Universitat Politècnica de Catalunya

Department of Computer Architecture

Ph.D. Dissertation

iDRM – Interoperability Mechanisms for Open Rights Management Platforms

Author: Carlos José Corredoura Serrão

Advisor: Professor Jaime Delgado

Co-Advisor: Professor Miguel Dias

December 2008

A dissertation submitted to the Department of Computer Architecture and the Committee on Graduate Studies of Universitat Politècnica de Catalunya in partial fulfilment of the requirements for the degree of Doctor.

Thesis Advisors
Catedràtic d'Universitat Jaime Delgado Mercé
Professor José Miguel Dias

Author
Carlos José Corredoura Serrão

iDRM – Interoperability Mechanisms for Open Rights Management Platforms

Abstract

Today's technology is raising important challenges in the Intellectual Property (IP) field in general and to Copyright in particular [Arkenbout et al., 2004]. The same technology that has made possible the access to content in a ubiquitous manner, available to everyone in a simple and fast way, is also the main responsible for the challenges affecting the digital content IP of our days [Chiariglione, 2000].

Technological solutions and legal frameworks were created to meet these new challenges. From the technological point of view, Rights Management Systems (RMS) and Copy Protection Systems (CPS) have been developed and deployed to try to cope with them. At first, they seemed to work however, their closed and non-interoperable nature and a growing number of wrong strategic business decisions, soon lead to a strong opposition. One of the strongest negative points is the lack of rights management interoperability [Geer, 2004].

The work presented on this thesis primarily addresses the RMS interoperability problems. The objective of the thesis is to present some possible mechanisms to improve the interoperability between the different existing and emerging rights management platforms [Guth, 2003a].

Several different possible directions to rights management interoperability are pointed in this thesis. One of the most important is openness. Interoperability between different rights management mechanisms can only be achieved if they are open up to a certain level.

Based on this concept, an open rights management platform is designed and presented in this thesis. Also, some of the interoperability mechanisms are presented and explained. This

platform makes usage of the emerging service-oriented architectures to provide a set of distributed rights management services.

Rights management solutions rely heavily on the establishment of authenticated and trust environments between its different elements. While considering different RMS, the establishment of such trust environments can be somehow complex. This thesis provides a contribution to the establishment of interoperable RMS trust environments through the usage of Public-Key Infrastructure (PKI) mechanisms.

Modern rights management systems have to handle with both keying material and licenses which are used mostly to define how content is governed by the system. Managing this is a complex and hard task when different rights management solutions are considered. This thesis presents and describes a generic model to handle the key and license management life cycle, that can be used to establish a global interoperable management solution between different RMS.

Acknowledgements

Becoming a Ph.D. student and being able to complete it with success has always been one of my very personal dreams. Concluding it was not an easy task, and without the help and the support of many key-persons I am sure that I would not reach the end. Therefore, I will use this opportunity to publicly thank to all of them.

First, I would like to thank my family for their unlimited support and their spirit of sacrifice. To my wife, Fátima, who has always supported and encouraged me to keep on going, even if that would mean having to spent almost a year away from home and far from her. As hard as it may seem, it was easy when compared to leaving her with the sole responsibility of taking care of our two beloved children. To both of them, David and Dinis, I apologise for my long absence. Please also apologise for the time we did not spend together and sometimes for my bad humour and stress.

To my parents, because they have provided me the opportunity and the financial support carry with my studies. Also, because they always stood beside me providing me the moral support to never give up even when things went a little darker.

To my parents in law, not only by their unquestionable support but also for their immense support in the education of their grandchildren, while I was not around.

A very special thank to both of my Ph.D. advisors, Professor Miguel Dias and Professor Jaime Delgado. Without them this would not have been possible.

Professor Miguel Dias is a highly energetic person that has the capability to motivate everyone around them. Thank you for your vision, advice and support.

To Professor Jaime Delgado, for the opportunity that he provided me to integrate his own successful research group in the beautiful city of Barcelona for a year. This was a crucial success factor for the conclusion of this work. Thank you very much for all the support, advice and commitment. Indeed, this was deeply appreciated. I sincerely wish that this is not the end of our collaboration.

To my former MSc. advisor, Professor José Cordeiro Gomes, for its unconditional support and advise on what concerned my Master thesis (one of the grounds of this work) and also for its

support and motivation to carry with my Ph.D.

To Dr. Panos Kudumakis, a friend, colleague and project manager of a European R&D Project (MOSES) and a very enthusiastic and positive person. As a very dedicated professional in this field, the ideas exchanged with him were quite helpful for this work.

To the memory of Eng. José Guimarães, that unfortunately is not among us any more. He was the main responsible for my decision to remain in the academy world and to start my R&D and lecturing career. He is also responsible for part of my professional training and to introduce me with the International R&D projects work. Wherever you are, thank you very much.

To ADETTI, in special to the NetMuST lab, who has provided me the opportunity to learn more and to work on International co-operative R&D since 1995.

To everyone else whose name is not mention here, but which were decisive for the conclusion of this work.

Thank you very much!

Table of Contents

Images Index

Terms and Definitions

AES	Advanced Encryption Standard
AUS	Authentication service
AXCS	AXMEDIS Certifier and Supervisor
AXCV	AXMEDIS Certification and Verification
AXMEDIS	Automating Production of Cross Media Content for Multi-channel Distribution
AXS	AXMEDIS Supervisor
B2B	Business to Business
B2C	Business to Consumer
C/P	Copy-Protection
CA	Certification Authority
CAMART	Core Accounting Manager and Reporting Tool
CAS	Conditional Access System
CCOL	Common Command Operation Language
CERN	European Organisation for Nuclear Research
CEK	Content Encryption Key
CFS	Configuration service
CI	Content Issuer
CMS	Content Management System
CO	Copyright Owner
CODEC	Coding/Decoding
CORBA	Common Object Request Broker Architecture

COS	E-Commerce Service
CPS	Content Preparation Service
CRA	Content Rendering Application
CSR	Certificate Signing Request
DCB	DMP Content Broadcast
DCF (OMA)	Digital Rights Management Content Format
DCF (DMP)	DMP Content File
DCI	DMP Content Information
DCOM	Distributed Component Object Model
DCS	DMP Content Streaming
DI	Digital Item
DID	Digital Item Declaration
DIDL	Digital Item Declaration Language
DII	Digital Item Identification
DMAG	Distributed Multimedia Applications Group
DMP	Digital Media Project
DOD	Digital Object Distributor
DOI	Digital Object Identifier
DOPS	Digital Object Provider Service
DORD	Digital Object Rendering Device
DORS	Digital Object Registration Service
DOUI	Digital Object Unique Identifier
DRM	Digital Rights Management
DRM-A	Digital Rights Management Agent
DVB-CP	Digital Video Broadcast, Copy Protection
e-IPR	Electronic Intellectual Property Rights
EAI	Enterprise Application Integration
EC	European Commission
EJBCA	Enterprise Java Beans Certification Authority
EKCS	Encryption Key Creation Service
ENVISAT	Environmental Satellite
EO	Earth Observation
EPG	Electronic Program Guide
ER	Event Reporting
ERR	Event Report Request
ESA	European Space Agency
FSF	Free Software Foundation
GPL	GNU Public License
GUI	Graphical User Interface

H2KBB	HICOD2000 Black-Box
hDRM	Horizontal Digital Rights Management
HICOD2000	High-Performance Coding, Protection and Trading of Satellite Images, Using JPEG2000
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
INDICARE	Informed Dialogue about Consumer Acceptability of DRM Solutions in Europe
IP	Intellectual Property
IPMP	Intellectual Property Management and Protection
ISMA	Internet Streaming Media Alliance
KEK	Key Encryption Key
LIS	License service
MAC	Message Authentication Code
MDS	Media Distribution Service
MIME	Multipurpose Internet Mail Extensions
MIPAMS	Multimedia Information Protection and Management System
MMI	Mother-May-I
MMS	Multimedia Messaging Service
MOM	Message Oriented Middle-ware
MOSES	MPEG Open Security for Embedded Systems
MPL	Mozilla Public License
NAVSHP	Networked Audio Visual Systems and Home Platforms
NEMO	Network Environment for Media Orchestration
OBEX	Object Exchange
OCP	OpenSDRM Capability Proxy
OCSP	Online Certificate Status Protocol
ODRL	Open Digital Rights Language
OFB	Output Feedback
OMA	Open Mobile Alliance
OMA-DRM	Open Mobile Alliance Digital Rights Management
OSI	Open-Source Initiative
OSS	Open-Source Software
P2P	Peer to Peer
PAV	Portable Audio-Visual device
PDA	Personal Data Assistant
PDCF	Protected DCF
PDS	Payload Data Segment
PGW	Payment Gateway service

PKCS	Public Key Cryptography Standards
PKI	Public-Key Infrastructures
PKIX	Public Key Infrastructure (X.509)
PMS	Protection Manager Support
PTS	Protection Tools service
RDD	Rights Data Dictionary
REK	Rights Encryption Key
REL	Rights Expression Language
RGS	Registration service
RI	Rights Issuer
RMI	Remote Method Invocation
RMS	Rights Management Systems
RO	Rights Object
ROAP	Rights Object Access Protocol
RSA	Rivest, Shamir, Adleman
RTP	Real Time Protocol
RTCP	Real Time Control Protocol
RTSP	Real Time Streaming Protocol
SAC	Secure Authenticated Channel
SAML	Security Assertion Markup Language
SAV	Stationary Audio-Visual device
SDK	Software Development Kit
SDMI	Secure Digital Music Initiative
SEP	Service Enabling Platform
SKSS	Secure Key Storage Service
SLS/TLS	Secure License Service/ Transport Layer Security
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SPKI	Subject Public-Key Info
SSL	Secure Sockets Layer
SWOT	Strengths, Weaknesses, Opportunities and Threats
UDDI	Universal Description Discovery and Integration
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
vDRM	Vertical Digital Rights Management
WAP	Wireless Application Protocol
WMA	Windows Media Audio
WMDRM	Windows Media DRM
WMDRM-ND	Windows Media DRM for Network Devices

WMDRM-PD	Windows Media DRM for Portable Devices
WMF	Windows Media Format
WMRM	Windows Media Rights Manager
WMV	Windows Media Video
WRMiM	Wallet Rights Management interoperability Middle-ware
WS	Web Services
WS-BPEL	Web Services Business Process Execution Language
WSDL	Web Services Description Language
WSP	Wireless Session Protocol
WWW	World Wide Web
XML	Extensible Markup Language
XrML	Extensible Rights Management Language

Notation and Conventions

This part contains some of the conventions and notation format that is used in this document. This is particularly important to understand some of the protocols and methods described in the document.

K_{pub}^{A}: the public-key of entity A;

K_{priv}^{A}: the private-key of entity A;

$Cert^{A}_{B}$: indicates that a digital certificate, issued by entity A, that certifies the entity (the public-key) of entity B;

K_{AES}: an AES cryptographic secret key;

$K_{AES}[info]$: represents some information ciphered with an AES secret key;

$MD5(info)$: represents some information hashed with an hashing algorithm (in this case is MD5);

$K_{pub}^{A}\{info\}$: represents some information ciphered with the public-key of an entity A;

$K_{priv}^{A}[info]$: represents some information signed (digital signature) with the private-key of an entity A.

Part I. Introduction

Chapter 1. Introduction

1 Introduction

Upholding intellectual proprietary (IP) in a growing global digital World is a daunting task. What seems to be effective in the analogue World is not adequate for the digital World [Duhl and Keroskian, 2003]. One of the major problems of this digital World is bits. Bits can be easily moved, copied, transferred, distributed and deleted [Schneier, 2001]. This raises important challenges in the IP management arena, because author rights, inherited from the old analogue age, cannot be properly safeguarded or enforced [Duhl and Keroskian, 2003].

Digital makes part of everyone's live especially on what concerns content. Users are constantly bombarded from different sources and with the most diverse types of stimulus to acquire different types of objects, in particular in digital format (digital objects), or devices to handle those digital objects.

Companies are launching new types of devices everyday, capable of handling more advanced digital objects. More and more devices are appearing on the market, with more capabilities, connectivity and processing power [Weber, 2007]. Also new mobile open development kits are appearing, promoting the development growth of new and interesting applications [OHA, 2007]. This powerful combination presents an opportunity for digital content producers and providers but at the same time, an enormous challenge [IFPI, 2004][IFPI, 2005].

At the same time, end-users give a new meaning to the word “end”. In the digital content value-chain, the role of the end-users is less and less, a passive one. They use content to create new content, and at a glimpse of an eye they suddenly change their role from passive

end-users to active content producers and providers [Kelly et al., 2002].

This is an opportunity because digital content producers and providers have new and attractive distribution channels for pushing their digital products to a wider range of potential users and customers eager for content to be available on their devices. This digital era offers innovative digital technologies to content producers, leading to the improvement and provision of more interesting and interactive content. Also, providers are able to distribute digital content in a much faster and reliable manner.

However, at the same time, it is also an important challenge. Digital content can be easily copied and illegally distributed without proper authorisation. In the old analogue age this was also a problem (vinyl discs, cassettes and video-tapes could also be copied) however, digital has aggravated this problem [Michael and Rosenblatt, 2005]. Digital makes possible the production of unlimited illegal copies without any quality loss (exact bit by bit copies can be easily obtained at low cost). More, with high availability of digital networks and new distribution architectures (such as P2P) [White, 2000], the illegal distribution of content is facilitated.

To make digital content distribution and commercialisation business models work, specific technological measures have been put in place, by the content industry to prevent the proliferation of uncontrolled illegal copying, usage and distribution. These technological measures have the capability to protect the digital object, manage the rights, both associated to it and to users, and prevent illegal usage and distribution [Bechtold, 2006].

The behind the scenes technology that is capable of providing the necessary means to uphold digital object governance business models is called Digital Rights Management (DRM). DRM is composed of a Rights Management System (RMS) and are often used together with Copy-Protection (C/P) mechanisms also to try to prevent illegal copying. Both, used individually or together, have the means to provide a way to make digital content business models viable. The most well known example of how these business models work is the iTunes content store. Currently, iTunes is a worldwide leader in digital music, audio books, TV series and movies trading. The Apple iTunes on-line store applies copy-protection measures to the digital objects and uses a built-in DRM technology to ensure the proper rights management [Lenzi et al., 2003].

These two concepts of C/P and RMS are most of the times confused. However, it is possible to have rights management without C/P, the same way it is possible to have C/P without any management of rights. The work presented in this thesis will only address the rights management issue and not C/P.

DRM is a technology that involves the description, layering, analysis, valuation, trading and monitoring of rights over an individual or organisation's digital assets. DRM covers the digital management of rights [Duhl and Keroskian, 2003]. DRM is composed by a chain of hardware and software services and technologies governing the authorised use of digital objects and managing any consequences of that use throughout the entire digital object life cycle.

The copyright environment consists of three main aspects: rights (what can be protected by copyright) and exceptions (e.g. copies for private use or for public libraries); enforcement of rights; and management of rights (exploiting the rights). In the on-line World, management of rights may be facilitated by the use of these DRM systems [Rosenblatt et al., 2001].

Currently, digital rights management technologies are facing an enormous challenge. They are being confronted by a strong opposition from many sources. DRM, as it exists today, limits the end-user freedom to use legally acquired content [Groenenboom et al., 2006]. Sometimes it is even easier to illegally obtain content than through a legal content business establishment – not only because content is free but because it does not impose any restrictions on the way they use the content [Russinovich, 2005].

One other example of user lower satisfaction is the fact that DRM-governed content acquired on a digital content store will only play on the specific content rendering applications and/or devices that were designed to support such DRM. If the user wishes to acquire content on any other store that uses some different DRM technology, the user will be unable to render the content on the very same applications and/or devices [Seki and Kameyama, 2003].

This is truly a problem for modern digital content users, but at the same time it is also a huge problem for content providers. With this current model, content producers and providers will either need to make their digital content available for a different set of platforms and systems or to make technological choices and provide their content for just a single platform [Geer, 2004]. For digital content users this obtrusive situation limits the user’s adherence to

just one of the available content stores and to one of the available content rendering applications and/or devices. This represents one of the hardest barriers in current DRM deployment – interoperability [Schmidt et al., 2004].

Interoperability is currently one of the most debated problems in the DRM field. In particular, there is a lack of interoperability between the currently commercially available DRM solutions. The actual trend are the one-to-one DRM systems, in which a vertical integration exists in the digital content value chain around a single DRM system. This old-fashioned business strategy tries to raise entry barriers [Porter, 2004] to new players in the same field and at the same time are quite penalising for end-users [Russinovich, 2005]. These DRM interoperability issues affect DRM in general, and will be addressed later on this document.

The work presented in this document will focus on the interoperability issues that RMS will have to address. As presented before, this is still one of the biggest issues that DRM technology will have to solve, and therefore this work will address the specific open RMS interoperability problems and it will draw some approaches to solve partly the question of the rights management layer interoperability on in the context of RMS platforms. The author is aware that his contributions, presented in this document, are like a drop of water in an ocean, and that many issues will still remain open and unsolved, towards the existence of truly and complete interoperability between RMS solutions.

2 Objectives

The overall objective of the work presented in this thesis is to study the RMS interoperability problems and to provide contributions towards the partial solution of this problem. The RMS interoperability problem is a vast and broad problem. Therefore the work has been focused on some specific are relevant areas which are described next.

First, the identification and description of the most significant DRM architectures and other initiatives that deals with RMS interoperability problems and the identification of the basic functions and processes of those RMS solutions. This first step will involve a study of several DRM initiatives, related to RMS interoperability, such as MPEG-21 [Bormans and Hill, 2002], the Digital Media Project (DMP) [Chiariglione, 2005], the Coral Consortium [Kalker et al.,

2007], Sun DReaM [SunLabs, 2007] and others.

Second, study the applicability of service-oriented architectures and web-services on the design of interoperable services for rights management processes. This is important, because the different components of the different rights management systems can interface with each other through a common interface and therefore improve the overall interoperability between the different services.

Third, study the complexity of the different rights management systems in the establishment of authentication and trust environments, and designing and defining a method, based on Public-Key Infra-structures (PKI) and brokerage mechanisms to create trust environments between different rights management solutions, therefore increasing the interoperability among them [Lee et al., 2004].

Fourth, the identification of the specific DRM processes that handle key management and license management. During this phase a generic for secure license and key management process for DRM will be defined. During this stage, the necessary requirements for secure license and key management will be identified and will be taken into account in the definition of this model.

Fifth, design and develop an open-source distributed service-based rights management platform that will enable interoperability between the different services of the content value-chain. This platform is content agnostic and business model independent, and can be used to test and deploy different governed digital content business scenarios. Moreover, this is an open platform that provides public open interfaces for interoperating with other existing rights management platforms, one of the approaches uphold in this thesis

Sixth, create an interoperability system that will enable the different content rendering applications to interoperate through a common interoperable rights management middle-ware. This will enable the different content rendering mechanisms to abstract from the complexities of the rights management processes, while using digital governed content. This system will allow a de-coupling between the content rendering application and the rights management system, through an abstraction layer capable of handling the rights clearance on behalf of the rendering application.

