribute information made available to it, the Service Provider allows or refuses the User access to the resource.

It is noteworthy that the attributes released to Service Provider are information about the User rather than that of the resource the User inquires to access. That part is solely decided by Service Provider.

PERMIS

PERMIS[7] is a privilege management infrastructure (PMI) developed by the ISSRG team of the University of Kent [CZO$^+$08] primarily for grid computing environments. Its policy-based authorization system can work with a number of different authentication systems (e.g. Shibboleth, Kerberos, PKI, and Username/PW). PERMIS deploys the 'Subject-Action-Target' paradigm together with the ISO attribute-based access control (ABAC) model, which can be considered as a more general model of role-based access control (RBAC). For instance, given a username, a target and an action, the PERMIS decision engine tells whether the user is granted or denied access to the given target based on the given policy. Thus, it can be said that PERMIS supports the concept of role; however, rather than considering it as a set of permissions as in RBAC, the roles in ABAC refer to a user attribute or certain properties of the context the policy is described with. PERMIS also supports a notion of hierarchical RBAC in which roles (or attributes) are organized in a partial hierarchy, with some being superior or subordinate to others.

The two main components of PERMIS' PMI are Policy Enforcement Point (PEP) which executes a policy and Policy Decision Point (PDP) which makes an access control decision based on a given policy. In addition, a number of different modules exists to provide services related to authorization. For instance, its enforcement infrastructure includes attribute repository, history-based decision-making PDP and credential management suite – credential issuing service, credential validation service (CVS), credential retriever and credential decoder.

Taking a modularized infrastructure, it can be integrated with various Grid softwares and user authentication components for virtual organizations. For instance, it can be integrated with Shibboleth and GT4, which has a number of tools that make development and deployment of Grid services easier [CZO$^+$08]. This component is called GridShibPERMIS as described in Figure 2.5. The core modules of the PERMIS infrastructure are PERMIS PDP and Credential Validation Service (PERMIS CVS) components. These two modules provide the interface to various authentication products such as Shibboleth, X.509, SAML, or a customized approach. It provides an interface called "GridShibPERMIS

[7]http://www.permis.org/

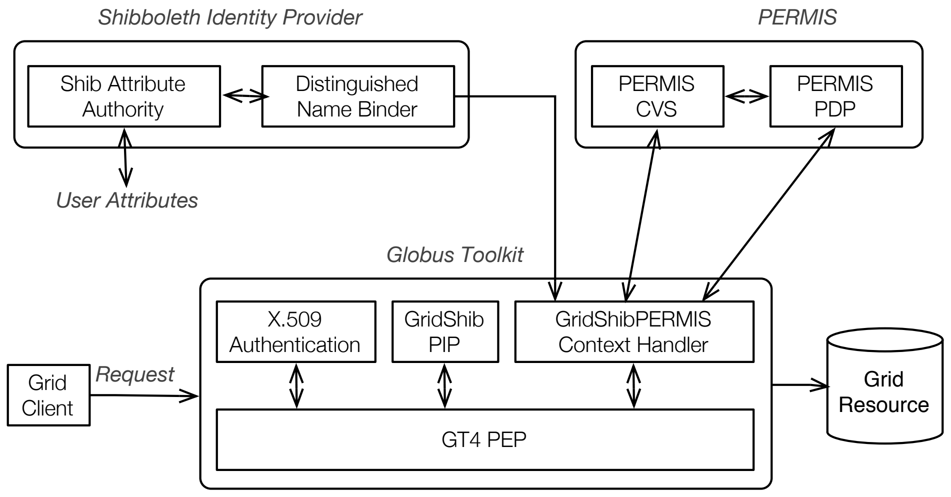


Figure 2.5.: The PERMIS Authorization Decision Engine [CZO+08]

Context Handler" which interacts with a selected authentication module. At the same time, the context handler has another interface to communicate with the policy enforcement point which is GT4 in the case with Globus. Based on the choice of the authentication application selected, it also integrates the matching credential retriever and decoder as well as the corresponding policy parsing modules. For instance, since Shibboleth utilizes the SAML standard, the SAML attributes must be also loaded up to the PERMIS authorization decision engine.

Although the PERMIS offers a 'hub' for a variety of authentication methods; the way the PERMIS authorization policies are specified is resource-centric. The owners of resources can compose or edit PERMIS policies through a tool called Policy Editor with a graphical user interface. From the back-end of the program, PERMIS policies in extensible markup language (XML) are generated based on the user input. This policy language is also not a standardized format. Additionally, the protocol by which the PEP and the PDP interact is also a proprietary protocol.

WS-* Security Components

The Web-services technology offers various security related components. Among these components are the building blocks for security policy specification and enforcement. Figure 2.6 depicts how some of the major building blocks can be positioned. The top layer components are related to the *Application* layer of the Open System Interconnection (OSI) Model while the second to the fourth layers from the top are related to the *Presentation* and the *Session* layers of the OSI Model, dealing with message format and data transformation. The second to the bottom layer of the figure can be linked to the *Transport* layer of the OSI while the lowest layer corresponds to the *Network* layer of the OSI.

The top left block of Figure 2.6 called "Security Management" contains the WS-Trust [LKNG07b] component, which enables Web-services to require incoming messages to provide a set of claims such as name, key, permission, and capacity according to the security policies in use. The goal of the WS-Trust is to enable "applications to construct trusted SOAP message exchange" [LKNG07b]. In the

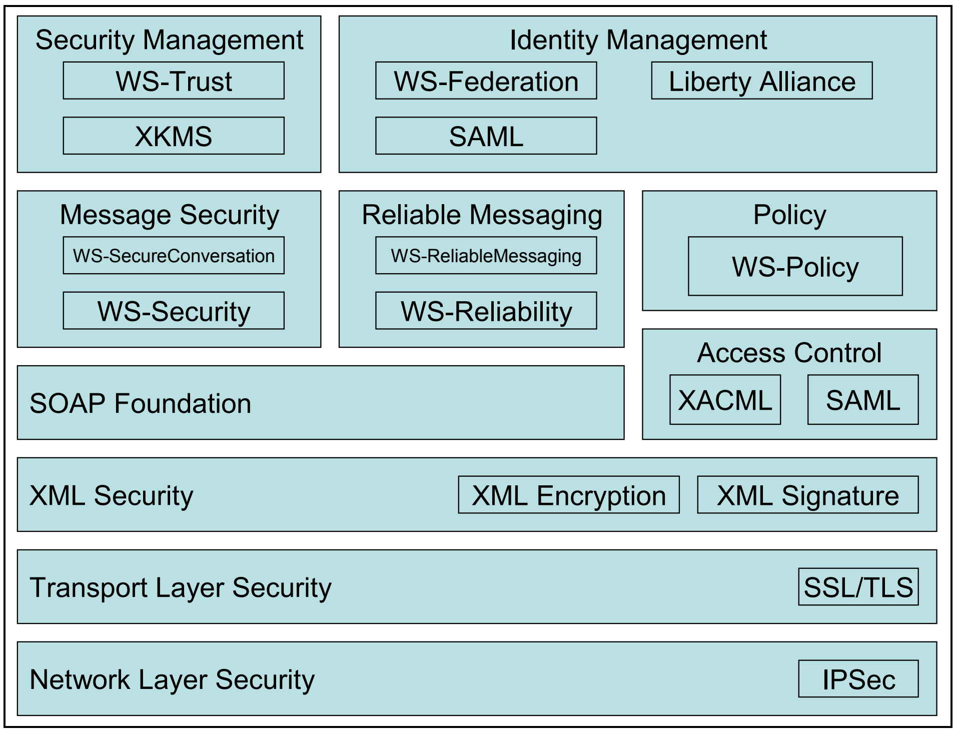


Figure 2.6.: Web Services Security Standards [SWS08]

specification, it provides Security Token Service Framework through which one can request and obtain a security token that will be trusted by the Web-services it wants to contact.

The next block called “Identity Management” contains WS-Federation [LABS06], which is built on top of the lower-level building blocks such as WS-Security [LKNM06] and WS-SecureConversation [LKNG07a]. From the conceptual point of view, the specification discusses federation scenarios and proposes possible federation protocols. On the infrastructural side, WS-Federation defines a federation meta-data document to exchange federation-related data between federation participants. In addition, a pseudonym service, an attribute retrieval service, and a sign-out mechanism are also defined. These elements basically enable a WS-Federation implementation to authenticate a user and perform Single Sign-On (SSO) and/or Single Log-Out (SLO) in a Web services environment. The principal action in such an environment is a federated service request. WS-Federation describes a basic role model for federated requests that consists of four actors and involves three communication actions between them. Nevertheless, WS-Federation is little concerned with the much broader concept of identity management and still has not made it into draft status at OASIS (Organization for the Advancement of Structured Information Standards).

The second layer blocks from the top are all related to SOAP message. WS-Security in the left-most block called “Message Security” aims to establish end-to-end message level security. It specifies “an abstract message security model in terms of security tokens combined with digital signatures” [LKNM06] and enables different security mechanisms to be used in order to provide message integrity and confidentiality such as PKI, Kerberos, and SSL. As for the format of specification, it utilizes

XML Signature and XML Encryption in conjunction with security tokens. WS-Reliability [IDR04] in the “Reliable Messaging” block enables specification of a critical set of requirements in order for application-level messaging protocol to guarantee reliable message transfers for some applications of Web Services. Some of those requirements are “guaranteed delivery, no duplicates, and guaranteed message ordering” [IDR04].

In the block called “Policy”, WS-Policy [BBCC06] is stated. It is note-worthy that the word “policy” here does not imply policy in the context of access control where it represents a set of rules referred to for deciding whether to grant an access or not. Rather, it refers to policies for securing communications. It is used to specify “a collection of policy alternatives” [BBCC06] where a policy alternative represents “a collection of policy assertions” [BBCC06]. For example, it specifies various conditions and contextual information of data exchange via the SOAP protocol such as the type of security token, digital signature algorithm, encryption mechanism for a SOAP message (for example, a purchase order message), and partial contents of a SOAP message (a credit card number, for example). In addition, it can also specify data-privacy or data-confidentiality rules.

WS-Policy does not specify how to discover policies nor how to attach a policy to a Web service. It relies on other WS-* specifications (for example, WS-PolicyAttachment, which is not shown in Figure 2.6) to provide full functionality of policy management. Under WS-Policy, there is a set of policy assertions for each policy domain. For example, the assertions for use with WS-Security [NKMHB06] are defined in WS-SecurityPolicy. Each specification or schema to be controlled or managed by WS-Policy requires definition of a new set of assertions. Under the WS-Policy model, a policy for Web services denotes conditions or assertions regarding the interactions between the endpoint Web-services. The service provider exposes Web-services’ policy for the services they provide. The service requester decides, using the policies, whether it wants to use the service. In other words, WS-Policy does not have the notion of a Policy Enforcement Point (PEP), which enforces policies, nor a Policy Decision Point (PDP), which also determines policies. It leaves the policy enforcement and decision to the service providers and service requesters.

The descriptions of the rest of the lower blocks are skipped since they are irrelevant to authorization while the explanation of XACML in the “Access Control” block is given in the following sub-section. The number of WS-* security related building blocks alone has increased rapidly over past years. Thus, to find out which standard is for which purpose and with what other specifications it must be developed is no longer a trivial task. NIST has published a security guideline for Web-services [SWS08] in which Table 2.2 has been provided, summarizing which security specifications are for which security dimensions and requirements.

XACML

The OASIS standard, eXtensible Access Control Markup Language (XACML) provides not only the XML based format to specify authorization policies but also an infrastructure to enforce the XACML formatted policies [Mos05]. The XACML Framework is shown in Figure 2.7 together with the explanation of step-by-step operations of the processing an XACML request:

Table 2.2.: Specifications and Standards Addressing Security of SOAs [SWS08]

Dimension	Requirement	Specifications
Messaging	Confidentiality and Integrity	WS-Security
		SSL/TLS
	Authentication	WS-Security Tokens
		SSL/TLS X.509 Certificates
Resource	Authorization	XACML
		XrML
		RBAC, ABAC
	Privacy	EPAL
		XACML
	Accountability	None
Negotiation	Registries	UDDI
		ebXML
	Semantic Discovery	SWSA
		OWL-S
	Business Contracts	ebXML
Trust	Establishment	WS-Trust
		XKMS
		X.509
	Proxying	SAML
		WS-Trust
	Federation	WS-Federation
		Liberty IDFF
		Shibboleth
Security Properties	Policy	WS-Policy
	Security Policy	WS-SecurityPolicy
	Availability	WS-ReliableMessaging
		WS-Reliability

1. The access requester sends a request for access to the PEP.
2. The PEP sends the request for access to the context handler in its native request format, optionally including attributes of the subjects, resource, action and environment.
3. The context handler constructs an XACML request context and sends it to the PDP.
4. The PDP requests any additional subject, resource, action and environment attributes from the context handler.
5. The context handler requests the attributes from a PIP.

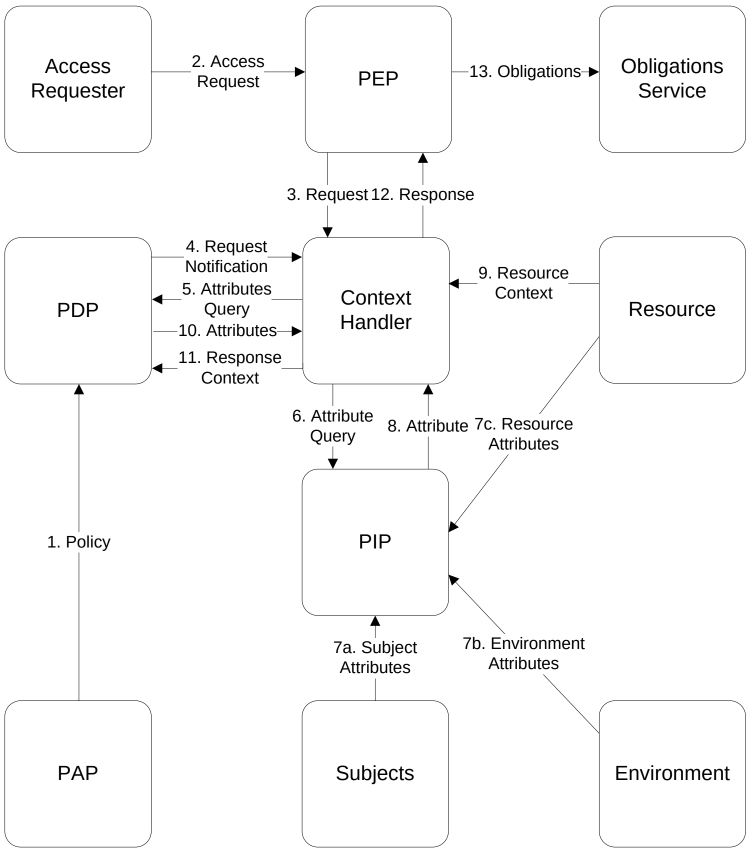


Figure 2.7.: Overview of XACML Framework [Mos05]

6. The PIP obtains the requested attributes.
7. The PIP returns the requested attributes to the context handler.
8. Optionally, the context handler includes the resource in the context.
9. The context handler sends the requested attributes and (optionally) the resource to the PDP. The PDP evaluates the policy.
10. The PDP returns the response context (including the authorization decision) to the context handler.
11. The context handler translates the response context to the native response format of the PEP. The context handler returns the response to the PEP.
12. The PEP fulfills the obligations included in the response.

(Not shown) If access is permitted, then the PEP permits access to the resource; otherwise, it denies access.

YAABA

YAABA (Yet Another Attribute Based Authorization) is an XACML-based Privilege Management Infrastructure (PMI) developed by Laborde et al [LKW$^+$09]. It implements the XACML authorization

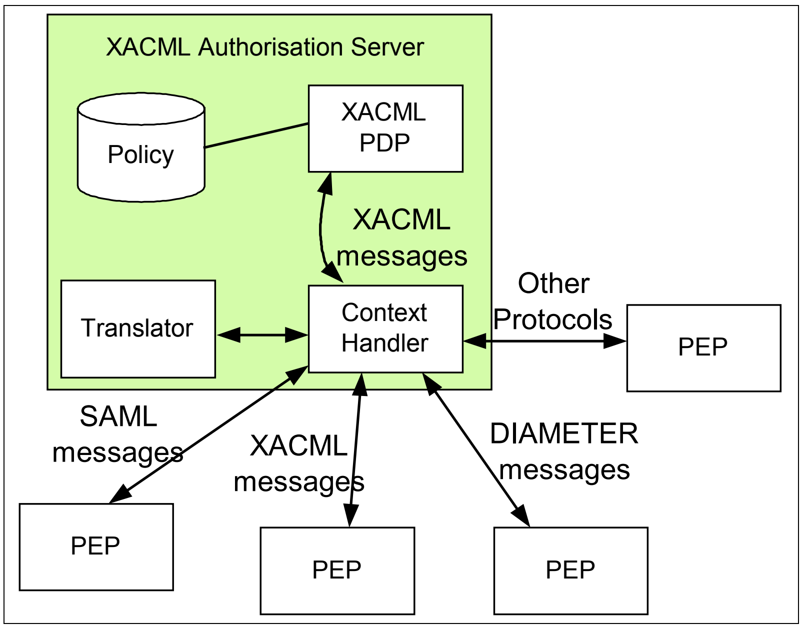


Figure 2.8.: Architecture of YAABA [LKW+09]

architecture that can fit in to the solution of Shibboleth.

YAABA is designed to be used for virtual organizations. While user administrators issue credentials containing users' attributes, the managers of a VO resource determine the authorization policy for target resources. These policies are written in XACML and stored in a way the PDP can understand. When a user requests an access to a resource, 'application gateway' of the target system performs user's authentication first according to the application within which the request comes from. Then the PEP agent that receives the access request forwards the request together with required attributes to the PDP of YAABA. Once the decision is made, it returns to PEP, and the PEP then can process the user's request based on the PDP's authorization decision. Their authorization policies are application-independent, and thus the policies can be used by different applications.

As depicted in Figure 2.8, the Context Handler accepts various forms of access requests from PEP, for the XACML standard does not specify which protocol is to be used between the PEP and the Context Handler [LKW+09]. Since the communication between PDP and PEP is in the XACML format, the incoming requests of the Context Handler are translated into the XACML request format by the Translator, which is a new component in YAABA. Unfortunately, YAABA is currently not supported any more, but its core components are developed together with policy developed using configurable data types and ontology instead of using the Translator.

Other Solutions

Needless to say, there are a number of commercial products that offer security services for inter-domain collaborations or so called cloud computing. IBM's Tivoli[8] is one of those examples. It is a full-fledged security solution for cloud computing offering services from user provision to service automation,

[8]http://www.ibm.com/software/tivoli

service request, and monitoring usage and accounting manager. Nonetheless, it requires an all-knowing centralized management of users and the access control policies of resources involved. This cannot be acceptable to autonomous collaborators that they must share all their internal resources' attributes with centralized administrators. It also includes proprietary communication protocols and infrastructures; as a result, any organization wanting to join a collaboration is required to buy and install the modules of Tivoli in use. It is not an exceptional case; most of the current commercial product solutions do come with similar conditions. This, however, is not the suitable choice for multi-domain e-Governmental collaborations. The commercial products, when well developed, are, in general, more reliable and provide advanced consulting services; yet, it does not offer scalable solutions.

There are, however, some minor percentage of commercial products that follow standard-based solutions. The entitlement management product by Axiomatics[9] is one of the examples. Axiomatics' product provides a client/server suite of entitlement management solution in terms of XACML with the ABAC paradigm. As described earlier in the description of Shibboleth, ABAC enables policy makers to compose fine-grained, context-aware security policies independent from specific applications.

As from research communities, there are a number of smaller-scaled solutions than Shibboleth and PERMIS. Some of the proposals published so far regarding multi-policy management schemes and authorization frameworks are worth mentioning. As some of the exemplary research work, [BBKKS05] proposes a subscribe/notify mechanism for automated conflict resolution between a VO and local policies and for informing related property changes to each other. They also state that they are investigating dynamic provisioning techniques to identify resources that match each VO's requirements.

Demchenko et al. propose an access control model for distributed collaborative applications, which is an extension to the major service-oriented and Grid-based access generic framework such as Acegi, Globus Toolkit Authorization framework, and GAAA authorization framework [DdLGvB06]. Their policy specification is an extension of RBAC with domain-based security context, and their implementation uses the XACML architecture.

In [US06], a VO is interpreted as a multi-agent system where its members are autonomous, heterogeneous, and proactive agent. Udupi and Singh propose a proactive policy specification for a VO, a hierarchical model of policy monitoring, and a formalization of VOs as a multi-agent policy architecture [US06].

2.4. Security Challenges

Much of the emphasis of current security services for inter-domain collaborations is on the actual run-time mechanism and components that refer to security policies that are assumed to be generated either for collaborative realms or by collaborators' domain specific policies in advance. However, the policies themselves are rather poorly created and managed, solely dependent upon administrators' manual actions. There is also a tendency of integrating different security solutions into collaborative systems such as how Shibboleth, PERMIS, and YAABA attempt to integrate their solutions in to Web-Service or Grid computing based collaborative systems. As far as the authorization is concerned, the

[9]http://www.axiomatics.com

resource owners delegate a part of their rights to the Shibboleth authorization server to make a new access control for the SSO community while PERMIS lets the resource owners come up with their own access control policies. PERMIS takes extra steps to find out the user attributes required for the PDP to make an authorization decision. Based on the survey of the existing security solutions and the authorization requirements elicited from the e-Government scenarios, the following requirements are drawn as security challenges that ought to be still researched:

Systematic methodology of building security policies Not only the mechanism of processing security policies, but a systematic way of building security policies that are consistent, and ease-of-maintenance must to be provided.

Privacy ensuring policy management Due to the independent existence of collaborating partners, privacy of end-user of collaborators should be preserved.

Systematic integration of security solutions Authorization specifications and their enforcement mechanisms are advised to be decoupled, and the architectures of the existing technologies do conform the guideline. However, integration is often out of the scope of development of partial solutions. This must be considered so that scalability and interoperability can be achieved without difficulties.

As depicted in Figure 2.9, security policies and related attributes and properties that must be considered by a PDP for an authorization decision are from various sources. Building a systematic methodology of forming and gathering the security policies required by a PDP increases not only the quality of the decisions but also administrations efficiency.

Security policies, as reflected from the discussions on PERMIS and YAABA above, should reveal no more than what the other collaborators need to know regarding their end-users. At the same time, it should be as fine-grained as possible to specify under which circumstances the policies ought to be enforced. These two contradicting challenges circumscribing security policies require a trade-off that cannot be resolved automatically.

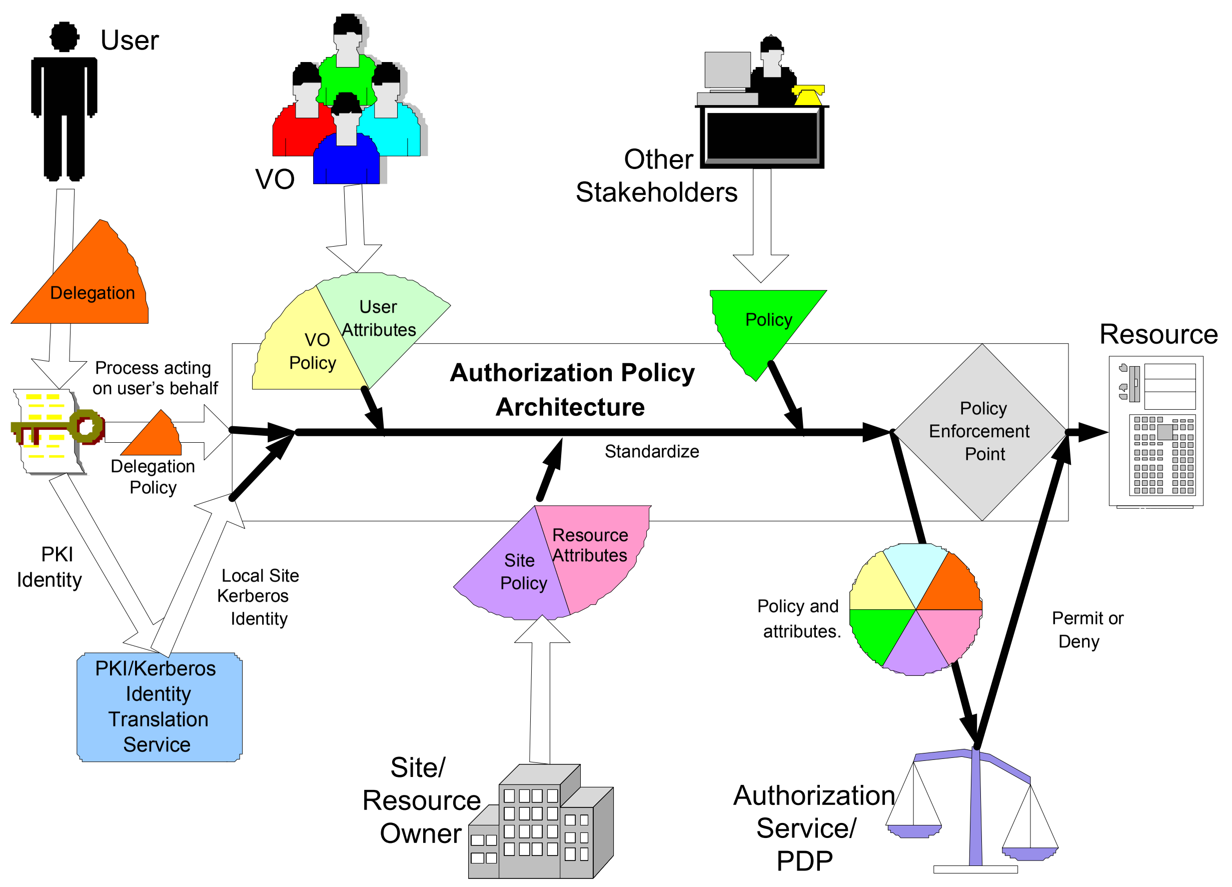


Figure 2.9.: Example Input for Policy Decision and Enforcement [FKSB05]

3. Decentralized Authorization with *Interactive Roles*

In computer science, problems of complexity are solved by [adding another layer of] indirection.
– David Wheeler

For a proper discussion on authorization, the terms used must be first introduced. Some of those basic terms are *principal*, *user*, *subject*, and *object*. The object of authorization or more closely of the access control field represents a resource such as file, devices, database, etc. The access to an object is specified in terms of operations and subject. Subject is a specific term for the entity that operates on behalf of "human users". This can be an application, device, etc. In the earlier chapters, we have casually used the term "user" and "principal" to refer to human users. From this point on, we will use principal to represent a name associated with a subject. As discussed in [Gas90],"access is based on the principal's name bound to the subject in some unforgeable manner at authentication time."

Access control service in the context of inter-domain collaborations can be defined as a mechanism to answer the following authorization question: "under which circumstances can a request be granted?" The phrase, "under which circumstances" includes a specific principal as well as the relevant conditions required by the security policy to produce a correct access control decision.

In its essence, *authorization* can be defined as a phase of defining the security policy by which access control can be determined. In this chapter, we introduce an authorization model to produce the security policy for the inter-domain collaborations. It is a logical extension of the Role-based Access Control model, which contains an extra layer of Role construct through an entity called *interactive Roles*. The additional abstraction not only allows decentralized user–role assignment in a scalable manner, but it also guides aligning functional roles defined in collaboration definitions with organizational roles in autonomous collaborative partners. As far as the definition of *role* is concerned, we follow the definition of the RBAC model; it is a collection of procedures, which in turn, is a 'high-level' access control method that specifies operations to objects [SCFY96]. A more in depth discussion of role is provided in Section 6.2.

