ETH zürich

ETH Library

Usage control in secure messaging

Master Thesis

Author(s):
Jotterand, Arnaud

Publication date:
2008

Permanent link:
https://doi.org/10.3929/ethz-a-005565539

Rights / license:
In Copyright - Non-Commercial Use Permitted

This page was generated automatically upon download from the ETH Zurich Research Collection.
For more information, please consult the Terms of use.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Arnaud Jotterand
ajotterand@student.ethz.ch

Usage Control in Secure Messaging

Master Thesis
April 2008

Supervisor: Professor David Basin

Abstract

In order to protect sensitive data that is more and more distributed over networks, one needs effective control techniques. The concept of usage control extends traditional access control by controlling both the access to data and the usage of it after access has been granted. A lot of work has been done in defining the concept of usage control, but it often stays at a theoretic level. This work presents an interesting case study of usage control in the area of secure messaging and it describes how usage control techniques may be used in a practical environment. It also proposes a design and the implementation of a solution for this use case.

Acknowledgment

I would like to thank the following people for the help they have provided in one way or another during the elaboration of this thesis:

- **Professor David Basin**, my supervisor at the ETH Zurich, for his support and advice
- **Doctor Ralf Hauser**, my supervisor at Privasphere AG, for his extensive personal knowledge and the experience he has shared with me.
- **Mr. Adrian Rufener** for his help in defining the requirements of a secure data room application
- **Messrs. Paul Frei, Beat Weber, Roger Willen, Michael Wiedemann and Jan Bielik**, workers at Privasphere AG, for their good temper and the sweets they provided during our coffee breaks
- **Mr. Paul F. Hartzell**, Senior Vice President of DataSite at Merrill Corporation, for his help in defining the requirements of a secure data room application
- **Miss Alexandra Eriksson** for her help in correcting my bad English
- **My family and my friends** for their support and their comprehension during these six months.

I would also like to apologize to the people I may have forgotten...

Contents

1. Introduction **7**
1.1. Context . . . 7
1.2. Related work . . . 8
1.3. Summary . . . 8
1.4. Contribution . . . 9
1.5. Overview . . . 10

2. Background **11**
2.1. Usage Control . . . 11
2.1.1. The UCON model . . . 12
2.2. eXtensible Access Control Markup Language (XACML) . . . 15
2.2.1. Architecture . . . 15
2.2.2. XACML context . . . 16
2.2.3. Policy language . . . 17
2.3. Role Based Access Control (RBAC) . . . 19

3. Use Case in Secure Messaging **21**
3.1. Secure Data Room . . . 21
3.1.1. Use case . . . 22
3.1.2. Requirements . . . 23
3.2. Usage control aspects . . . 24
3.2.1. Access control . . . 24
3.2.2. Usage control . . . 26
3.2.3. Legal issues . . . 26
3.2.4. Attractiveness of the use case . . . 27

4. Proof of Concept **29**
4.1. Design . . . 29
4.1.1. Web interface . . . 29
4.1.2. Architecture . . . 30
4.1.3. Identity management . . . 31
4.1.4. Policies . . . 33
4.1.5. Usage control . . . 37
4.1.6. Legal issues . . . 39
4.2. Implementation . . . 41
4.2.1. Framework . . . 41

4.2.2. Version control 42
4.2.3. Source code 42

5. Evaluation **45**
5.1. Benefits of Usage Control 45
5.2. Quality of design 45
5.3. Limitations 46
5.4. Experience 46
5.5. Extensions 48

6. Conclusion and future work **49**

A. Acronyms **51**

List of Figures

2.1. UCON model components 12
2.2. Major XACML model components 16
2.3. XACML context 17
2.4. XACML policy language model 18
2.5. RBAC profile principles 20

3.1. Secure Data Room - Use Case 22
3.2. Access control - Use case 25

4.1. Sections of the web application 30
4.2. Architecture of the application 31
4.3. Information flow for access request 32
4.4. Tree structure of the secure data room 42
4.5. Java packages of the application 44

Chapter 1.

Introduction

1.1. Context

Increasing amounts of information are available, which support our lives in many beneficial ways. However, data is collected with an ever increasing granularity and this can also threaten their owning individuals or subjects. Data protection laws were an early attempt to address these problems on a legal basis. Technically, so called Mandatory Access Control (MAC) has existed for decades in highly structured institutions such as defense and secret services. The inherent rigidity and inflexibility of such a system have limited significantly the pervasiveness of their use. Recently, some industries concerned with illegitimate utilization of protected assets (with intellectual property rights, e.g., music) have become interested in controlling the usage of such assets and mandate systems known under the term Digital Rights Management (DRM) that attempt to further develop similar copy protection mechanism that we remember from the 80ies and early 90ies to prevent software privacy. Such systems are often disliked, and thus are the source of conceptual objections such as *fair use* arguments; moreover they are the subject of relentless researches for presenting conceptual flaws and reverse engineering attempts by large global communities.

Ongoing research has, however, come up with promising frameworks to effectively model usage control with full acknowledgement of their limited enforceability. These frameworks allow data providers to formulate realistic usage control policies based on feasible provisions and consumer obligations. These can be complemented with effective observation and compensation strategies for detected violations if a high-level policy is not fully enforceable.

The field of secure messaging is a prime candidate to prototype usage control architectures. One the one hand, many basic ingredients to usage control such as authentication mechanisms at various levels, confidentiality and integrity methods are used daily. On the other hand, users of such systems normally entrust data with higher value to them and thus have an interest in adopting additional security measures such as usage control.

Therefore, they are more likely to accept the additional efforts required and adopt the process changes needed to implement some usage control, at least on the server side, and to accept possibles inconveniences on the client side.

The purpose of this work is to present an interesting use case for usage control in the area of secure messaging, and to describe how usage control techniques may be used in a practical environment.

1.2. Related work

A lot of work has been done in defining the concept of usage control and its main components, and many interesting papers have been published on the subject.
[18] presents an excellent general view of what is usage control, and the idea of provisions and obligations is explained and formalized in [7].
The theme of enforcement in usage control is addressed in [9, 13] by giving an overview of the existing control mechanisms in the area of DRM.
The problem of evaluating policies and the idea of negotiation are discussed in [11].
Usage control has been modelled in the UCON model; a good description of this is model can be found in [16, 17].
Many policy languages have been proposed, but often without any formalization. An effort has been made with Obligation Specification Language (OSL) [8] that supports the formalization of a wide range of usage control requirements and allows the translation to/from other languages.
[22] proposes solutions to control access to data in documents with the help of trusted computing.
[5] applies a semantic approach based on web ontologies to DRM.
[21] aims at designing an appropriate architecture to enforce dynamic access control by suggesting an approach based on a mathematical formalism, and [3] does an analysis of DRM architectures and proposes a refinement of these.

One can observe here that many aspects of usage control have been studied, and many solutions have been proposed. But as often in the academic world, there is a lack of practical work in the area.

1.3. Summary

When two companies decide to merge, the target company provides documents to the purchaser company during what one calls a due diligence process. These documents are generally put in a room (called data room) where the purchaser company can consult

them. The access to this room is controlled by traditional physical security equipments such as cameras or personal access cards, and the documents have to be consulted on site, without any possibility to do copies of any kind. Such data rooms are specially created during a due diligence process, and are closed at the end of it. The development of information technologies has allowed to transpose this idea of data room into an electronic version, available via a network: a secure data room. The goal is to simplify the due diligence process, without loosing the security offered by a traditional data room. Another goal is to be able to prove after the due diligence process what documents where at disposal of the auditors during this process; this is particularly useful at trial when the merger is not successful.

This work proposes to use the notion of usage control in order to meet these security requirements. First of all, one precisely defines what are these security requirements for such an application; this is done with the help of specialists and potential end-users. After the analysis of these requirements, one explains how usage control can be a solution to the problem. As a proof of concept, one describes the prototype of a web-based secure data room application, covering aspects such as access control to web pages, identity management, definition of permissions on documents and enforcement of these on the server side. One also proposes solutions to legal issues such as knowing and proving what information was effectively disclosed during the due diligence process using cryptographic tools such as digital signatures.

This thesis explains how usage control policies may be easily applied for controlling access to a web interface, allowing to centralize the permissions definitions. It also explains how usage control policies may be dynamically defined by a producer who is an end-user of an application. Moreover, this work shows that the enforcement of usage control policies requires sometimes very complex and costly technologies such as trusted computing, and that one has to find easier alternatives to implement them but where the enforcement is not guaranteed.

1.4. Contribution

As discussed in 1.2, a lot of work has been done in defining the concept of usage control and many interesting papers have been published, but they often stay at a theoretic level. This work contributes to the development of the concept of usage control by offering a discussion about the effective implementation of usage control mechanisms in a concrete use case, which is clearly needed. This work may be seen as a bridge between theory and practice, enabling to compare the advantages and disadvantages of some usage control mechanisms proposed in the literature.

In the literature, one defines the syntax and the semantic of policies and how they may be interpreted in order to take a decision, but not by whom and when these policies are written. These policies are often complex, and can therefore only be written by experts in usage control. Another contribution of this work highlights a use case where the end-user, who does not necessarily have good security notions, is asked to define these policies. The policies are then not statically written, but dynamically during the exploitation of the application. Therefore one has to find a way to present usage control policies in a simplified way, comprehensible to any user. Such a work of abstraction is interesting for anyone defining policy languages and definitions.

Moreover, in the literature of usage control, one often considers only the consumer to be watched and controlled. In the area of DRM for instance, one considers the producer to be trusted, and one therefore concentrates the usage control on the consumer side[1]. In this work, both parties have to be controlled for the legal reasons explained in 1.3. As a solution to this problem, one proposes for example to use an obligation stating that the event of posting a document (i.e. producing an object in the UCON terminology) has to be logged. This enables to know exactly what information was disclosed; observability is then achieved in that case.
By presenting a case where both the producer and the consumer have to be controlled, this work shows an interesting new perspective in the area of usage control.

Finally, the implementation proposed in this work as a proof of concept is delivered with an open source license, and can therefore serve as a basis for future implementations.

1.5. Overview

After having given a short overview of some of the concepts and technologies used in this work in chapter 2, one presents the chosen use case in chapter 3. As a proof of concept, one describes the design and the implementation of a solution to this use case in chapter 4. The evaluation of the proposed solution is then discussed in chapter 5.

[1] This is true from a technological point of view. The producer is also watched, but more from an ethical point of view in conceptual objections such as *fair use* arguments.

Chapter 2.

Background

2.1. Usage Control

Access control has been studied for more than 30 years now. Some well known access control models are the access matrix model, the lattice-based access control model, and the role-based access control model. These traditional access control models have found it difficult to address the needs of modern information systems. One of the main reasons is that traditional access control models have focused on authorization only. Here, authorization evaluates access requests based on subject attributes, object attributes and requested rights. However, modern information systems often require more than authorizations. For example, one may have to fill out a certain form or click the *yes* button for license agreement for usage allowance. We call these required actions as obligations. Obligations are requirements that have to be fulfilled by obligation subjects for usage allowance. Moreover, some digital objects can be played only on a certain device or location. These environmental restrictions are called conditions. Conditions are environmental and system-wide requirements that have to be satisfied for access. Obligations and conditions are rarely discussed in traditional access control models. In today's highly dynamic and distributed environment, obligations and conditions are also crucial decision factors for richer and finer controls on the usage of digital resources.