The major overall objective of this work is to provide a set of mechanisms that will provide some solutions to solve part of the interoperability problems that affect most of the existing vertical DRM solutions.

3 Document Structure

The work presented in this thesis is organised in four different parts composed by a set different chapters.

This first part introduces this work, providing a contextual overview and the description of the problem that it will address. It provides also an overall overview of the work objectives and describes the following document structure, in particular, the contributions part.

The second part of the document is devoted to the analysis and description of the state of the art. The state of the art part is composed by two chapters, one that presents and describes the different existing DRM platforms, its architectures and mechanisms, and a second chapter that presents and describes the RMS initiatives that handle directly with the DRM interoperability problems, assuming themselves as interoperability frameworks.

The following part of the document contains the different contributions of this thesis. Each of the contributions is detailed on a separated chapter. The first chapter on the contributions part, introduces the problems that are addressed by this thesis, and provides an overview of the different contributions proposed to face the stated problems. The next chapter presents a contribution about how web-services and service-oriented architectures can be used to create de-coupled rights management services, establishing a favourable environment for rights management interoperability.

The third chapter in the contributions part contains a contribution on how to the handle the non-interoperable trust mechanisms present on current rights management. This chapter provides contributions on how to use the well-established PKI to establish a common and interoperable trust environment between different rights management solutions.

The next chapter presents contributions on how an open model should be used to promote interoperability between the different rights management solutions. This open model,

proposes that the different rights management solutions provide a way, based on open-source implementations, open and public specifications or public interfaces, for their interoperability.

The following chapter of this part, contains a contribution on key and license management on rights management systems. The objective of this contribution is to derive a common model for managing the management of rights management systems keying material and licenses, and therefore contribute to the interoperability and security of different rights governance systems.

The sixth chapter presents a contribution on the specification, design and implementation of an open rights management system. This system implements the mechanisms presented on the previous chapters and aims to be an example of a truly open rights management platform and an interoperability promoter.

The following chapter of this contributions part, presents the specification, design and implementation of client-side middle-ware that provides an abstraction rights management layer between content rendering application and the rights management systems. This promotes interoperability ad the client-side between the different devices. Another contribution presented in this chapter refers to the definition and establishment of license templates, to allow an abstraction from the rights expression mechanisms used by the different rights management systems.

Finally, the last chapter on this part, presents a list of governed content use-cases that demonstrate the usage of the open rights management system developed and the remaining contribution on this thesis.

The final part of this document contains the major conclusions of this work. This part also contains some directions for future work and further research in this field, that emerged from the work conducted in this thesis.

Part II. State of the Art

Chapter 1. DRM Architectures

1 Introduction

In this chapter, an overview of the most relevant existing DRM architectures is presented. The presented DRM architectures are the result of international standardisation committees (ISO MPEG-21) [Bormans and Hill, 2002], international non-profit initiatives (Digital Media Project (DMP)[1] [Chiariglione, 2005], Sun DReaM [SunLabs, 2007], Open Mobile Alliance DRM (OMA-DRM)[2] [OMA, 2006b], AXMEDIS [Torres et al., 2006], OpenIPMP[3] [OpenIPMP, 2003], Coral[4] [Kalker et al., 2007], Marlin[5] [Marlin, 2006a]) and some commercial systems (Microsoft Windows Media Rights Management [Microsoft, 2004a] and Apple FairPlay [Venkataramu, 2007]).

The following sections on this chapter provide a small description of each of the DRM systems, with emphasis on the architectural and component organisation and interaction aspects.

2 Open Mobile Alliance (OMA) DRM

This is the DRM system defined by the Open Mobile Alliance (OMA), a forum of more than 200 industry partners leading activities on the mobile communications environment. This system is implemented by mobile devices provided by the different manufacturers and has to

1 http://www.dmpf.org
2 http://www.openmobilealliance.org
3 http://objectlab.com/clients/openipmp/
4 http://www.coral-interop.org/
5 http://www.marlin-community.com/

be interoperable among them [OMA, 2007].

The scope of OMA DRM (Open Mobile Alliance DRM) is to enable the controlled consumption of digital media objects by allowing content providers the ability, for example, to manage previews of DRM Content, to enable superdistribution [Chong and Deng, 2006] of DRM Content, and to enable transfer of content between DRM Agents. The OMA DRM specifications [OMA, 2006a] [OMA, 2006b] [OMA, 2006c] [OMA, 2006d] provide mechanisms for secure authentication of trusted DRM Agents, and for secure packaging and transfer of usage rights and DRM Content to trusted DRM Agents.

OMA DRM defines a set of Actors and Components in its reference architecture. The most relevant are the DRM Agent (DRM-A), Content Issuer (CI), Rights Issuer (RI), User and Off-device Storage.

The DRM Agent represents a trusted entity in a device. This entity enforces permissions and constraints associated with DRM content, controlling the access and usage of DRM content [OMA, 2006b].

The Content Issuer delivers DRM content. OMA DRM defines the DRM content format to be delivered to DRM-A, and also defines the way the content can be transported from a CI to a DRM-A using alternative transport mechanisms [OMA, 2006c]. The DRM content packaging may be handled directly by the CI or it may receive it from an external source.

The Rights Issuer is the OMA DRM entity that assigns permissions and constraints to DRM content, and generates Rights Objects (RO) [OMA, 2006c]. The RO is represented in XML and expresses permissions and constraints associated with DRM content.

A User is a human user of DRM content, which can only access DRM content through a trusted DRM Agent.

The Off-Device Storage allows DRM content to be stored off-device, for backup purposes, to free memory on the device. RO with stateless permissions can also be off-device stored.

The OMA DRM system enables the CI to distribute Protected Content and RI to issue Rights Objects for the Protected Content. The DRM system is independent of media object formats, operating systems and runtime environments. For User consumption of the Content, Users

acquire Permissions to Protected Content by contacting RI. RI grants appropriate Permissions for the Protected Content to User Devices [OMA, 2006d]. The Content is cryptographically protected when distributed and therefore the Protected Content will not be usable without an associated Rights Object issued and cryptographically bound to the User's Device.

The Protected Content can be delivered to the Device by any means. But the Rights Objects are tightly controlled and distributed by the RI. The Protected Content and Rights Objects can be delivered to the Device together, or separately. The system does not imply any order or “bundling” of these two objects. It is not within the scope of the DRM system to address the specific payment methods employed by the RI [OMA, 2006d].

The OMA DRM specifications define the format and the protection mechanism for DRM Content, the format and the protection mechanism for the Rights Object, and the security model for management of encryption keys. The OMA DRM specifications also define how DRM Content and Rights Objects may be transported to devices using a range of transport mechanisms [OMA, 2006b]. Any interaction between network entities is out of scope (Figure II.1.1).

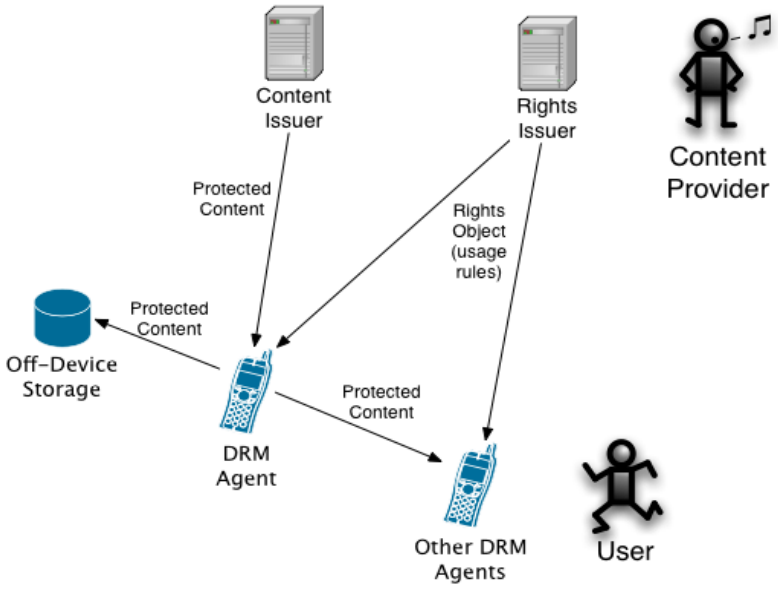

Figure II.1.1: OMA-DRM generic architecture

Before content is delivered, it is packaged to protect it from unauthorised access. A content issuer delivers DRM Content, and a RI generates a Rights Object. The CI and RI embody roles in the system. Depending on deployment they may be provided by the same or different

actors, and implemented by the same or different network nodes.

DRM Content cannot be used without an associated Rights Object, and may only be used according to the permissions and constraints specified in a Rights Object [OMA, 2006d].

Rights Objects are used to specify consumption rules for DRM Content. The Rights Expression Language (REL) defined by OMA DRM specifies the syntax (XML) and semantics of permissions and constraints governing the usage of DRM Content [OMA, 2006d]. An instance of a rights document is called a Rights Object, and has its own MIME (Multipurpose Internet Mail Extensions) content type. Rights Objects are made up of permissions (e.g. play, display and execute) and constraints (e.g. play for a month, display ten times). Rights Objects may also include constraints that require a certain user (user identity) to be present when the content is used [OMA, 2006d]. These permissions and constraints, along with other information embodied in the Rights Object, (e.g. copyright information) may be presented to the user. The Rights Object also governs access to DRM Content by including the Content Encryption Key (CEK) [OMA, 2006d].

Rights Objects associated with DRM Content have to be enforced at the point of consumption, which is modelled in the OMA DRM specifications by the DRM Agent [OMA, 2006b]. The DRM Agent embodies a trusted component of a device, responsible for enforcing permissions and constraints for DRM Content on the device, and controlling access to DRM Content on the device.

DRM Content can only be accessed with a valid Rights Object, and so can be freely distributed, as Rights Objects are cryptographically bound to DRM Agents. This enables super-distribution, as users can freely transfer DRM Content between them. To access DRM Content on the new device, a new Rights Object has to be requested and delivered to a DRM Agent on that device.

If rights issuers support it, a Rights Object may optionally be bound to a group of DRM Agents. This is known in the OMA DRM specifications as a Domain [OMA, 2006b]. DRM Content and Rights Objects distributed to a domain can be shared and accessed off-line on all DRM Agents belonging to that domain. For example, a user may purchase DRM Content for use on both her phone and her PDA (Personal Data Assistant).

In order to show how the previous concepts are linked among them, the following summarises the basic steps for distributing DRM Content [Pimenta and Serrão, 2006]:

1. Content packaging: Content is packaged in a secure content container (DRM Content Format or DCF). DRM Content is encrypted with a symmetric content encryption key. Content can be pre-packaged, i.e. content packaging does not have to happen on the fly. Although not required by the OMA DRM specifications or the OMA DRM architecture, it is recommended that the same CEK is not used for all instances of a piece of content. Using the same CEK for all content instances would pose a greater risk if a single device were to be hacked and a CEK stored on that device exposed. Using a different CEK for different deliveries or different devices will limit this risk;

2. DRM Agent authentication: All DRM Agents have a unique private/public key pair and a certificate. The certificate includes additional information, such as maker, device type, software version, serial numbers, and others. This allows the content and rights issuers to securely authenticate a DRM Agent. Any privacy aspects with releasing such information are addressed in the technical specifications [OMA, 2006b];

3. Rights Object generation: A Rights Object is an XML document, expressing the permissions and constraints associated with the content. The Rights Object also contains the CEK – this ensures that DRM Content cannot be used without an associated Rights Object.;

4. Rights Object protection: Before delivering the Rights Object, sensitive parts are encrypted (e.g. the CEK), and the Rights Object is then cryptographically bound to the target DRM Agent. This ensures that only the target DRM Agent can access the Rights Object and thus the DRM Content In addition, the RI digitally signs the RO;

5. Delivery: The RO and DCF can now be delivered to the target DRM Agent. Since both are inherently secure, they can be delivered using any transport mechanism (e.g. HTTP/WSP, WAP Push, MMS). They can be delivered together, e.g. in a MIME multipart response, or they can be delivered separately.

2.1 OMA Rights Object

In OMA DRM, rights are self-contained objects, separate from content and independent from the content type. These objects use the REL for defining their syntax and semantics which in turn is based on the Open Digital Rights Language (ODRL) [ODRL, 2002]. REL uses XML schema to define all the different elements inside a Rights Object (RO) [OMA, 2006d]. Elements are grouped according to their functionality in different models, namely the: Foundation (root/basis), Agreement (main container), Context (metadata), Permission (i.e. permission to: play, display execute, print, export), Constraint (time – DRM Time - and count based constraints), Inheritance (inheriting rights from a parent RO) and Security models (message digest and encryption data). Figure II.1.2 depicts a generic diagram for a RO outlining of the various models.

A RO may be associated to different content objects by having <asset> elements by referencing the content identifier, message digest (not present in REL 1.0 – no form of referencing a DCF by its message digest or "hash") of the content object using the SHA-1 algorithm and key information about the CEK. The confidentiality of the content associated with the RO is directly and solely dependent on the confidentiality of the CEK which in turn is wrapped using a Rights Encryption Key (REK) or Key Encryption Key (KEK) with the AES-WRAP algorithm [Pimenta and Serrão, 2006]. The integrity of the association is maintained through the message digest on the content end and through XML signatures on the RO end.

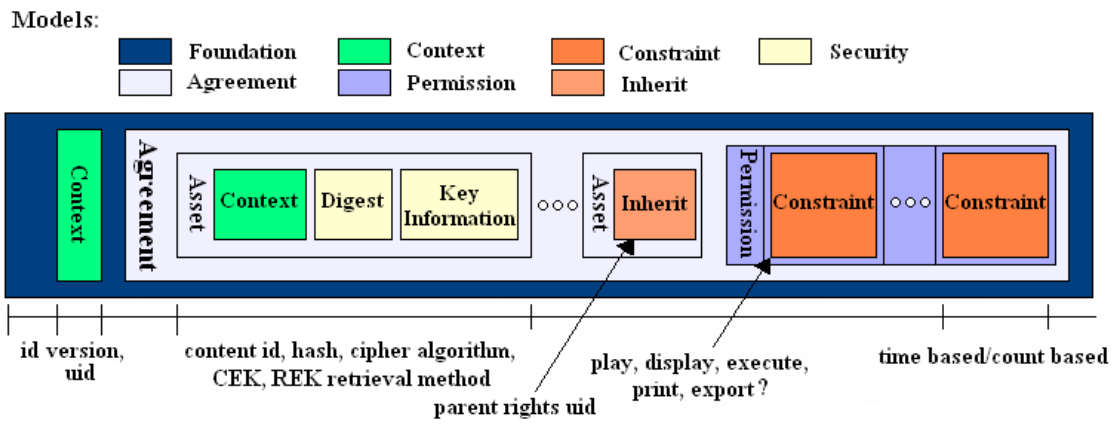

Figure II.1.2: Rights Object graphic representation

It may also be associated to a parent RO through the <asset> element which is one of the

other more visible differences from REL version 1.0 since previously there was no inheritance mechanism established. It should be noted however that parent ROs should not reference content objects directly themselves on their <asset> elements and parent-child inheritances must not exceed one level of depth.

There are also significant additions and changes within the remaining models. The permission model did not define exportation rules (<export>) since OMA DRM 1.0 relied on the forward-lock mechanism. The constraint model did not feature timed-count, accumulation, individual and system constraints.

Protected ROs are outside the OMA DRM 2.0 REL specification [OMA, 2006d] but are described in the OMA DRM 2.0 specification. They are referred to as “protected” since, unlike ROs that are encapsulated within <rights> tags. These tags include security information that prevents improper modifications/tampering and more: rights data authenticity and integrity information in association with the issuing RI (e.g. signatures); a time-stamp; the REK retrieval related data, (under the <encKey> element); association with the target DRM Subject Public Key Info (SPKI) through a digest value; canonicalisation algorithm identification and integrity data (signature) for the <ro> element.

A protected RO is obtained through a successful execution of the Rights Object Access Protocol (ROAP) [OMA, 2006a]. These can be device ROs which are associated with a sole device (user) through its public key or domain ROs which are associated a domain by means of a Domain Key.

The indicated procedure to be followed after receiving a domain or device protected RO includes verifying the existence of a valid Rights Issuer (RI) Context with the issuing RI or verifying the MAC and signature on the RO (if present). If the RO is stateful (i.e. contains any or all of the timed-count, count, interval, accumulated move type <export> constraints) the agent should perform a RO replay cache. A local replay cache should be stored by the agent to avoid replay attacks for this type of ROs. Succeeding with all the above mentioned checks wields a successful RO installation.

2.2 OMA DRM Protocols

OMA specifies a set of specific DRM protocols to obtain the rights object. These protocols are designated as Rights Object Access Protocol (ROAP) [OMA, 2006a] and will be described in this section.

The ROAP protocol suite defined in the OMA DRM 2.0 specification is the essential means by which the Agent entity makes contact with a RI, essentially in order to receive rights in a traceable and protected manner. This suite of security protocols, which is independent of the transport mechanism (e.g. HTTP), is primarily used as the means for acquiring ROs from RIs but, it is also composed by other sub-protocols that accomplish several other mechanisms within the DRM system, such as, joining/leaving a domain and providing connectivity for unconnected devices (i.e. ROAP over the IrDa Object Exchange Protocol or OBEX – Object EXchange).

There are two ways of initiating ROAP: 1) through receiving of a ROAP Trigger or 2) from a DCF [OMA, 2006b]. Initiating ROAP from a DCF object partakes to the analysis of the Box structure inside, particularly the Common Headers Box (present in both DCF and Protected DCF (PDCF)), which should contain the RI URL and prioritised textual headers. The DRM Agent then executes a HTTP Get request to the URL which may result in a (X)HTML session or a ROAP Trigger.

Upon first contact, and sequentially whenever deemed necessary (e.g. RI context expiration, loss of DRM Time synchronisation), the Agent and RI go through a handshaking process using the Registration Protocol [OMA, 2006b]. Successful registration results in the storage of RI context information, namely and at least: device ID, RI ID, RI, ROAP version, agreed upon algorithms (Hash, MAC, signature, key transport, key wrapping, canonicalisation), indication for RI certificate caching and RI context expire time (equal to the RI certificate expiry time or infinite for unconnected devices). After context storage, mutual authentication is made possible on the other ROAP protocol runs.

If executed successfully, including all the security checks (e.g., nonces, certificates) a RO Acquisition protocol run returns a valid RO in the RO Response message, otherwise an error message is returned in the status field of that message or the DRM Agent detects an error.

Similar status indications occur with the Join and Leave Domain Protocols, the remaining existing ROAP protocol runs.

2.2.1 Registration Protocol

As defined by the OMA DRM 2.0 specification [OMA, 2006b], the registration protocol is the means by which a DRM Agent and a RI establish a mutual context or RI context for the DRM Agent.