Introducing the *i*Role concept in the section below is followed by explaining how an *i*Role-based authorization policy can be applied to policy enforcement. We also discuss the possible conflicts the security policy might have with local policies of collaborators and the precautions we have taken in order to preserve the consistency property. In the end, we present the *i*Role specifications in the eXtensive Access Control Markup Language (XACML).

3.1. Model

3.1.1. Conceptual Overview

First let us look into the model in comparison to RBAC from which it has originated.

Role-based Access Control

The core idea of Role-based Access Control (RBAC) is introducing a semantic construct called *Role* between *User* and *Permission* sets so that through the abstraction security administration at a business enterprise level can be provided instead at a user identity level. This abstraction enables establishing permissions based on functional roles in the enterprise and decoupling assigning users to the appropriate roles. By separating user-to-role and role-to-permission relations, more efficient administration is possible as user-to-role relation has a higher turn-over rate in an organization than that of role-to-permission.

According to the standard ANSI RBAC model [ANS04], Role is defined as "a job function within the context of an organization with some associated semantics regarding the authority and responsibility conferred on the user assigned to the role". Furthermore, it defines *Permission* as "an approval to perform an operation on one or more RBAC protected objects" and *Operation* as "an executable image of a program, which upon invocation executes some function for the user." Here, the types of operations and objects RBAC controls are to be dependent on the type of system it is implemented for [ANS04]. Under this model, permissions are associated with roles, and users are assigned to appropriate roles. As such, a role is a means for naming many-to-many relations between users and permissions. The idea of using a similar indirection as role has been already suggested in the discussion of "Discretionary Access Control" (DAC) via a notion such as "group" [TNI87]. Based on the membership in a particular group, a user is authorized to access a certain object. The notion of role is more specific than group in a sense that they represent a set of permissions with respect to objects relevant to the context a security policy is based on. Of course, a role can be assigned to a group of users as well as to individual users.

There is also a notion called *Session* in RBAC. A session includes an activated subset of roles. In short, it maps a user to activated roles that are assigned to the user at the given time. Unlike other traditional access control models such as Mandatory Access Control (MAC) or Discretionary Access Control, RBAC is "policy neutral" [SCFY96]; this means a RBAC model can be utilized expressing other access control policies such as DAC, MAC, or Chinese Wall policies to name a few. For example, we have explained how DAC and RBAC can coexist in the previous paragraph. It is also apt to be used to support not only authorization specification but also describing well-known security principles such as information hiding, the Least Privilege Principle, and separation of duty constraint. NIST ANSI standard defines the core RBAC model as following [ANS04]:

- $USERS, ROLES, OPS$ and OBS (i.e. sets of users, roles, operations and objects respectively).
- $UA \subseteq USERS \times ROLES$, a user-to-role assignment relation.
- $assigned\_users : ROLES \rightarrow 2^{USERS}$, the mapping of role r onto a set of users.
  Formally: $assigned\_users(r) = \{u \in USERS | (u,r) \in UA\}$

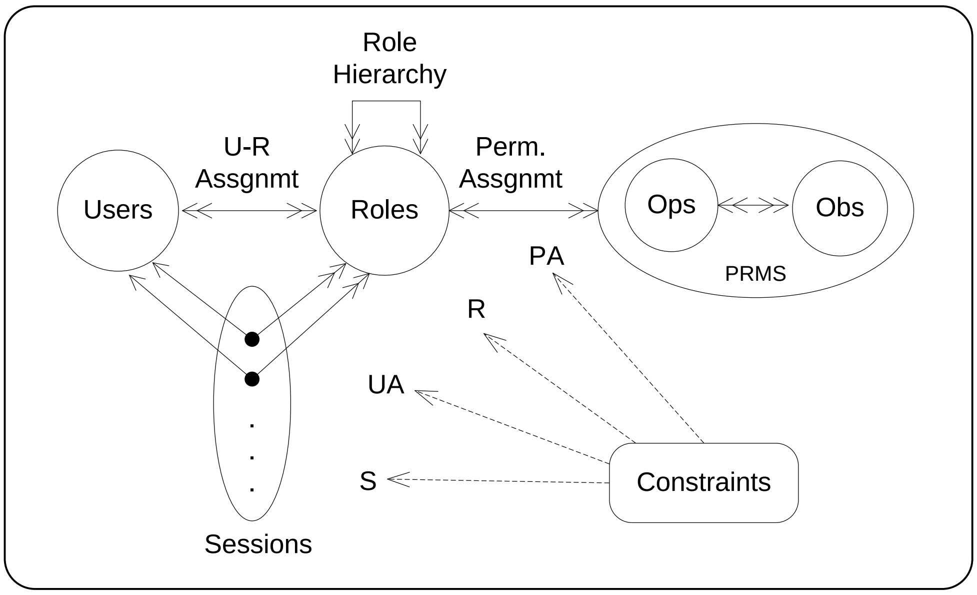


Figure 3.1.: RBAC Model

- $PRMS \subseteq 2^{OPS \times OBS}$, the set of permissions.
- $PA \subseteq PERMS \times ROLES$, a permission-to-role assignment relation.
- $assigned\_permissions : ROLES \rightarrow 2^{PRMS}$, the mapping of role r onto a set of permissions. Formally: $assigned\_permissions(r) = \{p \in PRMS | (p,r) \in PA\}$
- $assigned\_operations : PRMS \rightarrow 2^{OPS}$, the permission to operation mapping, which gives the set of operations associated with permission p.
- $assigned\_objects : PRMS \rightarrow 2^{OBS}$, the permission to object mapping, which gives the set of objects associated with permission p.
- $SESSIONS$, the set of sessions
- $session\_users : SESSIONS \rightarrow USERS$, the mapping of session s onto the corresponding user.
- $session\_roles : SESSIONS \rightarrow 2^{ROLES}$, the mapping of session s to a set of roles. Formally: $session\_roles(s) = \{r \in ROLES | (session\_users(s), r) \in UA\}$
- $avail\_session\_perms : SESSIONS \rightarrow 2^{PRMS}$, the permissions available to a user in a session, $avail\_session\_perms(s) = \cup_{r \in session\_roles(s)} assigned\_permissions(r)$

Figure 3.1 depicts the model with the related constraints to the associations involved. Constraints with respect to S (session), UA (user assignment), R (role), and PA (permission assignments) can be specified within the elaborated models; more specifically, RBAC$_3$ includes static separation of duty constraints amongst roles, and RBAC$_4$ includes dynamic separation of duty constraints amongst activated roles within a given session [SCFY96].

The ANSI RBAC model recognizes two types of role hierarchies—general role hierarchies and limited role hierarchies [ANS04]. General role hierarchies provide support for an arbitrary partial order to serve as the role hierarchy while limited role hierarchies impose restrictions resulting in a simpler tree structure, allowing only a single immediate descendent. And, the standard defines general role hierarchies as following:

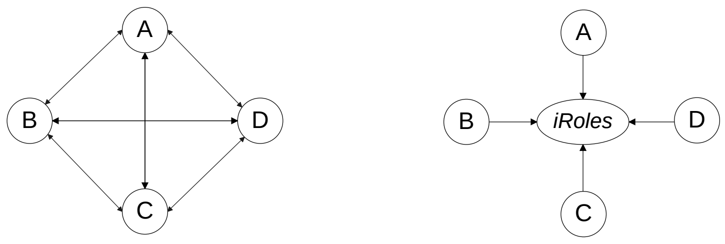
- $RH \subseteq ROLES \times ROLES$ is a partial order on ROLES called the inheritance relation, written as $\succeq$, where $r_1 \succeq r_2$ if and only if all permissions of r_2 are contained in r_1, i.e., $r_1 \succeq r_2 \Rightarrow authorized\_permissions(r_2) \subseteq authorized\_permissions(r_1)$.
- $authorized\_user : ROLES \rightarrow 2^{USERS}$, the mapping of role r onto a set of users in the presence of a role hierarchy. Formally it is expressed as: $authorized\_users(r) = \{u \in USERS | r' \succeq r, (u, r') \in UA\}$.
- $authorized\_permissions : ROLES \rightarrow 2^{PRMS}$, the mapping of role r onto a set of permissions in the presence of a role hierarchy. Formally: $authorized\_permissions(r) = \{p \in PRMS | r \succeq r', (p, r') \in PA\}$.

Interactive Roles

Inter-domain collaborations create, in nature, an over-arching umbrella out of participating collaborators. In such a case, permissions are often given to perform specific tasks or contact certain partners in order to request service provision or resource sharing. Roles defined in such collaborations represent rather functional roles, which is defined as representatives of specific "tasks or functions" [KSS03]; it is rightly so since they are defined as actors of a collection of interactions. If humans are to be assigned as these actors, the definition of the roles must be enhanced to convey additional attributes so that right personnel can be connected to the roles. Moreover, the domain boundaries of each collaborative partners must be respected. A direct access control specification of permissions and users are not favorable due to unnecessary disclosure of organization specific information.

Based on such observations, we have introduced an extra layer of abstraction to RBAC in [LL07] in order to enable decoupling of the organizational roles of each collaborator and the roles within their collaborations. This additional indirection also incorporates the connection between roles in organizational and functional contexts. The concept has been refined as *interactive Roles* (*i*Role). This entity can take other roles or actors in collaboration models as its permissions and can be assigned to other *i*Roles in other collaborations as users as well. The result is a creation of a chain of entitlements traversing different collaborative environments where inter-domain collaborations occur through this intermediation layer.

As an example, consider collaborations amongst European Union public sectors. One of their collaborative scenarios could include an *i*Role called "EU Border Controller". Suppose this role has a task to perform within a certain collaboration definition. For instance, the task could be checking extra personally identifiable information (PII) in order to verify the legitimacy of a person attempting to enter an EU territory. This would derive a privilege such as to access personal information of a certain



(a) Complexity of Role-mapping Approaches (b) Complexity of *i*Roles

Figure 3.2.: Comparison of Complexity of Role-mapping approach(a) and *i*Roles model(b)

database of an EU member state. Based on the description of the *i*Role, EU Border Controller, and its tasks and privileges, each participating EU public sector can then assign the role to someone appropriate in their organizations such as someone with a position analogous to frontier police officer, border guard, or coast guard. Utilizing the indirection layer of *i*Role with the collaborative definitions as their contextual environments, all 27 EU member state public sectors can interact with one another through a uniformed interface in a scalable manner.

The superior scalability of *i*Role model can be clearly illustrated when compared to the bilateral role mapping approach. The bilateral role-mapping approach, in its essence, connects a role of one domain to an analogous role of another domain based on the bilateral agreements [JBBG04, PJ05, DdLGvB06]. Thus, it is unavoidable that it needs to deal with various incongruent problems such as lack of analogous roles and security policy inconsistencies in both semantics and syntax. Attempts of linking existing heterogeneous policies, though many proposals exist, do not yield a scalable decentralized authorization specification. Based on role definitions within the context of collaboration definitions, *i*Roles enables a more scalable decentralized user–role assignment. Figure 3.2 illustrates the different complexities of setting up roles using a role mapping approach (3.2(a)) and doing it based on *i*Roles (3.2(b)).

From the collaborators' point of view, the description of *i*Roles is also closely related to their business scenarios, thus, mapping *i*Roles to their organizational roles can be easily done in an intuitive manner. In other words, roles are horizontally aligned across the organizational boundaries through the definitions of *i*Roles and vertically aligned as well through deriving access control permissions from collaboration processes and linking functional roles to *i*Roles as their permissions. Note, however, that *i*Role does not require the entity from a collaborative partner to be a role. It could be a user, a user group, or a role. It is different from role-mapping solutions for inter-domain authorizations where a role receives privileges of a role it is connected to. This also means when a collaborator internally uses a RBAC-based security policy, the *i*Role model does not scale better in this case but will become the same as bilateral role-mapping approaches.

While user–role assignments are to be done autonomously by each collaborative partner, the hierarchy of *i*Roles is administrated in a centralized manner amongst collaborative partners. Furthermore, the role hierarchies of *i*Roles represent user containment relations where role r_1 "contains" role r_2 as

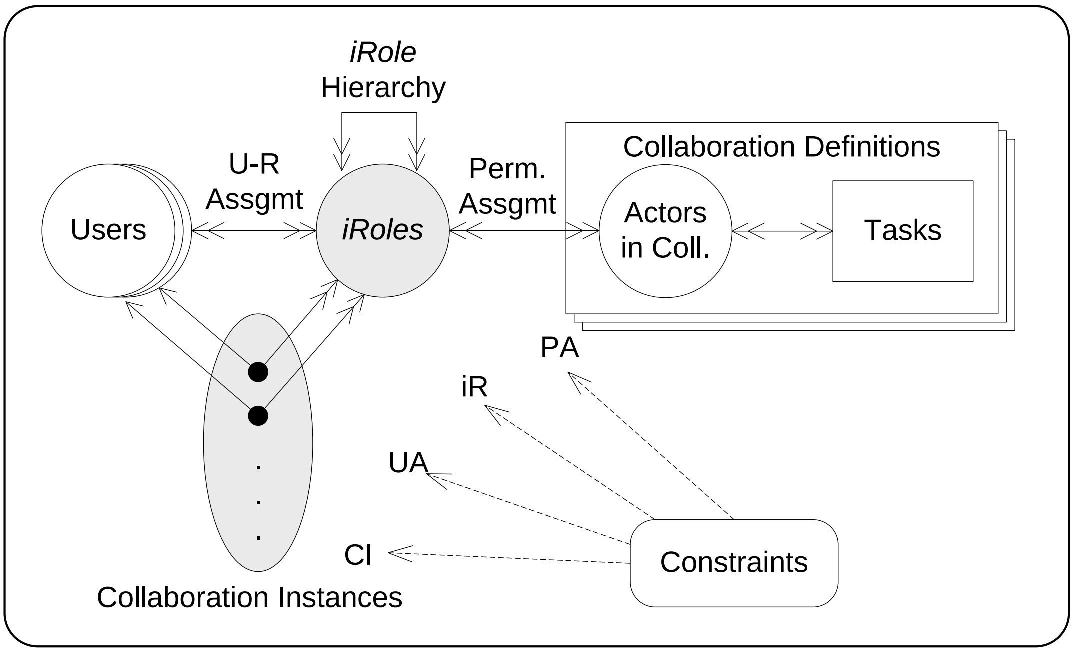


Figure 3.3.: *i*Role Model

all users authorized for r_1 are also authorized for r_2. Going back to the EU Border Controller example, the *i*Role EU Border Controller "contains" all the users authorized for frontier police officer, border guard, or coast guard in different collaborative systems or collaborative partners.

Figure 3.3 shows the logical extension of RBAC (refer to Figure 3.1) with *i*Roles. The new and modified entities introduced are shown as shaded. The pairs of an actor and a set of his or her tasks within a collaborative instance are defined as permissions. When a user is assigned to a certain *i*Role, he or she is entitled to be any of the actors associated with the *i*Role, in the collaborations where the actors play a role. Constraints that are depicted in the figure do not include static separation of duty and dynamic separation of duty. In this work, we deal with cardinality of users to a role and that of role to a permission. Separation of duties is left out for the purpose of simplicity.

Figure 3.3, however, does depict the possibility of organizing *i*Roles in a hierarchical manner and that constraints can be specified with respect to *i*Roles, user assignment, permission assignment, and collaboration instances. Due to its context being within the realm of inter-domain collaborations, *i*Roles do not necessarily show the distinct representation of an organizational position or rank. Therefore, the hierarchy of *i*Roles may contain more than a single tree structure. Moreover, the depth of the *i*Role hierarchy may not possess as deep a level as in a single administrative organization with RBAC policies. Dependent upon the scenarios or the nature of collaborations, it may be beneficial. For instance, when collaborations are on-going frequently, and if there exist enough seniority of interface job descriptions, forming a hierarchy of *i*Roles can be useful. The e-Governmental scenario depicted in Chapter 5 is one of those cases.

Note that *Operations* and *Objects* in RBAC are renamed as *Actors* (*ACT*) and *Tasks* (*TASK*) in the *i*Role model. Sets of actors and tasks are then denoted as Permissions (*PRMS*). Likewise *Sessions*

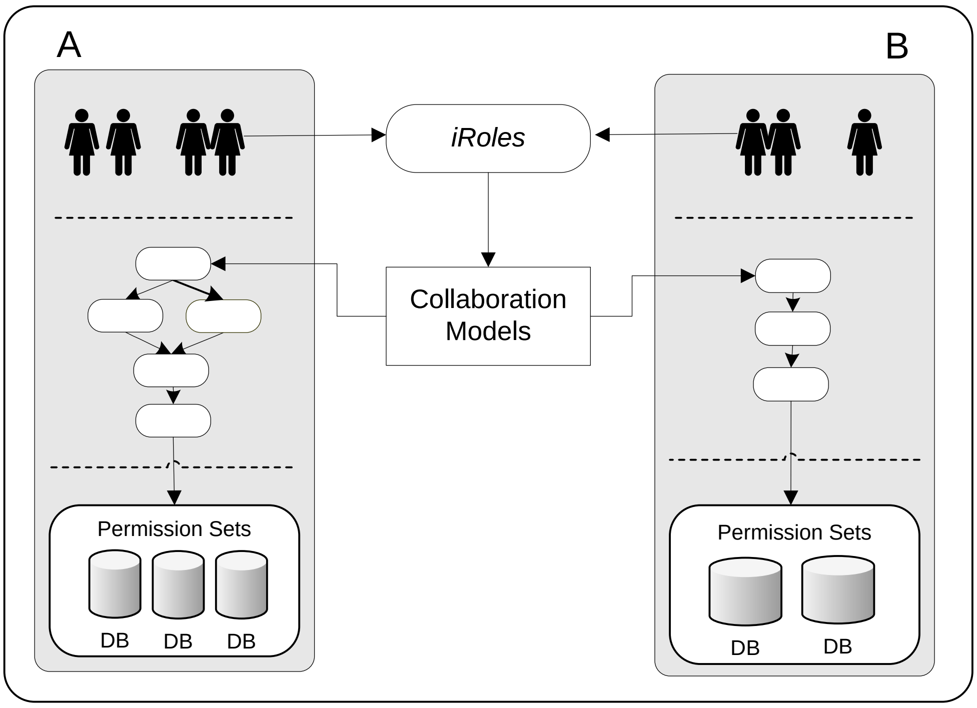


Figure 3.4.: Collaboration Overview with *i*Roles

in the core RBAC model has been replaced by *Collaboration Instances* (*CI*), which are analogous to sessions that map a user to activated roles of the user. Below defines the *i*Role model with role hierarchy, which can be analogous to the RBAC model we described in Figure 3.1:

- *USERS, ROLES, ACT, TASK, PRMS* and *CI* (i.e. sets of users, roles, actors, tasks, permissions, and collaboration instances respectively),
- $PRMS \subseteq ACT \times TASK$, an actor to task assignment relation,
- $PA \subseteq PRMS \times ROLES$, a permission to role assignment relation,
- $UA \subseteq USERS \times ROLES$, a user to role assignment relation,
- $user : CI \rightarrow USERS$, a function mapping each collaboration instance c to the single user $user(c)$ (constant for the collaboration instance lifetime),
- $roles : CI \rightarrow 2^{ROLES}$, a function mapping each collaboration instance c to a set of roles $roles(c)$ $= \{r|(r' \succeq r)user((c),r') \in UA\}$ (which can change with time), collaboration instance c has the permissions $\cup_{r \in roles(c)} = \{a|(a,r) \in PA\}$.

Now how can this model be utilized in an inter-domain collaboration? The overview of such collaboration with *i*Roles is depicted in Figure 3.4. In the figure, there is a collaboration definition between two heterogeneous domains A and B described as “collaboration models”. Some local principals are assigned to particular *i*Roles, which have permissions to carry out certain sets of tasks in the collaborations. Each task is again linked to a set of inter-domain sub-procedures, which eventually access necessary resources in order to complete the task. Users assigned to a particular *i*Role can request a certain resource or service to other administrative domains during the execution of the corresponding collaborations.

Permissions in the inter-domain collaborations represent rather coarse-grained access control. For instance, a valid permission could be able to perform a particular actor in a collaboration carrying out a set of tasks the actor is supposed to carry out. Nevertheless, the tasks defined in the collaborations can be decomposed to specify more fine-grained access control related to the collaborative environment. Any requests within collaborations are to be accompanied with a valid *i*Role certificate. The ultimate access control decision, nevertheless, is to be made by resource owners.

Consider the previous example with *i*Role EU Border Controller again. Suppose Alice from member state A may be assigned to the *i*Role, and thus she is entitled to request a certain PII of Bob whose passport claims he is a citizen of member state B. In order to get her request granted, she needs to present a valid certificate that proves that she is assigned to the *i*Role by a valid authority of domain A and that she actively participates in the corresponding collaborative scenario. Assuming the access control permissions in member state B have been already derived properly from the collaborative task definitions, the authorization can be done without exchanging user attributes or without specifying unified attribute semantics.

3.2. Representation of *i*Role in XACML

The final definitions of *i*Roles are written in XACML for the purpose of maximizing interoperability during the exchange of collaborative system security policies. In this section, the elements of *i*Role and its specification in XACML are explained.

3.2.1. Structure of *i*Role

As shown in Listing 3.1, the structure of *i*Role is divided into three parts. The first part is its name. Name is within a specific namespace similar to the URL structure so that each name can be uniquely identifiable. For instance, the EU Border Controller role's name can be defined as "EU_Border_Controller".

The second part is a list of attributes that must be satisfied by users who will be assigned to the role. These attributes will need to be specified by their types and values in a XML schema format. It can also utilize the "use" field and indicate if the attribute is "required" or "optional". For instance, a list of legitimate positions that can be assigned to *EU Border Controller* can be defined as an enumerated type of a XML schema, and it can be required that a local principal that will hold the *i*Role must hold one of the position listed in the enumerated type. The *EU Border Controller* *i*Role could be frontier police officer, border guard, or coast guard of any EU member state.

```
<Name> ::= <String>
<Description> ::= <String>

<Attributes> ::= <List-of-Attribute>
<List-of-Attribute> ::= <Empty-String>|<Attribute>|<Attribute><List-of-Attribute>
<Attribute> ::= <Enumerated-Type>
```

```
<Permissions> ::= <List-of-Permission>
<List-of-Permission> ::= <Permission>|<Permission><List-of-Permission>
<Permission> ::= <Collaboration-Model><Actor><Affiliation><Constraints>
<Collaboration-Model> ::= <String>
<Actor> ::= <String>
<Affiliation> ::= <String>
<Constraints> ::= <Empty-String>|<Cardinality>|<Static-SoD>|<Cardinality><Static-
    SoD>
<Cardinality> ::= <Roman-Numeral>
```

Listing 3.1: Specification of *i*Role

Lastly, the definition of *i*Role contains a list of permissions. Permissions consist of a collaboration model identifier, name of actor, affiliation and constraints on the actor of the collaboration.

When a collaborator wants to view the structures of *i*Roles and their role hierarchies, the information is transferred from a collaborative system to a participating collaborator's domain in the XACML-RBAC profile format. This allows the collaborators to choose their own selection of XML or XACML-based editors and viewers in addition to understanding the transferred semantics and syntax without extra instructions. Assigning users to *i*Roles on the collaborator's ends can be done by creating corresponding role assignment XACML-RBAC files; this can be also done with the aid of an administrative UI-based toolkit, which is described in more detail in the next chapter on the *i*RBAC framework. The next section describes how the definition of *i*Role is specified in XACML and its RBAC Profile.

3.2.2. XACML and the RBAC Profile

The OASIS standard, eXtensible Access Control Markup Language (XACML) is an applicable solution to express authorization policies. The overall structure of XACML is depicted as a UML class diagram in Figure 3.5 [Mos05]. The root element of a XACML policy is called *Policy Set*, which acts as a container for the rest of the other elements in the XACML specification. To allow generalization and specialization of policies, *Policy Set*s themselves can be arranged in a hierarchical order. Due to this possibility of having a nested structure, it is necessary, when evaluating XACML policies, that the results from several policies may need to be combined as one. Therefore, each *Policy Set* has a *Policy Combining Algorithm* element, giving instruction for logical operations on the sub-results of all involved policies, to lead to a single result. Along with *Policy* and *Policy Set*elements, there is *Rule* element in the top level of XACML specification. The *Rule* element contains a boolean expression that can be evaluated in isolation. These rules are intended to be managed by a Policy Administration Point (PAP) as a basic unit of management and to be used in multiple policies. As for rendering an authorization decision, it must be possible for a Policy Decision Point (PDP) to combine multiple policies.

The left subtree of the root node, as shown in Figure 3.5, is the *Target* element. It may have *Policy* elements, which apply *Rule* elements to its sibling *Target* node. Based on the *Condition* elements of a *Rule* element, which gets evaluated as "true" or "false", an access control verdict turns out to be either

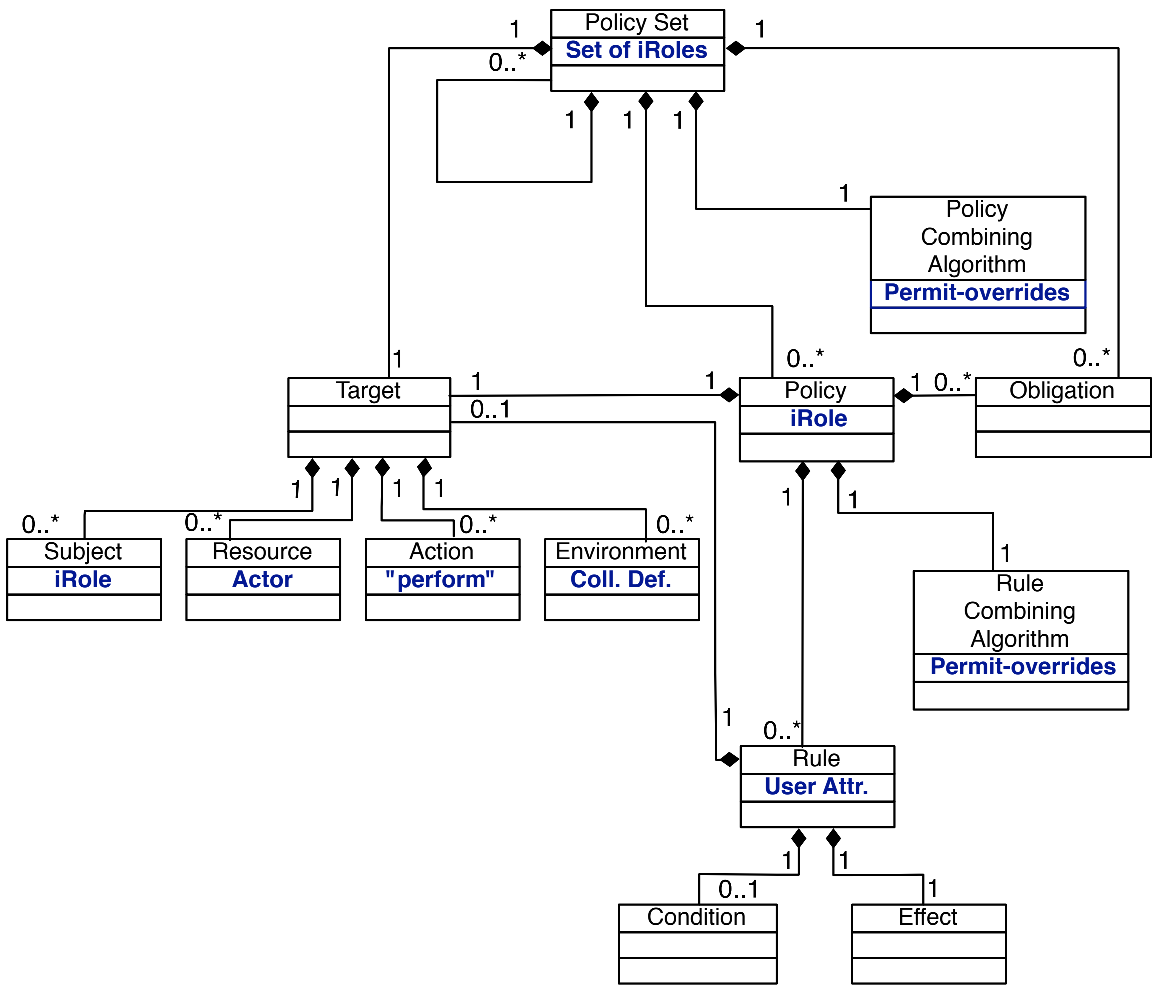


Figure 3.5.: Specification of XACML Core V. 2.0

permit or *deny*. A *Target* element includes four leaf nodes: *Subject*, *Resource*, *Action*, and *Environment*. *Subject* represents entities that want to get an access permission while *Resource* represents an object for which the access request is made. Description of the kind of access is specified in *Action* element. This allows multiple types of actions to be operated upon a resource. Lastly, the *Environment* element allows to define constraints about the environment where the access can be performed; these could be, for instance, temporal or spacial constraints. If any of the child elements of a *Target* element is missing, that means, any value would be permissible for that omitted element. For example, a missing *Subject* element indicates that any user can use the policy belonging to its *Target* to get access.