Also in traditional access control models, the authorization decision is made before access is allowed and no further enforcement is possible during the access. Hence there is no ongoing control concept considered. Another shortcoming is that consumable rights are not supported. In modern e-commerce systems, it is common to use consumable attributes or rights such as credit balance or a limited number of usages. More fundamentally, in traditional access control, rights are predefined and granted to subjects. This means that subjects hold granted rights for indefinite time whether the subjects actually exercise the rights or not. This might be fine for some authorization-based controls. However, this is not acceptable for obligation-based or condition-based controls as well as for other dynamic authorization controls.

2.1.1. The UCON model

As one can see, the notion of access control has shown its limits. In order to integrate obligation-based or condition-based controls, the notion of usage control has emerged. This notion of usage control has been modelled in the UCON model. The overview of the UCON model explained here is inspired from [16, 17].

The UCON model consists of three core components and three additional components that are mainly involved in the authorization process (see figure 2.1). Core components comprise subjects, objects, and rights. Each core component can be divided into several detailed components with different perspectives. Traditional access control policies also define similar components to UCON core components though the detailed definitions of these components in various access control policies are somewhat different from each other. Additional components include authorization rules, conditions, and obligations. In the UCON system at least the authorization rules (specifically rights-related authorization rule which will be discussed in this section) have to be included for authorization. Conditions and obligations can also be used in the authorization process.

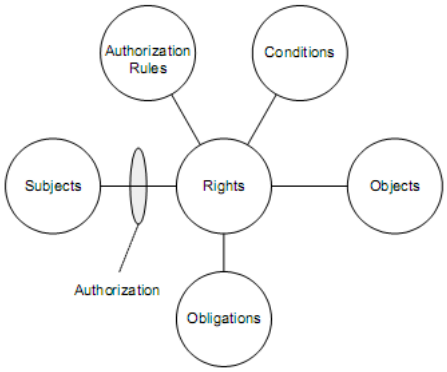

Figure 2.1.: UCON model components

2.1.1.1. Subjects

Subjects are entities associated with attributes, and they hold and exercise certain rights on objects. Attributes are properties of the subjects that can be used for the authorization process. Examples of attributes include identities, roles, credits, memberships, security levels, etc. A subject can be a user, a group, a role, or a process. A user is an individual entity that has certain rights on an object. A group is a set of users who hold the same rights as a group. A role is a named collection of users and relevant permissions. Groups and roles may have hierarchical relationships. The idea of role that one uses in this work is further discussed in section 2.3.

In UCON, the subjects can be consumer subjects (*CS*), provider subjects (*PS*), and identifiee subjects (*IS*). Consumer subjects are entities who receive rights and objects and use the rights to access the objects. An e-book reader, MP3 music player/listener and even a distributor of digital objects can be a consumer subject. Provider subjects are entities who provide an object and hold certain rights on it. Examples of provider subjects include an author of an e-book, a distributor of the book, a primary physician, etc. The identifiee subjects are entities who are identified in digital objects that include their privacy-sensitive information. A patient of a health care system is an example of an identifiee subject. Although the concept of identifiee subjects always exists in the case of privacy-sensitive information, identifiee subjects may or may not be included within UCON systems based on other control policies.

2.1.1.2. Objects

Objects are entities that subjects hold rights on, whereby the subjects can access or use objects. Objects are also associated with attributes, either by themselves or together with rights. Similarly to subjects, the attributes include certain properties that can be used for the authorization process. Examples of object attributes are security levels, ownerships, classes, etc. Object classes are used to categorize objects so that authorization can be defined based not only on individual objects but also on sets of objects that belong to the same class. In some cases, objects or objects with attributes (i.e. classes) are associated with attributes together with rights. Examples of the attributes for objects with rights are credits, roles, memberships, etc. The credits may be used to define how many credits are required to obtain a certain right on a specific object. For example, the *Harry Potter* e-book together with a read right may require CHF 10 or the book with an additional print right may require CHF 15.

In UCON, objects can be either privacy sensitive or privacy non-sensitive. A privacy-sensitive object includes individually identifiable information that can cause privacy problems if not used properly. An UCON object can be either original or derivative. The derivative object in UCON is different from the derivative object of the DRM literature. In the latter, the term *derivative* means derived (cited, quoted, or copied) from an original work to create another digital work that includes parts of the original work. In UCON, however, the derivative object is an object that is created in consequence of obtaining or exercising rights on an original object. For example, playing MP3 music file can create usage log information. This log data file is called a derivative object in UCON. Like the original object, this derivative object is also considered as an object and also holds UCON properties and relations with other components. Based on their format, objects can be documents (e.g. .doc, .pdf, .ps), audio (e.g. .mp3, .wav), video (e.g. JPEG, DVD, MPEG) or executable files (e.g. games) and so on. Each format may require the use of its own application tools. The objects may or may not have hierarchy on them.

2.1.1.3. Rights

Rights are privileges that a subject can hold on an object. Rights consist of a set of usage functions that enables a subject's access to objects. Examples of usage could be to listen to an audio file, burn it on a CD, or make a copy of it. The authorizations of rights require associations with subjects and objects. Rights may or may not have a hierarchy. Like subjects and objects, rights can also be divided into consumer rights (CR), provider rights (PR), and identifiee rights (IR). The rights include rights for the access and the use of objects and rights for delegation of rights.

2.1.1.4. Authorization rules

Authorization rules are a set of requirements that should be satisfied before allowing subjects' access to objects or use of objects. There exist two kinds of authorization rules. They are Rights-related Authorization Rules (RAR) and Obligation-related Authorization Rules (OAR). The RAR is used to check whether a subject has a valid privilege to exercise certain rights on a digital object. Examples include identities or roles verification, capabilities or properties checking, proof of payments, etc. The OAR is used to check if a subject has agreed on the fulfilment of an obligation which has to be done after obtaining or exercising rights on a digital object. Examples include metered payment agreement, usage log report agreement, etc. The authorization rules are different from conditions. The authorization rules are a set of decision factors used to assess whether a subject is qualified for the use of certain rights on an object, whereas the condition is used to check whether existing limitations and the status of usage rights on an object are valid and whether those limitations have to be updated.

2.1.1.5. Conditions

Conditions are a set of decision factors that the system should verify during the authorization process along with authorization rules before allowing usage of rights on a digital object. There are two types of conditions: Dynamic conditions and Static conditions. Dynamic conditions include information that may have to be checked for updates at each time of usage. Static conditions include information that does not have to be checked for updates. Dynamic conditions are stateful and the static conditions are stateless. Some examples of dynamic conditions are the number of usage times (e.g. can read 5 times, can print 2 times), and usage log (e.g. already read portion cannot be accessed again). Some examples of static conditions are accessible time periods (e.g. business hours), accessible locations (e.g. workplace), and allowed printer names.

2.1.1.6. Obligations

Obligations are mandatory requirements that a subject has to perform after obtaining or exercising rights on an object. In a real world implementation, however, this may have to be done by agreeing on the fulfilment of obligations before obtaining the rights and obligation-related authorization rules are checked. A possible example, a consumer subject would have to accept metered payment agreements before obtaining the rights for the usage of certain digital information. Another example, the consumer subject would have to agree on providing usage log information to a provider subject before reading an e-book or listening a music file. Traditional access control has hardly recognized the obligation concept. Recent DRM solutions are likely to include obligation functions though many of them implement the obligation functions only partially and implicitly.

2.2. eXtensible Access Control Markup Language (XACML)

As presented above in 2.1.1, the UCON model defines the notions of authorization rules, conditions and obligations that are involved in the authorization process. How these notions can effectively be expressed is however not included in this model. This can be done using a Rights Expression Language (REL). There exist many of these languages; the one chosen for this work is XACML, as it allows to easily express all these notions (as discussed in 2.2.3), it is extensible if needed, and it has been rigorously formalized using description logic [10]. The XACML 2.0 specification defines not only a policy and request/response language, but also a data-flow model for evaluating access requests. One gives here a general overview of this specification.

2.2.1. Architecture

The XACML 2.0 specification defines a data-flow model for evaluating access requests. The major actors in the XACML domain, shown in figure 2.2, are the following:

Policy Enforcement Point (PEP) The system entity that performs access control, by making decision requests and enforcing authorization decisions with the help of an obligations service entity. The latter manages operations specified in a policy that should be performed by the PEP in conjunction with the enforcement of an authorization decision.

Policy Decision Point (PDP) The system entity that evaluates applicable policy and renders an authorization decision. The context handler serves as a converter with the PEP and is responsible for providing context attributes necessary for the decision making.

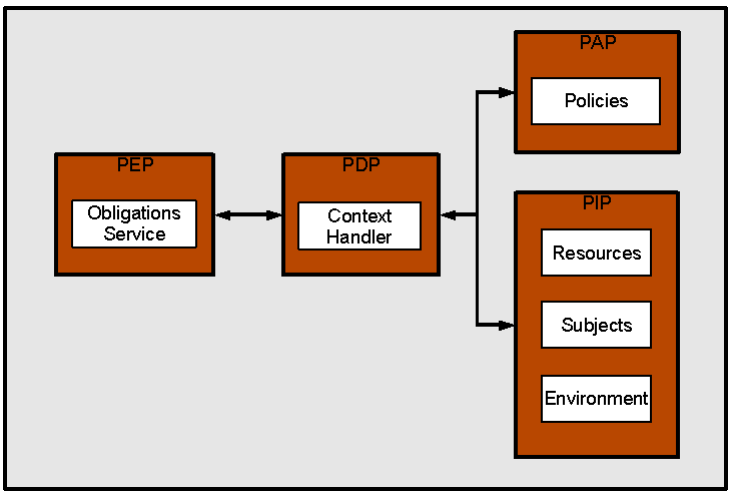

Figure 2.2.: Major XACML model components

Policy Administration Point (PAP) The system entity that creates and manages policies.

Policy Information Point (PIP) The system entity that acts as a source of attribute values. These attributes can be related to the resources, to the subjects, or even to the environment (i.e. the set of attributes that are relevant to an authorization decision and are independent of a particular subject, resource or action.).

Basically, the data-flow model can be described as follows:

- An application needs an authorization decision and will therefore ask the PEP.
- The PEP expresses this request as an XACML request that it sends to the PDP.
- The PDP retrieves the necessary policies and context attributes from the PAP and the PIP.
- Based on these, the PDP evaluates the policy and sends its decision (together with some optional obligations to fulfil) it to the PEP.
- The PEP fulfils the obligations (if any) and returns the authorization decision to the application.

2.2.2. XACML context

XACML is intended to fit diverse application environments. The core language is insulated from the application environment by the XACML context, as shown in figure 2.3, in which the scope of the XACML specification is indicated by the shaded area. The XACML context is defined in XML schema, describing a canonical representation for the inputs and outputs of the PDP. Attributes referenced by an instance of XACML policy may be

in the form of XPath expressions over the context, or attribute designators that identify the attribute by subject, resource, action or environment and its identifier, data-type and (optionally) its issuer.