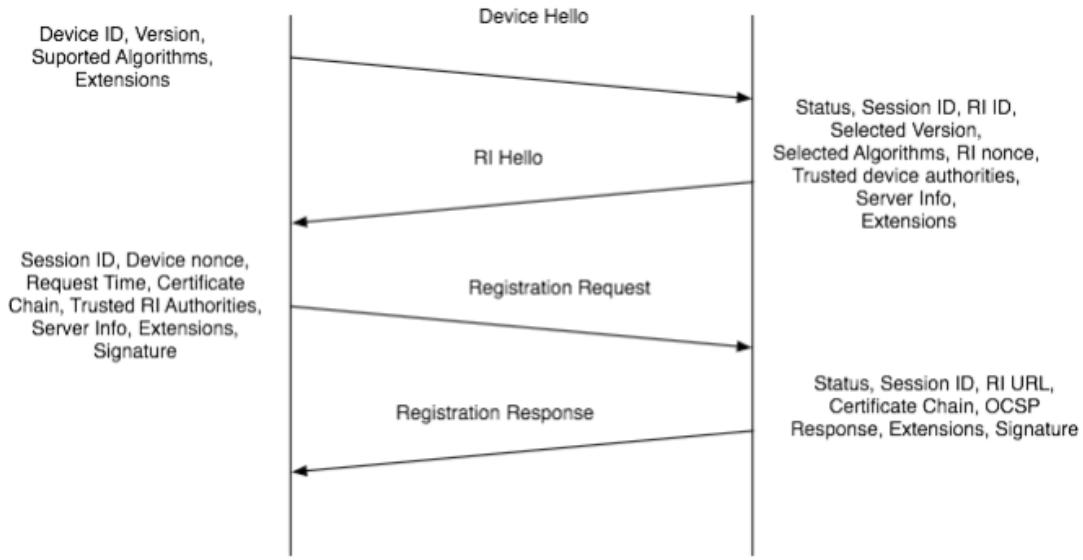

Figure II.1.3: ROAP – Registration Protocol

The Device Hello message initiating the ROAP Registration contains the Device ID parameter, the ROAP version (both are mandatory). This message contains also two optional fields. The list of supported algorithms by the device (hash algorithms, MAC algorithms, signature algorithms, key transport algorithms and key wrap algorithms) and the Extensions, for which one possible extension, called Certificate Caching is defined, indicating to the RI that the device has the capability to store information in the RI context whether an RI has stored device certificate information or not.

The RI Hello message is the answer from the RI to the device. This message is composed by a set of mandatory and optional fields. The Status indicates whether the Device Hello was executed with success or not. The Session ID identifies the protocol session identifier created by the RI. The Selected Version is the selected ROAP version. The RI ID identifies the RI to the device, and corresponds to the hash of the RI public key info as it appears in the RI

certificate. Selected Algorithms specifies the cryptographic algorithms to be used in the subsequent ROAP interactions. RI Nonce is a random nonce sent by the RI. Trusted device authorities are a list of device trust anchors recognised by the RI. Server info contains server specific information that the device needs to return unmodified in the next step of the protocol (Figure II.1.3). Three extensions are also defined. They are the Peer Key identifier, an identifier for a device public key stored by the RI; Certificate Caching indicating to the device that the RI has the capability to store information about the device certificate and; Device details, in which the RI can request the device to return device specific information in the next protocol message [OMA, 2006b].

After this, the Registration Request message is sent. This message contains the Session ID identical to the session id passed on the previous message, the Device nonce, which is a nonce generated by the device, the Request Time is the current time measured by the device, the Certificate Chain, the Trusted RI Authorities, the list of RI trust anchors recognised by the device, the Server Info, containing the same information that was passed previously, and some Extensions. The Extensions can be the Peer Key Identifier, No OCSP Response or OCSP Responder Key Identifier.

The final message is the Registration Response. This message contains the Status indicating if the Registration Request was successful or not, the Session ID that will be identical to previous ones, RI URL is the ROAP URL that should be stored in RI context, the Certificate Chain, the OCSP Response that includes the valid OCSP responses for the RI's certificate chain, and a list of Extensions. These Extensions include the Domain Name white list. Finally a Signature is applied to all the data in the message [OMA, 2006b].

2.2.2 RO Acquisition Protocol

The RO Acquisition is the OMA DRM 2.0 defined means of obtaining rights for rights-free DCFs or for getting new rights to replace old/expired ones.

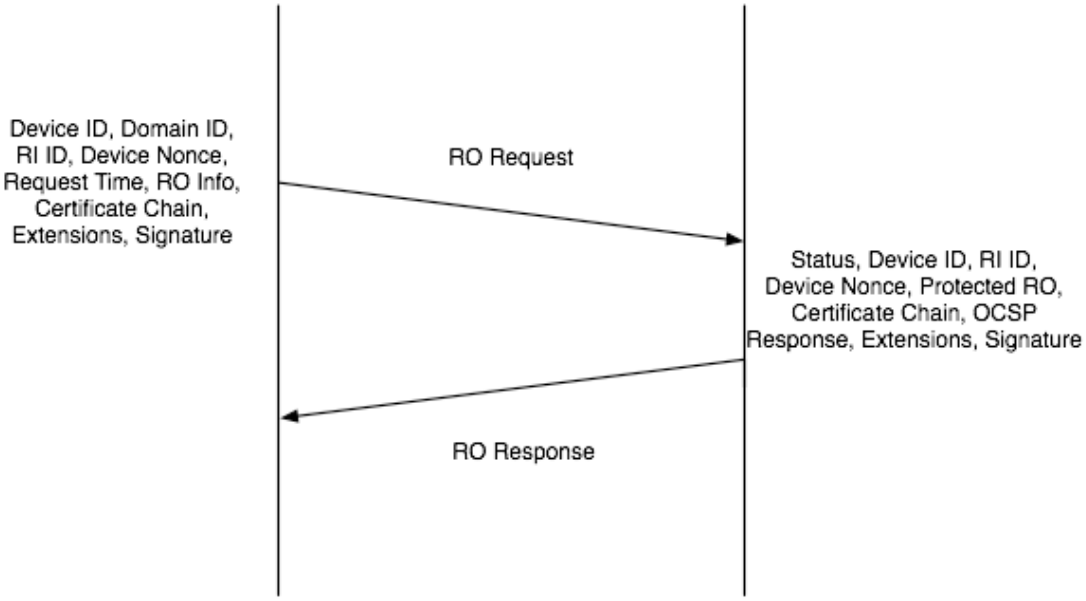

Figure II.1.4: ROAP – Acquisition Protocol

The message exchange for this protocol is simpler than for the Registration protocol as this is a 2-pass protocol (Figure II.1.4). There is also a 1-pass RO Acquisition Protocol defined by OMA [OMA, 2006b], which is initiated by a ROAP Trigger.

The first message to be exchanged in the protocol is the RO Request. This message contains a list of parameters, some mandatory and other optional. Device ID identifies the requesting device. The Domain ID, if present, identifies the domain for which the requested RO shall be used. The RI ID identifies the authorising RI. The Device nonce is a nonce chosen by the device. The Request Time is the current time as seen in the device. The RO Info contains the identification of the request RO. The Certificate Chain corresponds to the certificates held by the device. This message has also a set of extensions that can be used, and are composed by the Peer Key Identifier, an identifier for an RI public key stored in the device, No OCSP Response, indicating that there will be no need for an OCSP response validation, the OCSP Responder Key Identifier, which identifies an OCSP responder key stored in the device, and the Transaction Identifier, which allows a device to provide the RI with information for tracking of transactions. Finally, the Signature field corresponds to the signature of the entire message [Pimenta and Serrão, 2006].

After the RO Request has been sent to the RI, a second message is returned from the RI back to the device: the RO Response. This message contains the Status, indicating whether the

request was successful or not, the Device ID, identifying the device, the RI ID, identifying the RI, the Device Nonce, containing the same value that was passed in the previous message, Protected RO(s), are the rights objects in which the sensitive information is ciphered, Certificate Chain, the OCSP Response, if it was requested in the previous message, it will contain a set of valid OCSP responses for the RI's certificate chain, on Extension named Transaction Identifier, which can be used to track the operations. Finally, the Signature field is the signature of the entire message [OMA, 2006b].

2.2.3 Other protocols

Apart from the protocols for registering and acquiring, there are other DRM protocols inside OMA that are briefly described here. They are the Join and Leave Domain protocols.

The 2-pass Join Domain protocol is the protocol by which a Device joins a Domain. The protocol assumes an existing RI Context with the RI administering the Domain. Successful completion of the Join Domain protocol results in the establishment of a Domain Context in the Device containing Domain-specific security related information including a Domain Key. A Domain Context is necessary for the Device to be able to install and utilise Domain ROs [OMA, 2006b].

The 2-pass Leave Domain protocol is the protocol by which a Device leaves a Domain. The protocol assumes an existing RI Context with the RI administering the Domain.

3 DReaM

Project DReaM [SunLabs, 2007] is a Sun Microsystem initiative to develop a DRM solution focusing on open-standards. DReaM is based on Opera, a former DRM interoperability specification and implementation in the Opera Eurescom project [Wegner, 2003]. According to DReaM own information, whenever the market requires proprietary solutions DReaM will be capable of integrating with these solutions providing openness and interoperability that meets customer requirements [SunLabs, 2007]. DReaM is an initiative to leverage the methodology of Service Oriented Architectures (SOA) and introduce rights management services that leverage open standards and support cross-service capabilities [Fernando et al.,

2005].

The DReaM architecture supports the separation between the rights management components through the de-coupling of authentication, licensing, rights management and protection systems [SunLabs, 2007]. This disintermediation enables the choice and selection of these technologies independent of each other without any compromise for the overall solution. There are two key elements for disintermediation in DReaM:

- Separation of rights management from the content protection systems;
- Separation of identity and authentication services from individual hardware devices.

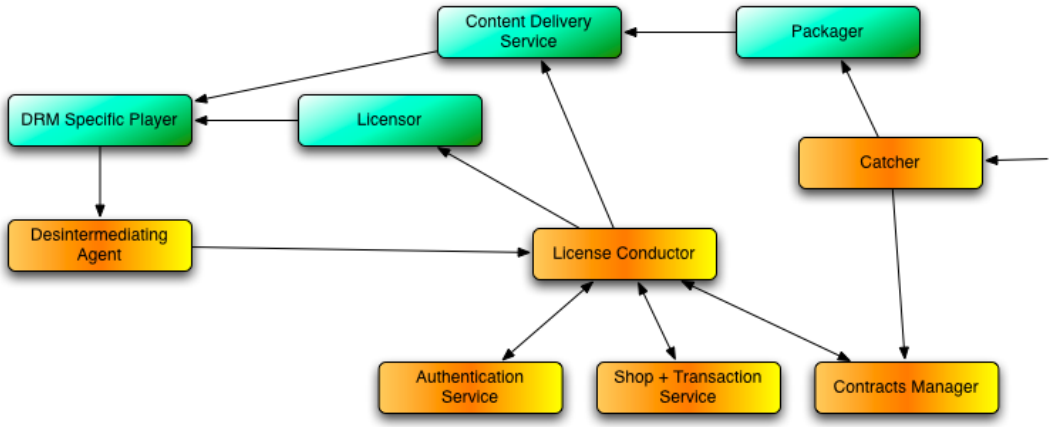

Figure II.1.5: The DReaM DRM disintermediation system

DReaM (Figure II.1.5) has a central objective towards the creation of an interoperable DRM, offering the capability to interoperate directly with other content protection technologies and supporting services that enable both Conditional Access System (CAS) and DRM [SunLabs, 2007]. A key-concept in the DReaM platform is the disintermediation concept – this enables multiple instances of these components to exist in a DRM/CAS system. The DReaM disintermediation system enables the coexistence of multiple instances of content protection specific components (player, licensor and packager) and components that are not content protection specific (disintermediation agent, conductor, catcher, licensing conductor, contracts manager, authentication service, shop and transaction system and content delivery system) [Serrão et al., 2006e].

The process of disintermediation happens as follows:

1. Client requests a license;

2. Front-end service redirects client to a client disintermediation agent;
3. Disintermediating agent contacts Conductor (back-end service);
4. Conductor contacts back-end services for authentication and rights verification:
5. Conductor signals front-end service with instructions to deliver license to client;
6. Front-end service delivers license.

Although DReaM's objective is to offer the capability to interoperate directly with other content protection technologies, it has developed a new method of expressing and controlling rights for content, which is termed DreaM-MMI (Mother-May-I) [SunLabs, 2006].

DReaM has produced for the moment the DReaM-CAS and DReaM-MMI specifications [SunLabs, 2006], which are available upon request.

According to Sun, the design philosophy underlying DReaM-MMI is that clients should be able to negotiate for rights through standardised protocols rather than downloading a license with an embedded expression of rights [SunLabs, 2006]. Access to content is requested under certain conditions, and the client software manages the use, according to the guidelines under which the content is requested. It is done in the following manner:

- A DReaM-MMI compliant client will request the use of given protected content under a specific set of usage terms (e.g. number of viewings, etc.);
- The DReaM-Licensor responds to the client's DReaM-MMI request after communicating with the DReaM Contracts Manager to determine whether the client should be allowed access to the content on those DReaM-MMI expressed terms;
- If the Licensor response is positive, the content keys are delivered to the client where it will be consumed according to the terms expressed in the DReaM-MMI request;
- The DReaM-MMI compliant client has the responsibility for enforcing that the content is only used under those specified terms.

If a client wishes to access content under different usage terms, the client could renegotiate with the DReaM-Licensor. No more access is allowed than the specific rights the client had requested.

An important issue in DreaM-MMI is that no Rights Expression Language is delivered to the client. This approach for the rights management can be seen as a Sun attempt to avoid the patent issue regarding Rights Expression Languages [Serrão et al., 2006e].

DReaM defines a set of Actors and Components in its architecture [SunLabs, 2007]. The most relevant are the following:

- Client - DRM Specific Player: It is a client-side player application that has DRM specific support for handling protected content and licenses;
- Client - Disintermediating Agent: It is a Java application that would perform the re-direction required for disintermediation;
- Licensor: The licensor is tightly bound to the DRM specific content protection technology;
- Licensing Conductor: It handles the role of managing the licensing processes involved in the DReaM solution. It has interfaces to the DReaM Client, Shopping and Transaction Service, Authentication Service, Contracts Manager and the Licensor. It performs the necessary e-commerce transactions and authentication of the user. It instructs the Licensor to generate the license for a given user for specific content;
- Contracts Manager: It stores business rules associated with content, as well as user rights. This component has interfaces to the Licensing Conductor and the Licensor. The Licensor will generate a license for a given piece of content based on the business rules and user rights that are available in the Contracts Manager;
- Authentication Service: It is where subscribers, users and devices are cleared for access to services and content. The methods of authentication vary from weak methods such as username and password challenge to stronger authentications such as smart cards or biometrics;
- Shop and Transaction Service (Business Support Services): The work flow functions of shopping and transacting purchases includes everything from collecting payments from buyers to paying sellers and making sure that everyone is appropriately compensated in a secure manner;

- Content Delivery Server: The content distribution server will receive protected content from the packager. Stream keys used for content protection in the packager may be optionally stored with the content in the content delivery server;
- Packager: The packaging process involves combining content data/files with associated metadata and creating logical packages that include the defined business rules. DRM packaging applications may have user interfaces for the human processing of content or the rights may be machine processed from business rules that are made available at the time of content ingestion. These business rules may be stored in a content management system (CMS), and the DRM packager would then read them through database queries;
- Catcher: The Catcher performs content ingestion. It receives content and associated business rules from the content supplier. The content, which is unprotected at this stage, is passed to the Packager. The business rules associated with the content are passed to the Contracts Manager.

4 Marlin

The aim of Marlin [Marlin, 2006a] is to create a DRM system that interoperates among devices from different vendors. It is based on previous work developed by Intertrust[6] in NEMO (Networked Environment for Media Orchestration) [Marlin, 2006b] and Octopus [Marlin, 2006c] projects. Whereas the first is a secure messaging architecture based on web services for digital media distribution and rights management, the latter is a software toolkit for developing lightweight DRM systems based on elementary graph theory.

In Octopus (Figure II.1.6), there are nodes for entities in a DRM scheme that represent users, domains, devices and subscriptions (licenses) [Marlin, 2006c].

A device has rights over a content governed by a subscription if there is a series of links that connect the device to a subscription through a user. Those links are created by e-commerce systems, which are Marlin compliant [Marlin, 2006a].

To determine if a user has rights over the content, there must be a series of links that

6 http://www.intertrust.com

connect the user to a subscription. The subscription node points to the content object, which contains on one hand a control program written in a byte-code language called Plankton and the keys used to decrypt the content. In this way, when a user wants to exercise a right over a content, the Marlin client will run the control program associated to that content to determine whether he is authorised or not by checking if there are existing links from the client node to the user's identity [Marlin, 2006a].

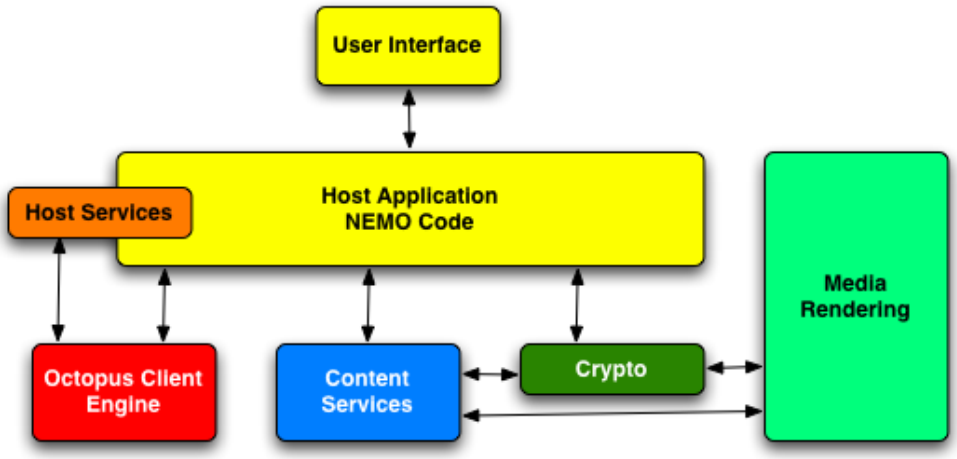

Figure II.1.6: Octopus DRM Client System Elements

One important aspect of Marlin is that it avoids the usage of Rights Expressions Languages (RELs) by avoiding a descriptive grammar, and thus, the patent issue regarding Rights Expression Languages [Marlin, 2006c].

Marlin includes an OMA Gateway, which enables Marlin Clients to act as OMA DRM Agents. The combination of the Marlin OMA Gateway with a Marlin DRM Client will satisfy all requirements for an OMA DRM Agent, and therefore, can be considered to be an OMA DRM Agent in all respects. For this reason, OMA content can be received, processed, and used as it would on any other OMA DRM compliant device. This means that no modifications are required on the part of OMA Rights Issuers to interact with "Marlin-based" OMA DRM agents. For example, Rights issuers do not need to add additional permissions nor would they have to modify the protocols used to communicate with Marlin-based DRM Agents [Marlin, 2006a].