The Role-based Access Control (RBAC) profile of XACML enables to specify the concept of role and role hierarchy from the RBAC model [And05]. With this profile, an authorization service runs in a chain procedure from permission to role and role to user; at the same time, it is still possible to give permissions directly to users as specified in the core specification. The RBAC profile basically extends *Subject* of a *Target* in a way that roles can be legitimate values of the *Subject* field. Roles represent sets of privileges [SCFY96]. Users may take a part of multiple number of roles, and multiple number of users can be assigned to a single role. A functionality called “multi-role permissions” [And05] is also specified in the profile so that more than one role can be required from a user to get an access to a resource. The profile uses the tag “&role;” [And05] in the attribute field of *Subject* or *Resource* element to indicate that this element is a role and not a user. By utilizing an attribute field, this profile introduces no changes to the original XACML schema file.

Figure 3.7 shows the way RBAC profile utilizes XACML entities. Figure 3.7(a) shows how user–role assignment is being specified in XACML format while Figure 3.7(b) shows how role and permission are specificed. As shown in these figures, multiple policy files are created by XACML-RBAC profile in order to specify the RBAC model. Role is defined within one policy set. The name of a role is defined as an attribute of *Target* element of *Policy* element. One role is defined within one *Policy* element. Permissions are defined as *Target* elements of *Rule* element of a *Policy* element. Role name is defined as an attribute of *Subject* element, and *Resource* attribute contains a resource in interest. As for the *i*Role related permissions, these four sub-elements describe a specific permission of a collaboration definition. The *Resource* field contains an actor in a collaboration definition, which can be performed by the given *i*Role. The *Action* entity specifies what kind of access is granted (e.g. “perform” would be one of the legitimate actions). The *Environment* entity in the *i*Role related permissions would hold the identity of collaboration definition in consideration.

Notice that each permission *PolicySet* represents a set of permission directly related to a role given as an attribute field of its *PolicySet*. Any other permission sets legitimate due to the seniority of role hierarchy is represented by *PolicySetIdReference* field which references a different permission policy. Thus, one could conclude that the role hierarchy of XACML-RBAC profile is indirect and not explicit. Figure 3.7(b) shows how a role is connected to its permission policy and how a senior permission policy includes its junior policy set by the arrows that point to different policy files by referencing. We foresee that this can cause problems in terms of reduction of management efficiency as well as a possible source of introducing errors into the policy specifications.

The collection of *Policy* elements each defining a single *i*Role of a single name space is grouped together under a *PolicySet* element. The *RuleCombiningAlgorithm* and *PolicyCombiningAlgorithm* elements are respectively used to combine multiple rules and policies. The default of both of those algorithms is "Permit-overrides" since the very basic rule to start with is "Deny-all". The *Rule* elements are evaluated for user–role assignment as well as for the authorization decision making process.

3.3. Preservation of Consistency of Policies

Whenever a role defined in one domain is being mapped to another in another domain, there is the potential risk of violating consistency properties. Within the context of access control, the consistency property can be simply defined as "There are no conflicting rules governing the access to a data item" [Gol11]. This section discusses the potential threat of inconsistency property using *i*Roles and a procedure to avoid encountering such violations.

3.3.1. Potential Violation of Consistency Property

Since *i*Roles are visible in the collaborative realm, individual local policies are completely transparent to them. Whether they use RBAC or any other access control model internally, *i*Roles enables collaborative partners to cope with RBAC-based security policies in the inter-domain realm where their collaborations occur. Thus, local security policies may map users, roles, or even user groups to *i*Roles. When a local security policy is also specified with RBAC with a role hierarchy, however, violation of security principle can be detected. The definition of this security principle and its potentially conflicting autonomy principle are defined as following according to Gong and Qian and Shafiq et al. [GQ96, SJBG05]:

- **Autonomy Principle:** If an access is permitted within an individual system, it must also be permitted under secure interoperation.
- **Security Principle:** If an access is not permitted within an individual system, it must not be permitted under secure interoperation.

One can surely argue that the principle of autonomy stated above is too strong a restriction in practice. For instance, local domain policy could have specified a dynamic separation of duty in such a way that when a particular role of a user is activated through an inter-domain collaboration, some local roles of the same user must not be activated. Such a case is valid; it is actually a fine usage of one of the major advantages of deploying security policies that allows fine-grained access control. However, these principles do point out the conflicting goals posed by autonomy and security.

A number of publications on this issue of security violation detection in secure interoperation has been proposed. In [KAMCM00], Kapadia et al. define two types of security violations when roles are mapped across domains—***infiltration*** and ***covert promotion***. Infiltration can occur when a role is mapped to an unintended role in another domain through yet another intermediate domain. In such case, the original role can get access to the third domain through its immediate mapping to the role in the intermediate domain. The root of this problem sprouts from the fact that a role mapping has crossed multiple boundaries of administrative domains while role-mapping is not transitive. Covert promotion is the case when a role mapping causes a role to return to its original domain with a more privileged role than the original role. This violation is also generally called as elevation of privilege.

In [SJBG05], Shafiq et al. respond to this problem with a policy composition framework to generate a secure interoperation policy by an automated generation of role mapping between two given domains and resolving detected conflicts. Their role mapping enables inter-domain information and resource sharing through the mapped roles. As for the inconsistency of the composite policies, they specifically deal with three types of conflicts: 1) role-assignment violation, 2) role-specific-SoD violation, and 3) user-specific SoD violation. An integer programming-based approach is being used to optimize maximal data sharing via cross-domain roles.

In [WSY$^+$08], Wang et al. propose an efficient method of detecting such security violation called *Minimum Detection Method*. In order to use their method, both global role system and local role system must be known where global role system is the conjunction of all roles from all domains collaborating. This is theoretically permissible, but in reality, to collect all role systems of collaborators is rather infeasible.

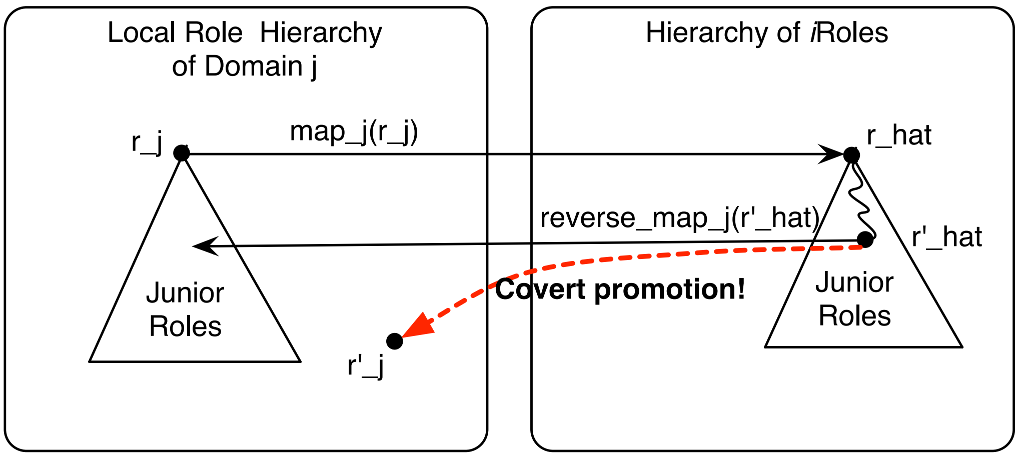
Khoury et al. also provide a method of checking consistency of security policies on role assignment for secure interoperability in [KCH08]. Their underlying assumption on a role translation across domains is to either import or export roles. And, given two sets of permissions representing two roles from two different domains that are to be mapped, they propose to use Description Logic formalisms to create *bridge rules* between the two given roles.

Aforementioned solutions of the interested security violations are providing detection algorithms or role translating algorithms. These may work for randomly generated arbitrary roles, but in the real world situations, they may not provide a remedy. First of all, the algorithms to detect such security violations require the entire role hierarchies of the involved domains as input. Such can not be provided in the first place. Moreover, roles may be defined in such a way that translating from one role to another by the hierarchical positions may not be sensible. In addition, generating an entire role hierarchical set of a domain may not be feasible. The next section explains how these potential violation of security policies are dealt with in relating to *i*Roles.

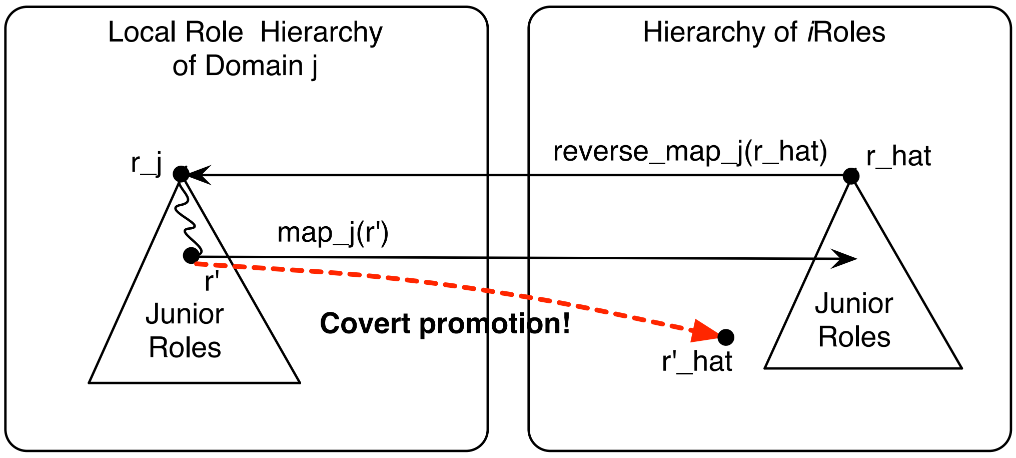
3.3.2. Inconsistency Prevention

Having a set of *i*Roles between the domain specific roles eliminates the necessity of sharing a domain specific role hierarchy with other collaborators. It also eliminates the possibility of having infiltration since role mapping is allowed only between *i*Roles and local roles. On the other hand, the covert promotion violation can occur if a domain decides to use the definition of *i*Roles as a part of their local roles' permission sets. In other words, when role–permission assignment is operated upon *i*Roles in such a way that *i*Roles are used as sets of permissions, covert promotion may occur.

Given the fact that the local administrators have full control over user–role assignments and though discouraged from doing so may use *i*Roles as local permission sets, the following paragraph provides an explanation of the situations when covert violations occur and a procedure to assign a user to an



(a) Case 1: When a junior role of $\hat{r}$ has been granted to a user of a non-junior role of r_j



(b) Case 2: When a junior role of $\hat{r}$ has been assigned to a non-junior role of r_j

Figure 3.6.: Possible Covert Promotion Cases

*i*Role without causing a covert promotion.

Figure 3.6 depicts two situations when covert promotion violation occurs. The first case is when a local role r_j is being mapped to *i*Role $\hat{r}$. In that situation, if all junior roles of $\hat{r}$ are being mapped to junior roles of r_j, there is no violation. However, if there is any junior role of $\hat{r}$ which has been granted to a non-junior role of r_j, then r_j gets more privileges than it should have through the junior role of $\hat{r}$ that has been reversely mapped to a non-junior role of r_j. Here, a non-junior role of r_j can be any role in the local role hierarchy of Domain j including senior or sibling roles of r_j as well as non-related roles existing in the role hierarchy. This case is depicted in Figure 3.6(a).

The second situation, which is illustrated in Figure 3.6(b), occurs when an *i*Role $\hat{r}$ is being mapped to a local role of Domain j r_j. In that case, if all junior roles of r_j are mapped to the junior roles of $\hat{r}$, there is no possibility of covert promotion. However, if there is any junior role of r_j which has been mapped to a non-junior role of $\hat{r}$, then with the similar reasoning as the former case, those who have been assigned to *i*Role $\hat{r}$ gets more privilege than it should from the *i*Role hierarchy when the security policy of Domain j is in use. In this case, the extra privilege $\hat{r}$ gets belongs to $\hat{r'}$.

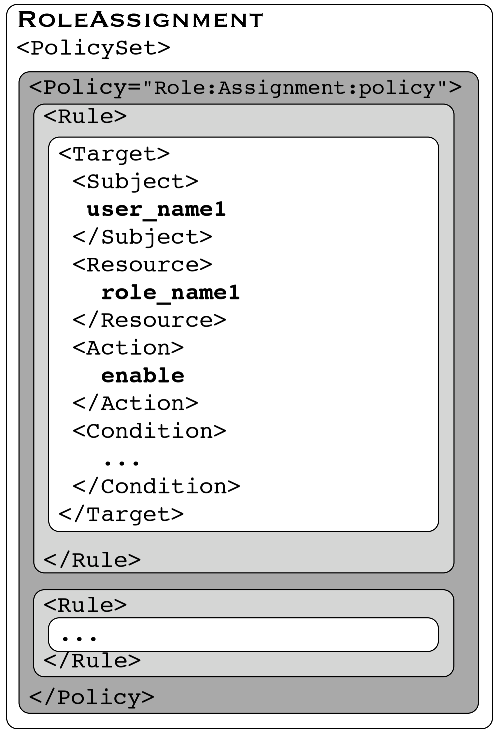
Below are the necessary definitions for the covert promotion violation checks:

- *iROLES*: is a set of *i*Roles, $\hat{\succeq}$ is the role hierarchy of *iROLES*
- $ROLES_j$: is a set of Domain j's local roles, $\succeq_j$ is the role hierarchy of $ROLES_j$
- map_j: $ROLES_j \rightarrow iROLES$
- $reverse\_map_j$: $iROLES \rightarrow ROLES_j$
- Junior roles of $\hat{r}$: $\hat{J}(\hat{r}) = \{\hat{r'} | \hat{r} \hat{\succeq} \hat{r'}\}$
- Junior roles of $r \in ROLES_j$: $J_j(r) = \{r' | r \succeq_j r'\}$

Given the definitions above, perform the following checks so that possible covert promotion security violations can be detected when assigning *i*Role $\hat{r}$ to Domain j's role r_j:

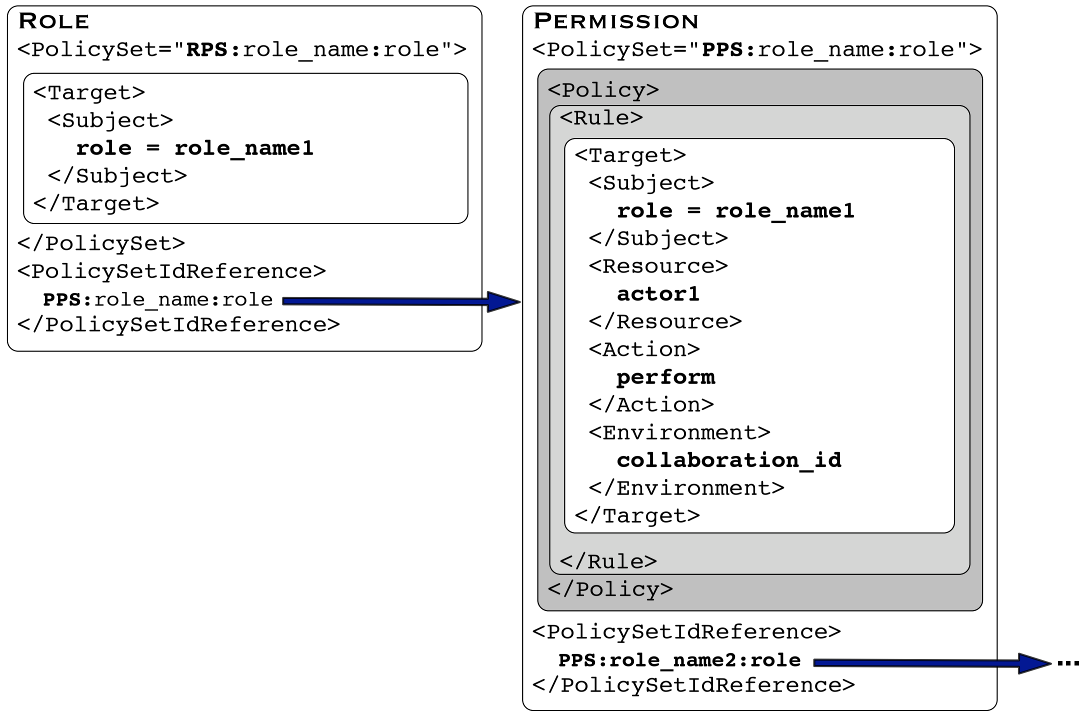
- **Case 1: All junior roles of $\hat{r}$ must have been granted to the users of junior roles of r_j:**
  $\{r | reverse\_map_j(\hat{r'}) = r, \hat{r'} \in \hat{J}(map_j(r_j))\} \subset J_j(r_j)$
- **Case 2: All junior roles of $\hat{r}$ must have been assigned to junior roles of r_j:**
  $\{\hat{r'} | map_j(r') = \hat{r'}, r' \in J_j(reverse\_map_j(\hat{r}))\} \subset \hat{J}(\hat{r})\}$

It is noteworthy to mention that, in general, importing permission sets from one domain to another causes various problems such as name clashes, syntax conflicts, and the security violations as discussed earlier. Thus, this option must be used with discretion. A couple of other important assumptions made here are 1) that the local role hierarchy is composed as a single tree cluster and 2) that there is trust relationship established amongst the collaborators. Thus, each partner entrusts the others to assign *i*Roles to appropriate local roles. This is a reasonable assumption since there is an incentive for all collaborators to wish to bring their collaborations to successful ends.



```
ROLEASSIGNMENT
<PolicySet>
<Policy="Role:Assignment:policy">
<Rule>
<Target>
 <Subject>
  user_name1
 </Subject>
 <Resource>
   role_name1
 </Resource>
 <Action>
   enable
 </Action>
 <Condition>
   ...
 </Condition>
</Target>
</Rule>
<Rule>
...
</Rule>
</Policy>
```

(a) Specification of User–Role Assignment



```
ROLE
<PolicySet="RPS:role_name:role">
<Target>
 <Subject>
   role = role_name1
 </Subject>
</Target>
</PolicySet>
<PolicySetIdReference>
  PPS:role_name:role
</PolicySetIdReference>
```

```
PERMISSION
<PolicySet="PPS:role_name:role">
<Policy>
<Rule>
<Target>
 <Subject>
   role = role_name1
 </Subject>
 <Resource>
   actor1
 </Resource>
 <Action>
   perform
 </Action>
 <Environment>
   collaboration_id
 </Environment>
</Target>
</Rule>
</Policy>
<PolicySetIdReference>
  PPS:role_name2:role
</PolicySetIdReference>
```

(b) Specification of Role and Permission

Figure 3.7.: Specification of XACML-RBAC

4. *i*RBAC Framework

Let everyone sweep in front of his own door, and the whole world will be clean.
– Johann Wolfgang von Goethe

This chapter presents how to realize the model described in the previous chapter. The first section explains the high-level design overview. It explains which parts are decentralized and which parts are shared among collaborators. The following sections then explain in detail how the *i*Role-based policies for inter-domain collaborations are constructed. Implementation of the modules that enable decentralized autonomous management are presented in the end.

4.1. High-level Design Architecture

The *i*RBAC framework consists of two major components: 1) a component which assists local security administrators to assign local principals to *i*Roles; and, 2) a component which builds an *i*Role-based RBAC system including a role hierarchy. While the former component resides on each collaborator's domain, the latter component is being administrated in a centralized manner and is to be available to all collaborators.

Figure 4.1 represents a high-level overview of the architecture of the *i*RBAC framework. The top part represents the former component which resides in each collaborator's domain. There the mapping between user–role is to be performed autonomously from each collaborator's site based on the comparisons of user attributes. This component is being explained in 4.3 with a UI-based tool local security administrators can use. The latter component is being depicted by the shaded middle part. It consists of computing entities that generate *i*Role definitions and a data repository that stores the collaboration models and the specifications of *i*Roles. More specifically, permission sets are extracted out of collaboration definitions such as WS-CDL and WS-BPEL that are represented by the lower part of the diagram.

In the following sections, the overview of our approach to role engineering is presented. Afterwards, each of the introduced components is described in detail. Lastly the administrative components to manage the resulting RBAC-based authorization policies are illustrated. Note that Section 6.2 explains in depth the existing methodologies for building an RBAC system.

4.2. RBAC System Builder

Role engineering, as outlined by Coyne [Coy96], is a systematic process of identifying the activities of a single user and defining this as a role, which is also known as "the costliest part of migrating to an RBAC system" [MLL$^+$09], often requiring manual intervention due to the input data being noisy and

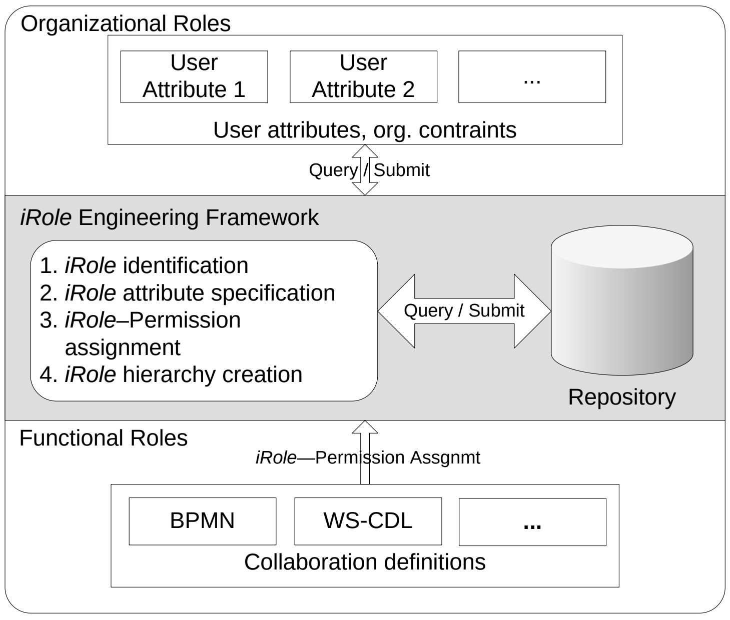

Organizational Roles
User Attribute 1
User Attribute 2
...
User attributes, org. contraints
Query / Submit
iRole Engineering Framework
1. *iRole* identification
2. *iRole* attribute specification
3. *iRole*–Permission assignment
4. *iRole* hierarchy creation
Query / Submit
Repository
Functional Roles
iRole—Permission Assgnmt
BPMN
WS-CDL
...
Collaboration definitions

Figure 4.1.: Architecture of *i*RBAC Framework

incomplete. At minimum, an RBAC system defines roles, permissions, and role–permission relationships.

4.2.1. Methodology

Aiming to provide a mechanism for developing consistent role sets not only from an organizational point of view in a business context but also from a viewpoint of technical specifications where collaborative operations are defined, we take a methodology of combining top-down and bottom-up approaches. From the operational level, we take inter-domain collaboration specifications. They could be a suite of specifications in BPMN, WS-CDL, WS-BPEL, etc. So long as the specifications include roles, the choice of the design languages would not matter; all of the aforementioned specifications, however, include the concept of roles in their specifications. From the organizational level, the entities that are to be connected to *i*Roles should contain matching attribute values in order to be assigned to proper *i*Roles. This process may require interactions between the inquiry entities and the framework to query required attributes and submit the values to the framework and receive the corresponding *i*Role. This is depicted in the upper part of Figure 4.1.

Given collaboration definitions, the definitions of *i*Roles and their permissions must be deduced. As described in the previous chapter, the *i*Role model takes the sets of actors of collaboration definitions and their tasks as permission set. The *i*Role–Permission assignments are to be done in a semi-automated method. Based on the pairs of actors and tasks extracted from collaboration definitions, potential *i*Roles are automatically defined. Duplicates are eliminated by reviewing actors, affiliations, and tasks. Human inspection is then necessary to check that the permission grouping has been done properly and to give meaningful names and attributes to the newly discovered *i*Roles. Once this step is completed, a role hierarchy can be generated. The resulting specifications are then stored in a shared repository so that local policy administrators can retrieve *i*Role definitions and a graphical view of their

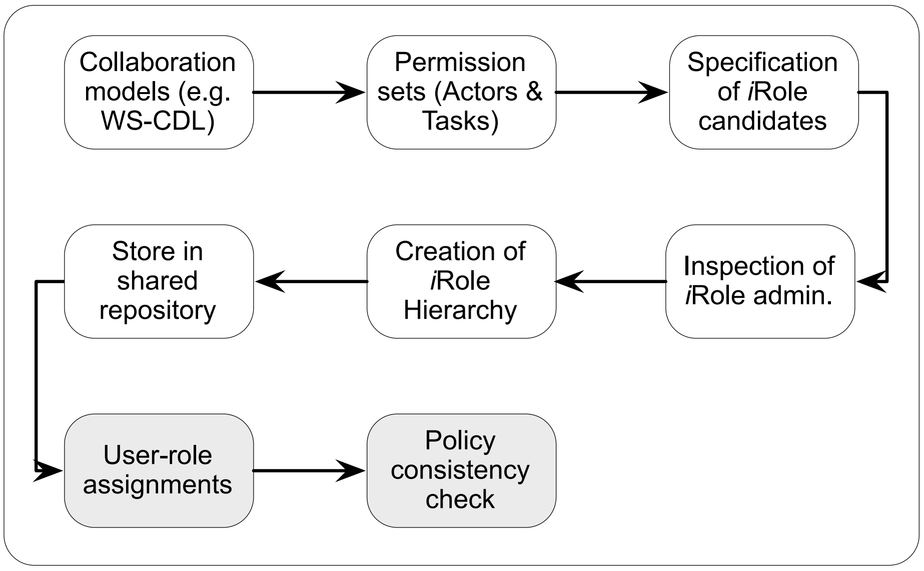

Collaboration models (e.g. WS-CDL)
Permission sets (Actors & Tasks)
Specification of *i*Role candidates
Inspection of *i*Role admin.
Creation of *i*Role Hierarchy
Store in shared repository
User-role assignments
Policy consistency check

Figure 4.2.: *i*Role Engineering Process

role hierarchy so that they can map proper local principals to the given *i*Roles. Figure 4.2 shows the overall *i*Role engineering process; note that the last two shaded steps are to be done in a decentralized manner by the collaborators' security administrators.