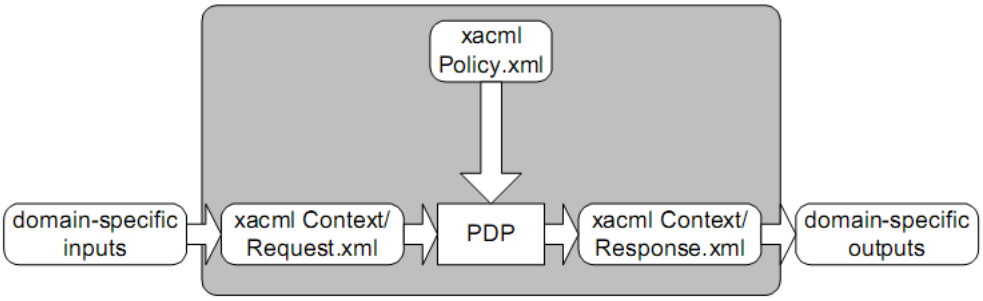

Figure 2.3.: XACML context

2.2.3. Policy language

The model of the XACML policy language is shown in figure 2.4. Many of its components are similar to the notions defined in the UCON model:

- The *Subject* and *Resource* components correspond respectively to the subjects and the objects of the UCON model.
- The *Action* component defines how a subject may use an object (i.e. usage); it is therefore somewhat similar to the rights of the UCON model which are basically sets of usages.
- The *Condition* and *Obligation* components correspond to their homonyms in the UCON model. However, the *Environment* component containing attributes such as current time or location used to define the conditions and obligations is not considered as a separate component in the UCON model.
- The *Rule* and *Policy* components correspond basically to the authorization rules of the UCON model, a policy being basically a set of rules, as explained below.

2.2.3.1. Rule

A rule is the most elementary unit of a policy and can be evaluated on the basis of its content. The main components of a rule are:

Target The target defines the set of resources, subjects, actions and environment to which the rule is intended to apply. The condition element of the rule may further refine the applicability established by the target. The PDP verifies that the matches

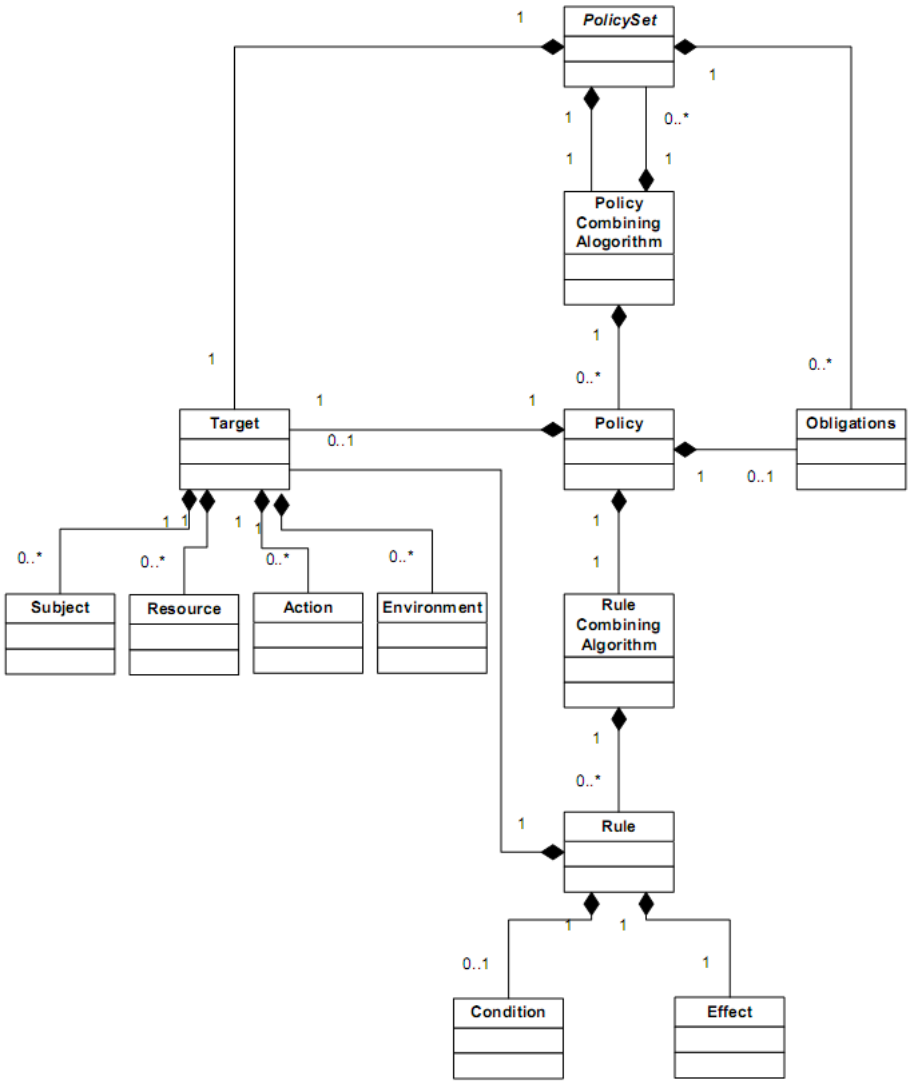

Figure 2.4.: XACML policy language model

defined by the target are satisfied by subject, resource, action and environment attributes of the request context.

Effect The effect of the rule indicates the rule-writer's intended consequence of a *true* evaluation for the rule. Two values are allowed: *permit* and *deny*.

Condition The condition represents a boolean expression that refines the applicability of the rule beyond the predicates implied by its target.

2.2.3.2. Policy

The PAP combines rules in a policy. A policy comprises four main components:

Target Similarly as in the rule, the target defines the set of resources, subjects, actions and environment to which the policy is intended to apply. It acts as a global target for all rules contained in that policy. This means that a rule that has the same target element as its policy may omit to declare it and simply inherits it from its policy.

Rule-combining algorithm The rule-combining algorithm specifies, when evaluating the policy, the procedure by which individual evaluations of the component rules are combined. There are many possible combining algorithms; the main ones are *deny-overrides* and *permit-overrides* indicating that the policy is evaluated to *deny* resp. *permit* as soon as one rule is evaluated to *deny* resp. *permit*.

Rules A set of rules as defined above.

Obligations Obligations may be added by the writer of the policy. When a PDP evaluates a policy containing obligations, it returns certain of those obligations to the PEP in the response context; the PEP is then responsible for fulfilling these.

Policies may be grouped in a Policy Set in a similar way rules are grouped in a policy.

2.3. Role Based Access Control (RBAC)

One has seen in 2.1.1 that a subject in the UCON model can be a user, a group, a role, or a process. In this work, one uses the idea of role as defined in RBAC, which is one of the most frequently cited access control models.
The basic concept of the RBAC model is that users are assigned to roles, permissions are assigned to roles and users acquire permissions by being members of roles. The user-role assignment can be a many-to-many relation in the sense that a user can be assigned to many roles and a role can have many users. Similarly, the permission-role assignment is also a many-to-many relation. The RBAC model is organized in four levels each including the requirements of the basic RBAC: the flat (or core) RBAC, the hierarchical RBAC that adds requirements for supporting role hierarchies and the constrained

RBAC that adds constraints on the hierarchical RBAC. The constraints may be associated with the user-role assignment (for static separation of duty) or with the activation of roles within user sessions (for dynamic separation of duty). The last level is the symmetric RBAC (also called consolidated) that adds a requirement for permission-role review. This is essential in any authorization management to identify and review the permissions assignment, i.e. the relation between permissions and roles.
The main benefit of this model is the ease of administration of security policies and its scalability. When a user moves inside an organization and has another function, the only thing the administrator needs to do is to revoke the existing user-role assignment and assign her a new role. There is no need to revoke the authorizations she had before and she will be granted new authorizations assigned to her new role. Adding to that, the role hierarchy defined in this model, where a given role can include all the permissions of another role, is a way of having a well structured access control that is the mirror of the organization structure. Finally the RBAC model supports the delegation of access permissions between roles. A role can delegate its role or part of its role to another role.

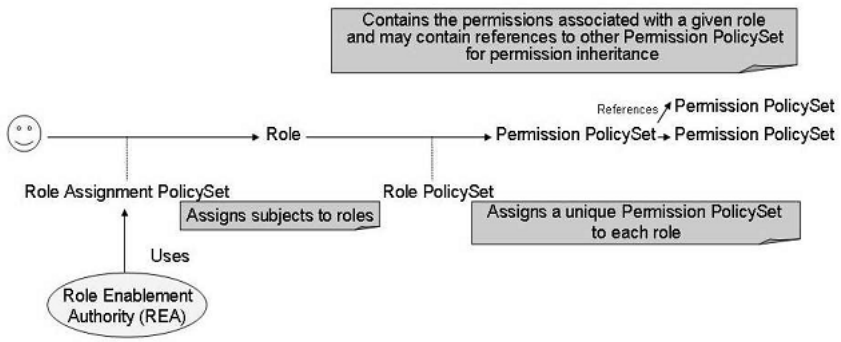

Figure 2.5.: RBAC profile principles

Chapter 3.

Use Case in Secure Messaging

3.1. Secure Data Room

In order to expand their operations often reflecting their willingness to increase their long term profitability, companies may resort to mergers. Usually these mergers occur in a consensual (occurring by mutual consent) setting where executives from the target company help those from the purchaser company (or auditors) in a due diligence process to ensure that the deal is beneficial to both parties. During this due diligence process, documents have to be shared by the target company with the purchaser company (or auditors). In order to do so, one generally uses a so called *data room*, which is a room where the documents are at the disposal of the purchaser company. The access to this room is controlled by traditional physical security equipments such as cameras or personal access cards, and the documents have to be consulted on site, without any possibility to do copies of any kind. Such data rooms are especially created during a due diligence process, and are closed at the end of it.

The development of information technologies has enabled to transpose this idea of data room into an electronic version, available via a network. Such a version has been called by many names such as *Electronic data room*, *Internet data room*, *Data site*, or *Secure data room*. For the remaining of this document, the term *secure data room* will be used.
Such secure data rooms have the same core functionalities as the traditional physical data rooms and offer many advantages. They transform due diligence into a streamlined, efficient process which reduces cost, eliminates aggravation and accelerates transaction cycles. They automate due diligence by facilitating easier information flow and collaboration and give corporate executives control of the process. By employing a centralized, shared repository of all due diligence data, vital information is readily available and easily accessible on-line by any and all transaction parties.

There already exist commercial secure data room applications. Two of the most known products are the *IntraLinks Virtual Data Room* and the *Merrill Datasite*. Both of them fulfil the requirements listed in 3.1.2, the former using the Adobe PDF format and the latter

using a specific format and a Java-based reader software to consult documents. The solution proposed in chapter 4 offers the same core functionalities using a web-based on-line reader. The main advantage of this solution is the use of usage control techniques such as the definition of conditions and obligations that have to be fulfilled before accessing or using a document, as discussed in section 3.2.

3.1.1. Use case

The use case corresponding to such a secure data room is shown in figure 3.1.