Marlin defines several Actors in his architecture:

- User: A user is an individual that interacts with Service Providers to acquire licenses

for digital content, and interacts with Marlin Clients to access or manage use of that content. In most instances, users are also “License Owners”, that is, they have purchased the required rights to use the content under given circumstances. In some use cases however, such as in “sharing” or “super-distribution” cases, the user does not have a license to use the content and must interact with a Service Provider or a valid License Owner to gain access to the content;

- Service Provider: A Service Provider is the generic term used to describe the entity or organisation that is responsible for selling or distributing digital media content and associated licenses. Service Providers are not limited to any particular type of business model for licensing or distributing content. They can choose to support “a-la-carte” individual content licenses, rentals, subscription-based services, or a hybrid of these;

- Content Provider: A Service Provider may serve the role of a Content Provider and aggregate content from Content Owners and distribute it to Users. However, Marlin is flexible and allows for a variety of mechanisms for content delivery. For example, content may be delivered in peer- to-peer fashion or from a Broadcaster via a broadcast channel.

Marlin defines several modular components that are designed to serve a particular purpose or role within the system [Marlin, 2006a]. Its main components are the following:

- Marlin Client: It is responsible for requesting licenses and links, and controlling access to protected content. A Marlin Client may be implemented in a hardware device (such as a portable media player) or as a client application (such as a PC software media player application). When the host device (or player application) requests access to content, the Marlin Client will execute the control program in the license and check for the presence of any required links. Then, if permitted by the license, the Marlin client will allow the content key to be decrypted and used to access the protected content;

- Domain Manager: It serves the function of creating a domain and managing the devices and users that are associated with the domain. To do this, a domain manager

issues Device-Domain Link Objects that associate devices to a domain, and Domain-User Link Objects that associate the domain to users that 'own' the domain. A domain manager can either be operated by a Service Provider at a remote location accessible via the Internet, or can be implemented on a Marlin device in the user's local network [Kamperman et al., 2007]. The 'rules' that a Domain Manager must use to administer the domain are defined via a Domain Policy;

- Registration Service: It is typically operated by a Service Provider, and serves the role of issuing Nodes and Links. During the purchase process (or a subscription renewal process), the Marlin Client receives a trigger to contact a Registration Server to request the appropriate nodes and/or links to support the transaction. For example, when purchasing content a Marlin Client would be instructed to request a User Node corresponding to the user, and a link from the Client to that User. Alternatively, in a subscription renewal, the Marlin client might be automatically triggered to request a new User- Subscription link from the Registration Server (e.g., one with an extended expiration date);

- License Service: A Service Provider also operates a License Service that is responsible for issuing licenses to Marlin Clients. After a service provider's e-commerce system processes a transaction for a purchase, the ecommerce system will trigger the Marlin Client to contact a License Service with a request for a license. The License Service will generate the necessary license objects and respond with a "License Bundle" that the Marlin Client will use to govern access to the content.

Additional functional components are also defined in Marlin. These other components serve functions such as ensuring that Clients have the most current versions of security metadata, trusted time values, and others [Marlin, 2006a].

5 AXMEDIS

One of the major aims of the European Project called AXMEDIS (Automating Production of Cross Media Content for Multi-channel Distribution) [AXMEDIS, 2007] is to create and exploit an innovative technological framework for automatic production and distribution of cross-

media contents over a number of different distribution channels (e.g., networked PC, PDA, kiosk, mobile phone, i-TV, etc) with DRM.

For that purpose, AXMEDIS has designed and implemented a DRM architecture (Figure II.1.7) that consists on several independent modules that interact as web services when they are located in different machines or directly in other situations [Torres et al., 2006]. The general description of the AXMEDIS architecture main modules is as follows:

- Protection Processor: This client tool module is responsible for estimating the client tool fingerprint, enabling or disabling the tool, verifying the tool integrity and unprotect protected multimedia objects;
- Protection Manager Support Client (PMS Client): This client tool module manages and stores protection information, licenses, reports regarding the offline performed actions and other secured information in a local secure storage system called secure cache. It is responsible for authorising users to perform actions over objects with respect to digital licenses during offline operation. It also delivers protection information to the protection processor, if present in the secure cache, or requests it to the AXCS (AXMEDIS Certifier and Supervisor) after a positive authorisation. It acts also as the intermediary module used by Protection Processor to contact AXCS to certify and verify tools;

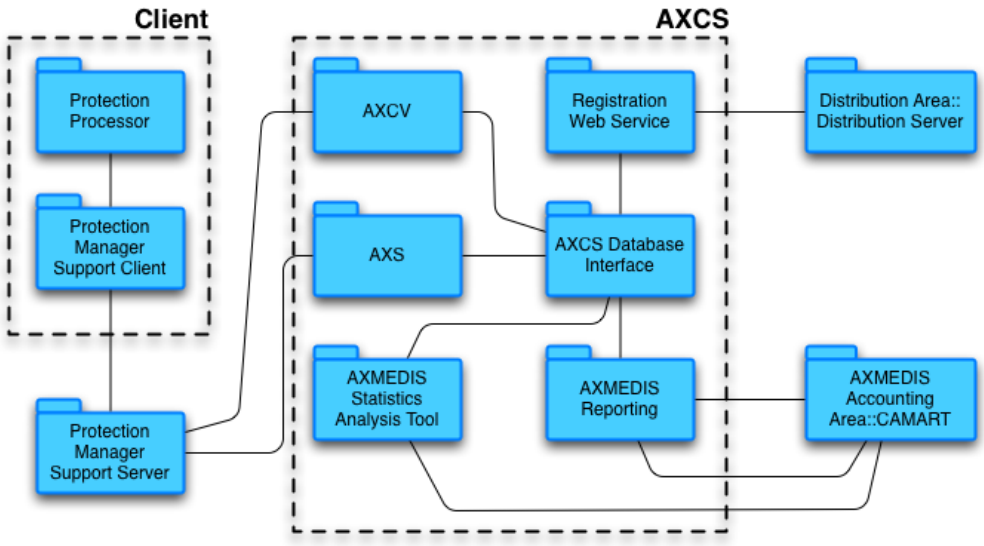

Figure II.1.7: AXMEDIS architecture: protection, rights management and accounting functionalities

- Protection Manager Support Server (PMS Server): The functionality provided by this module is the following: 1) creation and storage of rights expressions, 2) adaptation of rights expressions, including translation, 3) authorisation of content usage based on the licenses owned by the user, 4) requesting protection information to the AXCS if needed. Not all the functionality is available for all types of users. For instance, only content creators, content providers and distributors are allowed to create rights expressions. Nevertheless, the availability of creating rights expressions will depend on the rights a user has over the content. That means that a user with a distributor role can create licenses for final user if he has a distribution right over the content to be licensed [Torres et al., 2006]. Initially, MPEG-21 REL [MPEG-21-REL, 2004] and its authorisation model are provided for creation of rights expressions and authorisation of actions over content. Then, adaptation of rights expressions will be implemented, including translation to e.g. OMA DRM REL [OMA, 2006d];

- AXMEDIS Certifier and Supervisor (AXCS): AXCS is the authority in charge of user and tool registration (Registration Web service), user and tool certification (AXMEDIS Certification and Verification, AXCV), user and tool management (e.g. status supervision, automatic blocking, deadline supervision, etc.), user and tool unique identifier generation and object metadata collection. AXCS is also responsible for saving the Protection Information related to protected multimedia objects as well as the actions performed on them (AXMEDIS Supervisor, AXS), the so-called Action Logs. Action Logs are the particular implementation of MPEG-21 Event Reports in the AXMEDIS context [Rodriguez and Delgado, 2007a]. AXCS also includes a user Registration service, useful for registering new users in the system from distribution servers. All these data are stored in the AXCS database, which is accessed though the AXCS database interface module in order to keep the access independent from its implementation. Other functionalities provided by AXCS are those related to reporting and statistical analysis, which are performed by the Core Accounting Manager and Reporting Tool (CAMART module) by analysing the information stored in the AXCS database and collected in Action Logs. The integral modules of AXCS have been developed as web services or libraries [Torres et al., 2006].

Next, the most important aspects that are taken into account in the AXMEDIS architecture [Delgado et al., 2006] are summarised:

- Security and Trust on User Tools and Communication to Server: As it was previously mentioned, in the client side there are different modules as Protection Processor and PMS Client which are devised to communicate with the server part by providing not only security to the transactions but also trust from the server side perspective;

- Registration of Users: All the users in the system must register, which enables their interaction with the system and system tools. User information is stored in the server side (AXCS) and is used for further verification purposes. After the user registration, the corresponding AXCS issues a user certificate that will be used to authenticate the user when performing some specific operations as the certification of tools. Every AXMEDIS user has associated a status that is used to determine whether the user is blocked or not in the system when interacting with the server part. The user status can be modified by the AXCS if some critical operation attempt is detected;

- Registration of Tools: Tools using AXMEDIS framework must be verified to accomplish a series of guidelines, which are checked before registration is done. Once verified, each tool is registered for being used by AXMEDIS users. During registration phase, a fingerprint of the software tool is estimated so that its integrity can be checked later when the tool is installed and certified or verified on a specific device;

- Certification of Tools: The certification of a tool that uses AXMEDIS framework is a necessary step for that tool to work. Before a user is able to run and use a tool, the tool must connect to the AXCS to be certified as an “installed tool”. Before installation, AXCS verifies the tool integrity by comparing its fingerprint to the one stored during the tool registration process and, once installed, extracts some information (tool fingerprint) concerning the installation of the tool and the device where it is installed. A malicious user who tries to certify a tool whose fingerprint does not match the original registered tool fingerprint would be automatically blocked in the system so that he cannot continue performing other operations within the system. Moreover the tool would not be certified and thus it would not be operative. Once a user successfully certifies a tool, any user of the system who owns

a valid AXMEDIS user certificate can use it. Blocked users cannot use tools in the system. To perform the certification of a tool, the tool connects to the AXCS via PMS Client and PMS Server web service. In order to have a secure communication, the client certificate is used to authenticate the user against the PMS Server;

- Verification of Tools: Verification of tools is devised to cover two functionalities. First, it provides a means to ensure that client tools have neither been manipulated nor corrupted. Moreover, verification is used to resynchronise all the actions performed by users during off-line operation that were stored in the local secure cache. Verification of tools is performed periodically by the Protection Processor and every time the user tool resynchronises the off-line performed actions with the server part. It consists on the verification of the estimated tool fingerprint in the moment of the verification against the tool fingerprint stored in AXCS database during the certification of the installed tool. Regarding the tool integrity verification, if AXCS detects that critical parts of the tool or the device have been manipulated, it can adopt the pertinent measures as, for example, blocking the specific installed tool for which the verification failed. Regarding the resynchronisation of off-line performed operations, AXCS executes an algorithm to determine whether the received list of operations, which are called Action Logs in the AXMEDIS context, is complete with respect to the previous received operations. This integrity check is feasible thanks to the calculation of a fingerprint on the performed Action Logs, which is computed by PMS Client during the tool operation. This fingerprint is sent to AXCV when resynchronising the offline Action Logs and is verified by AXS.

6 Windows Media Rights Manager

Microsoft embraces DRM as the process of protecting digital media controlling its usage by licenses, which confer specific rights to the end-user [Microsoft, 2004a]. The digital media is encrypted using a key that blocks/unblocks the usage of the latter. After the encryption process, the media can only be played using a license (issued by a License provider) containing the key that unlocks it. The Windows Media DRM (WMDRM) is composed by the following components [Microsoft, 2004b]:

- Windows Media Rights Manager (WMRM) SDK for packaging content and issuing licenses;
- Windows Media Format SDK (WMF SDK) for building Windows applications which support DRM and the Windows Media format;
- Windows Media DRM for Portable Devices (WMDRM-PD) for supporting offline playback on portable devices;
- Windows Media DRM for Network Devices (WMDRM-ND) for streaming protected content to devices attached to a home network.

The Microsoft's DRM process can be represented in the following figure (Figure II.1.8).

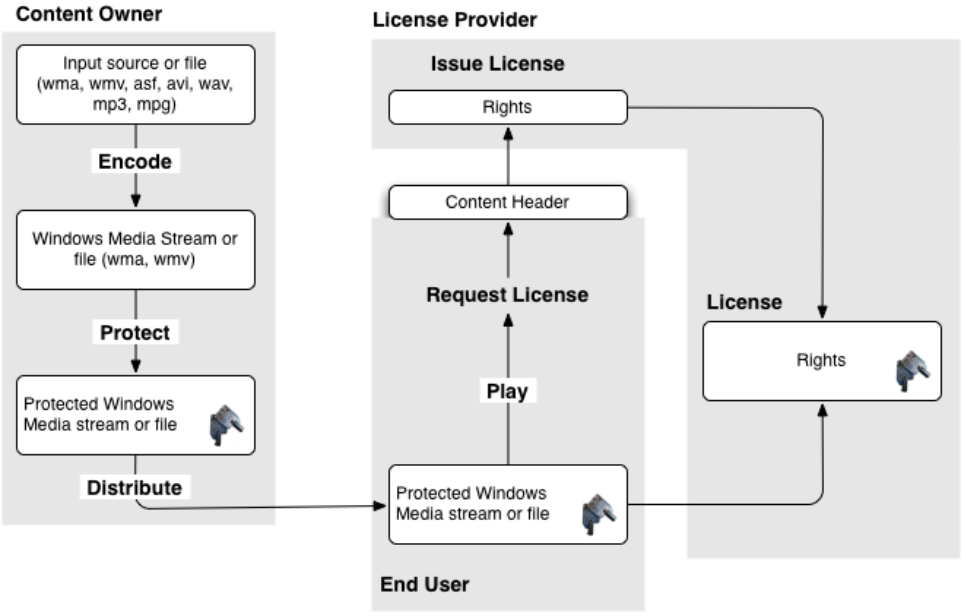

Figure II.1.8: The Microsoft DRM Process

The Windows Media Rights Manager has the following characteristics:

- Persistent Protection: Windows Media Rights Manager "locks" digital media files with a license key to maintain content protection, even if these files are widely distributed. Each license is uniquely assigned to each computer. This prevents illegal distribution of digital media files;
- Strong Encryption: Windows Media Rights Manager includes proven encryption schemes that ensure distributed digital media files are not exposed to piracy or other

illegal use;

- Individualisation: Rights Manager makes each player unique by linking the player to the host computer. This prevents a compromised player from being widely distributed over the Internet. With individualisation, any compromised player can be identified and disabled during the licensing process;
- Secure Audio Path: Rights Manager ensures content protection in the operating system from the player to the sound card driver in Microsoft Windows. This secure relationship reduces the likelihood that any unauthorised program will capture a digital media stream within a PC;
- Improved Revocation and Renewability: Windows Media Rights Manager enables the revocation of compromised players when new players become available;
- Secure End-to-End Streaming and Downloads: Digital media files are protected during download and on the consumer's PC through secure cryptographic protocols.

The most recent version of Windows Media (WMDRM 10) includes a new DRM functionality (code named “Janus”) such as DRM for portable and network devices, supporting synchronisation between any devices that Windows treats as a hard disk [Microsoft, 2004a].

The main key features of this new version are:

- WMDRM 10 SDK: Microsoft Windows Media 10 DRM is an SDK entirely written in ANSI C, enabling mobile device producers and developers to write Windows Media DRM dependent applications without having a Windows OS installed in the target machine. According to the capabilities of the target device, the developers use a subset of the Windows Media DRM. However, the OS running on the device is required to support a set of operations required by Windows Media;
- Modes of operation: The WMDRM 10 has two basic modes of operation:
    - Direct License acquisition – Under this mode, the portable device is capable of communicating directly with a license server via the Internet to obtain required licenses for content playback;

    - Indirect License acquisition – When using the mode, the portable device must be docked and connected to a PC in order to obtain licenses.

- License Chaining: The License Chaining structure is a new feature of Windows Media 10, which is used to model and define subscription rights. With this system, users can obtain a "root" license and install it on the device. This license represents basic usage rights for a subscription service, but does not grant any rights to a specific content item. When a user downloads a specific content, a "leaf" license is created and attached to the "root" license, where the latter defines all the usage rights. This system improves substantially interoperability with mobile devices, while satisfying content owners about security;

- Playback counting: The system will provide playback counting for a particular content, although users or devices are never identified;

- Secure clock: To use content with time-bound licenses, devices must synchronise a secure clock with a time provider over the Internet or on a computer. This prevents roll-back attacks on time-bound content, such as subscription services;

- Automated License Store Garbage Collector: With this feature, WMDRM 10 provides automatic clean up of expired licenses, in order to optimise storage space.

The different steps of the Microsoft DRM process [Microsoft, 2004b] are described next.

6.1 Coding/protection and content distribution

The content owner starts by encoding the content (e.g. a song recorded in MP3 format [Brandenburg, 1999]) into a Windows media stream or file such as WMA or WMV. The encoding is achieved using Windows Media Encoder, where the latter contains also the technology to protect the content, allowing coding and ciphering in one single step. In order to use Windows Media Encoder to protect contents, the content owner must own a DRM profile obtained at a License Provider [Liu et al., 2003]. This profile allows the content owner to generate keys in order to protect his media, as well as define the terms of usage. Once protected, the content can be distributed in the usual distribution channels.

During the protection process, the Windows Media Encoder generates the key (using the DRM profile) [Prunela, 2001], performs ciphering operations and adds DRM specific information to the content. The key is generated using a specific algorithm, which uses a license key seed in conjunction with a key id.

The license key seed is a value known only by the content owner and by the license authority. Both have to know and share this value in order to protect content. This value is generated when a new DRM profile is being created in a License server.

The “key ID” is a value generated by the content owner and it’s included in the header of the protected file/stream. This is an essential value for the License provider because it uses this value to re-generate the corresponding key [Liu et al., 2003].

In order to protect content using Windows Media Encoder, the content owner must choose the correct DRM profile in order to protect the content and a key ID is automatically generated by Windows Media Encoder [Microsoft, 2004a].

The content owner must also provide the following information on the content header:

- The URL of the license provider;
- A unique content ID. This is a value that can be used to identify the content in a database and related information, such as artist name, date of recording, among others;
- Attributes in the style “name-value” defined by the content owner (one can include the ciphering date of the content);
- The version of the individualised DRM component. The DRM component is the sub-module responsible for handling all the DRM functions in Windows Media Player. When a user installs a new player, the DRM component is individualised to that user, i.e., this component is unique and machine specific. This way, if a hacker cracked the DRM component would only crack that one, and not all.

6.2 Header signing

As an extra security measure, a public/private key pair is used to digitally sign the header of

the protected content, in order to ensure that the header and its information are legitimate [Microsoft, 2004a]. For example, a hacker could change the URL of License provider to a site where malicious scripts would be running. In this way, Windows Media player verifies, using the public key, if the header is legitimate or not. If the signature could not be verified, the end user would be warned about it. The end-user then chooses whether he wants to proceed with the license acquiring process [Microsoft, 2004a].

6.3 Playing protected content

In order to play protected contents, the end-user must own a player compatible with Microsoft's DRM technology, such as Windows Media Player [Microsoft, 2004b]. When the end-user tries to play the protected content, Windows Media Player starts by searching in the user's computer for a valid license for that content. If the latter is not found, the player analyses the content header trying to find the URL of the responsible License Provider. When this is found, the key ID is retrieved from the header and combining it with the license key seed (found at the License provider), the required key to unlock the content is re-generated. A new license is then produced including the key required for playing [Microsoft, 2004a].