The list below explains a step-by-step process that happens in the *i*RBAC framework to obtain the complete specification of *i*Roles. Note that the whole process can be reiterated multiple times before the final definition is specified:

1. Import a suite of inter-domain collaboration definitions from the data repository (refer to Figure 4.1).

2. Extract all the actors from the given collaboration definitions.

3. Identify identical actors by comparing their names and affiliations and remove duplicates.

4. Define a set of organizational roles required for the designated set of collaborations. Name the roles with unique name and affiliation using namespaces similar to how SPKI builds unique names [EFL+99].

5. Define user attributes as criteria of assigning user–role assignments. Each attribute consists of a pair of name and value where the entire set of possible user attribute values are defined as type.

6. The result from Step 3 is a set of permissions as functional roles while the result of Step 5 represents the organizational roles whose names are for the *i*Roles. Assign the sets from Step 3 to the results from Step 5 as *i*Roles' permission sets. This step represents the role–permission assignments.

7. Export the resulting *i*Role definitions in the XACML format into the data repository (refer to Figure 4.1).

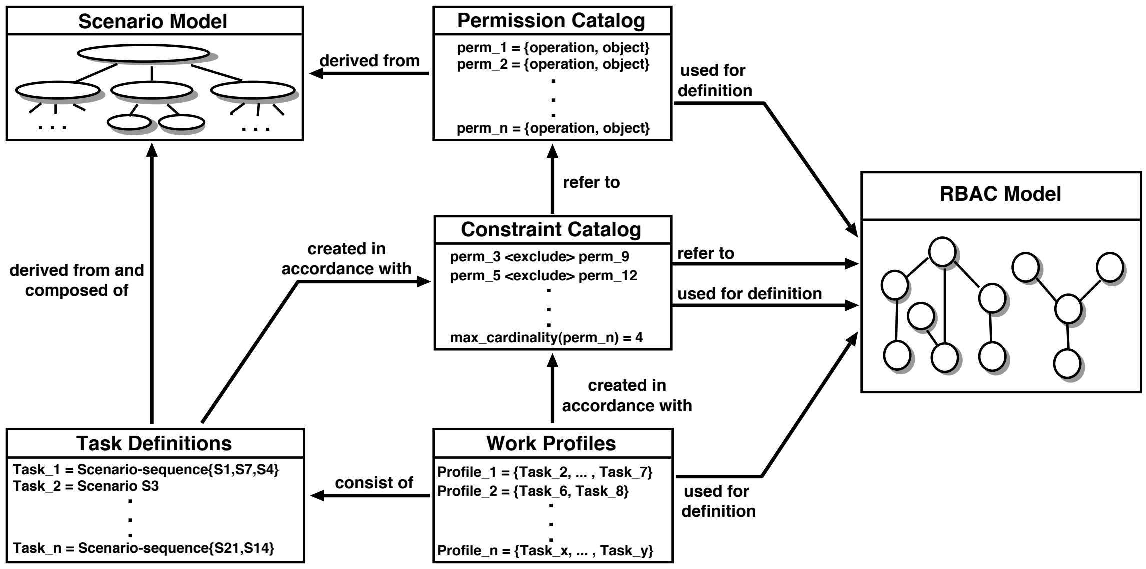

Scenario Model
Permission Catalog
perm_1 = {operation, object}
perm_2 = {operation, object}
perm_n = {operation, object}
derived from
used for definition
refer to
RBAC Model
Constraint Catalog
perm_3 <exclude> perm_9
perm_5 <exclude> perm_12
max_cardinality(perm_n) = 4
created in accordance with
refer to
used for definition
derived from and composed of
created in accordance with
Task Definitions
Task_1 = Scenario-sequence{S1,S7,S4}
Task_2 = Scenario S3
Task_n = Scenario-sequence{S21,S14}
consist of
Work Profiles
Profile_1 = {Task_2, ... , Task_7}
Profile_2 = {Task_6, Task_8}
Profile_n = {Task_x, ... , Task_y}
used for definition

Figure 4.3.: Scenario-based Role Engineering [NS02]

This methodology is similar to that of scenario-based role engineering by Neumann and Strembeck [NS02] in that the permission sets are derived from tasks detected from scenarios. Figure 4.3 depicts the overview of their methodology. Based on their model, task definitions and permission catalogues are derived from a scenario which visually breaks down tasks into a group of smaller operations. Another catalogue called Constraint Catalog is built along the way as Permission Catalog and Work Profile are derived. Afterwards, out of these three components, an RBAC system is built. Therefore, their scenarios are rather abstract descriptions of work and hierarchical break-down's of tasks. The derivation relies upon human understandings of the work profiles.

Unlike theirs, our methodology relies on computing power for parsing out the parts of scenarios composed in one of the standard formats and producing candidates of permission sets automatically. Grouping the candidate permission sets into meaningful *i*Roles is still relying on human supervision. As far as the user–role assignment is concerned, it is performed from within each collaborator's realm; it is explained in more detail in Section 4.3.

4.2.2. Role–Permission Assignment

The role–permission assignment is done based on the collaboration definitions. First possible permissions of *i*Roles are extracted from the given collaboration definitions; the actual components extracted from the definitions vary dependent upon which syntax has been applied to the given collaboration definitions. According to the overview of the methodology employed as described in Figure 1.2, this phase correspond to the steps boxed around the grey rectangle in the middle – it is responsible for 1) identifying permissions, 2) deriving roles, 3) building a role hierarchy, and 4) analysing policies.

As for the tool that supports policy administrators to select actors from a given collaboration definition as permissions of a given *i*Roles, Figure 4.4 shows the graphical user-interface developed in Java

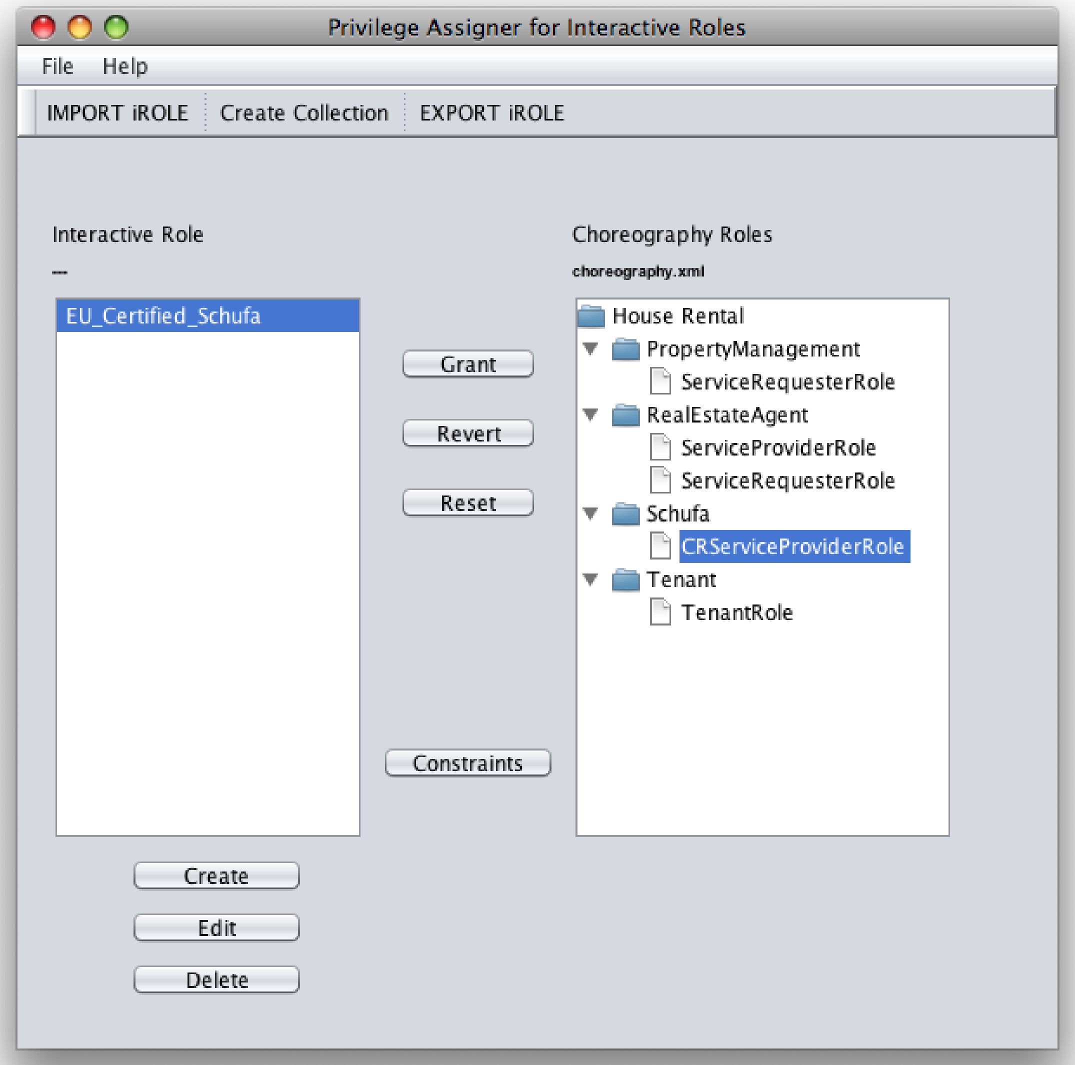


Figure 4.4.: User Interface of *i*Role Builder

with various libraries for XML processing such as AXIS2[1]. On the upper right pane of the figure, the extracted actors from collaboration definition are displayed. From the upper left pane, a new *i*Role can be created; or, existing *i*Role definitions can be loaded. Afterwards, the actors on the right panes can be assigned to a selected *i*Role as its permission. When saved, the corresponding XACML policy file gets created according to the UML class diagram presented in Figure 3.5.

4.2.3. Permission Extraction from Collaboration Process

Collaboration definitions specify how collaborations must proceed. The form of collaboration processes can vary from a workflow type to a set of dynamic interactions amongst different web-services. Therefore, different collaboration processes definitions entail different components to extract permis-

[1]http://axis.apache.org/axis2/java/core/

sions from the given processes. In this study, we have selected WS-CDL (Web Services Choreography Description Language) as a collaboration process definition language. WS-CDL is one of the Web-Services building blocks, which specifies a language to describe interactions between distinct web services. It takes a global point-of-view of, for instance, inter-domain collaborations without notion of a centralized coordinator as in WS-BPEL. The language is based on the XML syntax, and a graphical editor is available as an Eclipse plug-in by a tool known as *pi4soa*[2]. The elements of WS-CDL can be divided into two parts: 1) a dynamic part that defines interactions with structural activities and workunits, and 2) a static part that describes invariants such as role types and relationships.

The elements of WS-CDL strongly tied to actors are as following: *roleType*, *Relationshiptype*, and *participantType*. Below we introduce the structure of each of those element types [KBR+05]:

- *roleType*

  A *roleType* contains a list of observable behaviors a participant can exhibit in order to interact with other collaborative partners. For example, the "EU Border Controller" *roleType* is associated with requesting information on an EU member state citizen according to Need-to-Know based principle related to his or her working duty. Likewise, an "EU Police Information Base Unit" *roleType* is associated with providing the information requested.

  The syntax of the *roleType* construct is:

  ```
  <roleType name=NCName>
     <behavior name=NCName interface=QName />+
  </roleType>
  ```

  Listing 4.1: Construct of *roleType*

  The attribute name is used to specify a distinct name for each *roleType* element declared within a choreography package. Within the *roleType* element, a behavior element specifies a subset of the observable behavior a participant exhibits. A *roleType* must contain one or more behavior elements. The attribute name within the behavior element is used to specify a distinct name for each behavior element declared within a *roleType* element. The behavior element defines an optional interface attribute, which identifies a WSDL interface type. This is to indicate specific Web Service interfaces to be used.

- *RelationshipType*

  A *RelationshipType* identifies the *roleTypes* and behaviors, where mutual commitments must be made for collaborations to be successful. Back to the EU border control scenario, the *RelationshipTypes* between a "EU Border Controller" and a "EU Police Information Base Unit" could include:

  - A "Information Requesting" *relationshipType*, for the EU Border Controller to inquire a certain information to EU Police Information Base, and

[2]http://sourceforge.net/projects/pi4soa

  - A “Information Providing” *relationshipType* to allow the EU Police Information Base to provide the requested information to EU Border Controller.

  Although *RelationshipTypes* are always between two *roleTypes*, choreographies involving more than two *roleTypes* are possible. In that case, two *RelationshipType* definitions are required to describe the multi-lateral relationships completely. For example, a request for a certain piece of information can also be made by EU Border Controller to EU Police Liaison Officer. However, the final reply can be sent to EU Border Controller from EU Police Information Base Unit. In this case, a couple of more specific *RelationshipType* can be declared as following:

  - A “BCPLO_Requesting” *relationshipType* between the “EU Border Controller” and the “EU Police Liaison Officer”, and
  - A “BCIBU_Providing” *relationshipType* between the “EU Border Controller” and the “EU Police Information Base Unit”.

  The syntax of the *relationshipType* construct is:

```
<relationshipType name=NCName>
   <roleType  typeRef=QName behavior=list of NCName />
   <roleType  typeRef=QName behavior=list of NCName />
</relationshipType>
```

Listing 4.2: Construct of *relationshipType*

  The attribute name is used to specify a distinct name for each *relationshipType* element declared within a choreography package. A *relationshipType* element requires having exactly two *role-Type* defined. Each *roleType* is specified by the *typeRef* attribute within the *roleType* element. The “QName” value of the *typeRef* attribute of the *roleType* element ought to reference the name of a *roleType*. Within each *roleType* element, there are optional attribute called *behavior*. It basically identifies what kind of observable commitment a participant must comply with. The value of this attribute is a XML-Schema list of *behavior* types belonging to this *roleType*. If there is no *behavior* attribute specified, then all *behavior* attributes of this *roleType* must be considered as to be committed. If the *behavior* attributes are present, then only those listed are the ones to be committed.

- *participantType*

  A *participantType* groups together those parts of the observable behavior that must be implemented by a participant. A logical entity or organization may be represented by more than one *participantType* within a choreography definition.

  The syntax of the *participantType* construct is:

```
<participantType name=NCName>
   <roleType typeRef=QName />+
</participantType>
```

Listing 4.3: Construct of *participantType*

The attribute *name* is used to specify a distinct name for each *participantType* element declared within a choreography package. Within the *participantType* element, one or more *roleType* elements identify the *roleType* that must be implemented by this *participantType*. Each *roleType* is specified by the *typeRef* attribute of the *roleType* element. The “QName” value of the *typeRef* attribute of the *roleType* element must reference the name of a *roleType*.

An example of *roleType* element of *participantType* definition is as following:

```
<roleType typeRef=tns:EUBorderControllerType />
```

Listing 4.4: Reference to *roleType*

Notice that a reference to an existing *roleType* is made as above.

These constructs of WS-CDL are used to specify actors and their relationships within collaborations. Consequently, the definitions of actors are extracted and collected as possible permission sets that are to be assigned to *i*Roles involoved.

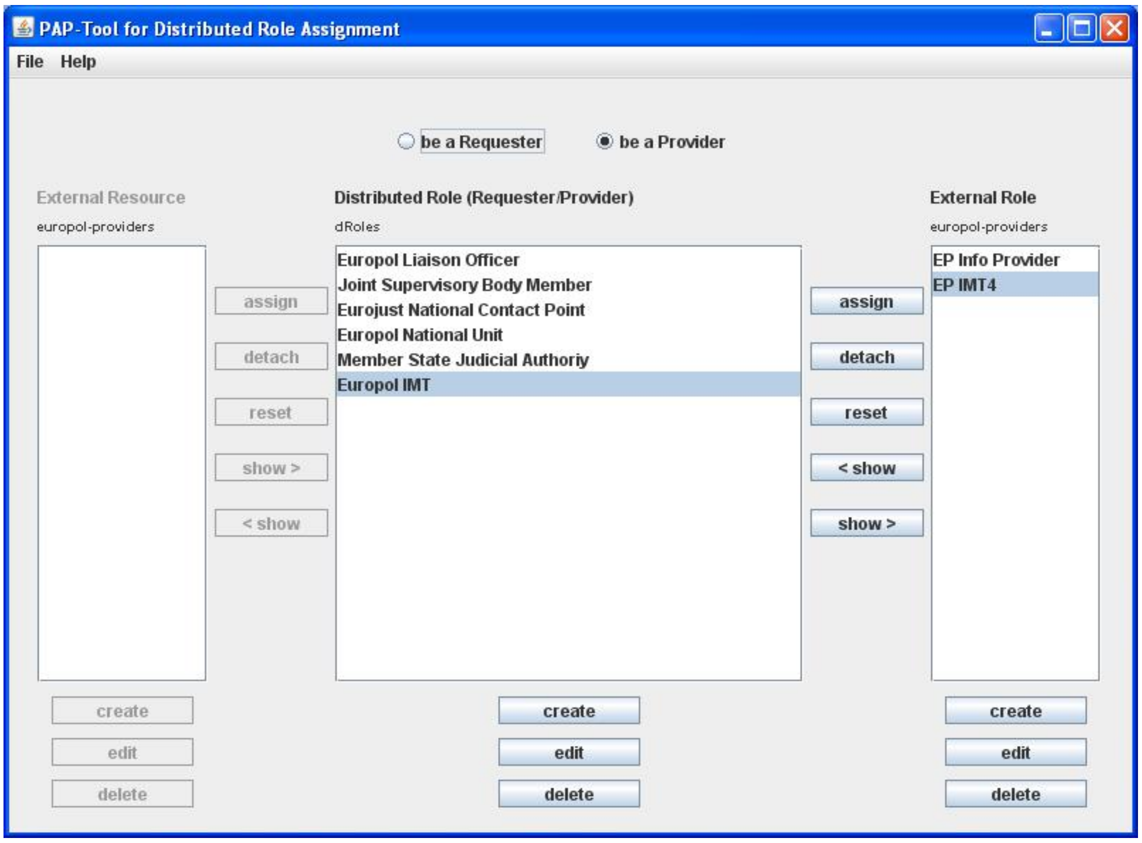
4.2.4. *i*Role Hierarchy

In theory, the RBAC role hierarchy should improve the efficiency of the policy evaluation as well as management by the inheritance relationship between junior and senior roles. However, in reality, its representation can be as ambiguous as the various definitions of roles. Some role hierarchies reflect the line of authority or rank in an organization while others represent the inheritance of functional permission sets. Depending upon which aspect has been more emphasized, the resulting hierarchies can form conflicting structures in other viewpoints. For instance, permission inheritance property in functional role hierarchy may breach a least privilege security principle or cause covert permissions.

4.3. User–Role Assigner

The user–role assignment process can be set up prior to the execution time of collaborations or dynamically during the collaborations. We present a UI-based tool that has been developed to assist collaborators' administrators to map local principals to the pre-defined *i*Roles. This process requires no end-user interaction, but the administrators need definitions of *i*Roles with their attribute fields to make proper mappings. If a local principal is to be approved dynamically, the required user attribute values must be collected for a proper assignment. For other than technical reasons such as privacy, collecting user attribute may not be automated even if a user profile is available at certain environments.

Figure 4.5 shows a screen shot of the Java implementation of the toolkit. The main elements of the tool are four columns; the two in the middle show the contents of the *i*Roles while the other two columns show elements from the local policies. The leftmost column represents the elements that can be used as requesters, and the rightmost column lists the ones that can be providers. Dependent upon the selection of the radio buttons on the top of the frame, an administrator can specify requesting side policies or providing side ones.



Europol Liaison Officer
Joint Supervisory Body Member
Eurojust National Contact Point
Europol National Unit
Member State Judicial Authoriy
Europol IMT

EP Info Provider
EP IMT4

Figure 4.5.: User – Role Mapping Tool

The rightmost column in Figure 4.5 is disabled because a requesting point of view has been selected. By selecting two elements of two lists next to each other and pressing the “assign” button between them, the connection between the selected local entry and the distributed role entry is defined. An entry can be created for any of the columns described by filling a textfield below a column and pressing “create” button below the text entered textfield. When all mappings between entries of the local and distributed role policies are set, the corresponding XACML instance can be generated by clicking the “Submit Policy Entry” button.

The well-known 80/20 Rule phenomena can be observed virtually everywhere where management is required. It was first articulated by the Italian economist Vilfredo Pareto in 1906 when he created a mathematical formula to describe the unequal distribution of wealth in his country, stating that twenty percent of the people owned eighty percent of the wealth. Some of the well-known application phrases of this rule would be, “80 percent of the result is created by 20 percent of the input,” “80 percent of the revenue comes from 20 percent of customers,” or “80 percent of the crashes are caused by 20 percent of bugs.” The key point of this observation is that things are not evenly distributed and that some contribute more than others.

4.4. Decentralizing Administration Enablers

Administration of *i*Roles is decoupled in such a way that creation of their definitions are done in the collaborative environment while assignments of them to end-users are done in a decentralized and autonomous manner within each collaborator's domain. The activity of *i*Role administration itself can be considered as an instance of an inter-domain collaboration, dedicated for specifying, conducting, and reviewing authorization for inter-domain collaborations. In theory, the administrative collaborations may even run in parallel to the actual collaborations they are administrating or they may run in advance to the subjected collaborations. This component not only enables timely updates of globally shared information, but it also provides necessary collaborative administrations for more efficient and dynamic courses of collaborations. It can be also used as a medium to foster assurance between specifications and mechanisms of inter-domain authorization solutions.

With the broad outlook of improvement in mind, we start with the essential functionalities the administration need to deal with. First, we classify two different types of administrations according to the scope of their administrations as following:

- **Collaborative administration**: Administrations on this scope deal with *i*Role creation, role–permission assignments, and maintaining a correct role hierarchy.
- **Autonomous administration**: Administrations on this scope deal with user–role assignment so that a user from a collaborative partner can be linked to a *i*Role. User revocation is also dealt with by these administrators.

In addition to those autonomous administration tasks, it is essential that updates on the *i*Role based policies are being communicated between collaborative and autonomous scopes of administrations. For instance, when an *i*Role is deleted, the administrators of the autonomous administration scope must be informed that the user–role mapping is out of sync.

First, let us look into the entire scope with the involved entities. Figure 4.6 depicts a bird's eye view of the administration scope of inter-domain collaborations. The three frames in the lower part represent three different domains of collaborators. Within the frames, a conventional, single domain reference monitor, which is being widely adapted from the IETF/DMTF model to the XACML framework, is in place. Each of these frames comprises of four components—Policy Repository (PR), which stores the policies of the organization, Policy Administration Point (PAP), which manages policy repository, Policy Decision Point (PDP), which evaluates the rules retrieved from the policy repository and makes an authorization decision, and Policy Enforcement Point (PEP), which sends a request to a PDP and executes the policy decision received from a PDP.

The upper part of Figure 4.6 shows the Collaborative administrative component. It consists of an *i*RBAC Builder with a shared repository of *i*Role based security policies and Decentralized Administrations Enabler, a knowledge base and a collaborative PAP. The PAP's from different domains communicate with the collaborative PAP so that their local policies for *i*Roles can be established without violating the consistency property.

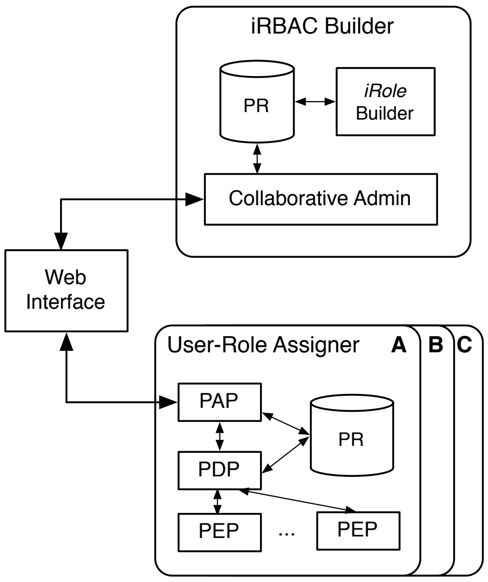


Figure 4.6.: Collaboration of Policy Administrations

The following sub-sections describe the essential functionalities of administrations in the scopes we have mentioned above, and we will also introduce a protocol that is to be used for the communications between the administrators.

4.4.1. Collaborative Administrations

Collaborative administrators have permissions to create, modify and delete the definitions of *i*Roles and collaborations. They have super user access to the data repository in a collaborative realm highlighted in the middle of Figure 4.1. The operations to be conducted by collaborative administrators are listed as following:

- **AddRole**: This command creates a new *i*Role to *i*Role data set maintained by collaborative administrator of a collaborative realm. The new role is broadcasted to all collaborative partners.
- **DeleteRole**: This command removes an *i*Role from the *i*Role data set maintained by collaborative administrator of a collaborative realm. The deleted role is announced to all the collaborative partners that assigned their personnel to that role.
- **GrantPermission**: This command grants an *i*Role the permission to perform a certain role in the collaborations.
- **RevokePermission**: This command removes the permission granted to an *i*Role previously.

In our implementation, these operations are carried out through the web-interface where the *i*Role definitions are stored in a XML database known to the collaborative partners. Through the interface, existing *i*Roles and their permissions can be browsed. For the web interface of the policy administration, we have utilized the following components to realize it:

- eXistDB: open-source, XML database for storing collaboration definitions such as in WS-CDL and XACML formatted authorization policies which are the definitions of *i*Roles.
- Spring: Jave Web application framework
- Tomcat: Java Servlet engine

In addition to the operations on the authorization policies for collaborations, collaborative administrators are entitled to manage web interface user administration tasks for autonomous administrators. Collaborative administrators work as super users who can create and regular user to the administrative web interface for autonomous administrators can do their tasks. Those tasks of autonomous administrators are explained in the following Section 4.4.2.

4.4.2. Autonomous Administrations

Autonomous administrators represent the local security administrators of collaborators' domains. The operations they are entitled to perform within their domains are all the operations related to the user–role assignment step. They are itemized as following:

- **AddUser**: This command creates a new user of User data set maintained locally in each collaborative partners.
- **DeleteUser**: This command removes a user from the User data set maintained locally by each collaborative partners.
- **AssignUser**: This command maps a user from an autonomous administrative scope to an *i*Role.
- **DeassignUser**: This command removes the link between a user from an autonomous administrative scope to an *i*Role he or she was assigned previously.

In addition to these, the autonomous administrators are also entitled to perform the following queries to the collaborative policy repository through the web interface. These are to assist autonomous administrators to assign proper users to appropriate *i*Roles:

- **Query permission sets**: This command selects an *i*Role and queries for its permission sets. Then, all the permission sets granted to the role are displayed as a result.
- **Query *i*Role**: This command selects a task in a specific scenario definition and queries for the list of *i*Roles that are entitled to perform the selected task within the selected collaboration definition.
- **Query user attribute sets**: This command selects an *i*Role and queries for its any constraints on user qualification sets. Then, all the user attribute sets required and/or recommended to the role are displayed as a result.