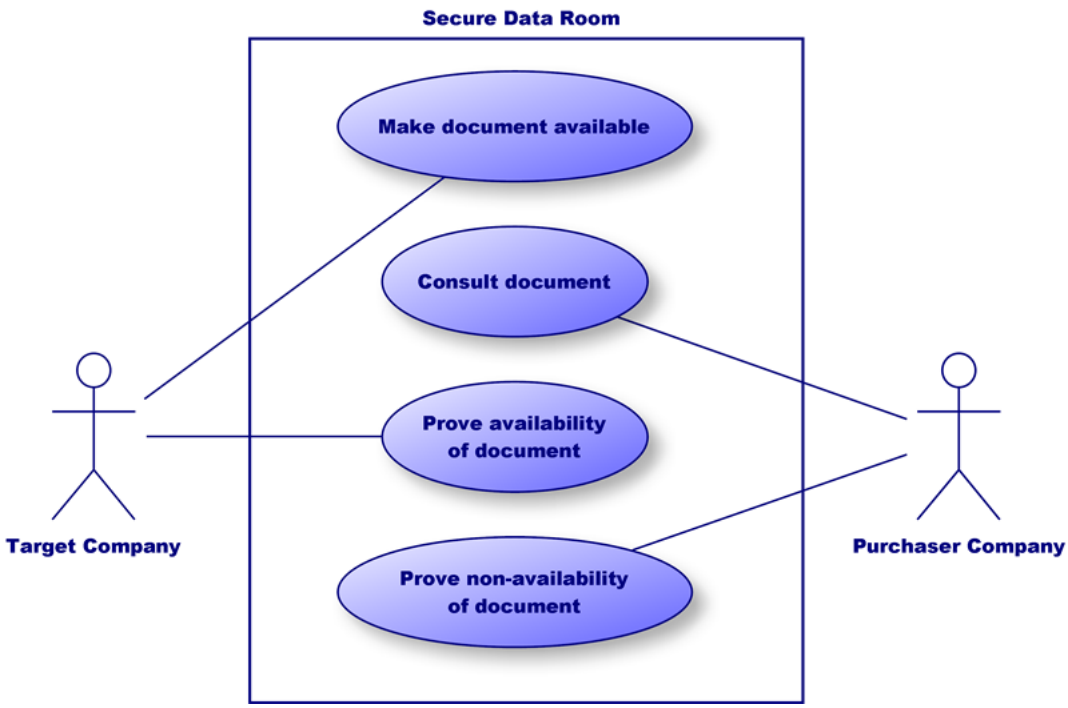

Figure 3.1.: Secure Data Room - Use Case

The main activities are:

Make a document available The target company makes documents available by posting them on the secure data room.

Consult a document The purchaser company (or auditors) consults available documents posted by the target company on the secure data room.

Prove availability of document The target company proves that a document has been made available to the purchaser company (or its auditors) during the due diligence process.

Prove non-availability of document The purchaser company proves that a document has not been made available by the target company during the due diligence process.

The first two happen during the due diligence process, i.e. during the exploitation of the secure data room. The last two happen after the exploitation of the secure data room when for instance the merger is not successful and a trial has to find out who is responsible for any money lost.

3.1.2. Requirements

Based on the definition of a secure data room application, one can define the basic requirements of a secure data room as:

- One should be able to make documents available to other users (target company).
- One should be able to consult available documents (purchaser company).
- The control over what users are allowed to do with the documents should be at least equivalent to what is guaranteed by a traditional physical data room.

If one only looks at the first two assumptions, one has requirements for a basic Content Management System (CMS). A secure data room is indeed a kind of CMS, but its particularity (and this is what is interesting in the context of this thesis) is the third requirement. In order to fulfil this requirement, one actually has to define a set of security requirements:

Access control An efficient access control mechanism should be in place at a document level.

Usage control The solution should prevent unauthorized printing, copying, forwarding, or print screening.

Reporting Each posting of a document or its access should be logged: who, what, when. This information should be available during and after the due diligence process, with a sufficient certification to serve as a proof in court.

Storage encryption Documents should be encrypted on the storage device.

Secure transactions All transactions with the application should be secured.

Network security The solution should incorporate firewall mechanisms, virus scanning and intrusion detection software.

Availability The documents should be available anytime. To enable this, the document container should have for example replicate servers, multiple connections and alternate power sources.

The last four ones are more about server configuration and architecture, and are therefore not of primary interest in the context of this thesis. The focus will be on the first ones in the context of usage control in section 3.2.

Of course, one can have some other usability requirements such as:

Simplicity of use The solution should be intuitive and straightforward to use. Users, who are not specialists in computer science, should be able to work with it without a complex training.

Format acceptance The solution should accept a wide range of document formats (e.g. PDF, Office documents, scanned documents and so on).

Scanning Scanning of paper documents should be possible using an integrated tool.

Search The solution should provide efficient search tools.

As these are not specific to secure data rooms (any CMS may have such requirements) and have nothing to do with usage control, they won't be further discussed in this document.

3.2. Usage control aspects

As one has seen in the previous section, many requirements of a secure data room application are related to the area of usage control. In this section, one describes each issue and tries to come up with possible solutions.

3.2.1. Access control

Basically, what one wants to achieve is shown in figure 3.2, where associations with an *X* are associations that should not be allowed.

Only registered users may access the application, and one should have a document-grained access control (i.e. some users can access a specific document, some others not). This can be done by writing policies defining exactly who can access what and how:

who The *subjects* as defined in UCON. In the context of a secure data room, this could be a single user or a group of users (by listing all members of the group or by using RBAC roles).

what The *objects* as defined in UCON. In the context of a secure data room, this could be the application (or sections of it) or posted documents (or pages of it).

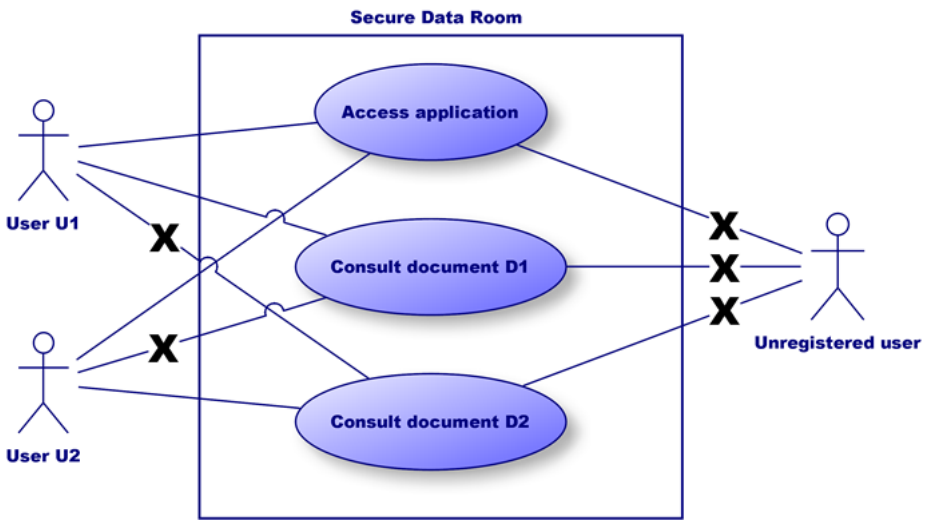

Figure 3.2.: Access control - Use case

how The *conditions* and *obligations* in UCON. In the context of a secure data room, one could imagine the following conditions:

- Access is granted only to users using a device located in the purchaser company's building.
- Access is not granted to users using a portable device such as a smart phone.
- Access is granted only if the user has been authenticated by a 2-factor authentication process.
- Access is granted only during working hours.

And the following obligations:

- Access is granted only if the user accepts a given license agreement.
- Any access has to be logged by the application.
- Access is granted only if the user notifies the target company right after having read the document.
- Access is granted only if the user deletes any copy of the document at the end of the due diligence process.

As one can see, usage control enables to extend traditional access control with conditions and obligations that can be necessary in the context of a secure data room.
Of course, these policies have to be enforced. Enforcement of such policies is discussed in chapter 4.

3.2.2. Usage control

One concentrates here on the aspects of usage control that are not about access control, i.e. how one can control what a user does with a resource once the access to it has been granted. In the context of a secure data room, the resource is a posted document (the application as a resource is only interesting for access control). What a user is allowed or not allowed to do with a resource (*rights* in the UCON model) may be defined in policies as well; in an XACML policy for instance, this is defined in the *action* part of a target rule.

One sees from the requirements that in the context of a secure data room, one has to control the following usages:

- Print the document
- Make an electronic copy of the document
- Forward the document to an unauthorized user
- Make a print screen of the displayed document

It is not an easy task to enforce the policies defining which usage is allowed and which one is not. One could for instance disable the *print* and *copy* functionalities of the viewer application used to read the document. This is done either by code running on the user device that disables these functionalities, or by options offered by the viewer application[1]. The problem is that there exists many tools and code manipulations cracking such disabling protections. One has therefore to be sure that the application used on the user device does not bypass these protections. This can only be done by using techniques such as *Trusted Computing*. Due to the complexity of these techniques, these are neither used nor discussed in this thesis. The interested reader can further read specialized articles on the subject, such as [19] or [20].
As it is impossible to guarantee 100% control of such usages (even if trusted computing is used, one cannot prevent the user to take a picture of his screen), one has to make life harder to mis-acting users. This can be done for instance by using watermarks on the documents or by allowing access to documents only page per page with a time latency between them. Such "hard-life-making" techniques are exposed in chapter 4.

3.2.3. Legal issues

Basically, the idea here is to be able to prove what was disclosed by the target company to the purchaser company during the due diligence process. One has to provide proofs of what documents were available to the auditors (mandated by the purchasing company); this represents a legal protection for the target firm. One also has to prevent users of the

[1] The PDF format allows for instance to define if the document may be printed or copied, and PDF viewers are supposed to respect these protections.

target company to change the content of a document after an auditor has inspected it, at least without the auditor's knowledge; this represents a legal protection for the purchaser firm. For that, one needs to know what documents have been published and to whom they where available, what was the content of these documents, who actually consulted these documents, and so on. One can define obligations (in the sense of the UCON model) in policies. For example, the access authorization has to be logged together with a hash of the accessed resource, or the user of the purchaser company who consults a document has to inform the target company of this act. Some possible techniques that could be used are:

- Logs of every action on the secure data room.
- Logs are signed by a third party.
- Automatic messages sent to the auditors when documents they have access to are modified.
- Documents cannot be deleted, just made unavailable.
- Hash of every mentioned document also inserted in logs.
- Snap-shots of the container at periodic times (also signed by a third party).
- No posted document can be removed, modified or deleted; one may make an old version unavailable and post a new version, but every version stays unmodified in the secure data room.
- The secure data room should be designed so that one may do an archive of it (e.g. on DVD) allowing to know exactly what was made available to whom and when (often, conflicts and trials happen a long time after the due diligence process).

The implementation of some of the above mentioned techniques is presented in chapter 4.

3.2.4. Attractiveness of the use case

This use case is particularly attractive, as one has seen in this section, because a secure data room offers plenty of examples where usage control can be used. Moreover, it has the following particularities that other typical usage control use cases do not have:

Dynamic policy management The rights defined for a document are susceptible to change, for instance after a negotiation[2] between the target company and the purchaser company.

[2]The principle of negotiation is explained in section 4.1.5.

Multilateral control Both parties (target company and purchaser company) have to be controlled. There is no assumption that one side is trusted to act fairly. In traditional use cases, the producer/owner side is trusted (for instance in the case of DRM[3] used with downloaded audio files).

[3]For more information on DRM, please see [1, 24, 13]

Chapter 4.

Proof of Concept

4.1. Design

In this section, one proposes a design for a secure data room application as defined earlier. After analysis of the requirements (i.e. access over Internet, user-friendly interface, and so on), the choice of a web application seems to be appropriate. One will therefore present here a design for such a web-based secure data room application.

4.1.1. Web interface

There are basically 3 kinds of users of the application:

Administrator users The users responsible for the administration of the secure data room. They basically manage the users and their roles (see 4.1.3).

Publisher users The users who post documents on the secure data room, basically the employees of the target company. They are the provider subjects (PS) of the UCON model.

Reader users The users who consult documents from the secure data room, basically the employees of the purchaser company or the auditors. They are the consumer subjects (CS) of the UCON model.