According to the license type, the end-user may perceive the licensing process or not. For example, if the content is for promotional purposes, a license can be issued without any need for registry, being all this operation transparent for the user. This process is called Silent registration.

On the other hand, if the content is to be paid, a mini-browser is presented to the end-user showing the licensing conditions and a form for payment information.

Once the License is obtained, the end-user can use the content according to what was defined by the content owner.

6.4 License issuing/generation

The process used in the license issuance and license generation involves two different stages.

First, when a user wishes to playback a protected content for which he does not own a valid license, a license request is automatically performed to the respective License Provider. The

location of the license provider can be found at the content's header. The License Provider can then identify the content and issue the according license.

The second is the license generation. From the point of view of the content owner, the latter must first choose a License Provider and next create a DRM profile. The creation of the DRM profile requires personal information as well as the rights to be included in the licenses to be issued. Once the DRM profile is created, it is imported to the system that is responsible for coding and ciphering [Microsoft, 2004a].

6.5 How licenses work

Each license contains the key to unlock the Windows Media file [Microsoft, 2004a]. The license also contains the rights, or rules, that govern the use of the digital media file. The content owner sets these rights to determine which actions are allowed from minimal control over playback to more restrictive licenses [Microsoft, 2004b]. The licenses in Windows Media Rights Manager can support a wide range of different business rules, including:

- How many times a file can be played;
- Which devices a file can be played or transferred on. For example, rights can specify if consumers can transfer the file to portable devices that are compliant with the Secure Digital Music Initiative (SDMI);
- When the user can start playing the file and what is the expiration date;
- If the file can be transferred to a CD recorder (burner);
- If the user can back up and restore the license;
- Which security level is required on the client to play the Windows Media file;
- And others.

Licenses can be delivered in different ways and at different times, depending on the business model. The content owner might want licenses pre-delivered, or they might want the license delivered after a consumer has downloaded and attempted to play a packaged file for the

first time. Licenses can be delivered with or without the consumer being aware of the process using silent or non-silent license delivery [Liu et al., 2003].

7 Apple FairPlay

Apple FairPlay is the DRM system used in iTunes [Lenzi et al., 2003], a music distribution service from Apple directed mainly to Mac and Windows users.

Apple iTunes is the application that allows the user to access the iTunes Music Store, an on-line store with a large selection of digital content (music, movies, audiobooks and other). Due to the direct integration, the iTunes Music Store becomes a part of the iTunes desktop software which allows users to search and browse the available content, and to acquire that content upon registration on the store [Sharpe and Arewa, 2007].

Before starting content acquisition from the iTunes Store, a user has to create an account with the Apple's servers and then authorise a PC or a Mac running the iTunes software (Figure II.1.9). During the authorisation process, iTunes creates a globally unique ID number for the computer it is running on, and then sends it to Apple's servers, where it is assigned to the user's iTunes account [Venkataramu, 2007]. Five different machines can be authorised. Before authorising a new machine the user has to de-authorise other, if the maximum has been reached.

When a user acquires content from the iTunes Store, a user key is created for the purchased file. The content is scrambled using a separate master key, which is then included into the protected content file. The master key is locked using the user key, which is both held by iTunes and also sent to Apple’s servers.

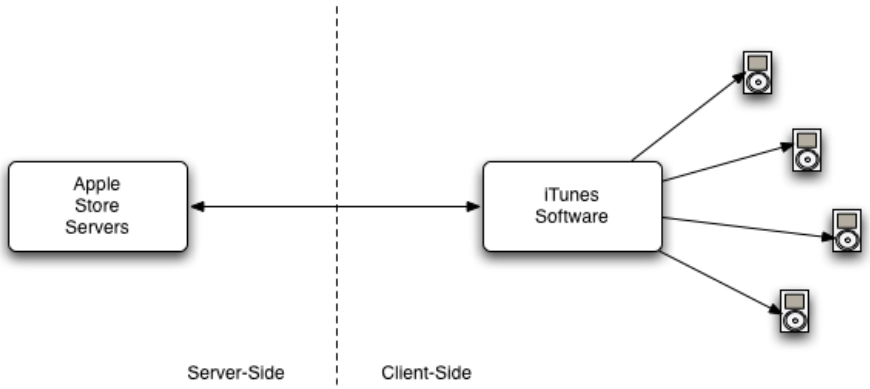

Figure II.1.9: Apple FairPlay system architecture

The protected, purchased content is locked within iTunes. The content is not scrambled on Apple's server. This speeds and simplifies the transaction by delegating that work to iTunes on the local computer. The result is an authorisation system that does not require iTunes to verify all the content with Apple as it plays. Instead, iTunes maintains a collection of user keys for all the purchased content in its library [Lenzi et al., 2003].

To play a protected content, iTunes uses the matching user key to unlock the master key stored within the content file, when is then used to unscramble the song data. Every time a new content is purchased, a new user key may be created. Those keys are all encrypted and stored on the authorised iTunes computer, and are also copied to Apple's servers. When a new computer is authorised, it also generates a globally unique ID number for itself and sends it to Apple, which stores it as one of the five authorisations in the user account [Lenzi et al., 2003].

Apple's server sends the newly authorised machine the entire set of user keys for all the content purchased under the account, so all authorised systems will be able to play all purchased content [Venkataramu, 2007].

8 DMAG MIPAMS

DMAG-MIPAMS is an architecture to manage multimedia information taking into account digital rights management (DRM) and protection [Delgado et al., 2005].

DMAG-MIPAMS is a service-oriented DRM platform and all its modules have been devised to be implemented using web services technology, which provides flexibility and enables an

easy deployment of the modules in a distributed environment, while keeping the functionality independent from the programming language and enabling interoperability [Delgado et al., 2005].

DMAG-MIPAMS (Figure II.1.10) encompasses an important part of the whole content value chain, from content creation and distribution to its consumption by final users.

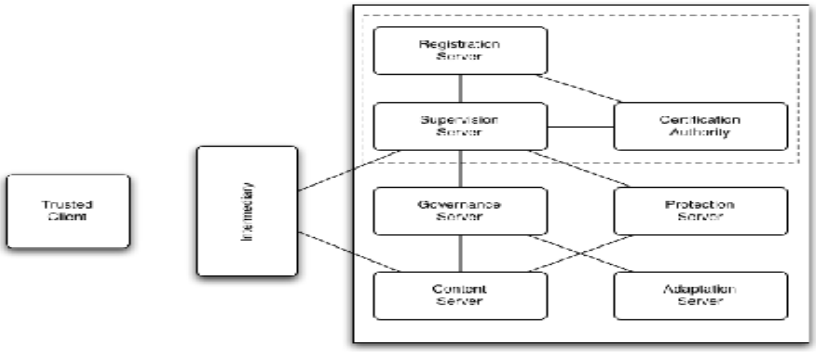

Figure II.1.10: DMAG-MIPAMS architecture

In the DMAG-MIPAMS architecture context, there are two important concepts: Components and Actors. An Actor is a user (person or an organisation) that uses a Component. A Component is module or set of software tools that offers a subset of DRM-related functionalities and which interacts with other components [Serrão et al., 2006e].

8.1 Content server

The Content Server offers the following functionalities: 1) enable users to browse/select content; 2) provide the content that final users may request to user applications; 3) encode and add metadata to received raw contents from providers; 4) register the digital items/objects describing resources (metadata) which can be stored independently from the resource itself (which could be stored e.g. in an external system). This functionality should be available for the Content Server itself as well as for client production tools. The generation of the encoded and protected objects with metadata is an operation that can be provided as a service in the server part [Torres et al., 2006]. However, it can be also done by means of specific client tools available for the production environment, which only register the content

into the server once it has been protected and once metadata has been included in the client tool.

8.2 Adaptation Server

The Adaptation Server performs the adaptation of contents and their associated metadata, depending on the transmission, storage and consumption constraints. It could be included in the content server as an integral part of it.

8.3 Protection Server

The Protection Server offers the following functionalities: 1) protect objects (for content server or client production tools); 2) protection tools description and download; 3) generate protection keys; 4) store Protection keys (optionally here). The protection of objects is an operation that can be provided as a service on the server side. However, it can be also done by means of specific client production tools available for the production environment, which protect the objects on the client side. Protection Server offers a service for protecting the content or digital objects, which become protected objects, mainly using encryption techniques and scrambling [Delgado et al., 2005]. The Protection Server creates and associates different data to the digital objects. First, the information related to the protection process that has been applied, and second, the protection techniques, keys or tools that have been used. In this way, it will be possible to determine the reverse process and unprotect the object, if the user is allowed to. The Protection Server also provides the functionality to download the tools for protection/unprotection in case a user does not have them available in the terminal or device. When client production tools are used for protecting the content, the client tool is responsible for the protection. However, it can contact the Protection Server to retrieve the list of available protection tools as well as their implementation [Torres et al., 2006]. The necessary keys used to protect the content and needed to unprotect it can be stored in this server, or they could be also stored in the Governance Server together with the corresponding license or even in the Content Server.

8.4 Governance Server

The Governance Server component includes the following functionalities: 1) generate licenses (end-user, distribution, etc.); 2) store licenses; 3) perform online license-based authorisation; 4) translate licenses. The license generation functionality is only available for content creators, content providers and distributors. In any case, any user that creates a license describing the use of some specific content must have the corresponding rights. The generated licenses are associated to a content or digital object (by means of the object ID), which becomes a governed digital object. Licenses are stored in a license database or repository for authorisation purposes, and delivered to the client when necessary [Torres et al., 2006]. The governance server offers authorisation functionality (in an online mode). This way it is responsible for authorising users to perform actions over resources. Every time a user tries to perform an action over a resource, this module checks in the license database if it has a license that authorises him to do so [Rodriguez and Delgado, 2007b]. As licenses can be written in different languages, as for instance ODRL or MPEG-21 REL, a translation functionality is necessary to provide interoperability. In this way, the system can work with a predefined language and convert licenses expressed in other languages to the predefined one under certain conditions. To facilitate this interoperability, the system can define "equivalent" profiles between different rights expression languages, to enable their translation in both directions [Delgado et al., 2005].

8.5 Certification Server

The Certification Server provides different functionalities. It is decomposed into: Certification Authority, Registration Server and Supervision Server. Any user must be registered in the system in order to be able to interact with it. Once a user is registered, a certificate can be optionally requested to the Certification Authority. The certificate can be used to authenticate him when necessary [Torres et al., 2006]. Any tool that interacts with the system must be trusted by the system. One way to ensure this consists on registering the tools in the Registration Server prior to its installation in user devices. Later, when a user installs and tries to use a tool for the first time, he will be forced to certify it, which consists on verifying its integrity (Supervision Server), registering it in the system as an installed tool

together with the device where it has been installed (Supervisor Server) and issuing a specific tool certificate (Certification Authority) which can be used to establish secure communications with the server part. Any tool in the user side must be trusted during its lifetime [Erickson, 2003]. One way to ensure this consists in periodically perform some checks against the Supervision Server to verify its integrity. Every time a user tries to perform an action over a protected and governed object, the client side module sends the necessary information to the server side so that it can verify not only the user status and the device where the tool is being run, but also the tool integrity to ensure that the module has neither been modified nor corrupted. By ensuring the tool integrity, it is possible to assure that it is still trusted from the system point of view. Any module of the system can issue some reports about specific events upon request or in a predefined manner. The Supervision Server receives these reporting messages that include information about different aspects of media usage. The information it collects is stored in a database. Then, it can be used by specific applications to keep track of what happens in the system by generating statistics and traces. For example, in a scenario where a content distributor and a client reach the agreement of monthly billing, it can provide the distributor the list of products that the user has consumed for billing purposes. With this information, it can also generate automatic reports to the authorised parties. Supervision Server needs to accept the reports generated during online operation as well as during off-line operation [Torres et al., 2006].

For the latter case, client tools need to provide a mechanism to securely store them locally and the Supervison Server needs to provide a means for checking the list integrity.

8.6 Certification Authority

The Certification Authority issues the necessary X.509 certificates for the different Components and actors in the system. The main functions of the component are: 1) issue X.509 installed tool certificate; 2) issue X.509 user certificates for registered users; 3) issue X.509 component certificates for the different architecture servers.

8.7 Registration Server

The Registration Server is used to register actors and potentially installable tools. The result

of the actor's registration is a user certificate, whereas tools registration is performed to be able to verify them once installed on client devices. The main functions of the component are: 1) register tools; 2) register actors.

8.8 Supervision Server

The Supervision Server authenticates and supervises the actors and system components. It is responsible for extracting and registering a fingerprint for the installed tools so that they can be verified during their whole life operation and requesting the tool certificate to the Certification Authority [Rodriguez and Delgado, 2007b]. It also verifies the user tool integrity during its operation by checking its fingerprint, registered during certification. Moreover it receives the action reports regarding content consumption or other relevant issues in the system as e.g. license generation. The main functions of the component are: 1) authenticate users; 2) authenticate installed tools; 3) authenticate architecture components; 4) verify tool installation attempts against registered tools features; 5) register new installed client tools and tool and device fingerprints; 6) request installed tool certificate to the Certification Authority; 7) receive and store action reports [Torres et al., 2006].

8.9 Trusted client

The Trusted client is a module with which the client application must interact to enforce the DRM operations [Torres et al., 2006]. It consists of a trusted software module and a secure local repository for the storage of licenses, protection information, offline operations reports and other critical data. Main functionalities include: 1) estimate trusted client and tool features; 2) estimate device features; 3) perform offline authorisation; 4) unprotect content; 5) retrieve offline action reports; 6) store offline action reports; 7) store content protection information; 8) install unprotection tools; 9) license local storage; 10) tool certificate storage [Rodriguez and Delgado, 2007b].

8.10 Intermediary

It is usually an integral part of the trusted client but could also be located in the server part to simplify the number of components the trusted client needs to contact. It can be seen as a

broker to whom the trusted client requires the authorisation and the keys needed to unprotect the content [Torres et al., 2006]. Main functionalities include: 1) require certification to Supervision Server; 2) require verification to Supervision Server; 3) require online authorisation to Governance Server; 4) license download from Governance Server; 5) send offline operations to Supervision Server; 6) download unprotection tools from Protection Server. The trusted client could be implemented as a specific client tool or also as a plug-in for an already existing tool. In the latter case, the trusted client is responsible for interacting with the client tool for delivering the unprotected contents of it [Erickson, 2003].

8.11 Client Application

The client application is the player or edition tool, which needs to deal with the protected content. In the case of the player, it needs to get the unprotected content to be reproduced. In the case of a production tool, it needs to request authorisation before allowing the user act upon the objects as e.g. to modify an image or embed it in another object [Rodriguez and Delgado, 2007b].

9 OpenIPMP

The OpenIPMP[7] ("IPMP" stands for Intellectual Property Management and Protection) system [OpenIPMP, 2003] is a collection of tools/services designed to deliver a robust and scalable DRM solution, whose main objective is to support the management and secure the delivery of digital assets during their entire life cycle. The OpenIPMP system is open-source and uses open standards. It comprises a set of standards for audio and video CODECs, intellectual property management and protection (IPMP) and a new multimedia file format [OpenIPMP, 2003].

7 http://sourceforge.net/projects/openipmp

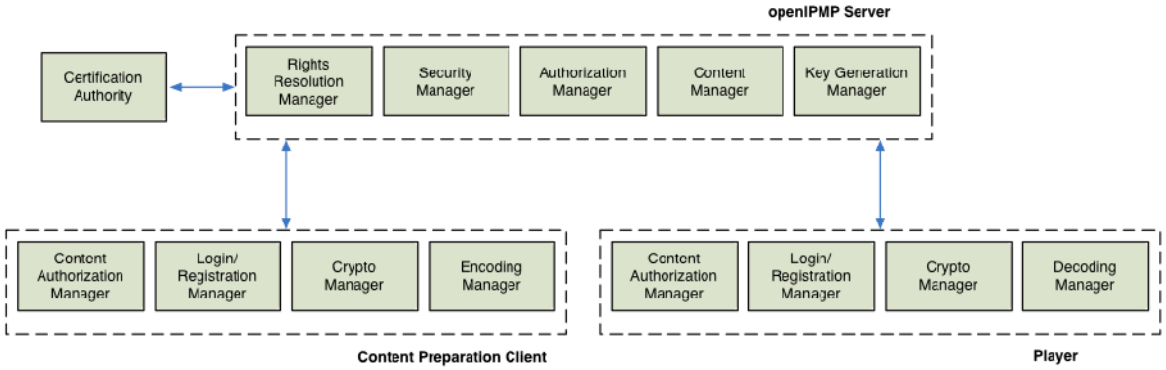

Figure II.1.11: OpenIPMP architecture

The OpenIPMP system (Figure II.1.11) comprises User management and identification, content encryption algorithms as well as distribution channel protection. OpenIPMP adheres to a set of open standards, including OMA-DRM v2, Internet Streaming Media Alliance (ISMA) encryption (ISMACrypt) [Lifshitz, 2003] and DRM signalling, and MPEG-2 and MPEG-4 IPMP specifications.

The following points present some of the key concepts of the OpenIPMP system [OpenIPMP, 2003]:

- User Management: The OpenIPMP user management infrastructure is the basilar stone of the system. In order to manage rights and enforce role based rules and permissions, the framework needs to be able to uniquely identify every user of the system. With this objective in mind, the system uses Digital Certificates issued by a Certificate Authority, which uniquely identifies each user, ensuring its identity. It also enables secure, confidential communications with the OpenIPMP server components helping to ensure that sensitive data is not compromised during transit;
- Content Identification and Management: The OpenIPMP system provides unique content Identification much to the similarity to what happens with physical media. The system implements one of the identification mechanisms proposed by the MPEG-21 standard. By uniquely identifying content, the system can provide protection and tracking mechanisms during the entire lifetime of the asset;
- Rights Management: Managing rights comprises the ability to create, modify or revoke usage rights for a given content in a networked environment. The OpenIPMP

system provides multiple usage rights ranging from simple to complex, while following the proposed format for rights definition (REL – Rights Expression Language) included in the MPEG-21 specification [Chiariglione et al., 2007].

Some more details about the technological aspects of the OpenIPMP platform are described on the following sections.

9.1 User Management

User Management is implemented by using Digital Object Identifiers (DOI) [Rosenblatt, 2002], which is considered to be an "actionable identifier" for identifying and managing intellectual property on the Internet. The DOI is one of MPEG-21 proposed mechanisms for user identification over networked environments and it is designed to work with past, present and future systems. The reason for being considered "actionable" is due to its ability to dynamically update meta-data information about a given digital asset [OpenIPMP, 2003] .

9.2 License Management

OpenIPMP supports the two leading XML-based Rights Expression Languages used in License definitions. A more detailed explanation about these is presented below:

- Open Digital Rights Language (ODRL) [ODRL, 2002]. Originally created by IPRSystems, it is based on a XML schema, allowing the definition of permissions, constraints, obligations, offers and agreements with rights holders;
- MPEG-REL. Originally created by ContentGuard, and latter adopted and modified by MPEG, the MPEG-REL is another Rights Expression Language enabling the definition of fine-grained usage rights to both digital content and services [MPEG-21-REL, 2004].