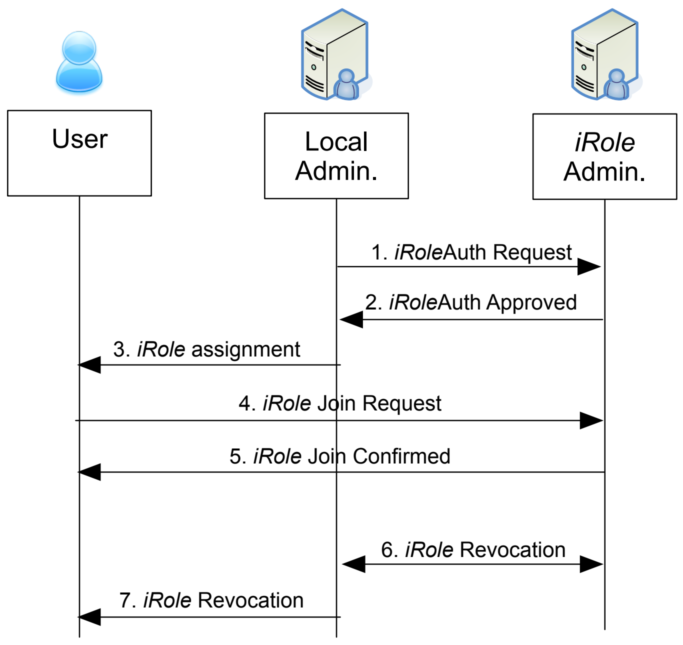


Figure 4.7.: *i*Role–User Assignment

The queries above allow autonomous administrators to identify permissions sets of a given *i*Role and vice versa. Query for user attributes for a given *i*Role would assist to select appropriate users. The user attributes, however, are optional and may not be always specified. In addition to the query functionality, a graphic representation of a role hierarchy of set of participating *i*Role in a selected collaboration definition is envisioned to be available for the autonomous administrators. This feature is helpful when *i*Roles form a hierarchical relations.

4.4.3. Administrative Communications

Collaborative administrators are responsible for maintaining the authorization policies to be in sync with collaboration definitions throughout the life time of *i*Roles. For instance, when a permission is revoked from an *i*Role, the change must be informed to all autonomous administrators to make sure that user–role assignments are still in sound relations. For cases as such, there must be a way of communicating between collaborative administrators and autonomous administrators. Typically, such communication can be modelled in a pull model in which autonomous administrators contact the collaborative repository and collect any updates on the system or a pull model in which messages are sent to the autonomous administrators. In our implementations, we use both mechanisms. The pull model is only used for urgent updates that require immediate attention from autonomous administrators.

Another essential communication required between those two administrators is a set of operations related to a proof of a user's association with a specific *i*Role. Figure 4.7 shows the step-by-step process, composed out of three indepedent operations, that takes place from the moment a user is assigned to an *i*Role till the user is deassigned. Note that for the step of issuing an *i*Role certificate to be used during the execution time of a collaboration, an additional cryptographic technology can be deployed such as zero-knowledge protocols.

In case a zero-knowledge protocol is being used as an anonymous group signature scheme where each *i*Role forms a group of users from different collaborative domains, the proof of a role assignment is presented by an *i*Role certificate [CD00]. In the figure, three separate major operations are illustrated that happen throughout the life-time of a user assigned to an *i*Role. Below each step in Figure 4.7 is explained:

1. Autonomous administrator requests an approval of *i*Role assignment (Required) to collaborative administrator.
2. Collaborative administrator replies to the request either with approval or with disapproval.
3. Upon the consent of the collaborative administrator, assign a user to the requested *i*Role.
4. User then sends a request to *i*Role group to collaborative administrator.
5. Collaborative administrator replies to the request either with approval or with disapproval based on the user's *i*Role certificate.
6. When a user is no longer entitled to the *i*Role, autonomous administrator requests to revoke the user corresponding to the *i*Role certificate which accompanies the request to collaborative administrator.
7. Collaborative administrator revokes the *i*Role certificate and informs the autonomous administrator.
8. Autonomous administrator informs the user.

If a different cryptographic method is chosen, the specifics of data exchange between autonomous administrator and autonomous administrator will change; however, the fundamental protocol depicted in Figure 4.7 will not change.

5. Case Study

The value of an idea lies in the using of it.
– Thomas Alva Edison

The goal of this chapter is to illustrate how to use our methodology of building *i*Role definitions and deriving their permissions sets in a real world inter-domain collaboration. The end-resulting *i*Role specifications are then expressed in XACML format to be used as authorization policies for inter-domain collaborations. The tools used for the different steps are introduced alongside.

5.1. Collaborations of Europol & Eurojust

5.1.1. Backgrounds

Europol, the European Police Office (EP) and Eurojust, the European Judicial Cooperation (EJ), are supra-national European agencies that have been set up to facilitate the EU member states in their fight against cross-border organized crimes. To accomplish their primary mission, both Europol and Eurojust carry out specific tasks in the context of joint efforts amongst the police, customs, immigration services, and justice departments of the EU member states [ABN07]. While Figure 5.1 sums up the bird's eye view of the generic interactions amongst participants of their collaborations, the following section explains the nature of their interactions, involved resources, roles, and regulations in detail.

Interactions. The nature of the interactions between Europol and Eurojust as well as with the 27 member states are best described as dynamic and ad-hoc. Dependent upon how an investigation unfolds, different member states join or leave in the middle of a collaboration. Thus, it is not always known in advance which path of execution a collaboration will take. While the path of execution is dynamic and unpredictable, the patterns of their interactions are rather static and easily categorized. They have basically two types of interactions: 1) a request/response based information exchange and 2) a direct access to a cross-organizational resource. In the former type of operation, a requester simply inquires about a certain set of information from an organization that owns the particular set of information. Upon the approval of such a request, a resource provider sends back a requested set of information. The latter type of interaction actually includes the former type of operation in order to receive an approval from a resource provider before executing a direct access across an organizational border. Though a significant amount of their activities are currently paper-based and done off-line, here we consider digital information exchanges only. Some of the examples of their collaborations are forming a joint investigation team, acquiring an European arrest warrant, and requesting for mutual legal assistance

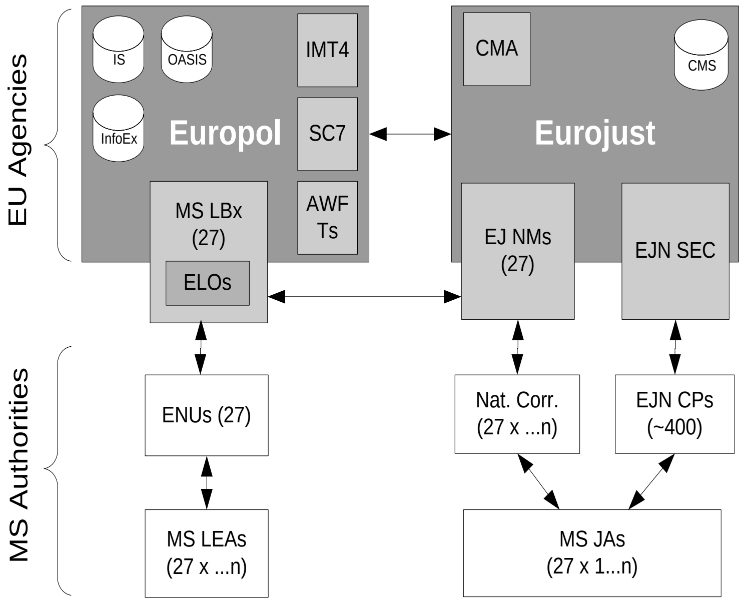


Figure 5.1.: Generic Overview of Europol & Eurojust Collaborations

for witness protection during a court proceeding.

Resources. Both Europol and Eurojust utilize a set of resource repositories to store the information related to on-going investigations. The information they keep are mostly highly confidential and often originally come from the member states. The life-time of the resources being cached in these supra-national agencies is bound to that of the related investigation cases. In both of the organizations, the access control rules with respect to these resource repositories are explicitly specified, requiring highly specific and restricted roles, both internally and externally, to obtain appropriate permissions.

Roles. Each of the 27 member states has an appointed Liaison Officer and a Contact Point to interact with Europol and Eurojust respectively. Within each member state, various judicial and law enforcement authorities collaborate to complete the task that has been assigned from the inter-domain collaboration level. The main roles visible at the cross-organizational level are: Europol's Information Management Unit (IMT4), Europol's Liaison Officers (ELOs), Member States' Liaison Bureaux (MS LBx), Europol National Units (ENUs), Member States' Law Enforcement Authorities (MS LEAs), Eurojust National Members (EJ NMs), Case Management Analyst (CMA), National Correspondents, Member States' Judicial Authorities (MS JAs), European Judicial Network Secretariat (EJN SEC), and European Judicial Network Contact Points (EJN CPs). Refer to Figure 5.1 to view how the roles are related. Through these roles, approximately 5,000 users from the law enforcement side (involving 250-300 Law Enforcement Agencies with 17-20 officials each) and 750 users from the judicial side (involving 400 EJN Contact Points, the 27 National Correspondents on terrorism, the 27 Eurojust National Members, and additional 10 users per Member State) are involved in the Europol and Eurojust collaborations [ABN07].

Rules. The majority of the overarching regulations amongst the involved parties are derived from European directives and laws. Related regulations cascade from the European Union level to the national level. In addition to the legislative regulations, an agreement exists between Europol and Eurojust regarding security issues of their interactions. For instance, the equivalence principle must be applied upon the data handling code of security level of any information that is being exchanged between them. A higher security code must be observed between a sender and a recipient of data throughout their collaborations [ABN07]. Another note-worthy regulation with respect to the resources stored in these organizations is the notion of ownership. Each set of information stored is associated with its owner, and the owner can specify access control rules regarding his or her own resources. It is, however, in a way restricted by the general overarching laws and regulations. For example, if the general regulation specifies that Eurojust national members may access analysis work files, and as an analysis work file owner, one can only specify which, out of the 27 Eurojust national members, may have access to his or her resource file. The owner, however, can not make his or her resource accessible to personnel not permitted to have access to the type of resources by the overarching rules.

5.1.2. Scenarios

To demonstrate the methodology of obtaining authorization policies from collaboration definition, in other words, the definitions of *i*Roles with their permission lists derived from collaboration definitions, we present a walk-through example based on the green paper, “Ne Bis In Idem” [CEC05]. Written in Latin, “ne bis In idem” means “not twice for the same.” This principle denotes that an accused person must not be prosecuted more than once for the same criminal incident, which is also known as double jeopardy.

According to the article 54 of the “Schengen Agreement” among the European Union dated the 22nd of Sept. 2000, the application of the *ne bis in idem* principle is stated as following:

“A person whose trial has been finally disposed of in one contracting party may not be prosecuted in another contracting party for the same acts provided that if a penalty has been imposed, it has been [either] enforced [or is] actually in the process of being enforced[; still other, it] can no longer be enforced under the law of the sentencing contracting party.”

Here the contracting party denotes in the context of that agreement one of the member states of the European Union. According to [CEC05], “creating a mechanism for allocating cases to an appropriate jurisdiction” is the most desiring provision to support the principle. The advised procedure of the steps are rather general but can be summarized as following:

1. **Exchange of information**: the competent authorities in the EU need to become aware of the proceedings and related decisions in each others’ jurisdictions. The step of exchange of information amongst the authorities is allowed by law, but it is actually more of an obligation of each authority to distribute the information to the others in a proactive way in order to ensure the awareness of the possible conflicts in proceeding of a jurisdiction.

2. **Holding own proceeding**: If more than one authority are engaged in the same case and if these authorities are aware of proceedings of other competent authorities, the member states are allowed by law to stop an ongoing prosecution or not to start a new prosecution.

3. **Identification of Interested Parties**: Any "member state prosecution authority (MSPA)" that has initiated or is about to initiate a criminal prosecution in a case which demonstrates significant links to another member state must inform the MSPAs of that other member state in due time.

4. **Consultation and discussion**: When more than two member states are interested in prosecuting the same case, the respective competent authorities should examine the "best place" to prosecute the case. If needed, Eurojust and/or other Union mechanisms of assistance should be sought.

5. **Dispute settlement and mediation**: When an agreement cannot be found, a body at EU level can act as a mediator for resolution of disputes.

While there is no rigid regulations on how and which information must be exchanged amongst involved member states, it is assumed that they cooperate, participate on discussions, and share necessary information to bring the case to the settlement.

5.1.3. Development of *i*Role-based Authorization Policy

Based on the scenario description in Section 5.1.2, we have applied our proposed methodology of building a RBAC system with *i*Roles as illustrated in Figure 1.2. First, we have created an inter-domain collaboration definition. Then, using the role-permission assignment tool introduced in Section 4.2.2, we have derived *i*Role definitions and assigned permission sets to them. The overview of these steps side by side with their associated tools are shown in Figure 5.2. In the following sections, the formation of WS-CDL definition, *i*Role specification, and user-role assignment steps are further explained.

Specification of Inter-domain Collaboration

The complete construction of a collaboration definition in WS-CDL can be found in Appendix C. Several different attempts have been made previously before this final version [Lue08]. They are designed in Eclipse editor using Pi4SOA[1] extensions as depicted on the upper row of Figure 5.2, and the resulting choreography view has been reconstructed for a better resolution as shown in Figure 5.3. It is one of the views available from Pi4SOA, illustrating the actors and types of relationships that are held between the actors.

Looking more closely into Figure 5.3, one can learn the major choreographies of the collaboration. For instance, "CaseAnnouncer" broadcasts the message to possibly interested member state prosecution authorities (MSPA) regarding the case of the interest. Then, all the interested MSPAs respond to the message and thus become "InterestedPartners." Amongst the "InterestedPartners," they communicate and decide which one of them would carry on the persecution as "Case Persecutor". If no one can be selected as a suitable one, then the collaboration takes a routine to go through an open discussion

[1]http://sourceforge.net/projects/pi4soa

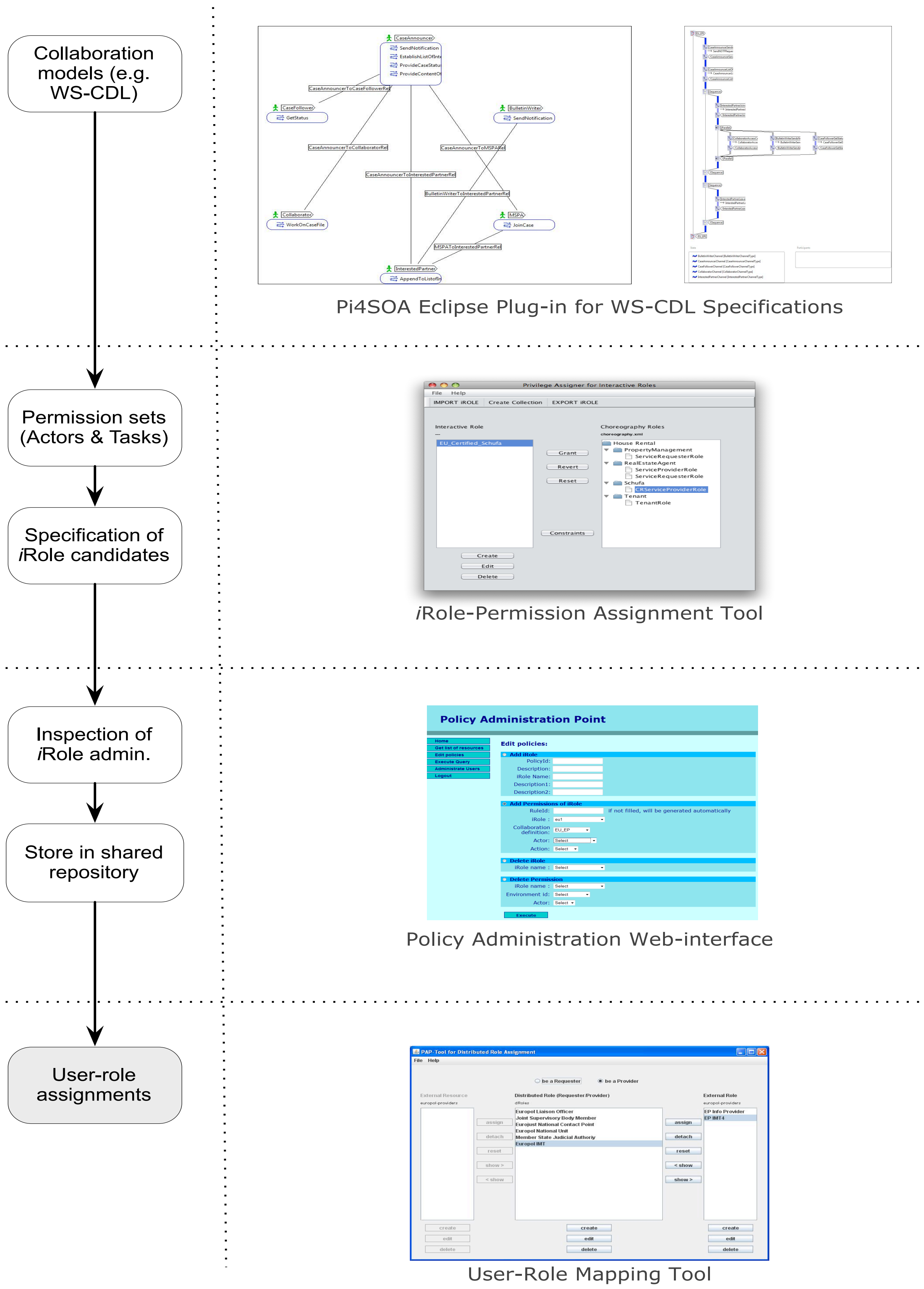

Collaboration models (e.g. WS-CDL)
Permission sets (Actors & Tasks)
Specification of *i*Role candidates
Inspection of *i*Role admin.
Store in shared repository
User-role assignments
Pi4SOA Eclipse Plug-in for WS-CDL Specifications
*i*Role-Permission Assignment Tool
Policy Administration Web-interface
User-Role Mapping Tool
Privilege Assigner for Interactive Roles
Policy Administration Point
PAP: Tool for Distributed Role Assignment

Figure 5.2.: Methodology of *i*Role Creation and Tool Supports

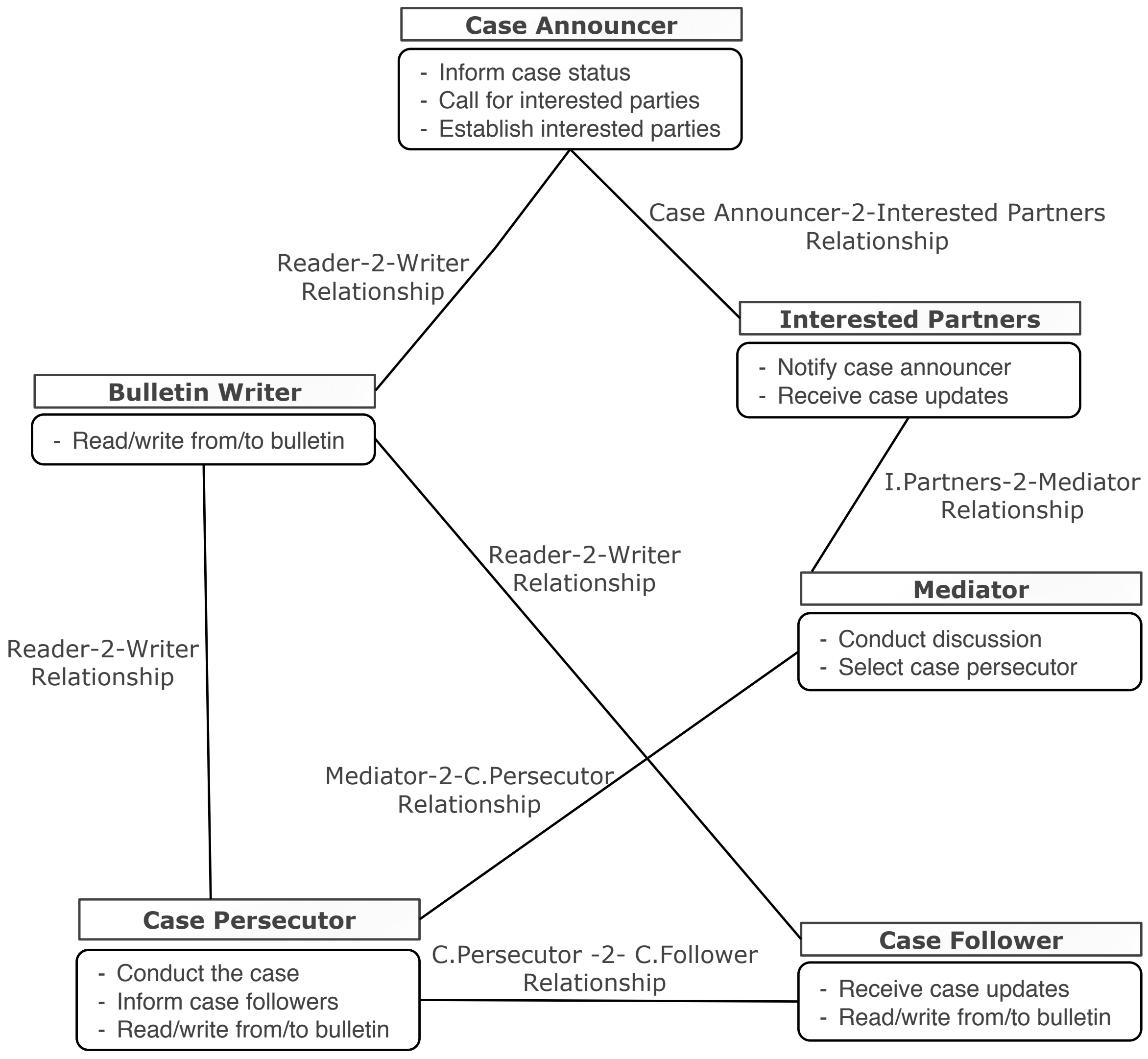

Case Announcer
- Inform case status
- Call for interested parties
- Establish interested parties
Case Announcer-2-Interested Partners Relationship
Reader-2-Writer Relationship
Interested Partners
- Notify case announcer
- Receive case updates
Bulletin Writer
- Read/write from/to bulletin
I.Partners-2-Mediator Relationship
Reader-2-Writer Relationship
Mediator
- Conduct discussion
- Select case persecutor
Reader-2-Writer Relationship
Mediator-2-C.Persecutor Relationship
Case Persecutor
- Conduct the case
- Inform case followers
- Read/write from/to bulletin
C.Persecutor -2- C.Follower Relationship
Case Follower
- Receive case updates
- Read/write from/to bulletin

Figure 5.3.: Choreography of the Case Study, “Ne Bis In Idem”

session amongst all interested parties using shared resource as “Bulletin Writer” participant . It can also take a “Mediator” to get involved in appointing a “Case Persecutor.” If one is selected, then the MSPA carries out the persecution while the other interested parties can voluntarily bail out from the collaboration or become “Case Follower.” The mediator gets also dismissed in due time.

Specification of Permissions and *i*Roles

From the process and choreography descriptions in Section 5.1.2, we derive possible permissions for *i*Roles as following. Note that they are the actors of the collaboration definitions; those represent “functional roles” and not *i*Roles:

- Case Announcer: A partner who initiates the awareness of the case. This partner broadcasts its intention of prosecution to other EU member states.
- Interested Partners: Any EU member state who is interested in the case. One becomes an interested partner by replying to the broadcast message by the case announcer.

- Case Followers: Based on the discussions amongst the interested partners, they shall pick the "best place" for prosecution. However, it may turn out that more than one party still wants to execute the case. All interested partners who still desire to remain on the case become case followers.

- Mediator: Based on the agreement amongst Case Followers, a body of EU level can be engaged as a mediator.

- Case Persecutor: As a result of more talk and discussions based on further pieces of evidence and information available, a case persecutor is being chosen.

- BulletinWriter: A bulletin writer has a permission to write on the shared space called "bulletin" and announce to the other partners his or her organization's decision or intention regarding the steps of collaboration. He or she can also use the space to share any resource of their own with the other collaborators.

All of the choreography roles described above require the actual end-user to hold a position of MSPA recognizable by the other EU member states. As for Mediator, one can be appointed by Eurojust.

Thus, a set of permissions can be defined for an *i*Role called MSPA as shown in Listing 5.1:

```
<Rule RuleId=‘‘Rule1’’ Effect=‘‘Permit’’>
      <Target>
        <Subjects>
                <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0
                    :function:string-equal">
                  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                      string">
                    &amp;externalResource;Eurojust National Member</AttributeValue
                        >
                  </SubjectMatch>
        </Subjects>
        <Resources>
          <Resource>
            <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
                equal">
              <AttributeValue
                DataType="http://www.w3.org/2001/XMLSchema#string">
                    CaseAnnouncer:NeBisinIdem:ws-cdl</AttributeValue>
            <ResourceAttributeDesignator
              AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
                                  DataType="http://www.w3.org/2001/XMLSchema#string"
                                      />
            </ResourceMatch>
          </Resource>
        </Resources>
        <Actions>
          <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
               equal">
            <AttributeValue
               DataType="http://www.w3.org/2001/XMLSchema#string">perform</
                   AttributeValue>
            <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0
                 :action:action-id"
               DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ActionMatch>
        </Action>
      </Actions>
      </Target>
    </Rule>
```

Listing 5.1: A Permission of *i*Role MSPA

Likewise, similar permissions to perform InterestedParty, CaseFollower, and CasePersecutor can be granted to MSPA. Each EU member state may have different positions that can be analogous to an MSPA, but all member states do have personnel that has to satisfy the duties belonging to MSPA. If another scenario is built to further execute the case, for instance, to issue an European Arrest Warrant and if the *i*Role, MSPA, should be entitled to participate certain actors in that scenario, it will also get permissions from the collaboration definition as well. Notice how the permission to be a CaseAnnouncer in Listing 5.1 contains an actor's name, name of collaboration, and the type of collaboration definition as following: "CaseAnnouncer:NeBisinIdem:ws-cdl"

The second part of this step of specification of *i*Roles is to finalize which *i*Roles to involve. The suitable *i*Roles may differ from collaboration to collaboration. In the case of the collaboration between Europol and Eurojust, it yields a great advantages to use publicly known positions of the states. Their names and responsibilities within the context of the applications are known to all partners and implicitely agreed by the collaborators. Therefore, the names of *i*Roles in this case studies are taken from the publicly known positions as depicted in Figure 5.1. Based on the collaboration definitions, however, only the *i*Roles that are relevant are being defined such as the Member State Persecution Authority (MSPA).

The *i*Role-Permission Assignment Tool is being utilized to carry out this phase. According to Figure 5.2, it is represented by the second row from the top where identification of permission sets and specification of *i*Role candidates are defined through the UI tool.

User-*i*Role Assignment

When supervision has been completed, and verification of the collaboration definition as well as the permission derivation and assignments to *i*Roles are finalized. They are to be placed in shared repositories where authorized partners have access to. Security administrators of involved collaborators can now assign the *i*Roles to the end-users of their domains. The step of inspection of *i*Role definitions are depicted on the third row of Figure 5.2 where storing of the resulting policies to the shared repository and offering a Web interface to inspect the *i*Role configurations are shown in a snapshot. The user-role

Table 5.1.: Summary of Resulting *i*Role Elements

Number of Permissions	6 Actors: Case Announcer Interested Partners Case Followers Mediator Case Persecutor BulletinWriter
Number of permission sets	3 Groups of Actors: Case Announcer, Interested Partners, Case Followers, BulletinWriter Case Persecutor Mediator
Number of *i*Roles defined	3 publicly known organizational positions: Member State Persecution Authority Europol Liaison Officer Eurojust Judicial Authority

assignment is done within the domain of each collaborator, and thus the step is shaded on the last row of Figure 5.2. The snapshot of the User-Role Assignment tool is provided on the right column. Each collaborator's security administrators act as brokers for their end-users to get proof-of-roles they have assigned to. The protocol used for the communication has been shown in Figure 4.7.