Therefore one takes into account 3 sections of the web interface, one dedicated to each type of users, as summarized in figure 4.1. The access to each page of the web application is of course controlled. One describes a technique to do that in section 4.2. Basically, it uses policies to define who can access what sections of the application, and every access to a page is controlled by evaluating these policies. How these policies are created is explained in section 4.1.4.

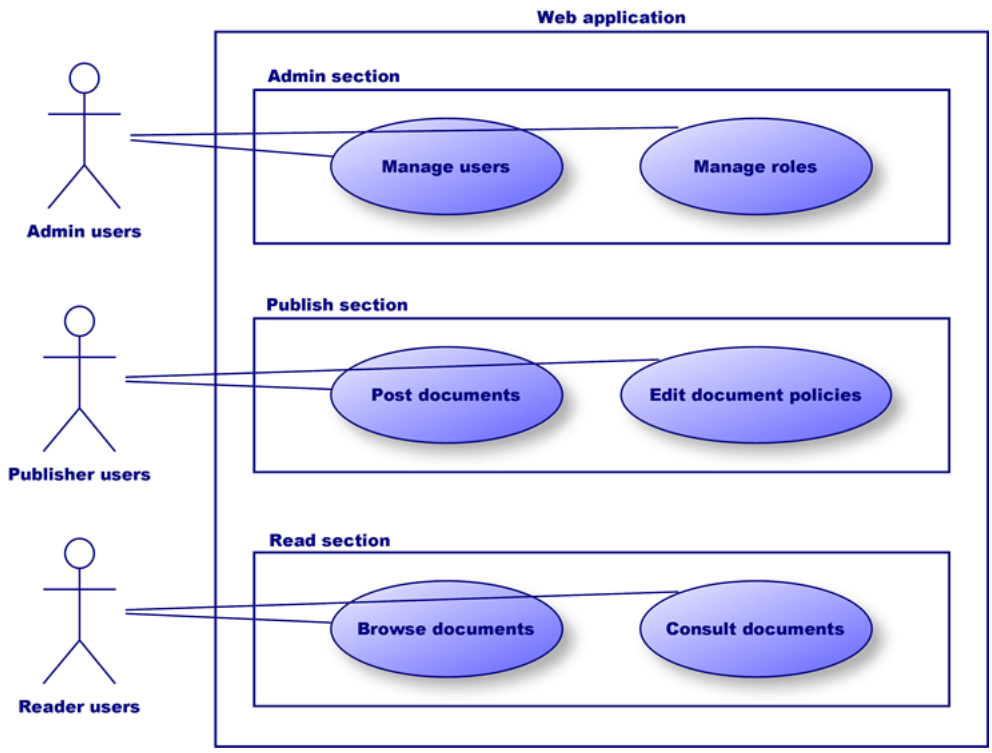

Figure 4.1.: Sections of the web application

4.1.2. Architecture

One has defined in section 4.1.1 how the web interface is organized. This interface allows the users to use the secure data room application; but this is only an interface. One describes here the architecture of the whole application.
The model shown in figure 4.2 defines the following components:

Web interface The presentation layer of the model, as described in section 4.1.1.

Identity manager The component responsible for managing users and roles by adding/deleting/modifying them in the database. It is therefore also responsible for retrieving information on users and roles for other components.

Resource manager The component responsible for managing resources (i.e. posted documents) by adding/deleting/modifying them in the database. It is therefore also responsible for retrieving information on the posted documents for other components.

Log manager The component responsible for managing log services.

Usage control engine The component responsible for controlling usage. Its architecture is greatly inspired by the XACML model: the PEP receives usage requests from other components of the application, asks the PDP for a decision and then enforces

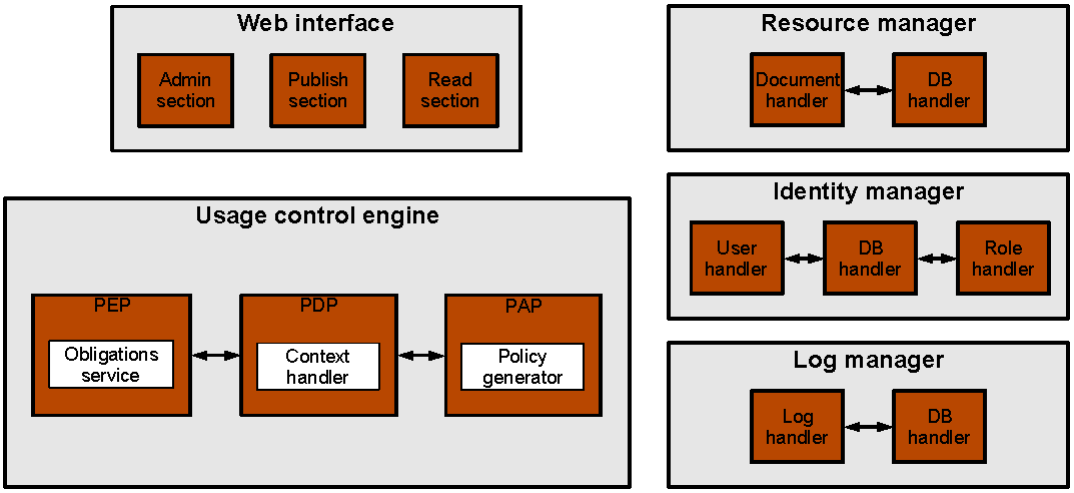

Figure 4.2.: Architecture of the application

it; its obligation service is responsible for applying obligations returned by the PDP. When receiving a request from the PEP, the PDP retrieves the policies from the PAP; its context handler retrieves the necessary attributes by asking the resource and identity managers. The PDP can then evaluate the policies and send a response to the PEP. The PAP is also responsible for generating new policies, as discussed in section 4.1.4. This information flow is depicted in figure 4.3.

4.1.3. Identity management

For the definitions of permissions, one uses the hierarchical RBAC model. The administrators of the secure data room define the roles and attribute them to the users. The permissions on posted documents are defined by the publishers using these roles. The advantages of using the RBAC model have been listed in 2.3: when a user quits his company, the administrators of the secure data room only have to remove the user from the system; the policies defining the permissions on documents do not have to be modified. The reason why one uses the hierarchical RBAC is because it simplifies the management of permissions. For instance, during the exploitation of the secure data room, one may need to create a new role. If one wants this new role to have more permissions than another (i.e. $newRolePermissions \supset oldRolePermissions$), one defines the new role as being a parent of the old role; it will then automatically have the same permissions. Any permission which is added to the old role will also be attributed to the new role, and none of the permissions added to the new role will be attributed to the old one.

The authentication of the users may be done in two ways:

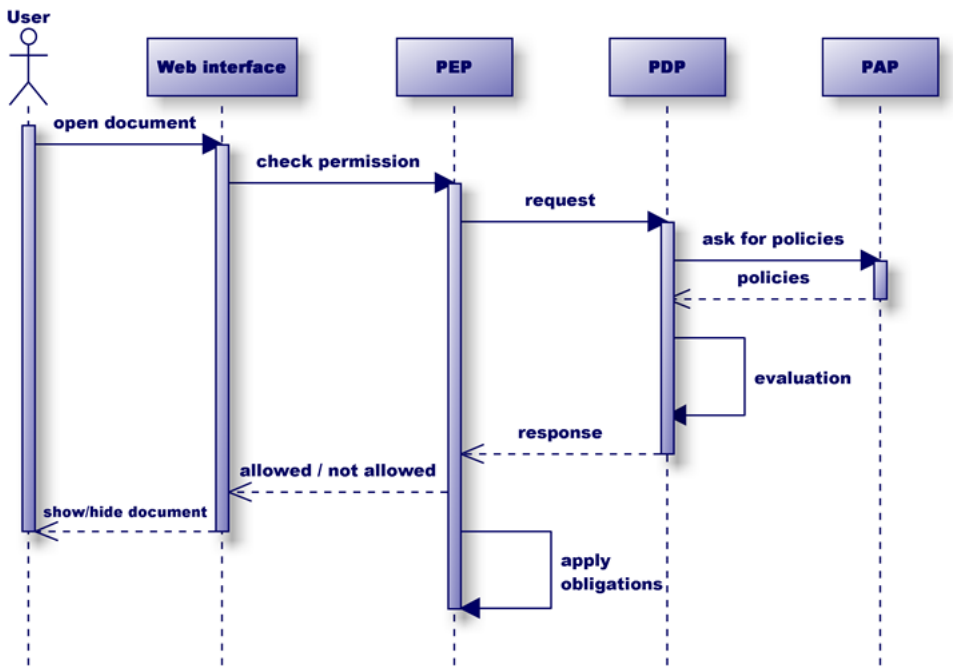

Figure 4.3.: Information flow for access request

- using traditional login (i.e. username and password)
- using certificates

The traditional login method is used when the administrator creates a new user: the new user becomes a randomly generated password and can log in with it. Once the user has logged, he can upload his personal certificate[1]. Once the certificate uploaded, the authentication procedure can be done using that certificate.
As the traditional login method is considered to be less secure than the one using certificates (passwords are vulnerable to dictionary attacks for example), one privileges the certificate-based authentication. This means that a user authenticated only with its username and password may have restricted rights on the application; for instance, publishers of documents may restrict access to documents depending on the authentication method used by the reader: a reader may consult the whole document if authenticated with his certificate, but only the first page if authenticated with traditional username/password. The restriction of the rights defined in the policies is discussed in 4.1.4.
The authentication method used here is what one calls *certificate-based mutual authentication*. It allows both the user and the server to be authenticated, providing of course a better security. The only inconvenient with that method is that the user has to have a personal certificate. If this certificate is not certified by a trusted authority, the user has to upload it before use. The first login after the creation of the user by the administrator has

[1] If this certificate is certified by an official certification authority, the application could automatically accept this certificate without the user having to upload it first.

then to be done using username/password[2]. Here one can imagine to use one-time (or *three-times*) passwords to constrain the user to upload a certificate rapidly. Of course, one may also use some 2-factor authentication methods using biometric data or smart cards. This could be easily added to the application presented in this thesis.

4.1.4. Policies

The policies used in this secure data room application are expressed using XACML. This choice has been motivated because of its rich expressiveness and its ability to be easily extended if needed. It has also been rigorously formalized using description logic [10].

The policies defining the access rights to the web interface are defined by the developer of the application. There are, therefore, statically defined before the secure data room is used. They define what roles are allowed to access each page of the web interface. There is no usage control here; the only usage that one has on web pages is to read them, i.e. load them in a web browser. One does not need any obligation in this context. The permissions are defined at a section level using regular expressions in the resource attribute of the XACML policies. For instance, if every page of the admin section is located at a path such as */Admin/*, one may use the regular expression */Admin/** to define the resource linked to the current rule. Once again, one sees the rich expressiveness of XACML.

The policies defining the usage rights on posted documents are dynamically created by the users posting the documents. Dynamically, in this context, means that they are created during the exploitation of the secure data room. As any authorized user may post a document with specific rights, one has to have document-level policies; one defines therefore one policy file for each document. In XACML, the resource target of the policy would then be it's corresponding document.
The user who posts a document has to define the content of its policy, i.e. who can do what and how with it:

Who The *subject* of a XACML rule. As explained above, here one would use roles.

What The *action* of a XACML rule. One may have the following usages in the context of a secure data room:

- **See** See that the document exists in the secure data room; this means for instance that the document is visible (i.e. listed) in a file explorer.
- **Read** Consult the document, i.e. access the content of it.