9.3 Cryptography

The OpenIPMP system uses some of the leading open standards for cryptography available [OpenIPMP, 2003]:

- Public Key Infrastructure (PKI). The system uses a PKI in order to ensure encryption

and digital signature for securing digital contents over networked environments. The PKI is a secure, extensible framework that allows for OpenIPMP installations to seamlessly work together;

- X.509 Certificates. Instead of a proprietary format, the X.509 format is used in the OpenIPMP to implement licenses. Signed by the OpenIPMP Certificate Authority service for authenticity, each digital certificate ensures each OpenIPMP transaction is certified cryptographically;
- Asymmetric Encryption. The system uses Asymmetric encryption algorithms such as RSA, to protect licences and other key sensitive data.
    - RSA keys. OpenIPMP supports configurable RSA key sizes allowing the client to determine the proper balance between security and speed. When a user registers with the OpenIPMP system he/she is given a pair of private/public keys. With this set, data encrypted with a user's Public Key can only be decrypted with that user's Private Key and vice versa.
- Symmetric encryption. The system uses Symmetric encryption algorithms in order to secure the digital assets. The two main algorithms employed are AES and BlowFish;
- Secure Sockets Layer (SSL). Designed and implemented by Netscape Corporation, the SSL network protocol ensures secure communication over the distribution channel. OpenIPMP utilises SSL for critical transactions such as License Acquisition and Content Registration ensuring that information cannot be tampered with during transit;
- Secure Storage. By using RSA Security's PKCS #12 file format, the OpenIPMP system ensures that all user-specific information (Private Key, User Certificate or Certificate Authority list) is stored securely.

9.4 Streaming

The OpenIPMP platform was also designed to support both playback methods supported by the MPEG-4 format [ISO/IEC14496-12, 2005]. These are the Local Playback (using the RTP –

Real Time Protocol) and the Streamed Playback (using the RTSP – Real Time Streaming Protocol).

9.5 Architecture/Infrastructure

The key components of the OpenIPMP DRM platform [OpenIPMP, 2003] are the following:

- Media Encoding tool. This is the tool used to protect content. It includes all the client side cryptographic algorithms required to provide persistent protection of the digital assets;
- Media Player tool. This is the tool used to playback OpenIPMP protected content. The tool also handles all the communication aspects with the server side components in order to acquire licenses for content playback and also enforces usage rights;
- User Registration service. This is the service used to register a new user in the system. Upon completion of the registration process, the user receives its credentials, which uniquely identify him/her and grant access to system resources;
- Content Management service. The content management service provides registered users with tools to manage their contents. Upon completing the content registration processes, assets are under the protection of the framework. All communications between the OpenIPMP client side components and the Content Management Service occur over a secure communication channel;
- Rights Authorisation service. With this service, registered users can define all the permissions for their registered contents;
- License Management service. This service is responsible for handling all the license requests from registered clients and deliver the respective authorisations;
- Administration tool. The OpenIPMP Administration Tool allows an administrator to configure the system according the specific needs of an organisation.

Chapter 2. DRM interoperability frameworks

1 Introduction

In this chapter, a list of DRM interoperability frameworks is provided. Some of these frameworks are not DRM systems by themselves (Coral [Coral, 2006a], MPEG-21 [Bormans et al., 2003]). Rather, these solutions claim to provide the necessary mechanisms to allow existing or new DRM solutions to interoperate among them.

The other solutions presented here are described has interoperability frameworks, but they also provide in itself a DRM solution. This is the case of the Digital Media Project (DMP) [Chen, 2007], which aims at the provision of a DRM interoperability platform, and at the same time it provides the open-source implementation of Chillout, a DRM platform.

The following sections of this chapter will start by providing a description of the MPEG-21 framework [Bormans and Hill, 2002], with particular emphasis on the parts more relevant to DRM, followed by a small presentation and description of the Coral Consortium interoperability framework and finally it will be presented the DMP platform for interoperability.

2 MPEG-21

MPEG-21 defines a normative open framework for multimedia delivery and consumption for use by all the players in the delivery and consumption chain. This open framework provides content creators, producers, distributors and service providers with equal opportunities in

the MPEG-21 enabled open market [Bormans and Hill, 2002]. This works also in the benefit of the content consumer providing them access to a large variety of content in an interoperable manner. MPEG-21 is based on two essential concepts: the definition of a fundamental unit of distribution and transaction (the Digital Item [Keukelaere et al., 2005]) and the concept of Users interacting with Digital Items [BUMETT et al., 2005].

MPEG-21 is a standard that defines a framework for dealing with the different aspects of multimedia information management (Figure II.2.1). The MPEG-21 standard [Bormans et al., 2003] normatively specifies different pieces and formats needed by a complete DRM system. These parts are MPEG-21 Digital Item Declaration (DID, Part 2) [MPEG-21-DID, 2005], that specify the model for a DI that is constituted by the digital content, referenced or embedded, plus related metadata that describes additional information regarding the content, i.e. protection, governance and processing information. MPEG-21 Rights Expression Language (REL, Part 5) [MPEG-21-REL, 2004] defines as a machine-readable language to declare rights and permissions using the terms as defined in the Rights Data Dictionary. MPEG-21 Rights Data Dictionary (RDD, Part 6) [MPEG-21-RDD, 2004] comprises a set of clear, consistent, structured, integrated and uniquely identified terms. The structure of the RDD is designed to provide a set of well-defined terms for use in rights expressions. MPEG-21 Intellectual Property Management and Protection Components (IPMP, Part 4) [MPEG-21-IPMP, 2006] deals with the standardisation of a general solution for the management and protection of Intellectual Property. Digital Items can be protected in order to ensure that the access to the contents is done according to the license terms. The solution lies in the use of digital signatures and encryption techniques over the digital content, which makes possible to deploy a business model that ensures the accomplishment of the license terms in a controlled way. MPEG-21 Event Reporting (ER, Part 15) [MPEG-21-ER, 2006] provides a standardised means for sharing information about Events amongst Peers and Users. Such Events are related to Digital Items and/or Peers that interact with them. The MPEG-21 standard has more parts, but these are the most relevant to DRM.

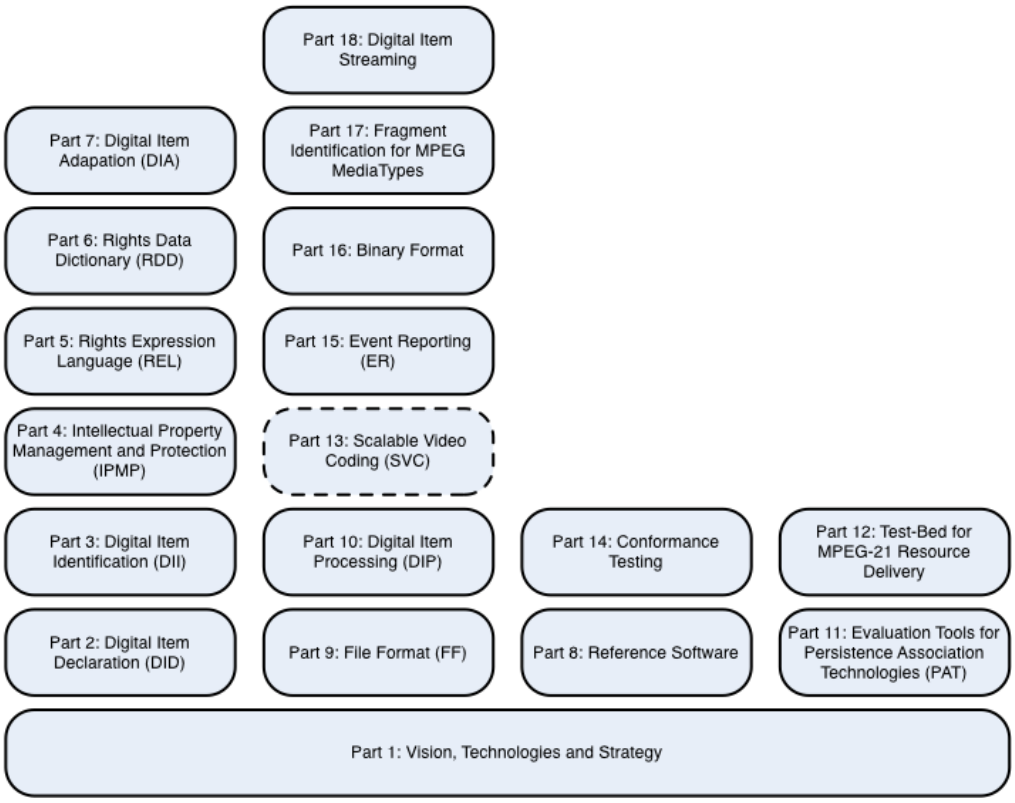

Figure II.2.1: The different parts that compose MPEG-21

2.1 MPEG-21 Rights Expression Language and Rights Data Dictionary

Parts 5 and 6 of MPEG-21 describe the Rights Expression Language (REL) [MPEG-21-REL, 2004] and Rights Data Dictionary (RDD) [MPEG-21-RDD, 2004], respectively. These are two of the most fundamental parts of the DRM based on MPEG-21 as they define the licenses, which include the rights that users have over Digital Items (DI). Digital Items (Part 2 of MPEG-21 standard) are the fundamental unit of distribution and transaction inside MPEG-21 [MPEG-21-DID, 2005].

Right Expression Languages (RELs) are languages devised to express conditions of use of digital content. They have been proposed to describe licenses governing digital content. Part 5 of the MPEG-21 standard specifies the syntax and semantics of a Rights Expression Language. MPEG chose Extensible Rights Markup Language (XrML) [ContentGuard, 2001][Wang et al., 2002] as the basis for the development of the MPEG-21 REL. It makes use of the RDD [MPEG-21-RDD, 2004] that comprises a set of clear, consistent, and structured

terms. The RDD defines the meaning for the terms defined in the REL.

MPEG-21 REL [MPEG-21-REL, 2004] specifies the syntax and semantics of the language for issuing rights for users to act on DIs. The most important concept in REL is the license that conceptually is a container of grants, each one of which conveys to a principal the sanction to exercise a right against a resource.

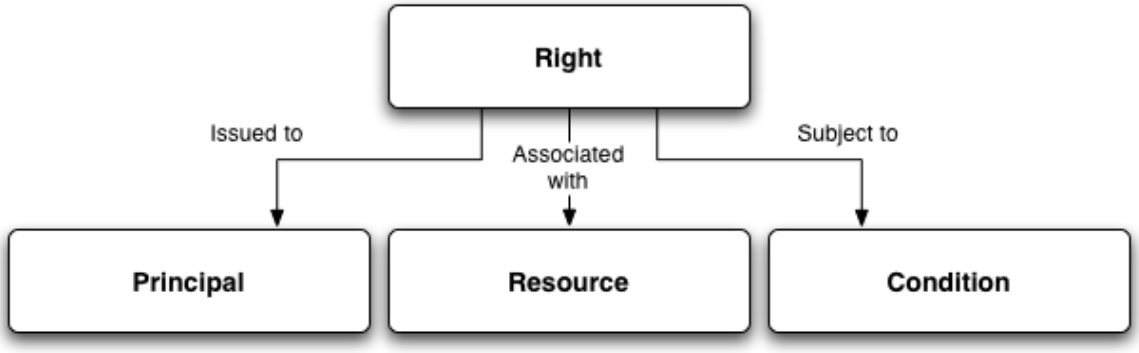

Figure II.2.2: The REL data model

The MPEG-21 REL (Figure II.2.2) can be extended to support new business models defining extensions. The extension mechanism that MPEG-21 REL specifies, allows the addition of new elements to address the requirements of a new application domain. Currently, MPEG-21 REL standard specification has five extensions: multimedia, standard, multimedia extension one, multimedia extension two and multimedia extension three.

2.2 MPEG-21 Intellectual Property Management Protection

The Intellectual Property Management and Protection (IPMP) Components, part 4 [MPEG-21-IPMP, 2006] of the MPEG-21 standard, deals with the standardisation of a general solution for the management and protection of Intellectual Property.

This part of MPEG-21 defines a language to provide protection and governance (i.e. control of content usage rights and conditions by means of a digital license) to any part of a Digital Item (DI), from a complete DI to a specific asset. The IPMP DIDL protects a part of the hierarchy of a DI, and provides mechanisms to associate appropriate identification and protection information to the protected part. Each of the IPMP DIDL elements contains the

elements Identifier, Info, ContentInfo and Contents. The Identifer element contains a unique identifier for the protected element. The Info element contains information about the protection tools and the rights expressions that govern the protected element. The ContentInfo element contains information about the protected element. Finally, the Contents element acts as a placeholder for the protected contents [MPEG-21-IPMP, 2006].

IPMP Components element also defines the information regarding the protection of a DI. This information falls into two categories: information about protection and governance related to the whole DI and information about the specific protection applied to a certain part of a protected DI. The general protection information includes the collection of licenses and lists of protection tools used, which can be later referred from specific protected elements. On the other hand, the specific information includes the specific tools and protection keys that have been applied, the licenses which are specific to that content, and others.

2.3 MPEG-21 Event Reporting

Part 15 of MPEG-21 Event Reporting (ER) [MPEG-21-ER, 2006], describes the mechanisms for informing about events occurred in the system. These events can be generated by Peers, applications, and others, and can be included inside the Digital Item to be sent under special circumstances or to be sent by the applications without DI intervention.

Event Reporting is required within the MPEG-21 Multimedia Framework to provide a standardised mean for sharing information about Events amongst Peers and Users. An Event, which can be defined as the occurrence of a reportable activity, is related to Digital Items and/or Peers that interact with them. In the MPEG-21 context, the reporting messages that include information about different aspects of media usage are called Event Reports [Rodriguez and Delgado, 2007a].

Fundamentally, Event Reporting facilitates interoperability between consumers and creators, thereby enabling multimedia usage information to be both requested and represented in a normalised way. Examples where Event Reports may be requested include usage reports, copyright reports, financial reports and technical reports [Rodriguez and Delgado, 2007a].

The basic model of Event Reporting indicates Events requiring reporting may be specified by interested parties through the use of an Event Report Request (ERR). An ERR is used to define the conditions under which an Event is deemed to have occurred. Events defined by ERRs trigger the creation of an associated Event Report (ER), which contains information describing the Event, as specified in the associated ERR. The ER purpose is to indicate which Peer created it, define the data items that need to be included in such Event Reports, provide a reference to the originating ERR and provide status information regarding its completion and creation, along with a free form description [MPEG-21-ER, 2006].

3 Coral

The objective of Coral Consortium [Kalker et al., 2007] is to facilitate the creation of DRM interoperability services, by third parties who have incentive to do so, which work among existing DRM technologies. Coral, by itself, it is not a DRM system [Coral, 2006c]. It uses the different existing DRM systems, through an interoperability framework (Figure II.2.3).

The Coral specifications consist on the Coral Core Architecture for DRM interoperability [Coral, 2006a] and a specification called Ecosystem-A [Coral, 2006d], which extends the Core Architecture for home networking interoperability.

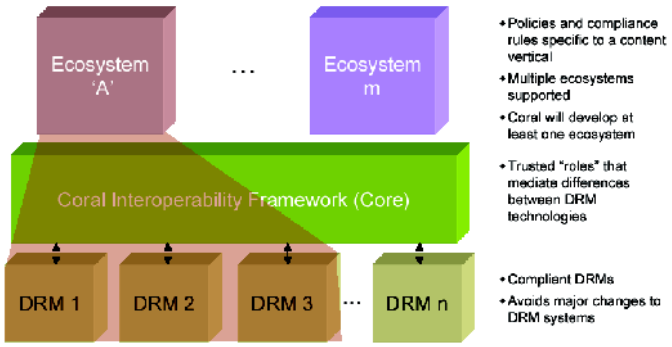

Figure II.2.3: Coral interoperability Core and Ecosystems

The Coral Core Architecture provides an underlying infrastructure that must be completed by the design of a content ecosystem responsive to the needs of a particular deployment. To look at an informative analogy, the Coral Core Architecture provides a measure of interoperability for DRM systems in the same way that the HTTP protocol and HTML language provide interoperability for the World Wide Web [Coral, 2006b]. These protocols

define behaviours that must be implemented by all web browsers and web servers, but do not constrain the uses to which these technologies are applied. Similarly, designers of ecosystems based on the Coral Core Architecture must define the specific application of the Core technologies [Coral, 2006a].

The elements that must be defined by a Coral ecosystem include:

- Usage Models: A common set of usage models that must be enforced by all content protection systems that are allowed to participate in the ecosystem. For example, if the ecosystem is designed to support time-based subscriptions, support for the secure management of time is required. This specification defines a domain model that requires specific domain semantics that are not specified in the Coral Core Architecture;

- Policies: An ecosystem may define certain parameters that modulate the behaviour of the system for security or usability reasons. An ecosystem might, for example, support the notion of periodic synchronisation with a certificate revocation list to ensure that illegitimate actors are quarantined. The frequency of such synchronisation might be a policy variable;

- Roles: The Coral Core Architecture defines a Role as an assertion that a particular entity engages in a standardised set of behaviours defined by the Coral Core Architectures or by ecosystems built upon it. Possessing a particular Role provides the predictability needed for interoperability across implementers. If a particular entity can be verified to possess a given role, then the other systems with which it interacts can be certain that it will behave according to a specification. The Coral Core Architecture defines some Roles; ecosystems may define their own Roles as necessary to provide ecosystem-specific semantics or functionality;

- Role Groupings: An ecosystem may additionally specify dependencies amongst Roles, both for Roles that it defines itself and Roles that are incorporated from the Coral Core Architecture. To take an example from this specification, an entity with the Ecosystem-A Domain Manager Role must also possess the Principal Relation Manager role defined in the Core Architecture. Role grouping allows ecosystem designers to

define their systems by extending and composing existing roles, adding ecosystem-specific functionality where required;

- Extensions: The Coral Core Architecture is designed to be extensible in as many ways as possible, allowing ecosystem designers to enhance the underlying framework for a specific application. In Ecosystem-A specification, for example, opaque DRM-specific information is conveyed in the extension fields of interfaces defined in the Coral Core Architecture;

- Third-party Technologies: Ecosystems may incorporate or require the use of third-party technologies not present in the Coral Core Architecture. For example, specific Secure Authenticated Channel or other output technologies may be incorporated into the ecosystem specification.

The Coral core architecture includes roles, which are functions, and nodes, which are physical realisations of collections of roles [Coral, 2006a]. Two important roles in the Coral architecture are the Content Mediator and Rights Mediator roles, which determine whether one system should be allowed to exercise rights on a particular piece of content from another system [Coral, 2006e]. Other roles include DRM Content Exporter, DRM Content Importer, and DRM License Mapper. The latter maps rights in one system to rights in another by making reference to a set of policies.

Nodes are combined into Ecosystems, which are collections of nodes that are considered mutually trustworthy so that they can interface with one another without security concerns that lead to diminution of functionality. The entire architecture is based on NEMO (Network Environment for Media Orchestration), a secure messaging scheme based on X.509 identity certificate and SAML security assertion standards [Coral, 2006a].