5.1.4. Summary of Results

While the complete policies resulting from this case study are presented in Appendix C, this section summarizes what has been derived and specified. Table 5.1 lists six permissions extracted from the given WS-CDL collaboration definition file and three mutually exclusive groups of those permissions. Grouping the permission is a step which requires human supervision. Three number of *i*Role definitions are made, and the names of *i*Roles are selected from the possible organizational positions known to other collaborators.

Europol Liaison Officer and Eurojust Judicial Authority can possibly take a role of Mediator in the collaboration. Mediator can be one more more. Member State Persecution Authorities are entitled to the first two permission sets stated in Table 5.1. Once one Case Persecutor has to be selected amongst collaborators, and the cardinality specification is included as an attribute of the permission (Refer to Listing 3.1 for the exact syntax of the structure).

Potentially, there are up to 27 Member State Persecution Authorities who can join the collaboration. In that regard, three *i*Role definitions in total appear to be a positive construction for efficient management and scalable membership inclusion. Especially considering the two *i*Roles other than Member State Persecution Authority is to be assigned to Europol or/and Eurojust personnel, this policy still brings the main leverage from the original RBAC model.

5.2. Lessons Learned

The evaluation of the *i*Role-based policies can not be done by a formal proof or by introducing a systematic performance indicator but rather observing the usage of the policies over a substantial time period. Nevertheless, one can detect if the *i*Roles are composed suitably or not during the process of the role–permission and role–user assignments. This section introduces a few lessons learned on defining good roles and usage of an RBAC-based approach with XACML.

5.2.1. On Defining Good Roles – Evaluation of Usage of *i*Roles

It is worth noting that the need of roles depends on the situations even for inter-domain collaborations. If the business environment is technical and does not require to be linked to the business worlds, a bottom-up approach can be sufficient as discussed in detail in Section 4.2. If collaborations require human involvements, in such cases, alignment of organizational roles and functional roles becomes vital, and it is then beneficial when both top-down and bottom-up approaches are incorporated into a role engineering process in order to yield more consistent results of role definitions. Criteria for achieving the good roles by Coyne and Davis are summarized as following [CD07]:

- Role names should be readily recognizable by personnel assigning users to roles and focused on a particular job functions.
- Permissions granted to each user via a role should enforce the principle of least privilege as well as separation of duties.
- The total number of roles should be considerably less than the number of users to be assigned to the roles; if the two numbers are comparable, the administrative advantage of RBAC is not being realized.

In the case study presented in the previous section, the suitable *i*Roles were available and known to all the partners. If it were not the case, defining an extra abstraction as *i*Roles can be rather obscure and even difficult to comprehend what they should represent. For instance, one can consider a typical e-commerce business scenario of a travel booking. Suppose a customer contacts a travel agency to book flight tickets, hotel, and a rental car. The travel agency may work as a broker and make further transactions with other business partners in order to offer the customer the most agreeable deal. In this scenario, the identities of collaborating partners are specific enough; one does not necessarily create different roles from collaborating partners. For example, "travel agency" is specific enough for a customer; there is no significant need for creating more specific roles to interact with. Thus, one can conclude that it is not always natural or necessary to form the layer of *i*Roles; such case study is not a suitable environment to use *i*Roles. This may be due to the simplicity of the problem set given within the context.

5.2.2. On Difference between RBAC and ABAC

Attribute-based Access Control (ABAC) has been introduced in the early 21st century as a more flexible and more general form of Role-based Access Control. However, many overlook the shift it has made

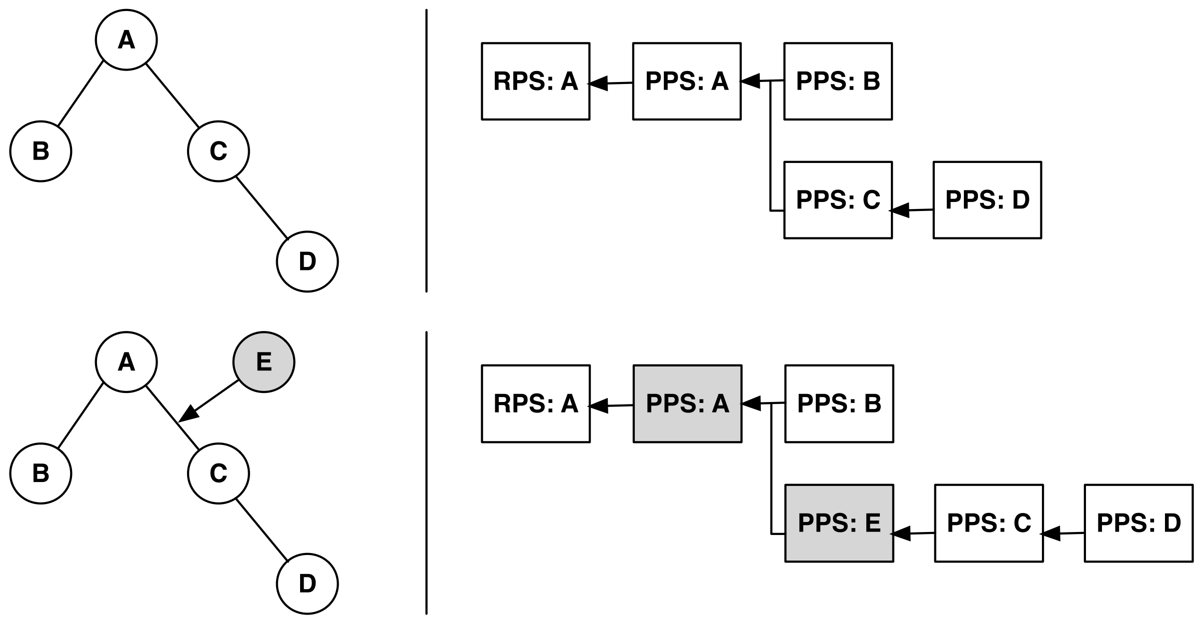


Figure 5.4.: XACML RBAC Profile Policy Reference Consequence

in the fundamental definition of role. While role is defined as a set of permissions in RBAC, it is a collection of attributes where attributes can be any labelled properties which can be used to describe any entity that needs to be considered for authorization purposes. For instance, it can represent a set of user attributes or a certain set of environmental conditions. It could, of course, include both of the types of attributes just mentioned. Therefore, it offers more fine-grained and context-awere access control in comparison to RBAC. However, this also means that a role in ABAC can be a set of heterogeneous attributes rather than representing a homogeneous set of permissions as in RBAC. The survey of current authorization specification models show that ABAC is being deployed and then being attempted to be mapped to principals. This is then a problem since principals do not hold some of the values of attributes such as a specification of time or space conditions.

It is clear that a misuse of RBAC may cause a role explosion[2], especially when roles are created from automated role mining techniques (e.g. selecting clusters of sets of permissions followed by theoretical permutation of the clusters). Nonetheless, if the permissions are to be assigned to principals in the end, a proper use of RBAC is advised. Other constraints in collaborations are necessary to be specified together with the collaboration definitions in the first place. Therefore, this also eliminates possible conflicts between the authorization policies built separately from the collaboration definition constraints. Additionally, it eliminates duplicates of policies that will only slow down policy enforcement schemes.

[2]http://www.axiomatics.com/is-rbac-really-deae.html

5.2.3. On XACML – Its Strengths and Weaknesses

Although XACML is the de facto standard used by many, its inborn problems cause a number of tricks to be invented to take a detour in terms of using the standard, to the point that it jeopardizes if an XACML request can be processed by a policy decision point as designed by the standard. One of the complications is the usage of references to different XACML policies possibly in remote locations. This reference mechanism has been heavily used by the RBAC-Profile in such a way that the actual permission sets of a role are specified as one policy, also designated as PPS and then are to be referenced by a role definition policy, also known as RPS.

Such referencing causes performance reduction due to searching and loading another policy for processing an access control decision. More importantly, when a role is to be inserted in to an existing role hierarchy structure, the senior roles' permission references must be changed as well. Figure 5.4 depicts such a situation. The upper left part of the figure shows a role hierarchy, and the upper right depicts the reference of policies between RPS's and PPS's of role A based on the role hierarchy diagram on its left side. As shown, the policy with the role definition, denoted as "RPS:A", references its corresponding permission policy "PPS:A" which include the permission sets exclusively given to role A. Then, it references its subordinate roles' permission policies. This is how XACML-RBAC manifests a role hierarchy.

The lower part of Figure 5.4 depicts the situation when a role is being inserted between role A and role C. The lower right part shows the changes required in order to correctly represent the role hierarchy updated by the change. The policy "PPS:A" must be modified so that instead of referencing "PPS:C," it now must point to "PPS:E." Also, "PPS:E" must be created with correct references to the PPS policies of its subordinate roles, namely "PPS:C" and "PPS:D". If the policies are organized as separate files, it requires painstaking effort to first construct the correct role hierarchy image thus to find out which policies must be searched for and get hold of the policies files that must be modified. Some point out that the focus of the XACML-RBAC profile is on how to use RBAC in an access control decision rather than how to define model control semantics [3]. It is definitely the case in terms of operating upon role hierarchy.

To this problem, our proposal is to provide a tool to visualize a role hierarchy when a set of role definitions written in the RBAC Profile are given. Usually, security administrators have to get involved to finalize the permission assignments in any case. Thus, assisting them with a visualization of hierarchy out of the multiple policy files they must deal with reduces their analysis time of the given policies and increases the correctness of modifications they manually introduce.

Another major problem is from the architecture of its infrastructure. The XACML Framework has been shown in Figure 2.7 together with the step-by-step explanation of the processing of an XACML request. The problem is between Step 2 and Step 3. If an access request is initiated from somewhere outside of the administrative domain where PEP and/or PDP reside, which should be the valid case, it is then required that the request knows which attribute values must be sent in order to receive an access grant. To do it properly, the requester must know not only the type of attributes required, but it

[3]http://www.openroleexchange.org/discuss/index.php?topic=7.0

also needs to know what type of values are acceptable by the PDP eventually since the request will be merely forwarded by PEP to PDP in the following steps.

This requires sharing of policies, and for various reasons, the framework cannot be set up that way or it will be very difficult to come to an agreement on following the same set of vocabularies. Therefore, in reality, the PEP translates requests coming from outside of the domain or PDP gets extra information from back door entities such as PIP which communicates with the originating requesting party for the extra information gathering. Thus, this makes the standard work rather proprietary manner when it is implemented. Though this can be argued to be an implementation flaw, the feasibility of making the XACML framework across organizational boundaries exists as a real challenge. In this sense, it might be more sensible to use XACML for specifying an access control policy for interoperability purpose only and to leave the enforcement of the policies to be revised to cope with domain-specific access control policies.

These issues of XACML are surprisingly unknown or ignored among the standard communities. One of the few proposals made for a remedy is enhancing the PIP to be more like a transformer of various formats of attributes to the kind that can be understood by PDP rather than a simple module of interface to the environment and subjects [LKW+09].

6. Related Work

What has been will be again,
what has been done will be done again;
there is nothing new under the sun.
– King Solomon

In this chapter, we discuss two major fields of work related to this thesis: the first is authorization models used for distributed systems. Secondly, we discuss role engineering methodologies and compare them to the approach we have chosen.

6.1. Authorization Models for Distributed Systems

In this section the authorization models developed for distributed systems are explained. The first category introduced in the following sub-section is the models derived from RBAC whereas the next sub-section describes the models relying on credentials to carry policy information.

6.1.1. RBAC-based Authorization

To the best of our knowledge, none of the current inter-domain authorization models based on the RBAC model satisfies all of the security requirements elicited from the case studies as stated in Section 2.4 with the security challenges still open as highlighted in Section 2.2.2. For instance, Shafiq et al. [SJBG05] introduce a framework in which different domains with different RBAC policies can integrate their policies to allow for secure interoperation; but, according to this model, all of the participating domains' security policies must be known to the others. Moreover, such a methodology does not scale. Kern at el. [KKSM02] have proposed the so called *Enterprise Roles* model, which proposes an extra layer between user and role abstraction of the RBAC model. However, it provides a centralized administration and requires the complete set of user attributes in order to build the corresponding RBAC system. Note that our work focuses on the specification of authorization policies rather than on how that is enforced. As Sterne mentions [Ste91], policy specifications ought to be independent from their enforcements.

This type of solutions specify authorization constraints and conditions as a set of rules (e.g. authorization policy) shared by all collaborators. Role-based Access Control and various extensions of the model are predominant examples of this category [SCFY96, PCND04, PJ05, DdLGvB06, BEM03] as described individually below:

- X-RBAC
  Joshi et al.'s XML Role-Based Access Control (X-RBAC) is the first model to use the RBAC

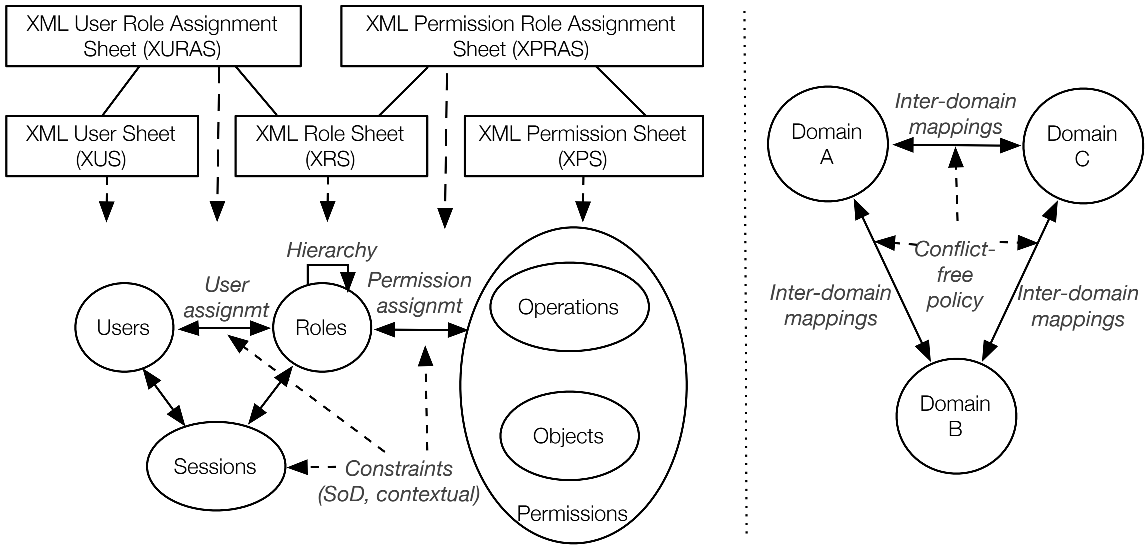


Figure 6.1.: X-GTRBAC in a Multidomain Enterprise Environment [BSB+05]

model in the context of inter-domain collaborations [JBBG04, BJBG03]. It provides XML syntaxed language to specify security policies for multi-domain environments, which is similar to the one in which inter-domain collaborations occur. It allows to express user credentials, roles and permissions in XML format where user credentials are a set of user attributes, and roles are identified by an associated set of credentials that must be satisfied by the users who are assigned to that role. It also enables to specify temporal constraints and environmental context within the XML-formatted policies. The main drawback of this approach is that the resulting meta policies are bilateral; therefore, if it is to be deployed to multilaterally collaborative environments, the meta policies must be specified between all possible pairs of collaborators within the group. Moreover, local policies including roles and permissions must be shared with the other collaborators in order for the roles to be mapped to local roles of the corresponding collaborator's domain. In addition, the XML-formatted specification being not a standard, makes it challenging for a new collaborators to join and adopt the X-RBAC policies.

- X-GTRBAC – Generalized Temporal Role-based Access Control with XML
  The Generalized Temporal Role Based Access Control (GTRBAC) model [PJ05] provides a set of language constructs for specifying various temporal constraints on roles in the RBAC model. These include role enabling, disabling, activation, and assignments of user-to-role and permission-to-role within a single administrative domain. The temporal framework in the GTRBAC model allows to specify periodic time expression in order to express the role activation, assignment constraints as well as relevant run-time events. The X-GTRBAC model extends the GTRBAC model so that the model can be used as enterprise access control policy base. The constraints of GTRBAC are expressed in a context-free grammar called X-Grammer; the notions used are similar to the ones in Backus-Naur Form (BNF). Being compatible with XML schema syntax, each element tag can contain attribute fields. In addition to the aforementioned

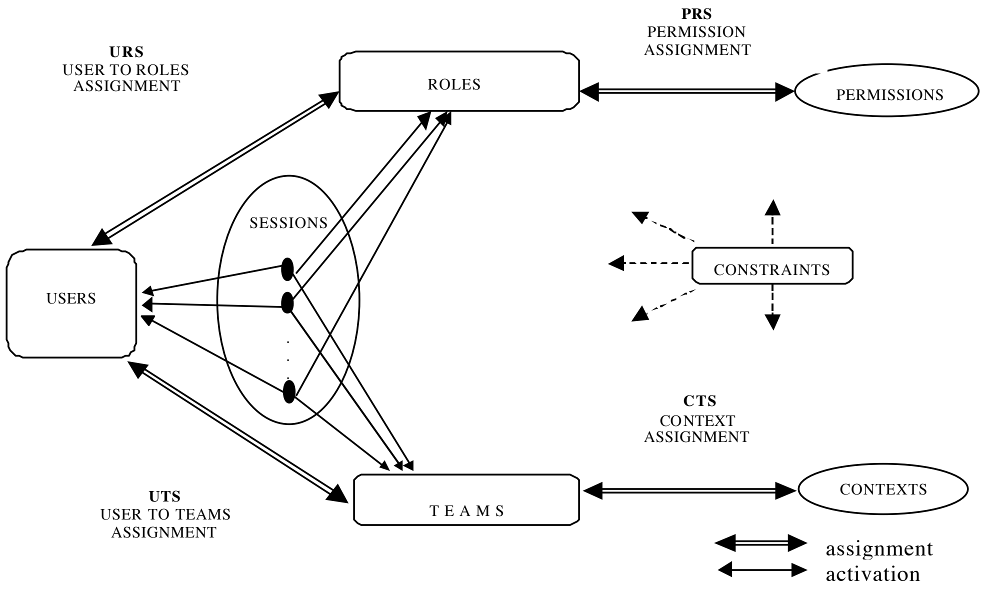


Figure 6.2.: The C-TMAC Overview [GMPT01]

constraints, separation of duty constraints on roles can also be specified in X-Grammer. As shown on the left side of Figure 6.1, the X-GTRBAC policies contain the XML User Sheet (XUS), XML Role Sheet (XRS), XML Permissions Sheet (XPS), XML User Role Assignment (XURAS), XML Permission Role Assignment Sheet (XPRAS), and XML separation-of-duty definition (XSoDDef) [BSB$^+$05]. On the other hand, the right side of the figure shows how different domains interact in terms of enforcing the policies. Inter-domain mapping techniques are given, and each domain exchanges necessary policies with the others in a bilateral way.

- C-TBAC – Context-based Team-based Access Control

  The Team-based Access Control (TMAC) model formulated by Thomas in [Tho97] is one of "active security models" that are aware of the context associated with an ongoing activity. It defines two aspects of the collaboration context called user context and object context. Extending on the focal idea of TMAC that the contextual parameters must be considered for an access control decision, C-TMAC [GMPT01] incorporates additional parameters such as time and space context of collaborations. In both of the models, a concept called "team" has been introduced. It represents a "collection of users in specific roles" within a certain instance of a collaboration. When a team is instantiated, the user context can be referred to in addition to role-based permissions to finalize access rights to team's resources.

  Figure 6.2 shows the overview of how the concept of roles and teams are working together within a given session. Although the contextual information is a necessary addition to provide a fine-grained access control, how the team concept can be incorporated to RBAC is not clearly developed [TAPH05]. It also lacks the self-administration of assignment relations between entities according to Tolone et al. [TAPH05], which is a real hurdle to overcome to deploy in a real collaborative systems. That includes two major challenges this thesis is dealing with, namely, 1)

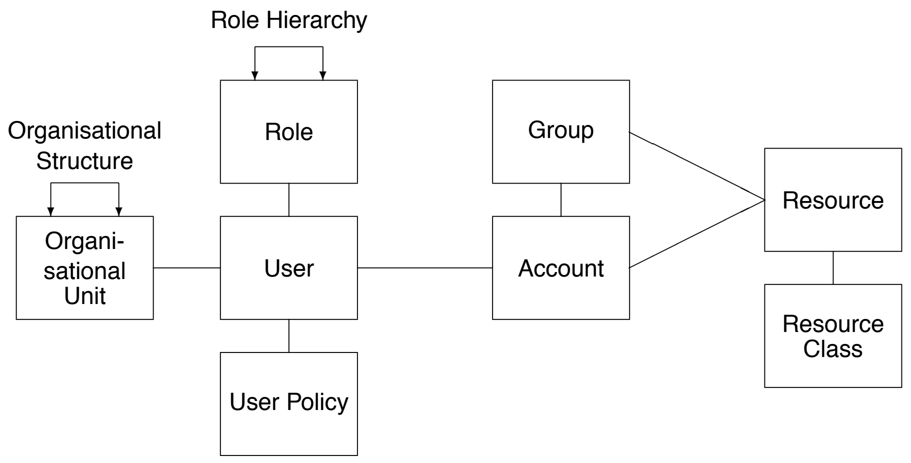


Figure 6.3.: Entity Relationships in ERBAC[Ker02]

autonomy of user administrations and 2) preservation of end-user privacy.

- ERBAC
  The Enterprise Role-based Access Control model has adopted the ANSI RBAC model so that the roles can be incorporated with an enterprise IT environment in which a variety of platforms and applications exist. Different operating systems, databases, and standard and business specific applications are some of the categories of the variety. Dependent upon the target systems, the specifications of permissions can be of various natures. Considering the characteristic of a typical enterprise, it elaborated the notion of role as "Enterprise Role" which consists of permissions in multiple systems. As shown in Figure 6.3, its permission can be a membership to a user group, another role to a database, or a generic set of ACL. Its downside is that in order to implement the model, complete knowledge of participating systems and user attributes are required in order to create "Enterprise Roles" and assign appropriate users to them.

- dRBAC
  dRBAC stands for distributed Role-based Access Control system, and it claims to provide a decentralized access control mechanism through a complete delegation solution that delegates roles to other roles or entities in another local name space where entities may refer to users or resources. Its infrastructure includes *proof monitors*, among others, which verify delegation chains at each entity [FPP$^+$02]. This model takes a similar approach to *RT* (Role-based Trust Management) in the usage of a delegation chain for establishing trust and deferring authorization decisions. Ma and Woodhead have applied this model to resolve the problems of identity management inherent in a distributed subscription-based resource sharing environments [MW06]. Using Distributed Role Markup Language (DRML), they specify distributed roles related entities, role constraints, and predicates. This application of dRBAC for subscription-based resource providers does get its mileage out of the abstract layer of external roles as a number of requesters per a given resource increases.

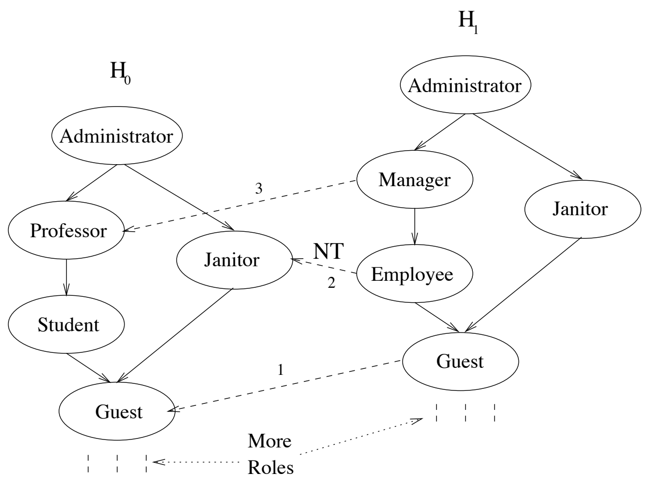


Figure 6.4.: Dynamic Role Translation in IRBAC2000 [KAMCM00]

- IRBAC 2000

  The Interoperable Role Based Access Control 2000 model aims to provide interoperability between two or more different domains with the policy framework that creates dynamic security policies to interconnect the local policies of the involved domains. It assumes each local security policy is specified in RBAC with hierarchy. By associating two different roles in two role hierarchies of two different domains, it achieves the localization of foreign roles, so called “dynamic role translation [KAMCM00].” Kapadia al et. explain that such domain crossing can cause several security hazards such as covert promotion and infiltration (refer to Section 3.3.1 for further discussion). The IRBAC 2000 model limits the role mapping to be done no more than across one domain boundary. However, no further mechanism is given to prevent them other than relying on “cooperation” amongst interacting domains. Figure 6.4 depicts the dynamic role translation in the IRBAC 2000 with the dotted lines as the role association from one domain to the other.

The downside of forming security policies shared amongst collaborative partners is that they tend to be static and cumbersome to administrate. More specifically, it is unavoidable to face conflicts amongst involved policies, and erecting a common set of attributes and a common interpretation of the attributes remains to be a challenge beyond the scope of technical solutions. There is also a risk of a single point of failure if the centralized policies are placed in one location. Nevertheless, they can efficiently describe globally known roles and their relationships, which produces a more lightweight solution in the long run than solely relying on delegation-based authorization solutions as described in the following Section 6.1.2 where more sophisticated controlling such as separation of duties cannot be specified. The core difference of the *i*Roles model from the other existing RBAC related models is the possibility of autonomous, decentralized policy management by letting the collaborators take control of assigning user–role and role–permission relations.

6.1.2. Credential-based Authorization

Authorization in a distributed system can be carried on by relying on credentials issued by the participating parties. In general, credential-based authorization models do not require the existence of security policies shared and agreed amongst interacting systems; rather, they assume each party to issue or check credentials that are being requested or produced by other collaborating parties. Below are some of credential-based authorization models:

- KeyNote / PolicyMaker

  PolicyMaker by Blaze et al. is the first *trust management* system, and KeyNote [BFIK99] is the successor. The concept of trust management is defined as "a unified approach to specifying and interpreting security policies, credentials, and relationships that allows direct authorization of security-critical actions" [BFIK99]. At the heart of a trust management system is a *trust management engine*, which receives authorization queries and evaluates requested actions against local security policies. From the requester, it receives an action string, which is application specific, and credentials, which is a chain of public keys. Here, both credentials and policies are referred to as *assertions*. An assertion is a construct that delegates authorizations to perform actions to a principal from its signer. An assertion has the following syntax:

  Source **ASSERTS** *AuthorityStruct* **WHERE** *Filter*

  Source is the source of the assertion, which can either be the keyword **POLICY** in the case of policy assertions or a public key of the principal who confers the authority implied by the assertion in the case of credentials. *AuthorityStruct* specifies a list of principals to whom the assertion applies. Each principal could be specified as a single public key, or as a threshold structure, e.g. K-out-of-N keys. *Filter* specifies the conditions that an action string must satisfy for the assertion to be valid. *Filter* are, by design, interpreted programs, and the prototype of PolicyMaker allows three different programming languages. Credential is, however, different from policy assertions, in a sense that it is signed, for it is intended to exist outside of the trust management engine, and therefore its integrity must be protected. In essence, the evaluation of a query is to find a chain of delegation from a policy as a trust root to the public keys requesting the action in which all filters along the chain are satisfied.

  KeyNote is based on the same concept as PolicyMaker but with improvements on issues such as standardization and ease of integration into applications. For instance, KeyNote provides a single, unified assertion language and shifts the responsibility of applications to verify signatures to trust management engines. It also uses a more efficient query evaluation algorithm.