[2] If the administrator could also generate the certificate for the new created user, he would know the private key of the user, which is conceptually wrong. One therefore has to use a traditional username/password login first.

Edit Modify the document by updating its content or its policy, or even remove the document from the secure data room.[3]

No other usage is allowed: the requirements specify that no copy or no printing of any document should be allowed; one as therefore to ensure that users reading documents will not be able to do anything except read them. The way this is ensured is discussed in 4.1.5.

How The *condition*s and *obligation*s of an XACML rule. One may have many conditions and obligations, as discussed in section 3.2.

In order for the user to define the document's policy, the application has to have an interface allowing the definition of the rules. This interface may propose to select individually the parts of the rules subject, action and conditions/obligations, or for the sake of simplicity and usability, to select for instance predefined templates[4].
After discussion with possible users of this application, one has chosen for this project to propose the two following conditions for the user to select from:

- The rule applies only if the user has been authenticated using a given authentication method. This could be for instance a traditional login, a login using certificates, or a 2-factor (using an external smart-card for instance) login process.
- The rule applies only for a given period of time. This can be useful to automatically make unavailable a resource that one considers to be useful only for a finite period of time. This condition can be understood as an auto-deletion mechanism.

Of course, one can add many other conditions or obligations to the application, depending on the requirements of the client. One has therefore to design the application in such a way that it is easy to add them, by using interfaces or factories for instance.

4.1.4.1. Web pages policy example

The policy shown in listing 4.1 allows the access to pages of the Admin section of the web interface only to users having the role WEB_Admin. The target of the policy states that the resource has to match the regular expression */Admin/**; this is the case for all pages stored in the Admin directory, i.e. all pages of this section. This policy has a rule that allows the access for subjects having the role *urn:sdr:role:WEB_Admin*. There is a second rule denying access for any subject; as one uses a permit-override combining algorithm, the *deny* decision is only returned if the first rule does not apply. One has therefore the exactly desired behavior.

[3] The exact meaning of editing a document in this context is discussed in section 4.1.6.2.
[4] For instance, a template named *public* may be defined as allowing any role to see and read a document, and allowing the owner (i.e. the publisher) to edit it.

Listing 4.1: A sample XACML policy for the Web interface

```
<Policy
 xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
 PolicyId="urn:sdr:webPolicy:resource:/Admin/"
 RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">
 <Target>
  <Resources>
   <Resource>
    <ResourceMatch
     MatchId="urn:oasis:names:tc:xacml:2.0:function:anyURI-regexp-match">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      urn:sdr:webResource:/Admin/.*
     </AttributeValue>
     <ResourceAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
      DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
    </ResourceMatch>
   </Resource>
  </Resources>
 </Target>
 <Rule
  RuleId="ruleId:sectionAccessPermission"
  Effect="Permit">
  <Description>Admin users can access the page</Description>
  <Target>
   <Subjects>
    <Subject>
     <SubjectMatch
      MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">
       urn:sdr:role:WEB_Admin
      </AttributeValue>
      <SubjectAttributeDesignator
       AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
       DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
     </SubjectMatch>
    </Subject>
   </Subjects>
  </Target>
 </Rule>
 <Rule
  RuleId="ruleId:defaultRule"
  Effect="Deny">
  <Description>Any other user cannot access the page</Description>
 </Rule>
</Policy>
```

Listing 4.1: A sample XACML policy for the Web interface

4.1.4.2. Document policy example

The policy shown in listing 4.2 defines permissions on the posted document */Finance/Report.pdf*. The target of the policy says therefore that the id of the resource has to match *urn:sdr:resource:/sdr:docRepository/Finance/Report.pdf*, which is the internal path of the document */Finance/Report.pdf*. All the rules defined in this policy then concern this document. The first rule allows subjects having the role *urn:sdr:role:Publisher* to edit (action matches *urn:sdr:action:edit*) the resource if they have been authenticated with a

certificate. The second rule allows subjects having the role *urn:sdr:role:Publisher* or the role *urn:sdr:role:Reader* to see and read the resource without any conditions. If the PDP does not find any matching rule, it returns the decision *not applicable*; the implementation has then to decide what to do with such a decision. In the case of the prototype discussed in 4.2, this decision is considered equivalent to a *deny* decision. This means that access will be denied to any other subjects wanting to use the resource in any other way.

Listing 4.2: A sample XACML policy of a posted document

```
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
  PolicyId="urn:sdr:documentPolicy:urn:sdr:resource:/sdr:docRepository/Finance/Rapport.pdf"
  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <Description>Policy for document /Finance/Report.pdf</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">
      urn:sdr:resource:/sdr:docRepository/Finance/Report.pdf
    </AttributeValue>
          <ResourceAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
   DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
  <Rule RuleId="ruleId:293114832" Effect="Permit">
    <Target>
      <Subjects>
        <Subject>
          <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">
                urn:sdr:role:Publisher</AttributeValue>
            <SubjectAttributeDesignator
     SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
     AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
     DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
          </SubjectMatch>
        </Subject>
      </Subjects>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">
     urn:sdr:action:edit
   </AttributeValue>
            <ActionAttributeDesignator
     AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
     DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
    <Condition>
   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
     <EnvironmentAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:subject:authentication-method"
    DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">
    urn:sdr:environment:auth_method:certificate
     </AttributeValue>
```

```
 </Apply>
    </Condition>
  </Rule>
  <Rule RuleId="ruleId:473956394" Effect="Permit">
    <Target>
      <Subjects>
        <Subject>
          <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">
                urn:sdr:role:Publisher</AttributeValue>
            <SubjectAttributeDesignator
     SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
     AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
     DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
          </SubjectMatch>
        </Subject>
  <Subject>
          <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">urn:sdr:role:Reader
                </AttributeValue>
            <SubjectAttributeDesignator
     SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
     AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
     DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
          </SubjectMatch>
        </Subject>
      </Subjects>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">
     urn:sdr:action:see
   </AttributeValue>
            <ActionAttributeDesignator
     AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
     DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
          </ActionMatch>
        </Action>
  <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">
     urn:sdr:action:read
   </AttributeValue>
            <ActionAttributeDesignator
     AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
     DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
  </Rule>
</Policy>
```

Listing 4.2: A sample XACML policy of a posted document

4.1.5. Usage control

As seen above, publishers of documents define their usage rights in policy files. The application has then to enforce these; this is discussed in that section.

One has seen above that users may use documents in 3 ways: see, read, or edit[5] them. In order to control a user that has a permission to *see* a document, one simply uses access control mechanisms when listing the available documents in a file explorer or during a search. The same applies for a user having a permission to *edit* a document; one simply controls the access to the page where editing features are available. The case where the user has a permission to *read* the document is the most interesting: one has first to control access to the document, of course, but how does one make the document available to the user for reading without allowing him to do anything else, such as making a copy of it or print it? There are basically two ways:

Prevent any unauthorized usage by using a client-side application The best way to control usage is to have a client-side application that prevents copying and printing. This would allow many more obligations such as, for instance, erasing a downloaded document at the end of the due diligence process. But as discussed in 3.2, this means that one has to use technologies such as Trusted Computing to be sure that the client application has not been corrupted. This would imply that users of the secure data room should possess special hardware components, and this is not a solution desired by the people having defined the requirements.

Make any unauthorized usage difficult The idea here is to use traditional technologies on the client side, i.e. an Internet browser, and to make life harder for the people wanting to overpass their rights. In order to do so, one uses the following techniques:

- One cannot download an entire document, but only access it's content online.
- The online viewer displays the document page per page.
- A page is displayed as an image, so that it is difficult to use copy/paste tools on the text.
- Every image contains a watermark on it, so that any simple copy or print of the image won't be considered as a legitimate document. The *no-printing* obligation is therefore observable.

One could also add some time delay when the pages are changing or allowing the person to read only a few pages during a session, but such techniques are really deprecating the usability of the application. One has tried here to find a good balance between security and usability; this balance may of course be modified depending on the price users are ready to pay for the security.

As defined in [9], the principle of negotiation means that the consumer of a resource may negotiate the exact content of a policy with the producer. In our case, this could happen for instance if the publisher allows a user to read the first pages of a document, but not more. The reader may ask the publisher to extend his rights in order to read other pages of the document. In this application, the reader may send a mail to the publisher to do this demand.

[5]The exact meaning of editing a document in this context is discussed in section 4.1.6.2.

4.1.6. Legal issues

As discussed in 3.2.3, one needs techniques in order to fulfil the requirements of a secure data room. For the proposed application, the chosen techniques are discussed in the next sections.

4.1.6.1. Logging of events

Basically, every event is logged into log files. The format chosen for these log files is the Extensible Markup Language (XML) format, which allows easier automated researches of any information. There are two types of log messages:

- Messages concerning the application itself (i.e. from a developer point of view) which are useful for resolving problems, bugs or errors.
- Messages concerning the legal issues of a secure data room, such as what was made available to whom.

The former ones are not specific to a secure data room application, therefore in the rest of this section one discusses only the latter ones. The distinction between them may be done for instance using specific levels (in the sense of Java.Logging.Level[6]): the traditional messages may have traditional levels, and the legal ones may have a specific level called *LEGAL*. This allows a better understanding of the log files, and allows to log these messages in separate log files.
The following events are logged as legal ones:

- A document is posted, modified or deleted
- A policy is created or modified
- A document is consulted
- A user is created, modified or deleted
- A role is created, modified or deleted
- A role is assigned to or withdrawn from a user
- Of course, the opening and closing of the secure data room is also logged.

Depending on the event, one may log information such as the ids of the user, the role, the document or the policy respectively, the date and time of the event, or a hash of the document or policy file.

[6] `http://java.sun.com/javase/6/docs/api/java/util/logging/Level.html`

4.1.6.2. Version control

If one allows users to modify posted documents or their rights, there is a risk that the target company posts a falsified document for instance, and after control of it by the auditors, replace it with the real document so that after the due diligence process, one thinks that the real document was presented to the auditors. Moreover the risk could be for instance that a posted document is not available to any user during the whole time but is made available just before the end of the due diligence process. In order to counteract such attacks, one could say that the edition of documents and their rights is not possible, but this would result in a bad usability. A better way to handle this is to use a version control technique[7]: every document and every policy file is kept on the secure data room; if one modifies a file, instead of replacing the old version, one simply keeps both versions. Of course, if changes are often made, this could use a lot of storage space; but modern version control systems (such as the one described in 4.2) have incremental mechanisms optimizing this, so this is not really an issue. By using version control, one can allow users to edit the documents and their rights without losing trace of what exactly was available to whom and when during the due diligence process.
The technique of version control is also used in the management of user and roles.

4.1.6.3. Archive and snapshots

The whole secure data room can be archived. By *the whole secure data room* one means:

- The posted documents (every version of them)
- The policy files (every version of them)
- Every stored information about users and roles (such as ids, role attribution and so on)
- The log files

The archive (in fact, a hash of it) may then be signed by a trusted third party, for instance the organization managing the application. This signed archive can then be burned on a disc and delivered to the client who can, in this way, keep a trace of what was made available during the transaction for several years.
This archive feature is also used to do some snapshots of the container during its use. One does backup archives at regular intervals (cron jobs running on the server), and signs a hash of it; this hash and its signature are then kept in the container as snapshot.

[7] For more information on version control, please read the introduction chapters of [4]

4.1.6.4. A possible threat

A possible threat against the legal measures exposed above would be for instance that the target company creates a fake user allowed to consult some sensitive documents. The company can then say that these documents were available to at least one user of the purchaser company. The purchaser company won't be able to prove that this is not true, because this user and his rights on the documents are effectively logged. One sees here that the entity responsible for managing users and roles of the application has to be trusted by both parties. One could also imagine to force users to log into the application using a personal certificate such as eID [6] proving that the user is real.