Ecosystem-A instantiates and extends the Coral core architecture for the specific application of content interoperability in home networks [Coral, 2006d]. It implements, among other things, two important concepts:

- Rights lockers: repositories of licenses associated to content that principals (users and/or devices) hold;

- Domains: groups of devices that act as single principals for licensing purposes.

In order for Coral to interoperate between different DRM systems it is necessary that those vendors modify their technologies to make them Coral-compliant [Coral, 2006d].

When a user acquires certain content, the Coral system registers a right token in a rights locker designed by the user. When requested, the rights locker sends the rights token to the license service with which a device interacts in order to check the rights fulfilment. If the same content needs to be consumed on a different device that works with a different license service, then a rights mediator will request to the rights locker the token and send it to the new license service, which will use it to construct the corresponding native DRM license and send it to the new device.

The Coral approach for solving DRM license interoperability in order to obtain equivalent licenses for different systems is to derive them from a common rights token. This approach supposes the definition of a new rights expression language from which other languages can be derived. In fact, Coral provides in its specifications the XML Data Type Definition for the creation of such rights tokens [Coral, 2006d].

4 Digital Media Project

The Digital Media Project (DMP) [DMP, 2007] approaches the problem of DRM Interoperability [Choi et al., 2007] through the specification of technologies (called Tools) required to implement “Primitive Functions”, i.e. “smaller” functions obtained by breaking down the Functions Value-Chain Users perform when they do business between themselves. It is expected that, while Functions may undergo substantial changes as a consequence of the evolution of the media business, Primitive Functions will generally remain stable [Chiariglione and Merrill, 2006].

Therefore, far from targeting a universal “DRM standard” capable of providing Interoperability between Users in arbitrary Value-Chains or across different Value-Chains, DMP only provides:

- Specifications of Tools enabling Primitive Functions;

- Examples of how Value-Chains serving specific goals can be set up using the standard Tools.

DMP has developed three versions of the Interoperable DRM Platform specification (IDP-1, IDP-2 and IDP-3 [DMP, 2007]. Chillout [Chiariglione and Liao, 2006] is the name of the IDP Reference Software, released as Open Source Software under Mozilla Public Licence 1.1, which is currently under development.

According to DMP, a typical Value-Chain (Figure II.2.4) has the following main components: Creation (including Adaptation), Instantiating, Production, Content Provisioning, Service Provisioning and Consumption [Chen, 2007].

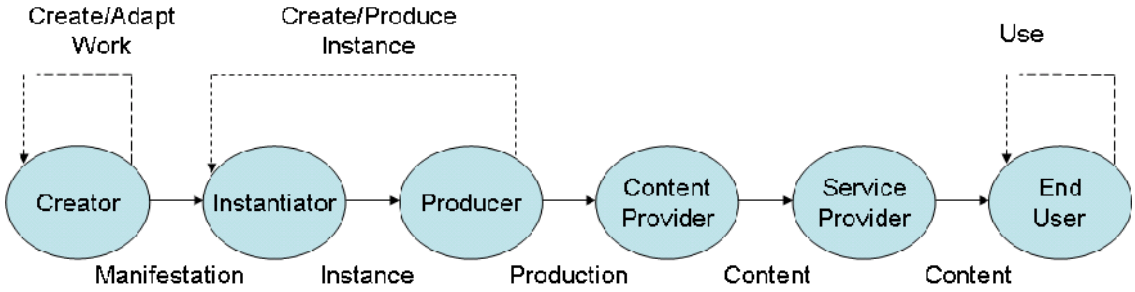

Figure II.2.4: General case of a Value-Chain

For proper management in a Value-Chain it is useful to combine different types of Resources with different types of Metadata and possibly other information types. DMP calls this combination Content. The digital Representation of Content, needed to Use Content on Devices, is called DMP Content Information (DCI) [DMP, 2007].

A User wishing to express conditions to Use a Content Item can associate a Licence to it Granting Permissions under specified Conditions. In this case the Content is called Governed Content. The party Granting Permissions is referred to as Licensor and the party receiving them is referred to as Licensee. A User who does not wish to express Conditions to Use a Content Item can do so by releasing it without a License.

To enable a Device to interpret Permissions without human intervention, a REL is needed. The Licensee can be a Device, a User or a Domain and the Licence can be bundled within the Content (i.e. it is part of the DCI) or not Bundled within the Content (i.e. the DCI references an external License). Other Content Elements (e.g. Resources) can be in-line or referenced

[DMP, 2007].

The Content Elements in Governed Content can either be in a form that allows immediate Processing, e.g. for Rendering by a Device (so-called Clear-text) or in a protected (i.e. Encrypted) form. Keys and related DRM information can be conveyed by various means.

When the Device has limited capabilities (as in Portable Audio-Visual devices (PAV)) it is useful to be able to make reference to a basic selection of Encryption Tools. However, when the Device does not have such restrictions (as in Stationary Audio-Visual devices (SAV)) it is important to be able to convey blocks of executable code (called DRM Tools) in a DCI that are required to Process various types of Governed Content [Chiariglione and Merrill, 2006].

XML is the technology selected by DMP to Represent Content. XML is very powerful and flexible but Content Represented by means of XML can easily become bulky and unwieldy. Therefore DMP has selected an XML binarisation technology that not only reduces the size of a DCI but also allows simpler Processing in a Device without loss of information.

To Deliver Content between Device Entities it is necessary to Package a DCI (in binary form) and its referenced Resources. Two forms of Delivery are possible: as File (DMP Content File - DCF) and as Stream (DMP Content Broadcast - DCB - in the case of an MPEG-2 Transport Stream, and DMP Content Streaming - DCS - in the case of Real Time Protocol on Internet Protocol) [DMP, 2007].

In general Users require Devices to perform the Functions proper of their role in the Value-Chain. To entrust their Content to a Device, Rights Holders must have assurance that the Device will execute Functions as expected. Device Certification is the process that provides that assurance. This is performed by a number of organisations (Certification Agencies) that are properly appointed and overseen by a single root authority (Certification Authority). DMP appoints the Certification Authority after approving the Authority's Certification policies.

In general interacting Devices require the establishment of a Trust relationship between them, e.g. when they Deliver Content between them. A precondition for this to be possible is that a Device be Identified. The task of Assigning Identifiers to Devices is performed by a number of organisations (Registration Agencies) that are appointed and overseen by a single

root authority (Registration Authority). DMP appoints the Registration Authority after approving the Authority's Registration policies. The same three-layer arrangement used for Certification is also used for Identification [DMP, 2007].

Content Items also require Identification. When Identifiers are assigned appropriate Metadata is also typically generated.

DMP IDP-2 provides the following models:

- Creation Model: identifying the major entities for which Intellectual Property (IP) is attributed and describing their relationship to the digital objects involved in Content Creation. The following objects referred to as IP Entities are defined formally in the DMP Terminology: Work, Adaptation, Manifestation, Instance, Expression;
- Distribution Model: identifying and describing the role of the principal Value-Chain Users engaged in distribution. In the general DMP Distribution Model the following Users operate: Content Providers, License Providers, DRM Tool Providers and Service Providers;
- Delivery Model: describing the two basic ways (File and Stream) in which Content can be delivered between Devices. DMP defines Delivery as a File (DCF), on an MPEG-2 Transport Stream (DCB) and on a Real-Time Protocol transport (DCS);
- DRM Tool Model: describing the general operation of DRM Tools in Devices. DRM Tools required to e.g. decrypt Resources may be natively embedded in a Device and executed when such DRM Tools are required. Alternatively DRM Tools may be delivered as part of a DCI. If the DCI does not contain the required DRM Tool, the DRM Processor in the device tries to Access it from the local Secure Storage. If the required DRM Tool is not found there, the DRM Processor Accesses the missing DRM Tool from the DRM Tool (or Service) Provider. In addition to individual DRM Tools, IDP-2 gives the possibility to convey DRM Tool Packs in a DCI. A DRM Tool Pack is composed of a DRM Tools Agent and a set of DRM Tools called DRM Tool Group. A DRM Tool Pack may contain all the DRM Tools required or require other DRM Tools external to the Tool Pack;

- Domain Model: describing the operation of Domains of Devices and Users. Using Domains it becomes possible to implement more flexible Licensing modalities, e.g. to License a Content Item to all Devices or Users in a Domain;
- Device Model: identifying and describing the principal Devices employed by Value-Chain Users, e.g. Content Consumption Device (PAV), Content Consumption Device (SAV), Content Creation Device, Content Identification Device, Content Provider Device, Device Identification Device, DRM Tool Identification Device, Domain Identification Device, Domain Management Device, DRM Tool Provider Device, License Identification Device, License Provider Device, PAV external Device;
- Import/Export Model: describing how governed Content can be converted to Governed Content from a different value chain and vice versa, by using mechanisms such as translating licenses;
- Data Model: identifying and describing the different types of Data specified by DMP, as for example, Content, Identifier, Resource, Metadata, DRM Information, DRM Tool, DRM Tool Body, Licence, Key, DRM Message, Device Information, Domain Information, Use Data, DCI Signature, DCI Hash [DMP, 2007].

Part III. Contributions

Chapter 1. Introduction to Contributions

1 Introduction

The time for the "old" electronic Intellectual Propriety Rights (e-IPR) management systems is coming to an end. This is the panorama for the current commercially dominated rights management systems market upholding a model that is not fair and acceptable by both end-users and content authors/producers [Bechtold, 2006].

Today, these digital rights management technologies face an enormous challenge. They are being confronted with a strong opposition coming from many sources. Perhaps, one of the most visible and active anti-DRM forces is embodied by the DefectiveByDesign[8] (DbD) activist group. DbD is a broad-based anti-DRM campaign that targets primarily the Big Media, unhelpful manufacturers and DRM distributors [Chiariglione, 2005]. According to DbD, DRM-enabled products have been crippled from an end-user perspective and therefore are "defective by design". DbD is not the only organised campaign and the anti-DRM feeling is growing everywhere [Craig and Graham, 2003].

In part, this anti-DRM feeling has a point. DRM, has it exists today, limits the end-user freedom to use legally acquired content [Mulligan et al., 2003]. It is easier to obtain illegal content not only because it is free, but also because it is not restricted in any way that limits user freedom to use it [Russinovich, 2005]. A clarification is needed here. The "freedom to use the content" does not refer to the fact that the user can use content for free and make it available to all of its friends, or that he can share it on his favourite P2P network. The

8 http://www.defectivebydesign.org

sentence refers to simpler issues, such as the ability to use the content on every digital rendering devices that the user possesses. This is something that should be brought from the analogue content World to the digital content World.

Another classical example that exemplifies why current DRM is not working is the fact that DRM-governed content acquired on a digital content store will only play on specific content rendering applications and/or devices that were designed to support such DRM regime. If the user wishes to acquire content on a different store that uses an alternative DRM regime, the user will be unable to render the governed content on the very same applications and/or devices [Kwok, 2002]. This is the case of iTunes[9] and URGE[10] digital content stores. iTunes uses the FairPlay rights management technology while URGE uses the Microsoft Windows Media DRM, so any DRM governed music acquired in the iTunes store will not play in Windows Media Player and any DRM governed music acquired in URGE will not play on the iPod [Fetscherin, 2006].

This is a problem for modern digital content users, but at the same time it is also a huge problem for content providers. With this model, content producers and providers will either need to make their digital content available for a different set of platforms and systems or stay with just a single platform [Odlyzko, 2007]. For digital content users this situation is obtrusive because they are limited to adhere to just one of the available content stores and to one of the available content rendering applications and/or devices. Partly, this is one of the hardest problems in current DRM deployment - interoperability.

Interoperability is currently one of the most discussed and debated open issues in the rights management field. The actual DRM systems trend is one-to-one. In this, there is a store that trades digital content that can be used in a particular content rendering application and on a specific flavour of portable media player [Ahamed and Ravinuthala, 2007]. This is the strategy followed by some DRM providers that deliberately use incompatibility mechanisms to lock-in users to acquire particular hardware/software content rendering devices (portable or not). This old-fashioned business strategy tries to raise entry barriers to new players and at the same time penalise end-users [Bellovin, 2007]. These DRM interoperability issues affect rights management systems in general [Senoh et al., 2004].

9 http://www.itunes.com
10 http://www.urge.com

Current DRM implement a vertical DRM (vDRM) system. vDRM is a type of DRM that is unique throughout the entire value chain, from the content producer to the end-user. On the other side there are horizontal DRM systems (hDRM). hDRM is a type of DRM that is open enough to support a multiplicity of DRM-enabling mechanisms in all the DRM governed content value chain (Figure III.1.1) [Serrão et al., 2005b].

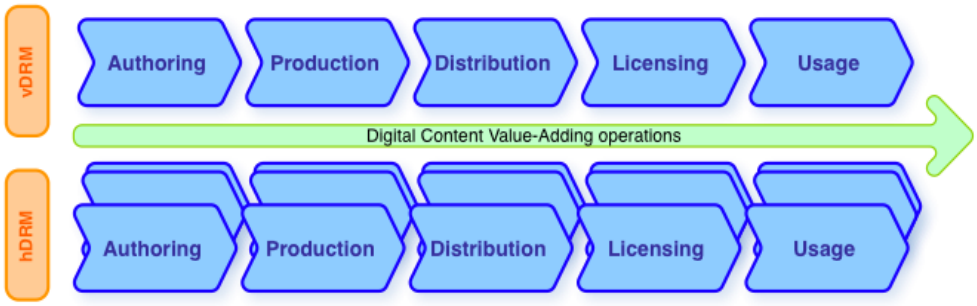

Figure III.1.1: Vertical DRM versus Horizontal DRM

Rights management systems, as they are today, are far from perfection. Most of them reflect the industry attempts to build barriers on outdated business models. Using Michael Porter's terminology [Porter, 2004], they build barriers to the entry of new competitors. Doing business in the digital World using the same analogue World business rules is almost impossible. But which are the main reasons for the DRM 1.0 failure? [Stamp, 2002]

First, until now all the DRM systems are vulnerable [Schneier, 2001]. There is no single mainstream DRM solution (Apple FairPlay and Windows Media DRM included) that has never been circumvented [Marson, 2005]. They all have been circumvented, and more than once, and they will continue to be in the near future. This is “natural” on the information security World. Many tools populate on the WWW (QTFairUse, DeDRMS, PyMusique, SharpMusique , NSCdec and others) offer the opportunity for users to remove DRM enforcement from legally acquired digital objects and therefore circumventing the whole purpose of copy protection and rights management [Petitcolas et al., 1998].

Second, they limit the user's digital content experience [Sharpe and Arewa, 2007]. Current DRM affects in many ways the user experience. They impose a restrictive usage of DRM-governed digital objects in such a way that is far from what users are used to on what concerns analogue content. Consider the following situation as an example. Wal-Mart in USA has started selling movies via-download (DRM-protected). The downloaded movies can only

be played on a PC and on a specific media player software [Ahamed and Ravinuthala, 2007]. If the user wants to watch the movie on a TV screen it has to be plug on the PC. This is restrictive for the end-user, and with some extra dollars the user can order the same movie in DVD format and use it without any restrictions, on any compliant device and even on the PC.

Third, they are not available everywhere [Chen, 2007]. Today DRM is not available for all platforms. The two most deployed DRM solutions that are responsible for the protection and rights management of most of the digital content available for legal acquisition (Apple Fairplay and Windows Media DRM) are only available on Windows and Apple systems. Linux users are not considered on this equation and therefore are left out of the process. This has caused in the past, the emergence of circumvention mechanisms to allow Linux users to get digital content from the Apple iTunes music store [Fetscherin and Schmid, 2003].

Fourth, do they protect anything? Current DRM systems are extremely weak [Schneier, 2001]. They have been broken and fixed several times and the general sense is that they offer little protection. In fact, it is known (and admitted) that it is possible to acquire a music track through the iTunes music store, record it on a CD-ROM and afterwards rip it without any kind of DRM protection. Is this the kind of protection content providers and authors are looking for? [Schneier, 2001] Robustness should be considered as one of the most important aspects for this type of rights management solutions.

Fifth, DRM cannot be imposed to end-users [Bechtold, 2006]. Until now, end-users have been left out of the DRM process, and most of the DRM schemes for digital object governance have been imposed by the content industry [Bellovin, 2007]. This gives origin to barriers on the acceptance of DRM as a legitimate technology to uphold copyright on digital objects. Rather, it creates the awareness that DRM cripples the end-users rights and imposes unacceptable restrictions on the way they interact and use digital content. DbD is just one of the many examples of an on-line association (promoted by the Free Software Foundation) to eradicate DRM from digital content. DRM should not focus so much in the restriction of user's rights but more on the positive aspects of the rights protection and on exploring on how it may improve the user experience and on the creation of new and improved business models.

Sixth, they do not interoperate [Koenen et al., 2004]. This is one of the biggest issues on current DRM. Existing DRM solutions do not interoperate and do not allow DRM-protected digital objects to flow from device to device to be enjoyed by end-users. It is currently impossible to "legally" acquire content on a specific DRM-governed store and to render that same content on another specific DRM-governed device. These types of DRM restrictions do not make much sense, and are extremely crippling for the end-users experience.

These six points identified allow to conclude that this scenario does not meet the current requirements and has to be changed in the future. The path to the DRM future needs to go in a direction that offers answers to some of these issues [Nuetzel and Beyer, 2006].

New rights management solutions are being studied and developed either by the academic community, international standards organisations and private joint organisation initiatives [Cheng and Rambhia, 2005]. These groups are trying to address the major drawbacks of the old rights management systems. A new rights management system era is emerging (DRM 2.0).

The work presented on this thesis will be focused on the interoperability issues around rights management systems. As presented before, one of the biggest issues still to be solved on DRM concerns the lack of interoperability between the different rights management solutions [Christopher, 2006]. This work will address the problem of DRM interoperability in general and the specific question of interoperability on the rights management layer in the context of DRM platforms.

Interoperability is a broad expression that can assume different meanings according to the context where it is applied. According to ISO/IEC 2382, interoperability can be defined as "the capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units" [ISO/IEC-2382-20, 1990].

In the context of DRM, interoperability can be defined has the mean to use governed digital content regardless of the specific characteristics of the DRM technology that is used to govern the content and manage its rights. In general, there are two different approaches to address interoperability: the vertical and the horizontal approach. The vertical approach to

interoperability is based on the assumption that every device uses one unique format and type of solutions - the format may be closed and proprietary or open. The horizontal approach to interoperability is based on the specification of some common interfaces and mechanisms while the definition of remaining elements stays open – the solution is necessarily open.

However, the interoperability problem in DRM, cannot be simplistically reduced to these two dimensions [Picot, 2003]. The DRM interoperability problem is multi-dimensional and can be approached from several sides (Figure III.1.2): Metadata, Rights Management, Content Format, Content Protection, Rights Format, License Acquisition, Key Management, Domain Management, Devices and others. According to a recent study conducted by INDICARE [Groenenboom et al., 2006], two of the most important issues on the use of governed content are device and rights management interoperability.