- SDSI / SPKI

  Simple Public Key Infrastructure (SPKI) by Ellison et al. [Ell99] is a standard specified by the Internet Engineering Task Force (IETF) SPKI working group. Producing a certificate structure and operational protocol, it was to support the needs of authorization management in Internet applications. Especially this work was motivated by the inflexibility and inadequacy of the naming scheme of X.509. Separately proposed, Simple Distributed Security Infrastructure (SDSI) by Rivest and Lampson was also design to improve the same global naming problem. Therefore, the two projects were later merged and published as SDSI / SPKI 2.0 [EFL$^+$99].

There are three types of certificate in SPKI: *authorization certificate*, *name certificate*, and *access control list (ACL)*. An authorization certificate transfers specific access rights from one principal to another, which is also known as a delegation certificate. As delegation propagates, the certificate includes a chain of public keys. A name certificate binds a public key with a name. An access control list certificate contains a security policy of a particular service.

In a nutshell, SPKI authorization mechanism works this way: when a service, S, grants access rights to a principal, C_1, it issues an authorization certificate that holds delegated rights from its ACL. C_1 could issue and sign an authorization certificate to further delegate to C_2, and so on. When S is requested by C_n, a certificate with the delegated rights is presented to the service as illustrated below. Here the double arrows indicate delegations, and a single arrow indicates a service request.

$$S \Rightarrow C_1 \Rightarrow C_2 \Rightarrow \ldots \Rightarrow C_n \rightarrow S$$

A name certificate claims a name of a principal within a local namespace where the principal belongs. Thereby, local names do not need to be globally unique but need to be unique in the given local namespace. Consequently, a globally unique version of a name can be obtained by linking a local name with its namespace, called a *linked name*. A name can be either resolved into its subject's public key or a reference to another name. SPKI requires either chains of names to be provided or only those names with a binding to a public key get processed.

- Role-based Trust Management
  Role-based trust management (*RT*) combines the strengths of role-based access control and trust management systems and is capable of expressing attribute-based access control [LMW02]. This is not the case with the traditional trust management systems. In addition to more flexible expressiveness of semantics, the advantages of *RT* over the previous trust management solutions include: a declarative, logic-based semantic language specification, strongly-typed credentials and policies, more flexible delegation structures, and more expressive support for separation of duty constraints [LMW$^+$04]. It utilizes delegation chain for establishing trust and deferring authorization decisions. While *RT* systems do offer a variety of features for a decentralized access control, they require significant supporting infrastructures and extra installations for their framework to work properly.

On one hand, credential-based authorization models promote prompt decentralized, ad-hoc style interactions amongst different systems; on the other hand, it is difficult to picture how a set of interactions will progress since the authorization decisions are based on dynamically created credentials. Although some of these models incorporate the concept of trust management to improve this shortcoming, these models are still more suitable for one-time transaction for ad-hoc style interactions.

6.2. Role Engineering Methodologies

Role is an abstract concept; therefore, it may represent different sets of semantic groups. Thus, it often accompanies further description of roles in use. For instance, Kuhlmann et al. define “functional roles” as representatives of specific “tasks or functions” [KSS03]. These functional roles created from the technical specifications are not necessarily compatible with the roles reflected in business scenarios where influences of organizational positions and line of authority are significant. Furthermore, Kuhlmann and et al. define “organizational roles” as basic roles of users including “all basic accounts and access rights in different systems” [KSS03] of an organization. In the original context of the RBAC standard, role is a representative of a group of permissions; thus, it is closer to the meaning of “functional roles” than “organizational roles” in terms of the definitions given by Kuhlmann et al.

As far as specifying roles and their permission sets are concerned, there are two major approaches: a top-down approach, which develops roles from organizational structure or business context, and a bottom-up approach, which takes existing permission sets and uses data mining techniques to derive roles as sets of permissions by merging the permissions.

Most of top-down approaches attempt to extend the RBAC model and provide a role engineering framework in order to find the roles. Poniszewska-Maranda proposes in [PM05] to extend the RBAC model and insert an entity called *Function* between role and permission so that functional aspects from UML’s Use Cases can be used to narrow down which permissions should be granted to which roles. Fernandez and Hawkins, on the other hand, propose to extend UML Use Case to specify role’s rights in order to find the linkage between roles and the corresponding privileges [FH97].

Instead of extending existing models or standards, Thomsen et al. introduce a seven-layered RBAC framework, which is to be developed by local system administrators and application developers [TOB98]. This framework is for network enterprises, and a framework has been proposed to offer fine-grained access control and ease of management. This framework has been only partly implemented as the NAPOLEON tool-set, which deals with the four layers of application developers’ part. Because the former approach involves much of human inspections and manual effort, it is costly and error-prone. Besides, roles from top-down approaches often represent organizational positions or business actors, resulting in difficulties in mapping those roles to the ones in a technical operational level.

The latter technique is also known as “role mining”; roles created from bottom-up approaches are generated through more automatic processes. For instance, Mendling et al. rely on the processes written in BPEL4WS to derive RBAC roles [MSSN04]. Specifically, they transform the entity called *Partner Roles* in BPEL to RBAC roles using XSLT. Neumann and Strembeck also present a methodology in [NS02] that takes a scenario model and decomposes the tasks to identify permissions and constraints involved, and from those they define role sets and hierarchy. Based on their proposal of role engineering, [SAI04] have developed a role engineering case study on eHealth. In [SAI04], users are first assigned to “Role Group” that are mapped to perform particular “Work Profiles”. Work Profiles are basically a hierarchical structure of tasks based on scenarios. This structure consists of tasks, scenarios, and steps in that seniority order. Work Profile elements are associated with permissions to resources in use. In the eHealth case study, the HL7 Reference Information Model [1] was adopted as the reference object model against which standard permissions are defined.

Roeckle at el. also apply a process-oriented approach to finding roles in large industrial organizations [RSW00]. They create a role-based administration layer between business layer and access control layer, which is a part of the technical infrastructures; then, they make use of a directory service called DINGO and data mining techniques [RSW00]. All these proposals tend to yield functional roles that are strongly bound to the technical worlds and not reflecting the roles in the business world. Consequently, a major challenge remaining with this approach is that the resulting roles may turn out to be too obscure to be mapped to a real role in the business context and that the rate of false positive can be high, resulting, for instance, "ghost roles". Therefore, for the meaningfulness and correctness in business scenario contexts, those roles are still required to go through some manual review and human intervention for finalization.

In short, we can say that there is no single perfect mechanism in role engineering process. Both top-down and bottom-up approaches have their own weaknesses in order to bridge business context and technical specifications. If a bottom-up approach is being used, the outcome of the roles does not correspond to the roles of the business layer where organizational constraints and regulatory laws are enforced. If, on the other hand, a top-down approach is being used, it needs to be connected to the existing access control solutions at the operational layers since access control mechanisms exist with or without RBAC models in the enterprises.

Kern et al. have pointed out the advantages of considering both approaches and proposed a role engineering process that considers both top-down and bottom-up approaches [KW05]. Based on so called *Enterprise Roles*, the permissions of their roles can be a specification of access to not only resource but also to other roles or to various types of target systems [KKSM02]. The user-role assignments are done by a rule engine that imports a complete set of user information from a given HR database and matches to roles based on rules composed [KW05]. Their solution is available as a commercial product called Security Administration Manager Jupiter [KKSM02, KW05]. Although this is a highly sophisticated solution, it requires a single point of administration with availability of entire user profiles. This condition, however, is not feasible for inter-domain collaborations of autonomous partners.

In reality, a suitable solution differs by various factors. One generic factor is the input to the process. All of the existing role engineering mechanisms take a certain set of input as it is the case with all man-made creations that they are made out of something else. Some of the examples of those input are the existing permission sets, user attribute sets, and/or the relations of user–permission sets. With this input, the interesting part is how to automate the process of finding useful roles. As an example, Dana et al. proposed using the result of **cluster tendency analysis** of permissions to form roles [ZRV09]. More specifically, through the analysis, they proposed to predict a number of possible roles and its possible hierarchy through observing a resulting number of clusters and tendencies of overlapping

[1]http://www.hl7.org/implement/standards/rim.cfm

clusters amongst them [ZRE07]. Others start with already existing role sets to minimize role sets and/or to build a role hierarchy [VAG07] since the resulting role candidates from the data mining techniques do contain a high level of noise.

In the context of inter-domain collaborations, the set of input are collaboration process and participating partners. The collaboration process consists of a collection of tasks and actors; actors are often identifiable as functional roles associated with the activities involved in the collaborations. Out of participating partners, organizational roles of their external business context can be derived. The output of the role engineering is then a collection of *i*Roles, its role hierarchy, and a set of role–permission sets.

7. Conclusions

The biggest room in the world is the room for improvement.
– Unknown

After twenty years of work on Role-based Access Control, why do companies still rely on consultants to implement their access control system based on RBAC? Theoretically, the essence of using RBAC is for the ease of administrative effort. Why is it then that the research communities still strive to publish yet another methodology, language or mechanism to improve the efficiency of administrating an RBAC system? As stated in Chapter 2 within the study of the nature of inter-domain collaborations and the security requirements elicited, an RBAC model appeared to be the ideal choice as an authorization model for inter-domain collaborations, and there are a number of proposals available as explained in 6.1.1 [JBBG04, PJ05, GMPT01, Ker02, MW06, KAMCM00, LMW02]. Nevertheless, what we have described is that the models are not sufficient enough to satisfy the elicited security requirements and to be deployed to a real-world inter-domain collaborative environment.

Our conclusion on this is twofold: first, what is lacking currently is **practical engineering support** for building an RBAC system. Role is an abstract notion, representing different concepts based on where it is being deployed [MLL$^+$09]. Thus, it is true that the creation of roles ought to be tailored case by case. However, role is meant to represent a set of permissions according to the original RBAC model. Thus, supports on identifying permissions and methods to group them in a meaningful way within the given environment are essential to developing an RBAC system. Secondly, administration of RBAC systems or RBAC-based security policies must embrace the management of the entire life-cycle of roles. This requires again reasonable engineering support to sustain an RBAC based authorization system. Our take is that too many work has been focusing on theoretical issues ignoring its practical implications. For instance, a number of algorithms are available to eliminate security violations caused by incorrect role mapping from one domain to another. But, these algorithms can not resolve the fundamental problem underlying this case, and thus they can not be utilized in a real-world setting. What is more important is to provide comparable roles that can be mapped across domain boundaries rather than connecting roles representing different concepts in the first place.

Based on these understandings, we have provided inter-domain collaborative environments with 1) a modified RBAC model with an additional *interactive* Role layer and substantial engineering support for building an RBAC system as well as managing the resulting RBAC based policies in a decentralized manner. The following sections summarizes the contributions of this thesis. We then end with further directions of improvements.

7.1. Summary of Contributions

The major achievement of this work is giving the full coverage of support throughout the different stages of life-cycle of an RBAC-based authorization policy for inter-domain collaborations. From their creation process to their continual change management policies are administrated by a decentralized policy administration framework called *i*RBAC. This differentiates itself from other works that focus on specific areas of the complete picture such as decentralized policy enforcement mechanisms, role engineering, and the XACML implementation as discussed in Section 2.3 and 6.1.1. Those solutions for the specific areas tend to cause difficulties in integrating with solutions of the other specific areas in such a way that they can be ready for real-world systems. For instance, roles created by a specific role engineering mechanism represent either functional roles or organizational roles that can not simply be integrated into a real-time collaboration system.

We have considered inter-domain collaborations in their characteristics and authorization requirements, proposed an additional indirection layer called *interactive Roles* to the RBAC model that satisfies those requirements, and presented how to build *i*Role-based authorization policies. As a part of the solutions, we have also implemented the necessary tools required for autonomous policy administration points and tools required for collaborative policy administrators so that policy administrations in all stages of the life-cycle of RBAC-based authorization policy can be managed. These have been assembled as *i*RBAC Framework. The end-result of using the *i*Role layer is providing alignment of functional roles defined in collaboration definitions with organizational principals in local domains of collaborators. Highlights of the contributions of this work are listed as below:

- Provision of a model of an authorization policies for inter-domain collaborations
- Framework for decentralized administration on *i*Roles
  - Role engineering methodology enabling alignment of business actors with technical roles
  - Decentralized user–role assignments, satisfying both autonomy and security requirements
  - Visual assistance of policy administrators by showing a role hierarchical structure of a given set of *i*Roles
  - A toolset which supports policy administration points (PAPs) at collaborators' sites as well as at a collaborative environment
- An XACML-based reference monitor which does not need to exchange user attributes across domain boundaries

It can be also speculated that this model could be used not only to form an inter-domain collaborations out of different domains but that it bridges different collaborative systems such as virtual organizations.

7.2. Future Work

Although a case study has been conducted, applying the *i*RBAC framework to a large-scaled enterprise inter-domain collaboration system would be a valuable next step to test its practicality within a real-world scenarios. Another important area of research is to evaluate the various properties of the *i*Role-based policies. For example, a number of usage of role definitions can be a valuable asset to determine how well the roles are specified. In addition to that, evaluation of performance of enforcement of the policies would be an interesting field of study. This area itself contains numerous elements of investigation, and a number of studies are presently going on focusing on the evaluation of XACML in its conformance of specification [HMHX07] as well as its PDPs' performance [TC08].

A. Acronyms & Abbreviations

ABAC	Attribute-based Access Control
ADFS	Active Directory Federation Service
B2B	Business-to-Business
CAS	Community Authorization Service
CVS	Credential Validation Service
DMTF	Distributed Management Task Force
DNS	Domain Name System
DTD	Data Type Definition
ebXML	Electronic Business using eXtensible Markup Language
EPAL	Enterprise Privacy Authorization Language
ESB	Enterprise Service Bus
GTCP	Trade Register of the Paris Business Court
GAAA	Generic Authentication, Authorization and Accounting
GGF	Global Grid Forum
HTTP	Hypertext Transfer Protocol
HTTPS	Secure HTTP
IdMI	Identity Management Infrastructure
IETF	Internet Engineering Task Force
IOP	Interoperability
IP	Internet Protocol (also synonym for IP-Address)
IdP	Identity Provider
J2EE	Java 2 Platform Enterprise Edition
LDAP	Lightweight Directory Access Protocol
NIST	National Institute of Science and Technology
OASIS	Organization for the Advancement of Structured Information Standards
OM-AM	Object, Model, Architecture, Mechanism
OS	Operating System
OSI	Open System Interconnection
OWL	Web Ontology Language
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PKI	Public-Key Infrastructure
PMI	Privilege Management Infrastructure
PIP	Policy Information Point
RBAC	Role-based Access Control
SAML	Security Assertion Markup Language
SLO	Single Log-out
SOA	Service-oriented Architectures

SOAP	Simple Object Access Protocol
SP	Service Provider
SSL	Secure Session Layer
SSO	Single Sign-on
TLS	Transport Layer Security
UDDI	Universal Description, Discovery and Integration
URL	Uniform Resource Locator
VO	Virtual Organization
VOMS	Virtual Organization Membership Service
WAYF	Where Are You From
W3C	World Wide Web Consortium
WS	Web Service
WS-CAF	Web Services Composite Application Framework
WS-CDL	Web Services Choreography Description Language
WSDL	Web Services Description Language
WSO2	Open Source technology company
XACML	eXtensible Access Control Markup Language
XKMS	XML Key Management Specification
XML	Extensible Marke Language

B. Term Definitions

Authorization

Is the function of the policy definition phase which precedes the policy enforcement phase where access requests are granted or rejected based on the previously defined authorization policies. This uses the premise that the access control process is divided into two phases:

i) policy definition phase

ii) policy enforcement phase

Collaborator

Is being used as a short form of a domain of a collaborative partner.

Consistency

is a condition in which “there are no conflicting rules governing the access to a data item” [Gol11].

Credential

Is a digitally signed set of assertions issued by an recognized authority regarding a user’s privileges.

Domain

Is considered to be “a collection of hosts and routers, and the interconnecting network(s), managed by a single administrative authority [Mal96].” In this thesis, it is a collection of resources that are subject to be managed by a single administrator.

Principal

Is “an entity that can be granted access to objects or can make statements affecting access control decisions” [Gas90]. It is to be mapped to either a user, role, or a member of a group within a collaborator’s domain. This term has been used so that there is no confusion that within a local administrative realm, any of those entities above that are described as principal can be mapped to roles defined in inter-domain collaborative realm.

Similar term often used in access control field is *subject*. It is defined as “an active entity within an IT system [Gas90],” and it “operate[s] on behalf of human users we call *principals*” [Gas90] in this thesis. In order for subjects to operate on behalf of principals, they are bound to principals’ names in some unforgeable manner.

Privilege

refers to a right that can be granted to a principal. This and the term “permission” has been used interchangeably in this thesis.

Role

According to ANSI RBAC, a role is defined as "a job function within the context of an organization with some associated semantics regarding the authority and responsibility conferred on the user assigned to the role [ANS04]." What is stated as "job function" above can be defined as *procedure* which represents "a 'high-level' access control method with more complex semantic than read or write [Gol11]."

Reference Monitor

An access control concept that refers to an abstract machine that mediates all accesses to objects by subjects [Gol11].

Security policy

Is interchangeably used as authorization policy throughout this thesis

C. Collaboration Definition for Case Study "Ne Bis In Idem" in the WS-CDL Format

```
<package xmlns="http://www.w3.org/2004/04/ws-chor/cdl" xmlns:cdl="http://www.w3.
    org/2005/10/cdl" xmlns:tns="http://www.pi4soa.org/EU_EP" xmlns:xsd="http://www.
    w3.org/2001/XMLSchema" xmlns:cdl2="http://www.pi4soa.org/cdl2" author="
    epej_collabotion_designer" name="EU_EP" targetNamespace="http://www.pi4soa.org/
    EU_EP" version="0.1">
    <description type="documentation">
        The choreography description for EU_EP
    </description>
    <informationType name="BooleanType">
        <description type="documentation">
            This is the information type BooleanType
        </description>
    </informationType>
    <token informationType="tns:BooleanType" name="CollaboratorRef">
        <description type="documentation">
            This is the token CollaboratorRef
        </description>
    </token>
    <roleType name="CaseAnnouncer">
        <description type="documentation">
            This is the role type CaseAnnouncer
        </description>
        <behavior name="CaseAnnouncerBehavior">
            <description type="documentation">
                This is the behavior CaseAnnouncerBehavior
            </description>
        </behavior>
    </roleType>
    <roleType name="CaseFollower">
        <description type="documentation">
            This is the role type CaseFollower
        </description>
        <behavior name="CaseFollowerBehavior">
            <description type="documentation">
                This is the behavior CaseFollowerBehavior
            </description>
        </behavior>
    </roleType>
    <roleType name="Collaborator">
```

```
        <description type="documentation">
            This is the role type Collaborator
        </description>
        <behavior name="CollaboratorBehavior">
            <description type="documentation">
                This is the behavior CollaboratorBehavior
            </description>
        </behavior>
    </roleType>
    <roleType name="InterestedPartner">
        <description type="documentation">
            This is the role type InterestedPartner
        </description>
        <behavior name="InterestedPartnerBehavior">
            <description type="documentation">
                This is the behavior InterestedPartnerBehavior
            </description>
        </behavior>
    </roleType>
    <roleType name="BulletinWriter">
        <description type="documentation">
            This is the role type BulletinWriter
        </description>
        <behavior name="BulletinWriterBehavior">
            <description type="documentation">
                This is the behavior BulletinWriterBehavior
            </description>
        </behavior>
    </roleType>
    <relationshipType name="CaseAnnouncerToCaseFollowerRel">
        <description type="documentation">
            Relationship between CaseAnnouncer and CaseFollower
        </description>
        <roleType typeRef="tns:CaseAnnouncer"/>
        <roleType typeRef="tns:CaseFollower"/>
    </relationshipType>
    <participantType name="CaseAnnouncerParticipant">
        <description type="documentation">
            This is the participant type CaseAnnouncerParticipant
        </description>
        <roleType typeRef="tns:CaseAnnouncer"/>
    </participantType>
    <participantType name="CaseFollowerParticipant">
        <description type="documentation">
            This is the participant type CaseFollowerParticipant
        </description>
        <roleType typeRef="tns:CaseFollower"/>
    </participantType>
```

```
<participantType name="CollaboratorParticipant">
    <description type="documentation">
        This is the participant type CollaboratorParticipant
    </description>
    <roleType typeRef="tns:Collaborator"/>
</participantType>
<participantType name="InterestedPartnerParticipant">
    <description type="documentation">
        This is the participant type InterestedPartnerParticipant
    </description>
    <roleType typeRef="tns:InterestedPartner"/>
</participantType>
<participantType name="BulletinWriterParticipant">
    <description type="documentation">
        This is the participant type BulletinWriterParticipant
    </description>
    <roleType typeRef="tns:BulletinWriter"/>
</participantType>
<channelType name="CaseFollowerChannelType">
    <description type="documentation">
        This is the channel type CaseFollowerChannelType
    </description>
    <roleType typeRef="tns:CaseFollower"/>
    <reference>
        <token name="tns:CollaboratorRef"/>
    </reference>
</channelType>
<choreography name="EU_EPProcess" root="true">
    <description type="documentation">
        Choreography flow for the EU_EP process
    </description>
    <relationship type="tns:CaseAnnouncerToCaseFollowerRel"/>
    <variableDefinitions>
        <variable channelType="tns:CaseFollowerChannelType" name="
            CaseFollowerChannel">
            <description type="documentation">
                Channel to facilitate interaction to CaseFollower
            </description>
        </variable>
    </variableDefinitions>
    <interaction channelVariable="tns:CaseFollowerChannel" name="
        CaseAnnouncerRequest" operation="AnnounceCase">
        <participate fromRoleTypeRef="tns:CaseAnnouncer" relationshipType="
            tns:CaseAnnouncerToCaseFollowerRel" toRoleTypeRef="tns:CaseFollower
            "/>
        <exchange action="request" name="CaseAnnouncerRequestRequestExchange">
            <description type="documentation">
```

```
                This is the exchange details for the request exchange
                    associated with interaction CaseAnnouncerRequest
            </description>
            <send/>
            <receive/>
        </exchange>
    </interaction>
  </choreography>
</package>
```

Listing C.1: Specification of Europol and Eurojust Collaboration in WS-CDL

```
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" RuleCombiningAlgId="
    urn:oasis:names:tc:xacml:2.0:rule-combining-algorithm:first-applicable"
    PolicyId="1288181567373">
    <Description>Member State Persecution Authority</Description>
    <Target>
        <Subjects>
            <Subject>
                <SubjectMatch MatchId="urn:oasis:names:tc:xacml:2.0
                    :function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                        string">&iRole;MSPA</AttributeValue>
                </SubjectMatch>
            </Subject>
        </Subjects>
    </Target>
    <Rule Effect="Permit" RuleId="1288181567374">
        <Target>
            <Subjects>
                <Subject>
                    <SubjectMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                            #string">iRole;MSPA</AttributeValue>
                    </SubjectMatch>
                </Subject>
            </Subjects>
            <Resources>
                <Resource>
                    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                            #string">wscdl:CaseAnnouncer</AttributeValue>
                        <ResourceAttributeDesignator DataType="http://www.w3.org
                            /2001/XMLSchema#string" AttributeId="
                            urn:oasis:names:tc:xacml:2.0:resource:resource-id"/>
                    </ResourceMatch>
                </Resource>
            </Resources>
            <Actions>
                <Action>
                    <ActionMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="any-URI">PERFORM</AttributeValue
                            >
                        <ActionAttributeDesignator DataType="any-URI" AttributeId=
                            "urn:oasis:names:tc:xacml:2.0:action:action-id"/>
                    </ActionMatch>
                </Action>
            </Actions>
        </Target>
        <Environments>
            <Environment>
                <EnvironmentMatch MatchId="urn:oasis:names:tc:xacml:2.0
                    :function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                        string"/>
                    <EnvironmentAttributeDesignator DataType="http://www.w3.org
                        /2001/XMLSchema#string" AttributeId="
                        urn:oasis:names:tc:xacml:2.0:environment:environment-id"/>
                </EnvironmentMatch>
            </Environment>
        </Environments>
    </Rule>
    <Rule Effect="Permit" RuleId="1288181567375">
        <Target>
            <Subjects>
                <Subject>
                    <SubjectMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                            #string">iRole;MSPA</AttributeValue>
                    </SubjectMatch>
                </Subject>
            </Subjects>
            <Resources>
                <Resource>
                    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                            #string">wscdl:CaseFollower</AttributeValue>
                        <ResourceAttributeDesignator DataType="http://www.w3.org
                            /2001/XMLSchema#string" AttributeId="
                            urn:oasis:names:tc:xacml:2.0:resource:resource-id"/>
                    </ResourceMatch>
                </Resource>
            </Resources>
            <Environments>
                <Environment>
                    <EnvironmentMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                            #string">wscdl:EU_EP</AttributeValue>
                        <EnvironmentAttributeDesignator DataType="http://www.w3.
                            org/2001/XMLSchema#string" AttributeId="
                            urn:oasis:names:tc:xacml:2.0:resource:resource-id"/>
                    </EnvironmentMatch>
                </Environment>
            </Environments>
            <Actions>
                <Action>
                    <ActionMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="any-URI">PERFORM</AttributeValue
                            >
                        <ActionAttributeDesignator DataType="any-URI" AttributeId=
                            "urn:oasis:names:tc:xacml:2.0:action:action-id"/>
                    </ActionMatch>
                </Action>
            </Actions>
        </Target>
    </Rule>
    <Rule Effect="Permit" RuleId="1288181567376">
        <Target>
            <Subjects>
                <Subject>
                    <SubjectMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                            #string">iRole;MSPA</AttributeValue>
                    </SubjectMatch>
                </Subject>
            </Subjects>
            <Resources>
                <Resource>
                    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                            #string">wscdl:InterestedPartners</AttributeValue>
                        <ResourceAttributeDesignator DataType="http://www.w3.org
                            /2001/XMLSchema#string" AttributeId="
                            urn:oasis:names:tc:xacml:2.0:resource:resource-id"/>
                    </ResourceMatch>
                </Resource>
            </Resources>
            <Environments>
                <Environment>
                    <EnvironmentMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                            #string">wscdl:EU_EP</AttributeValue>
                        <EnvironmentAttributeDesignator DataType="http://www.w3.
                            org/2001/XMLSchema#string" AttributeId="
                            urn:oasis:names:tc:xacml:2.0:resource:resource-id"/>
                    </EnvironmentMatch>
                </Environment>
            </Environments>
            <Actions>
                <Action>
                    <ActionMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="any-URI">PERFORM</AttributeValue
                            >
                        <ActionAttributeDesignator DataType="any-URI" AttributeId=
                            "urn:oasis:names:tc:xacml:2.0:action:action-id"/>
                    </ActionMatch>
                </Action>
            </Actions>
        </Target>
    </Rule>
    <Rule Effect="Permit" RuleId="1288181567377">
        <Target>
            <Subjects>
                <Subject>
                    <SubjectMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                            #string">iRole;MSPA</AttributeValue>
                    </SubjectMatch>
                </Subject>
            </Subjects>
            <Resources>
                <Resource>
                    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                            #string">wscdl:BulletinWriter</AttributeValue>
                        <ResourceAttributeDesignator DataType="http://www.w3.org
                            /2001/XMLSchema#string" AttributeId="
                            urn:oasis:names:tc:xacml:2.0:resource:resource-id"/>
                    </ResourceMatch>
                </Resource>
            </Resources>
            <Environments>
                <Environment>
                    <EnvironmentMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                            #string">wscdl:EU_EP</AttributeValue>
                        <EnvironmentAttributeDesignator DataType="http://www.w3.
                            org/2001/XMLSchema#string" AttributeId="
                            urn:oasis:names:tc:xacml:2.0:resource:resource-id"/>
                    </EnvironmentMatch>
                </Environment>
            </Environments>
            <Actions>
                <Action>
                    <ActionMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="any-URI">PERFORM</AttributeValue
                            >
                        <ActionAttributeDesignator DataType="any-URI" AttributeId=
                            "urn:oasis:names:tc:xacml:2.0:action:action-id"/>
                    </ActionMatch>
                </Action>
            </Actions>
        </Target>
    </Rule>
</Policy>
```