4.2. Implementation

As a proof of concept, one has developed a prototype of such a secure data room application. The source code, published under a BSD license and therefore free to use for personal and commercial purposes, is delivered as an annex to this document. The main technologies used in this implementation as well as some interesting topics of the implementation are presented in this section.

4.2.1. Framework

For the implementation of this secure data room Web application, one has chosen the JavaServer Faces (JSF) framework, which is a Java-based Web application framework that simplifies the development of user interfaces for Java EE applications. The goal here is not to present this framework; the interested reader may consult [12, 23] for an overview of this framework and would probably understand better the choice made here. One actually uses the Apache MyFaces Project[8] that provides an implementation of the JSF framework.
As explained above, one controls the access to the web pages of the application. This control is done every time a user requests a page. Without going into the details of JSF, when a user makes a request for a page, many phases are happening on the server side (*life cycle*) and these will result in the response sent back to the user. At the beginning of this life cycle, a request is made to the PEP in order to check whether the page can be accessed by the user making the request. If the PEP denies access to the requested page, the user is redirected to the entry page of the application (i.e. the login page) by returning the corresponding response to the user.

[8] `http://myfaces.apache.org/`

4.2.2. Version control

As discussed in 4.1.6.2, one needs a version control tool to store the documents and their policies. The storage of documents and policies, as well as users and roles, is done in the prototype application using Apache Jackrabbit[9]. Apache Jackrabbit is a fully conforming implementation of the Content Repository for Java Technology (JCR) API. A content repository is a hierarchical content store for structured or unstructured content, full text search, version control, transactions, observation, and more. Typical applications that use content repositories include content management, document management, and records management systems. For more information on JCR, please consult [2].
The Apache Jackrabbit tool allows to easily structure the data as a tree. The documents are then nodes of this tree containing a child node for the document file and another child node for the policy file; other information about the documents are stored as properties of the node. The users and roles are also nodes of the tree. The whole secure data room is then structured as a tree that can be easily backed up in order to do the archives and snapshots described in 4.1.6.3. The schema of this data model is shown in 4.4.

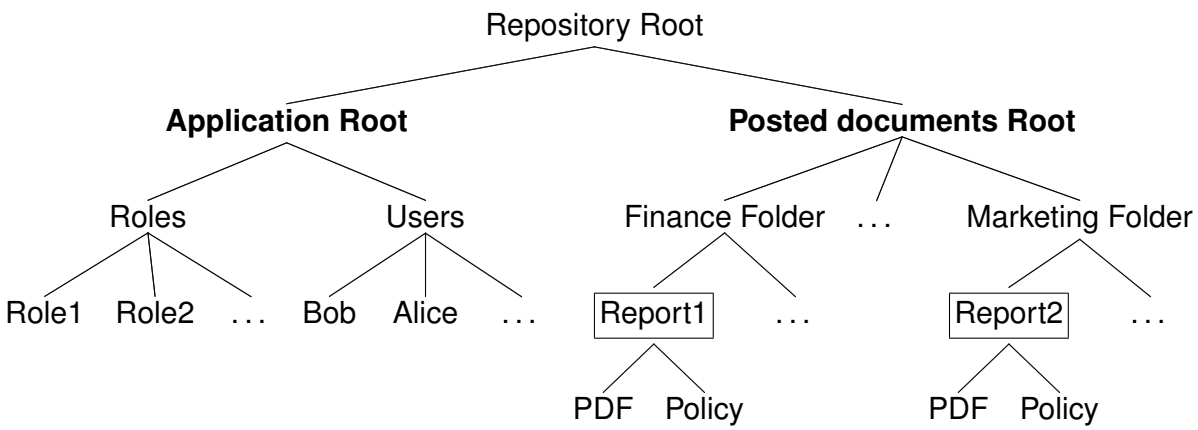

Figure 4.4.: Tree structure of the secure data room

4.2.3. Source code

One does not describe in this sub-section the source code, since this is not the purpose of this document; however, one can explain how it is basically structured based on the architecture presented in 4.1.2.
As in any Web application using the JSF framework, one has some XML files necessary to the configuration, and one has JavaServer Pages (JSP) pages responsible for the presentation layer. As this is not specific to a secure data room application, one will not further discuss them. What is interesting here is the model of this application which is

[9] `http://jackrabbit.apache.org/`

implemented using Java code. The division of this code in packages, as shown in figure 4.5, is the following:

ajotterand.uc.* The usage control engine. It uses the *SunXACML* library to handle policies.

- **ajotterand.uc.pap** Contains tools for creating, editing and managing policies.
- **ajotterand.uc.pdp** Contains tools to decide if a request from the PEP has to be accepted or refused. Therefore it contains tools to retrieve information about resources, users, or environment (so called Context Handler).
- **ajotterand.uc.pep** Generates the requests to be sent to the PDP and controls the application of the decisions.
- **ajotterand.uc.utils** Various implementation tools.

ajotterand.web.* The classes specific to the web application.

- **ajotterand.web.uc** The classes playing the role of a bridge with the usage control code.
- **ajotterand.web.beans** JavaBeans used in the implementation (as managed beans for example).
- **ajotterand.web.validators** Various validators as defined in JSF.
- **ajotterand.web.converters** Various converters as defined in JSF.
- **ajotterand.web.resource** Resource files containing messages and labels presented to the user (used for internationalization for example).
- **ajotterand.web.utils** Various implementation tools.

ajotterand.id The management of identities, i.e. users and roles.

ajotterand.doc The management of resources, i.e. posted documents.

ajotterand.db Contains tools for retrieving/adding/modifying data from/to the repository.

ajotterand.log.* Contains logging methods and tools.

- **ajotterand.log.threads** Thread classes containing possibly time costing logging operation such as hash or signatures (using the Java Cryptography Architecture (JCA)).

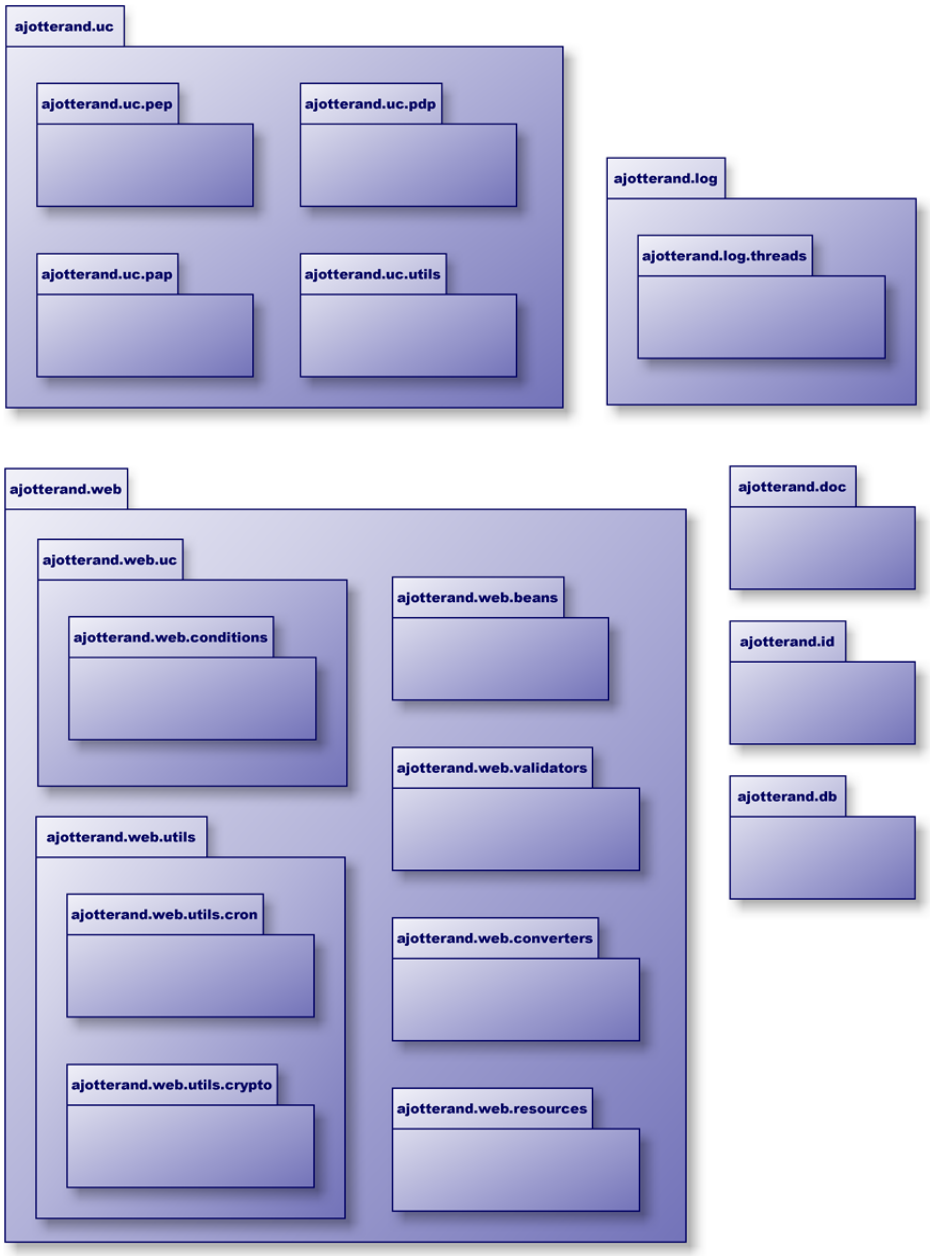

Figure 4.5.: Java packages of the application

Chapter 5.

Evaluation

5.1. Benefits of Usage Control

The fundamental requirement of a secure data room application is that it should, at least, offer the same security than a traditional data room. As seen in 3.2, this is not possible with traditional access control. In that case, usage control fills this gap with the notions of conditions and obligations. Theoretically (by using client-side enforcement and known techniques such as Trusted Computing for example), one can achieve the same level of security than with a traditional data room, with of course all the benefits offered by an electronic version of a secure data room (there is no need for auditors to go to a specific place, no need for costly surveillance equipment and so on).

5.2. Quality of design

One has proposed as a proof of concept a possible implementation of such a secure data room application. During the design phase of the development, choices have been made, as discussed in 4.1. Evaluating these choices is not an easy task. One bases one's judgment on the following criteria:

Are the requirements fulfilled? The requirements listed in 3.1.2 are partially fulfilled. Only partially because as explained in 4.1.5, one only makes life harder to the bad handling user that wants to copy or print documents.

Is the client satisfied with the proposed solution? [1] The client seems satisfied with the proposed solution; of course this is only a prototype, and there would be some work to do for a complete product. But the design used seems to be a good basis for such a product.

[1] The client may not be satisfied even if the requirements are fulfilled, due to misunderstanding of the requirements or bad listing of the requirements for example.

Are the usage control measures transparent enough to the end-user? The prototype application is relatively user-friendly. However, the definition of the policies for each document may be sometimes wearying for a user having to post many documents.

How good is the quality of the implementation? The implementation has successfully passed many series of tests and seems to be relatively exempt of bugs. It could easily be extended with new features and some components may be relatively easily reusable for other applications. Moreover, all the tools and technologies used are open standards and can therefore be easily evaluated or modified.