Listing C.2: Definition of an iRole-MSPA (Member State Persecution Authority) in XACML

```
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" RuleCombiningAlgId="
    urn:oasis:names:tc:xacml:2.0:rule-combining-algorithm:first-applicable"
    PolicyId="1288181567380">
    <Description>Member State Persecution Authority</Description>
    <Target>
        <Subjects>
            <Subject>
                <SubjectMatch MatchId="urn:oasis:names:tc:xacml:2.0
                    :function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                        string">&iRole;MSPA</AttributeValue>
                </SubjectMatch>
            </Subject>
        </Subjects>
    </Target>
    <Rule Effect="Permit" RuleId="1288181567378">
        <Target>
```

```
            <Subjects>
                <Subject>
                    <SubjectMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                            #string">iRole;MSPA</AttributeValue>
                    </SubjectMatch>
                </Subject>
            </Subjects>
            <Resources>
                <Resource>
                    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                            #string">wscdl:CasePersecutor</AttributeValue>
                        <ResourceAttributeDesignator DataType="http://www.w3.org
                            /2001/XMLSchema#string" AttributeId="
                            urn:oasis:names:tc:xacml:2.0:resource:resource-id"/>
                    </ResourceMatch>
                </Resource>
            </Resources>
            <Environments>
                <Environment>
                    <EnvironmentMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                            #string">wscdl:EU_EP</AttributeValue>
                        <EnvironmentAttributeDesignator DataType="http://www.w3.
                            org/2001/XMLSchema#string" AttributeId="
                            urn:oasis:names:tc:xacml:2.0:resource:resource-id"/>
                    </EnvironmentMatch>
                </Environment>
            </Environments>
            <Actions>
                <Action>
                    <ActionMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="any-URI">PERFORM</AttributeValue
                            >
                        <ActionAttributeDesignator DataType="any-URI" AttributeId=
                            "urn:oasis:names:tc:xacml:2.0:action:action-id"/>
                    </ActionMatch>
                </Action>
            </Actions>
        </Target>
    </Rule>
</Policy>
```

Listing C.3: Definition of an iRole-MSPA with Case Persecutor Privilege in XACML

```
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" RuleCombiningAlgId="
    urn:oasis:names:tc:xacml:2.0:rule-combining-algorithm:first-applicable"
    PolicyId="1288181567381">
    <Description>Europol Liaison Officer</Description>
    <Target>
        <Subjects>
            <Subject>
                <SubjectMatch MatchId="urn:oasis:names:tc:xacml:2.0
                    :function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                        string">&iRole;EPLO</AttributeValue>
                </SubjectMatch>
            </Subject>
        </Subjects>
    </Target>
    <Rule Effect="Permit" RuleId="1288181567382">
        <Target>
            <Subjects>
                <Subject>
                    <SubjectMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                            #string">iRole;EPLO</AttributeValue>
                    </SubjectMatch>
                </Subject>
            </Subjects>
            <Resources>
                <Resource>
                    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                            #string">wscdl:</AttributeValue>
                        <ResourceAttributeDesignator DataType="http://www.w3.org
                            /2001/XMLSchema#string" AttributeId="
                            urn:oasis:names:tc:xacml:2.0:resource:resource-id"/>
                    </ResourceMatch>
                </Resource>
            </Resources>
            <Actions>
                <Action>
                    <ActionMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="any-URI"/>
                        <ActionAttributeDesignator DataType="any-URI" AttributeId=
                            "urn:oasis:names:tc:xacml:2.0:action:action-id"/>
                    </ActionMatch>
                </Action>
            </Actions>
        </Target>
        <Environments>
            <Environment>
                <EnvironmentMatch MatchId="urn:oasis:names:tc:xacml:2.0
                    :function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                        string"/>
                    <EnvironmentAttributeDesignator DataType="http://www.w3.org
                        /2001/XMLSchema#string" AttributeId="
                        urn:oasis:names:tc:xacml:2.0:environment:environment-id"/>
                </EnvironmentMatch>
            </Environment>
        </Environments>
    </Rule>
    <Rule Effect="Permit" RuleId="1288181567383">
        <Target>
            <Subjects>
                <Subject>
                    <SubjectMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                            #string">iRole;EPLO</AttributeValue>
                    </SubjectMatch>
                </Subject>
            </Subjects>
            <Resources>
                <Resource>
                    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                            #string">wscdl:Mediator</AttributeValue>
                        <ResourceAttributeDesignator DataType="http://www.w3.org
                            /2001/XMLSchema#string" AttributeId="
                            urn:oasis:names:tc:xacml:2.0:resource:resource-id"/>
                    </ResourceMatch>
                </Resource>
            </Resources>
            <Environments>
                <Environment>
                    <EnvironmentMatch MatchId="urn:oasis:names:tc:xacml:2.0
                        :function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
                            #string">wscdl:EU_EP</AttributeValue>
                        <EnvironmentAttributeDesignator DataType="http://www.w3.
                            org/2001/XMLSchema#string" AttributeId="
                            urn:oasis:names:tc:xacml:2.0:resource:resource-id"/>
                </EnvironmentMatch>
            </Environment>
        </Environments>
        <Actions>
            <Action>
                <ActionMatch MatchId="urn:oasis:names:tc:xacml:2.0
                    :function:string-equal">
                    <AttributeValue DataType="any-URI">PERFORM</AttributeValue
                        >
                    <ActionAttributeDesignator DataType="any-URI" AttributeId=
                        "urn:oasis:names:tc:xacml:2.0:action:action-id"/>
                </ActionMatch>
            </Action>
        </Actions>
    </Target>
  </Rule>
</Policy>
```

Listing C.4: Definition of an iRole-EPLO (Europol Liaison Officer) in XACML

Acknowledgements

Although pursuing a PhD is known as a self-taught study, I realize that certainly I would not have come to the completion of the study unless I had received help from numerous people. First and foremost, I would like to thank my advisor Prof. Dr. Dieter Gollmann for his outstanding mentorship in multiple areas. He has not only offered intellectual enlightenment and guidance throughout the past years of study, but he has also taught me a great deal of how to write, think, and articulate ideas. I am also grateful for his prompt reviews and consistent encouragement to bring this work to the completion. I would also like to thank Dr. rer. nat. Joachim Posegga for his supervision especially during my time at University of Hamburg, working for the EU project R4eGov. I appreciate his trust in my ability to take an active role in the project as well as his complementary advice in terms of making progress in the PhD work.

I would also like to thank the requirement engineering team of the R4eGov project for their valuable input in terms of assessing readiness of e-Governmental inter-domain collaborations. Additionally, parts of the work here have been implemented by my fellow students. I am thankful for their participation. Especially, I would like to name, in chronological order, Heiko Leudemann, Kunbuma Galega-Sabum, and Margarita Podskrobko for their contributions.

Colleagues at the Institute of Security in Distributed Applications (SVA) at the Technical University of Hamburg (TUHH) and those at the University of Hamburg have given me opportunities to talk about trivial to grand thoughts. These small talks have helped me to develop the topic. Especially, I would like to thank Michael von Riegen, Harald Sauff, Jan Meier, Tobias Jeske, Assadarat Khurat, Christopher Alm, Dr. Martin Johnes, and Daniel Schreckling for their listening ears and feedback.

I would also like to express my utmost gratitude to my family for their unconditional care and support. Last but not least, I thank my husband Dr. David Baker for his practical guidance, insightful questions, and consistent encouragement to bring this thesis into being.

List of Figures

Listings

List of Tables

Bibliography

[AAAB07] Alexandre Alves, Assaf Arkin, Sid Askary, and Charlton Barreto. Web Services Business Process Execution Language Version 2.0. [Online], Available: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html (Last viewed on 02/02/2011), April 2007.

[ABN07] A. Dalamanga A. Boujraf and M. Noble. Final Master Case Study of Collaborative Public Sector. *Towards e-Administration in the Large (R4eGov), Deliverable WP3-D7*, 2007.

[ACKM04] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web Services: Concepts, Architecture and Applications*. Springer Verlag, 2004.

[And05] Anne Anderson. Core and Hierarchical Role-based Access Control (RBAC) Profile of XACML v2.0. [Online], Available: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-spec-os.pdf (Last viewed on 02/02/2011), Feb. 2005.

[ANS04] American National Standards Institute, Inc.: American National Standard for Information Technology – Role Based Access Control (ANSI INCITS 359-2004). 2004.

[BBCC06] Siddharth Bajaj, Don Box, Dave Chappell, and Francisco Curbera. Web Services Policy 1.2 - Framework (WS-Policy). [Online], Available: http://www.w3.org/Submission/WS-Policy/ (Last viewed on 02/02/2011), April 2006.

[BBKKS05] S. Bharathi, A. Chervenak B. K. Kim, and R. Schuler. *Combining Virtual Organization and Local Policies for Automated Configuration of Grid Services*, chapter Combining Virtual Organization and Local Policies for Automated Configuration of Grid Services, pages 194–202. IOS Press, 2005.

[BEM03] András Belokosztolszki, David M. Eyers, and Ken Moody. Policy Contexts: Controlling Information Flow in Parameterised RBAC. *Policy 2003: IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, pages 99–110, 2003.

[BFIK99] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The KeyNote Trust-Management System Version 2. [Online], Available: http://www.ietf.org/rfc/rfc2704.txt (Last viewed on 02/02/2011), 1999.

[BJBG03] Rafae Bhatti, James B. D. Joshi, Elisa Bertino, and Arif Ghafoor. Access Control in Dynamic XML-based Web-Services with X-RBAC. *International Conference on Web Serivces*, June 2003.

[BSB$^+$05] Rafae Bhatti, Basit Shafiq, Elisa Bertino, Arif Ghafoor, and James B. D. Joshi. X-GTRBAC Admin: A Decentralized Administration Model for Enterprise-wide Access Control. *ACM Trans. Inf. Syst. Secur.*, 8:388–423, November 2005.

[CD00] Jan Camenisch and Ivan Damgård. Verifiable Encryption, Group Encryption, and Their Applications to Separable Group Signatures and Signature Sharing Schemes. In *Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT ’00, pages 331–345, London, UK, 2000. Springer-Verlag.

[CD07] Edward J. Coyne and John M. Davis. *Role Engineering for Enterprise Security Management*. Artech House, Inc., Norwood, MA, USA, 2007.

[CEC05] Green Paper on Conflicts of Jurisdiction and the Principle of Ne Bis in Idem in Criminal Proceedings. Dec. 2005.

[CHvRR04] Luc Clement, Andrew Hately, Claus von Riegen, and Tony Rogers. UDDI Version 3.0.2. [Online], Available: http://uddi.org/pubs/uddi-v3.0.2-20041019.htm (Last viewed on 02/02/2011), October 2004.

[CJ99] Thea Clark and Richard Jones. Organisational Interoperability Maturity Model for C2. *1999 Command and Control Research and Technology Symposium*, 1999.

[CKPM05] Scott Cantor, John Kemp, Rob Philpott, and Eve Maler. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. [Online], Available: http://docs.oasis-open.org/security/saml/v2.0/ (Last viewed on 02/02/2011), March 2005.

[CMRW07] Roberto Chinnici, Jean-Jacques Moreau, Arthur Ryman, and Sanjiva Weerawarana. Web Services Description Language (WSDL) Version 2.0. [Online], Available: http://www.w3.org/TR/2007/REC-wsdl20-20070626/ (Last viewed on 02/02/2011), June 2007.

[Coy96] Edward J. Coyne. Role Engineering. page 4, 1996.

[CZO$^+$08] David Chadwick, Gansen Zhao, Sassa Otenko, Romain Laborde, Linying Su, and Tuan Anh Nguyen. PERMIS: a Modular Authorization Infrastructure. *Concurr. Comput. : Pract. Exper.*, 20(11):1341–1357, 2008.

[DdLGvB06] Yuri Demchenko, Cees de Laat, Leon Gommans, and Rene van Buuren. Domain Based Access Control Model for Distributed Collaborative Applications. *E-SCIENCE ’06:*

Proceedings of the Second IEEE International Conference on e-Science and Grid Computing, page 24, 2006.

[EFL$^+$99] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI certificate theory. [Online], Available: http://www.ietf.org/rfc/rfc2693.txt (Last viewed on 02/02/2011), Sept 1999.

[Ell99] C. Ellison. RFC 2692: SPKI Requirements. [Online], Available: http://www.ietf.org/rfc/rfc2692.txt (Last viewed on 02/02/2011), September 1999.

[FCK95] D. Ferraiolo, J. Cugini, and R. Kuhn. Role-based Access Control (RBAC): Features and Motivations. *Proceedings of the 11th Annual Computer Security Applications Conference, IEEE Computers & Security Press*, pages 241–248, 1995.

[FH97] E. B. Fernandez and J. C. Hawkins. Determining Role Rights from Use Cases. In *RBAC '97: Proceedings of the second ACM workshop on Role-based access control*, pages 121–125, New York, NY, USA, 1997. ACM.

[FKSB05] I. Foster, H. Kishimoto, A. Savva, and D. Berry. The Open Grid Services Architecture, version 1.0. Jan. 2005.

[FP08] Ludwig Fuchs and Anton Preis. BusiROLE: A Model for Integrating Business Roles into Identity Management. In *TrustBus '08: Proceedings of the 5th international conference on Trust, Privacy and Security in Digital Business*, pages 128–138, Berlin, Heidelberg, 2008. Springer-Verlag.

[FPP$^+$02] E. Freudenthal, T. Pesin, L. Port, E. Keenan, and V. Karamcheti. dRBAC: Distributed Role-Based Access Control for Dynamic Coalition Environments. In *In Proceedings of the Twenty-second IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2002.

[Gas90] Morrie Gasser. The Role of Naming in Secure Distributed Systems. In *Proceedings of the CS'90 Symposium on Computer Security*, pages 97–109, Rome, Italy, November 1990.

[GHMM07] Martin Gudgin, Marc Hadley, Noah Mendelsohn, and Jean-Jacques Moreau. SOAP Version 1.2. [Online], Available: http://www.w3.org/TR/soap12-part1/ (Last viewed on 02/02/2011), April 2007.

[GMPT01] Christos K. Georgiadis, Ioannis Mavridis, George Pangalos, and Roshan K. Thomas. Flexible team-based access control using contexts. In *Proceedings of the sixth ACM symposium on Access control models and technologies*, SACMAT '01, pages 21–27, New York, NY, USA, 2001. ACM.

[Gol11] Dieter Gollmann. *Computer Security*. John Wiley and Sons Ltd., 3rd edition, 2011.

[GQ96] Li Gong and Xiaolei Qian. Computational Issues in Secure Interoperation. *IEEE Trans. Softw. Eng.*, 22(1):43–52, 1996.

[HMHX07] Vincent C. Hu, Evan Martin, JeeHyun Hwang, and Tao Xie. Conformance Checking of Access Control Policies Specified in XACML. In *Proceedings of the 31st Annual International Computer Software and Applications Conference - Volume 02*, COMPSAC '07, pages 275–280, Washington, DC, USA, 2007. IEEE Computer Society.

[IDR04] Kazunori Iwasa, Jacques Durand, and Tom Rutt. WS-Reliability 1.1. [Online], Available: http://docs.oasis-open.org/wsrm/ws-reliability/v1.1/wsrm-ws_reliability-1.1-spec-os.pdf (Last viewed on 02/02/2011), November 2004.

[JBBG04] James B.D. Joshi, Rafae Bhatti, Elisa Bertino, and Arif Ghafoor. Access-Control Language for Multidomain Environments. *IEEE Internet Computing*, 08(6):40–50, 2004.

[JNT] UK Federation: How It Works. JNT Association.

[KAMCM00] Apu Kapadia, Jalal Al-Muhtadi, Roy H. Campbell, and Dennis Mickunas. IRBAC 2000: Secure Interoperability Using Dynamic Role Translation. Technical report, Champaign, IL, USA, 2000.

[KBR$^+$05] Nickolas Kavantzas, David Burdett, Gregory Ritzinger, Tony Fletcher, Yves Lafon, and Charlton Barreto. Web Services Choreography Description Language Version 1.0. [Online], Available: http://www.w3.org/TR/ws-cdl-10/ (Last viewed on 02/02/2011), November 2005.

[KCH08] Paul El Khoury, Emmanuel Coquery, and Mohand-Said Hacid. Consistency Checking of Role Assignments in Inter-organizational Collaboration. In *SPRINGL '08: Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS*, pages 82–88, New York, NY, USA, 2008. ACM.

[Ker02] A. Kern. Advanced Features for Enterprise-wide Role-based Access Control. In *Computer Security Applications Conference, 2002. Proceedings. 18th Annual*, pages 333 – 342, 2002.

[KKSM02] Axel Kern, Martin Kuhlmann, Andreas Schaad, and Jonathan Moffett. Observations on the Role Life-Cycle in the Context of Enterprise Security Management. In *SACMAT '02: Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 43–51, New York, NY, USA, 2002. ACM.

[KSS03] Martin Kuhlmann, Dalia Shohat, and Gerhard Schimpf. Role Mining – Revealing Business Roles for Security Administration Using Data Mining Technology. In *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 179–186, New York, NY, USA, 2003. ACM.

[KW05] Axel Kern and Claudia Walhorn. Rule Support for Role-based Access Control. In *SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 130–138, New York, NY, USA, 2005. ACM.

[LABS06] Hal Lockhart, Steve Andersen, Jeff Bohren, and Yakov Sverdlov. Web Services Federation Language (WS-Federation). [Online], Available: http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-fed/WS-Federation-V1-1B.pdf (Last viewed on 02/02/2011), December 2006.

[Lee07] H.K. Lee. Unraveling Decentralized Authorization for Multi-domain Collaborations. *International Conference on Collaborative Computing: Networking, Applications and Worksharing, 2007. CollaborateCom 2007.*, pages 33–40, Nov. 2007.

[LKNG07a] Kelvin Lawrence, Chris Kaler, Anthony Nadalin, and Marc Goodner. WS-SecureConversation 1.3. [Online], Available: http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.html (Last viewed on 02/02/2011), March 2007.

[LKNG07b] Kelvin Lawrence, Chris Kaler, Anthony Nadalin, and Marc Goodner. WS-Trust 1.3. [Online], Available: http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html (Last viewed on 02/02/2011), March 2007.

[LKNM06] Kelvin Lawrence, Chris Kaler, Anthony Nadalin, and Ronald Monzillo. Web Services Security: SOAP Message Security 1.1. [Online], Available: http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf (Last viewed on 02/02/2011), Feb. 2006.

[LKW$^+$09] Romain Laborde, Michel Kamel, Samer Wazan, Francois Barrere, and Abdelmalek Benzekri. A Secure Collaborative Web-based Environment for Virtual Organisations. *Int. J. Web Based Communities*, 5(2):273–292, 2009.

[LL07] Hannah Lee and Heiko Luedemann. A Lightweight Decentralized Authorization Model for Inter-domain Collaborations. In *Proc. ACM Workshop on Secure Web Services*, Nov. 2007.

[LMW02] Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a Role-based Trust Management Framework. In *Proc. IEEE Symposium on Security and Privacy, Oakland*, May 2002.

[LMW$^+$04] Ninghui Li, John C. Mitchell, William H. Winsborough, Kent E. Seamons, Michael Halcrow, and Jared Jacobson. RTML: A Role-based Trust-management Markup Language. Technical report, Purdue University, 2004.

[Lue08] Heiko Luedemann. Modeling Authorization Policy for Inter-domain Collaborations. Technical report, University of Hamburg, 2008.

[Mal96] G. Malkin. Internet Users' Glossary - RFC 1983, Network Working Group. [Online], Available: http://www.rfc-editor.org/rfc/rfc1983.txt (Last viewed on 02/02/2011), August 1996.

[MLL$^+$09] Ian Molloy, Ninghui Li, Tiancheng Li, Ziqing Mao, Qihua Wang, and Jorge Lobo. Evaluating Role Mining Algorithms. In *SACMAT '09: Proceedings of the 14th ACM symposium on Access control models and technologies*, pages 95–104, New York, NY, USA, 2009. ACM.

[Mos05] Tim Moses. eXtensible Access Control Markup Language (XACML) Version 2.0. [Online], Available: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf (Last viewed on 02/02/2011), Feb. 2005.

[MSSN04] Jan Mendling, Mark Strembeck, Gerald Stermsek, and Gustaf Neumann. An Approach to Extract RBAC Models from BPEL4WS Processes. In *WETICE '04: Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 81–86, Washington, DC, USA, 2004. IEEE Computer Society.

[MW06] Mingchao Ma and Steve Woodhead. Constraint-Enabled Distributed RBAC for Subscription-Based Remote Network Services. In *CIT '06: Proceedings of the Sixth IEEE International Conference on Computer and Information Technology (CIT'06)*, page 160, Washington, DC, USA, 2006. IEEE Computer Society.

[NKMHB06] Anthony Nadalin, Chris Kaler, Ronald Monzillo, and Phillip Hallam-Baker. Web Services Security: SOAP Message Security 1.1. [Online], Available: http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf (Last viewed on 02/02/2011), Feb. 2006.

[NS02] Gustaf Neumann and Mark Strembeck. A Scenario-driven Role Engineering Process for Functional RBAC roles. In *SACMAT '02: Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 33–42, New York, NY, USA, 2002. ACM.

[PCND04] Joon S. Park, Keith P. Costello, Teresa M. Neven, and Josh A. Diosomito. A Composite RBAC Approach for Large, Complex Organizations. *SACMAT '04: Proceedings of the ninth ACM symposium on Access control models and technologies*, pages 163–172, 2004.

[PJ05] Smithi Piromruen and James B. D. Joshi. An RBAC Framework for Time Constrained Secure Interoperation in Multi-domain Environment. *IEEE Workshop on Object-oriented Real-time Databases (WORDS-2005)*, 2005.

[PM05] Aneta Poniszewska-Maranda. Role Engineering of Information System Using Extended RBAC Model. In *WETICE '05: Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*, pages 154–159, Washington, DC, USA, 2005. IEEE Computer Society.

[PWF$^+$02] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke. A Community Authorization Service for Group Collaboration. In *POLICY '02: Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*, page 50, Washington, DC, USA, 2002. IEEE Computer Society.

[RNFJ09] Ian Robinson, Eric Newcomer, Max Feingold, and Ram Jeyaraman. WS-Coordination v1.2. [Online], Available: http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec-os.doc (Last viewed on 02/02/2011), Feb. 2009.

[RNFL07] Ian Robinson, Eric Newcomer, Tom Freund, and Mark Little. Web Services Business Activity V1.1. [Online], Available: http://docs.oasis-open.org/ws-tx/wstx-wsba-1.1-spec-os.pdf (Last viewed on 02/02/2011), April 2007.

[RNLW09] Ian Robinson, Eric Newcomer, Mark Little, and Andrew Wilkinson. WS-Atomic Transaction v1.2. [Online], Available: http://docs.oasis-open.org/ws-tx/wsat/wstx-wsat-1.2-spec-os.doc (Last viewed on 02/02/2011), Feb. 2009.

[RSW00] Haio Roeckle, Gerhard Schimpf, and Rupert Weidinger. Process-oriented Approach for Role-finding to Implement Role-based Security Administration in a Large Industrial Organization. In *RBAC '00: Proceedings of the fifth ACM workshop on Role-based access control*, pages 103–110, New York, NY, USA, 2000. ACM.

[SAI04] Science Applications International Corporation SAIC. Role-based Access Control (RBAC) Role Engineering Process (V.3). *Developed for: the Healthcare RBAC Task Force*, 2004.

[San00] Ravi Sandhu. Engineering Authority and Trust in Cyberspace: The OM-AM and RBAC Way. In *In Proceedings of 5th ACM Workshop on Role-Based Access Control*, pages 111–119. ACM, 2000.

[SCFY96] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47, 1996.

[SJBG05] Basit Shafiq, James B.D. Joshi, Elisa Bertino, and Arif Ghafoor. Secure Interoperation in a Multidomain Environment Employing RBAC Policies. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1557–1577, 2005.

[Ste91] Daniel F. Sterne. On the Buzzword 'Security Policy'. *Security and Privacy, IEEE Symposium on*, page 219, 1991.

[SWS08] Anoop Singhal, Theodore Winograd, and Karen Scarfone. Guide to Secure Web Services: Recommendations of the National Institute of Standards and Technology. *NIST Special Publication 800-95*, Aug. 2008.

[TA2] Technical Annex of R4eGov: Towards e-Administration in the large (IST-2004-026650).

[TAPH05] William Tolone, Gail-Joon Ahn, Tanusree Pai, and Seng-Phil Hong. Access Control in Collaborative Systems. *ACM Comput. Surv.*, 37:29–41, March 2005.

[TC08] Fatih Turkmen and Bruno Crispo. Performance evaluation of XACML PDP implementations. In *Proceedings of the 2008 ACM Workshop on Secure Web Services*, SWS '08, pages 37–44, New York, NY, USA, 2008. ACM.

[Tho97] Roshan K. Thomas. Team-based access control (TMAC): a primitive for applying role-based access controls in collaborative environments. In *Proceedings of the second ACM workshop on Role-based access control*, RBAC '97, pages 13–19, New York, NY, USA, 1997. ACM.

[TNI87] National Computer Security Center. Trusted Network Interpretation, NCSC-TG-005, Version 1.0, 1987.

[TOB98] D. Thomsen, D. O'Brien, and J. Bogle. Role-Based Access Control Framework for Network Enterprises. In *ACSAC '98: Proceedings of the 14th Annual Computer Security Applications Conference*, page 50, Washington, DC, USA, 1998. IEEE Computer Society.

[un208] UN E-Government Survey 2008: From E-Government to Connected Governance. page 246, 2008.

[US06] Y. B. Udupi and M. P. Singh. Multiagent Policy Architecture for Virtual Business Organizations. In *Proceedings of the IEEE International Conference on Services Computing*, pages 44–51, Washington, DC, USA, 2006.

[VAG07] *The Role Mining Problem: Finding a Minimal Descriptive Set of Roles*, SACMAT '07, New York, NY, USA, 2007. ACM.

[vRR09] Michael von Riegen and Norbert Ritter. Reliable Monitoring for Runtime Validation of Choreographies. In Mark Perry, Hideyasu Sasaki, Matthias Ehrmann, Guadalupe Ortiz Bellot, and Oana Dini, editors, *The Fourth International Conference on Internet and Web Applications and Services, ICIW 2009, 24-18 May, Venice/Mestre, Italy*, pages 310–315. IEEE, 5 2009.

[WSY^{+}08] Xinyu Wang, Jianling Sun, Xiaohu Yang, Chao Huang, and Di Wu. Security Violation Detection for RBAC Based Interoperation in Distributed Environment. *IEICE - Trans. Inf. Syst.*, E91-D(5):1447–1456, 2008.

[ZRE07] Dana Zhang, Kotagiri Ramamohanarao, and Tim Ebringer. Role engineering using graph optimisation. In *SACMAT '07*, pages 139–144, 2007.

[ZRV09] Dana Zhang, Kotagiri Ramamohanarao, Steven Versteeg, and Rui Zhang 0003. Rolevat: Visual assessment of practical need for role based access control. In *ACSAC '09*, pages 13–22, 2009.