5.3. Limitations

One has remarked that in order to fully ensure the enforcement of the policies, one needs technologies such as Trusted Computing. These technologies are relatively new, complex and very costly (for instance, the necessary hardware components are very costly). This can be seen as a limitation of the idea of usage control, in particular, the cost. The principle of usage control is well defined in the literature, and more and more papers are exploring aspects of it, but this often stays at a theoretic level. The implementation of these theoretical usage control mechanisms may sometimes be very difficult to do.
The prototype proposed in this thesis tries to recommend the best possible solutions without using technologies that a client could not possibly afford. This does result in some lacks of control on the usage of the documents. For instance, one does not prevent a user to copy the images of every page of a document, to remove the watermark using appropriate tools, and then to use Optical Character Recognition (OCR) tools to recreate a document that may then be easily modified or reproduced. One should therefore do a risk evaluation in order to define how much the client is ready to invest for the security of the documents. In the context of this thesis, the investment had to be poor; the measures taken may therefore be considered as sufficient.

5.4. Experience

One has discussed the advantages of using XACML in this work. Another advantage resides in the implementation phase: there are open source implementations of the XACML 2.0 specification available. One of these is the Sun's XACML Implementation developed in the Sun Microsystems Laboratories. By using this implementation, access control based on XACML policies is relatively straightforward to implement. The major difficulty resides in the creation of the policies. If one only needs a few static policies, one can easily write them by hand; but if an application needs to create the policies dynamically (as in a secure data room application for the documents), the creation of

these policies by the user is a tough job. One has then to find user-friendly tools in order to help the user define the policies. In the proposed prototype, the user creates each rule of the policy by selecting the attributes of the subject, action and condition elements. This allows to exactly define what one wants, but it can be wearying if there are lots of rules to define. One may imagine a solution using templates or default rules, as discussed in 5.5, in order to make life easier to the end user.
Consequently the access control to the pages of the web application is relatively straightforward to implement by using the Sun's XACML implementation: all the permissions one needs are defined in a couple of policy files, and the decision engine proposed in this implementation is the same because both web pages and documents of the secure data room are considered as resource. Once again one remarks on the beauty of the XACML specification that is generic enough to be used in a variety of contexts. One could therefore recommend this technique for controlling access to web applications.

As explained in 4.1.5, one has made the decision to let the documents be accessed only on-line using a document viewer that creates watermarked images of the pages of the documents. From the point of view of the developer, this is easily done with the help of the graphic libraries of Java and of the Multivalent library handling many document formats such as Portable Document Format (PDF) or Rich Text Format (RTF). From the point of view of the end user, however, this results in some constraints. For instance, one cannot download the document before getting on a train and reading it during a travel without the access to an Internet connection; but this is also not possible with a traditional data room, and solutions with a client-side application often also need an Internet connection in order to decide if the usage has to be allowed, as with the DRM technologies. Moreover, many people do not like to read a document on a screen and therefore prefer paper versions; one sees here a limitation to the concept of an electronic data room that does not allow printing documents. Another disadvantage with a document being presented as an image is that one cannot do annotations or copy/paste some parts for personal notes; but this is actually part of the requirements of a secure data room application.
As one can remark, the lack of usability of the proposed application is basically due to the security requirements of a secure data room. As often, the usability is in concurrence with the security; one has then to decide how much one is ready to invest (in term of lack of usability) for the security of the documents. But from the point of view of the user, the solution proposed in this work is not less user-friendly than another client-side-based solution. And one also has to keep in mind that the usability of a traditional data room is not so good either: identity controls, restricted access hours, no right to take documents out of the room and so on. As one can remark, the lack of usability of the proposed application is basically due to the security requirements of a secure data room. As often, the usability is in competition with the security; one has then to decide how much one is ready to invest (in term of lack of usability) for the security of the documents. But from the point of view of the user, the solution proposed in this work is much more user-friendly than another client-side-based solution, even if the definition of rights by a publisher may be sometimes wearying. And one also has to keep in mind that the usability of a traditional

data room is not so good either: identity controls, restricted access hours, no right to take documents out of the room and so on.

5.5. Extensions

For usability concerns, one could add the following features to the prototyped secure data room application (the list is non-exhaustive):

- The policy definition could be eased by the use of templates. Instead of defining manually each rule of the policy, one could have templates with predefined rules; the user would only choose the desired template from a list, and the rules may automatically be generated.
- For a user having to post many documents with the same permissions, one could add a mechanism to have the same policy for a group of documents. This could be done by specifying that the resource part of the policy target should match any of the documents within the group. However, one has chosen to have one policy file per document; if one wants to keep this design choice, the application could also automatically generate one policy file for each document, these policy files having the same properties except the resource matching part.
- One of the defined usages is *see*; this means that the user can know that a document exists, but cannot consult it. In order for the user to know if he wants to negotiate the right to consult the document with the publisher, one could add a description (or any kind of metadata) to each document; this metadata would then be visible to users having only the right to see the document.

Chapter 6.

Conclusion and future work

One has presented in this work an interesting use case of usage control in the context of secure messaging. This use case is a secure data room application, an electronic version of a data room used for example during a due diligence process where documents are exchanged between two companies.
One has defined what these security requirements are for such an application with the help of specialists and potential end-users. After analysis of these requirements, one has explained how usage control can be a solution to the problem.
As a proof of concept, one has described a prototype of a web-based secure data room application, covering aspects such as access control to web pages, identity management, definition of permissions on documents and enforcement of these on the server side. One has also proposed solutions to legal issues such as identifying and proving what information was effectively disclosed during the due diligence process using cryptographic tools such as digital signatures.

One has observed that usage control policies may be easily applied for controlling access to a web interface and allowing to centralize the permissions definitions. One has also observed how usage control policies may be dynamically defined by a producer who is an end-user of an application. Moreover, this work has shown that the enforcement of usage control policies requires sometimes very complex and costly technologies such as trusted computing, and that one has therefore to find sometimes easier alternatives to implement but where the enforcement is not guaranteed.

This work contributes to the development of the concept of usage control by offering a discussion about the effective implementation of usage control mechanisms in a concrete use case, which is clearly needed. It may therefore be a basis for future practical work exploring ways to enforce more effectively usage control policies, both on the server and on the client sides.

One has also highlighted a new perspective in usage control where both the producer and the consumer sides have to be controlled. The current models do not consider that case

and should therefore be extended in the future. This could be done for example by adding a trusted third party responsible for the enforcement of usage control on the producer's side. One could also try to restrict the rights of a producer to define rights on objects.
One sees here a really promising direction for future work.

The concept of usage control is a really interesting concept. As one has remarked, it enables to solve many problems encountered in modern applications. However, a lot of work has still to be done in order to fully benefit from its features.

Appendix A.

Acronyms

XACML eXtensible Access Control Markup Language

REL Rights Expression Language

PEP Policy Enforcement Point

PDP Policy Decision Point

PAP Policy Administration Point

PIP Policy Information Point

RBAC Role Based Access Control

CMS Content Management System

DRM Digital Rights Management

XML Extensible Markup Language

BSD Berkeley Software Distribution

JCR Content Repository for Java Technology

API Application Programming Interface

JCA Java Cryptography Architecture

JSP JavaServer Pages

JSF JavaServer Faces

OCR Optical Character Recognition

RTF Rich Text Format

PDF Portable Document Format

MAC Mandatory Access Control

OSL Obligation Specification Language

Bibliography

[1] Open Mobile Alliance. Digital rights management - short paper. Technical report, Open Mobile Alliance Ltd, 2003.

[2] Titus Barik. Introducing the java content repository api. Technical report, IBM, 2005. Available at `http://www.ibm.com/developerworks/java/library/j-jcr/`.

[3] Koen Buyens, Sam Michiels, and Wouter Joosen. A refined architecture for drm. Technical report, Department of Computer Science, K.U. Leuven, 2006.

[4] Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato. Version control with subversion. Available to download at `http://svnbook.red-bean.com/`.

[5] Roberto Garcia Gonzalez. A semantic web approach to digital rights management. Technical report, Universitat Pompeu Fabra, Barcelona, 2005.

[6] Belgian government. Eid hierarchy and certificate profiles. Technical report, FedICT (Belgian government entity), 2006.

[7] Manuel Hilty, David Basin, and Alexander Pretschner. On obligations. Technical report, Information Security, ETH Zurich, Switzerland, 2005.

[8] Manuel Hilty, Alexander Pretschner, Christian Schaefer, and Thomas Walter. Enforcement for usage control - a system model and an obligation language for distributed usage control. Technical report, Information Security, ETH Zurich, DoCoMo Euro-Labs, 2006.

[9] Manuel Hilty, Alexander Pretschner, Christian Schaefer, and Thomas Walter. Enforcement for usage control - an overview of control mechanisms. Technical report, Information Security, ETH Zurich, DoCoMo Euro-Labs, 2006.

[10] Vladimir Kolovski, James Hendler, and Bijan Parsia. Analyzing web access control policies. Technical report, International World Wide Web Conference Committee, 2007.

[11] Matthias Leumann. Policy evaluation and negotiation in distributed usage control. Technical report, Department of Computer Science, ETH Zurich, 2007.

[12] Qusay H. Mahmoud. Developing web applications with javaserver faces. Technical report, Sun Developer Network, 2004. Available at `http://java.sun.com/developer/technicalArticles/GUI/JavaServerFaces/`.

[13] Sam Michiels, Wouter Joosen, Eddy Truyen, and Kristof Verslype. Digital rights management - a survey of existing technologies. Technical report, K.U. Leuven, 2005.

[14] OASIS. extensible access control markup language (xacml) version 2.0. Technical report, OASIS Open, 2004. Available at `http://docs.oasis-open.org/xacml/access_control-xacml-2.0-core-spec-cd-02.pdf`.

[15] OASIS. Core and hierarchical role based access control (rbac) profile of xacml v2.0. Technical report, OASIS Open, 2005. Available at `http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-spec-os.pdf`.

[16] Jaehong Park and Ravi Sandhu. Towards usage control models: Beyond traditional access control. Technical report, George Mason University, USA, 2002.

[17] Jaehong Park and Ravi Sandhu. The ucon abc usage control model. Technical report, George Mason University, USA, 2004.

[18] Alexander Pretschner, Manuel Hilty, and David Basin. Distributed usage control, 2006.

[19] Ravi Sandhu, Kumar Ranganathan, and Xinwen Zhang. Secure information sharing enabled by trusted computing and pei models.

[20] Ravi Sandhu, Xinwen Zhang, Kumar Ranganathan, and Michael J. Covington. Client-side access control enforcement using trusted computing and pei models. *Journal of High Speed Networks, IOS Press*, page 229Ű245, 2006.

[21] Thierry Sans, Frederic Cuppens, and Nora Cuppens-Boulahia. A flexible and distributed architecture to enforce dynamic access control. Technical report, GET/ENST Bretagne, France, 2005.

[22] Paul E. Sevinc. Securing information by controlling access to data in documents. Technical report, Department of Computer Science, ETH Zurich, 2007.

[23] Inc. SunMicrosystems. The java ee 5 tutorial. Technical report, SunMicrosystems, Inc., 2007. Available at `http://java.sun.com/javaee/5/docs/tutorial/doc/JavaEETutorial.pdf`.

[24] Pasi Tyrväinen. Concepts and a design for fair use and privacy in drm. *D-Lib Magazine*, 2005. Available at `http://www.dlib.org/dlib/february05/tyrvainen/02tyrvainen.html`.