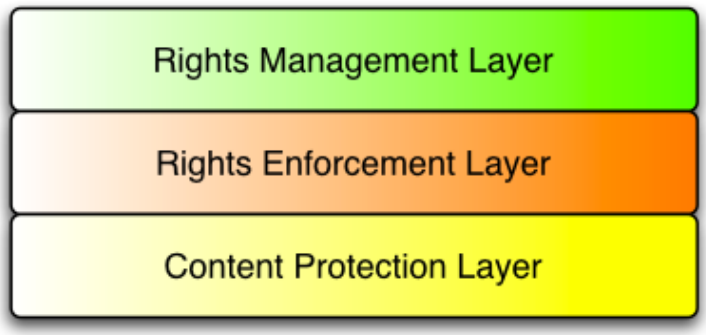

Figure III.1.2: The three different DRM layers

The Rights Management layer (the topmost layer) defines the conditions (rights) under which the content access is granted. These rights are generally expressed using a Rights Expression Language. The licence is a representation of the rights.

The Rights Enforcement layer (the middle layer) provides the protection of the licence and of the keys that will be used to open the content. This layer must also ensure that the rights granted are being respected.

The Content Protection layer (the lower layer) defines how the content is protected by specifying all cryptographic material that are used, including the scrambling and encryption algorithms, the type and size of the keys, the crypto-period and, optionally, the watermark algorithm and the keys.

As DRM 1.0 solutions are non-interoperable, each of them implement their own different DRM layers and these DRM layers are incompatible between the different existing DRM solutions. This is the typical scenario (Figure III.1.3).

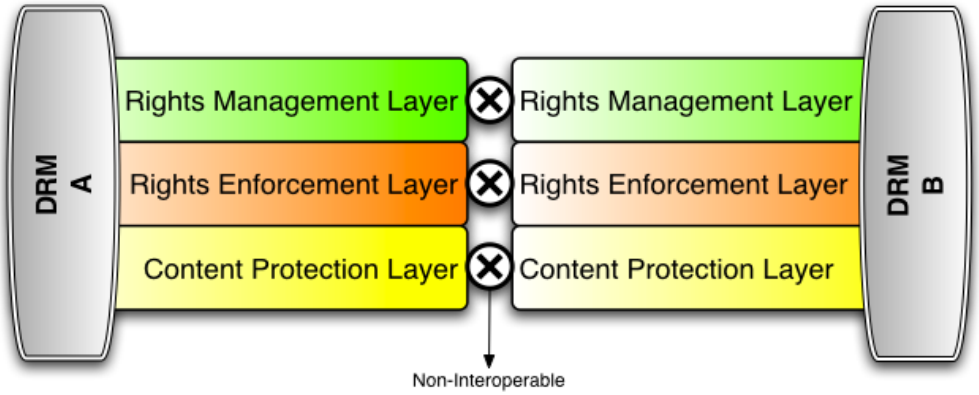

Figure III.1.3: Non-Interoperable rights management systems and layers

On the other hand this scenario although simpler to implement and deploy is not desirable and causes more problems than it solves, both for consumers, device producers, content providers and content distributors [Kalker et al., 2007]. Consumers are put of by content and services that do not work with all of their devices. Device producers must choose to use a single DRM technology or to support multiple ones. Distributors have to choose which is the most popular DRM technology and to distribute only for that platform leaving out some potential consumers. This model completely breaks apart the two DRM solutions, building incompatibilities on all the different layers, from the Content Protection Layer up to the Rights Management Layer.

The alternative to this is to have a model in which each of the different layers implemented by the different DRM systems can interoperate (Figure III.1.4) [Heileman and Jamkhedkar, 2005]. In this model it would be possible to use, for instance, the Content Protection Layer from one DRM system A and the Rights Management Layer from another different DRM system B [Jamkhedkar and Heileman, 2004].

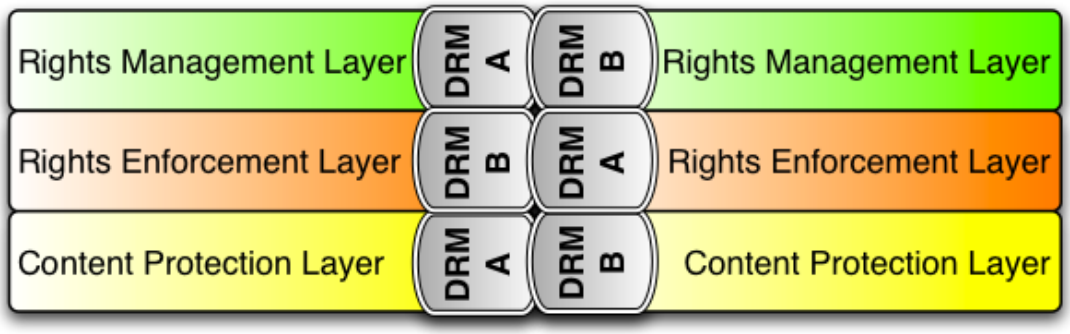

Figure III.1.4: DRM interoperability scenario, with all three interoperable layers

Some authors like [Koenen et al., 2004] and [Kalker et al., 2007] have suggested a set of approaches to the DRM interoperability problem, based on International Standards: full-format interoperability, connected interoperability and configuration driven interoperability. In the full-format interoperability case, all protected content conforms to some unique globally standardised format. This is hard to accomplish, since all of the content providers and DRM software manufacturers would have to agree on the same file format to use. This is nevertheless the strategy that is being followed by Open Mobile Alliance (OMA) [Buhse and van der Meer, 2007]. In OMA, the DRM Content Format (DCF) is a format that all the devices need to know and implement. In a second alternative approach, translation third parties are used to translate operations from one DRM regime to another. This seems to have a more solid background and a set of translation entities may actually exist in the future, to allow the translation between different DRM functionalities to accomplish the same objective. In this approach a peer-to-peer architecture may need to be established in which each node allows an interface to its peers, and if it cannot satisfy a direct request then it redirects the search to other peers. Another option for this approach is the "intermediated digital rights management" that identifies a set of tasks to be carried out by the intermediary in transferring content in the format used by the content provider to the format required by the end-user. Rights management tasks would have to be executed by a third party server (the intermediary) on behalf of the content scripts and end-users. The third and final approach for DRM interoperability upholds the downloading of adequate tools that allow any DRM system to get the ability to process protected content on end users devices. This is also a valid alternative for the DRM interoperability problem, allowing devices and digital content rendering applications to "grow" in terms of capabilities and functionalities to enable different DRM regimes according to the ones governing the protected content. For instance,

this is the DRM interoperability model that is uphold by MPEG-4 IPMP-Extensions. These approaches will be further discussed on the following chapters.

2 Specific contributions of this work

In this thesis the rights management interoperability problem will be addressed and some mechanisms will be proposed to improve the interoperability between different open rights management solutions.

The thesis author is convinced that true rights management interoperability will only be possible if these solutions follow an open approach. This means either that the specifications, interfaces or source-code are freely available. The interoperability model should be open, to enable the interaction between the existing rights management solutions and to provide the means for new and emerging solutions to interoperate as well.

Some of the contributions of this thesis include the design and operation of some mechanisms to allow different open rights management solutions to interoperate at different levels. Since the interoperation between these platforms is a rather complex problem, some specific processes have been selected to develop some interoperability mechanisms.

The first contribution presented is about how the emerging service-oriented computing paradigm could be used to promote rights management interoperability. In this contribution, the concept of a generic rights management framework is introduced. This framework is composed by different components that provide a set of functionalities to different actors: End-Users, Content Owners and Content Providers. In this, a set of different rights management services can be used to build different rights management ecosystems provided by different suppliers and using different services. In this contribution, it is proposed to expose the different rights management services as web services and allow them to integrate and interoperate among each other. Each of the services exposes their functionalities through WSDL and published on a public repository (UDDI) where they can be searched and redirected. An example illustrating how this integration may actually work in a scenario where different rights management service providers can co-operate in the service integration to create new interoperable rights management solutions is also presented.

Another contribution presented in the context of this thesis refers to the usage and applicability of standard Public-Key Infrastructure solutions (in particular PKIX) as a DRM interoperability driver. This contribution presents some ideas on how PKI services can be helpful has an interoperability driver for different rights management services and components, in particular for the establishment of trusted environments [Erickson, 2003]. Two different strategies are considered, one using a single and central PKI service and a second using multiple PKI services. This contribution proposes the deployment of this second strategy through the usage of a PKI services broker that is responsible for managing the interoperation between the different PKI services, therefore providing the establishment of a general trust environment.

Any rights management system has to deal with a relevant amount of keying material. Such keying material, if not properly managed may lead to security breaches. Therefore, another contribution was presented on this thesis on the design of a mechanism to securely manage the licenses and the keys associated to the digital content governance. In this contribution, an analysis of the current open rights management platforms, about how the key management life-cycle is implemented, was also conducted.

A contribution is also presented on the study, design and development of an open rights management service-based architecture, called OpenSDRM, that incorporates the interoperability mechanisms and strategies discussed before. OpenSDRM is an open rights management architecture designed using a service-oriented paradigm, providing the separation of the different rights management processes governing the digital content value chain, as different, independent, distributed services. In this contribution not only the different components of the open DRM architecture are detailed, but also the different services, which are part of OpenSDRM, are presented. This architecture has been used to create different experiences in the field of governed digital content e-commerce, that will be reported has a contribution of this thesis as well.

Another contribution presented refers to the design and implementation of a client-side rights management middle-ware mechanism, called Wallet Rights Management interoperability Middle-ware (WRMiM). This mechanism implements a client-side interoperability mechanism, allowing the abstraction of the Content Rendering Applications

(CRA) from the rights management complexities required by the governed digital content rendering. Another contribution presented, related to the previous one refers to the establishment of a rights expression language independent template mechanism that allows rights templates to be instantiated by the content providers without having to deal with the complexity of the many existing REL. This way, the Content Provider, can define the conditions and terms that best suit their own digital content business model, without having to make any assumption about the specific rights expression mechanism that will be used.

Finally, another contribution is presented with a series of experiences/use-cases where the mechanisms uphold on this thesis have been experimented, and tested with real live trials. There were a total of four different use-cases in which these ideas were tested together with the OpenSDRM. First, a digital music e-commerce business scenario was setup to demonstrate the applicability of the platform in a scenario, where the content provider directly uploaded content to the platform, defined the rights, and the users were able to access such content, governed by such rights. Second, a large JPEG2000 remote sensing images e-commerce site was set-up. The images were governed by the digital rights management platform, and the rights were defined to allow the user access according to a certain resolution level. Third, another scenario were the platform was used. This time, the OpenSDRM platform was used to govern the access to video-surveillance video, both for stored video-surveillance data and for real-time video-surveillance data. Finally, a fourth scenario, for the usage of the rights management platform for home-networks was setup. The objective of this scenario was to have a mean to control the access to governed content, from an home network appliance, in each content could be shared within the home network through a residential gateway.

All of these contributions will be presented and further described on the following chapters of this document.

Chapter 2. Rights Management interoperability and Service-oriented Architectures

1 Introduction

In a largely interconnected World, the Web Services (WS) computing paradigm is gaining momentum. Most Web Services applications existing today are being developed in the E-Business or E-Commerce context, mainly for Enterprise Application Integration (EAI). This chapter describes the applicability of the WS technology to control and manage the access to digital content through service-oriented rights management architecture [Michiels et al., 2005], deploying critical rights management elements such as device and user identification and authentication, content registration and protection, license representation and production and payment [Denaro et al., 2006].

The service oriented computing model, which allows the distribution of applications and services over open networks, represents one of the most important trends in the present days. The paradigm model behind the Service Oriented Architecture (SOA) encompasses the possibility to potentially have completely heterogeneous application architectures in which each of the services provided can be implemented in their own environment and providing their own functionalities in their own language, but contributing globally for the same end [Kleijnen and Raju, 2003]. One of the applications that can benefit enormously from this approach is rights management [Serrão et al., 2005c].

Breaking with the traditional monolithic programming paradigm, distributed programming

introduced a fresh plethora of solutions to common problems like language interoperability, remote data exchange and remote procedure execution. Distributed programming has been around for quite some time now, but existing solutions found it hard to convince users of their simplicity and lightweight functionality. All of the previous solutions like CORBA, DCOM, RMI and several more, were heavier and bulkier from the performance perspective, more complex, more dependent on shared business context information between intercommunicating parties and generally required more agreement among the several business systems involved [Serrão et al., 2005c]. The WS paradigm came in the natural order of things to offer simplicity, portability and interoperability – all in an existing-standards compliant manner, using tested technology and tried on underlying open networking infrastructures. The whole WS paradigm eventually highlighted the advantages of the Internet when applied to services, instead of just information. Instead of a network of documents the new Internet reality is now more of a network of (increasingly relevant) information, a network of semantically meaningful micro-content. This new shift in application development, and in particular in web application development, has in great part contributed to what is now becoming known as the Web 2.0. Leaving behind the "read-only" web environment provided by the original World Wide Web, the Web 2.0 paradigm is now a platform where users participate in information generation, where services consumption blends with the traditional information usage and where they can be accessed and integrated seamlessly by third parties interested – facilitating the exchange and remixing of the micro-content clusters that are disseminated around the web. This new Web 2.0 is not domain-centric anymore. It is content-centric. The word services in WS imply modularity – a component or building block like approach – usually used to build bigger services. And that is precisely the very essence of the WS paradigm, and precisely the key factor from which rights management systems can benefit from [Serrão et al., 2006c]. The actual platform that comprises the term WS is achieved through a combination of several technologies. The basic technology framework is laid out by HTTP and XML. The former is a well-established, ubiquitous protocol that is behind the World Wide Web while the latter is a very powerful semantic mark-up language that enables the accurate description of any content. While this basic framework is sufficient for a generic WS set-up, it is its augmented state – built resorting to other technologies, which enable remote invocation, directory services,

description of services, authentication, security, transactions, etc. – that actually harnesses the true power that lies at the very core of a WS platform. These base technologies are SOAP, WSDL and UDDI.

The most important WS approach benefits are interoperability, distribution and reusability. Therefore, they potentially can help DRM overcome one of its biggest barriers: interoperability. Currently, interoperability between different rights management systems is virtually inexistent and an open and distributed service-oriented approach can help reducing the existing interoperability barriers [Serrão et al., 2006c]. This is something that will be exploited later on this and other chapter.

This chapter starts by introducing a generic rights management framework concept composed of different components that provide a set of functionalities to different actors: End-Users, Content Owners and Content Providers. In this, a set of different rights management services can be used to build different rights management ecosystems provided by different suppliers and using different services.

The following section on this chapter exploits the possibility of using the emerging service oriented computing architectures to expose the different rights management services as web-services and allow them to integrate and interoperate among each other. Each of the services expose and describe their functionalities through WSDL and publish them on a public repository (UDDI) where they can be searched and redirected. This section also presents an example illustrating how this integration may actually work in a scenario where different rights management service providers can co-operate in the service integration to create new interoperable rights management solutions.

At the end of this chapter some conclusions are presented as well as some future work that can be performed in this field.

2 Generic DRM framework

Although it is possible to find many different DRM solutions and architectures today, they all share some commonalities between them. This makes possible the identification of the main actors and services that make up a generic DRM framework (Figure III.2.1) [hwan Suh et al.,

2006]. Generally, any rights management framework addresses the End-User, the Content Owner and the Content Provider offering a set of functionalities that allow them to conduct their business operations through the value chain [Hwang et al., 2004]. Such generic rights management framework has been originally proposed by [Verslype and De Decker, 2006] and identifies seven key DRM services: the Content Service (e.g. search for content), the License Service (e.g. issue licenses), the Access Service (e.g. authenticate consumers), the Tracking Service (e.g. produce usage statistics), the Payment Service (e.g. billing), the Import Service (e.g. submit content to the DRM system), and the Identification Service (e.g. reveal abusers) [Verslype and De Decker, 2006].

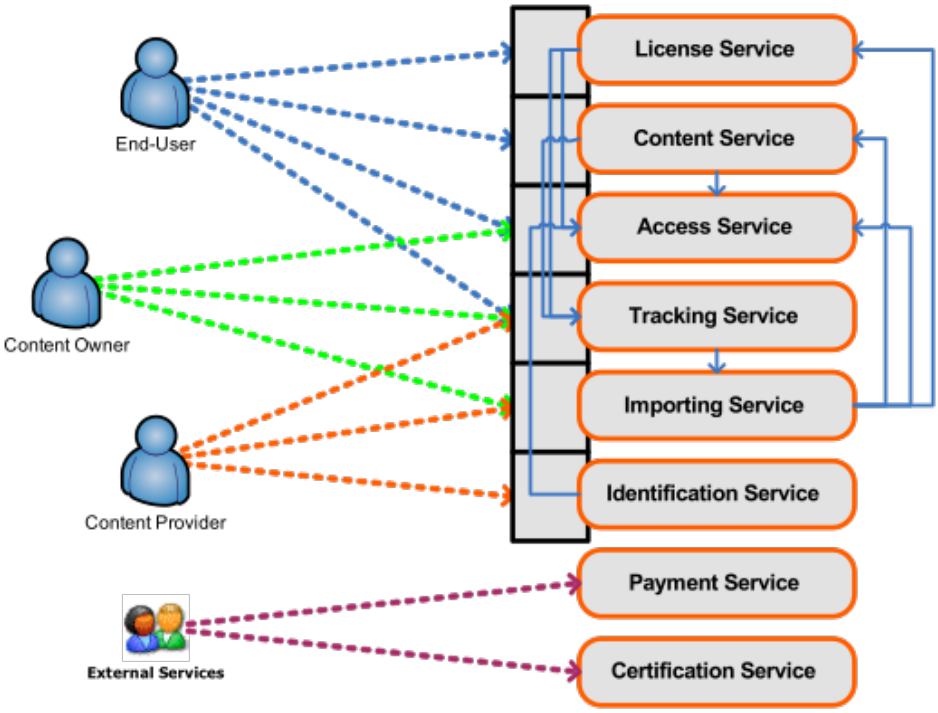

Figure III.2.1: Generic DRM framework

Each of the different generic services establishes relations between each other and with some of the actors. This was the approach that it was followed on this thesis and will be further reported on a letter chapter (Part 3, Chapter 6). However, there is a difference between this approach and the one that this contribution proposes. While the authors of this framework tend to see DRM in a layered organisation to provide interoperability between the different key services of different DRM products, the approach followed on this document is different. Instead of layers, the different rights management key-services are seen as generic distributed services, which hide their implementation details from the

different DRM actors exposing publicly their functionalities. This approach allows the distribution of the different key-DRM services over distributed open-networks, such as the Internet (using proper protection), and DRM key-services to be developed regardless of their own specific and proprietary implementation. This way it can be possible that DRM-services provided by vertical DRM solutions (examples of such include Windows Media DRM, Helix DNA or Apple FairPlay) to implement a specific interface to expose their basic functionalities to other services, without revealing any internal implementation details [Zhang et al., 2004].

Following such approach, new rights management solutions providers (either commercial or not, closed or open) can flourish in the market and offer their own particular DRM-services without having to implement a complete end-to-end solution [Serrão et al., 2005c]. The integration between all the services can provide interoperable environments where each can specialise on a particular type of service (or group of services) that will enable the implementation of different business models (Figure III.2.2).

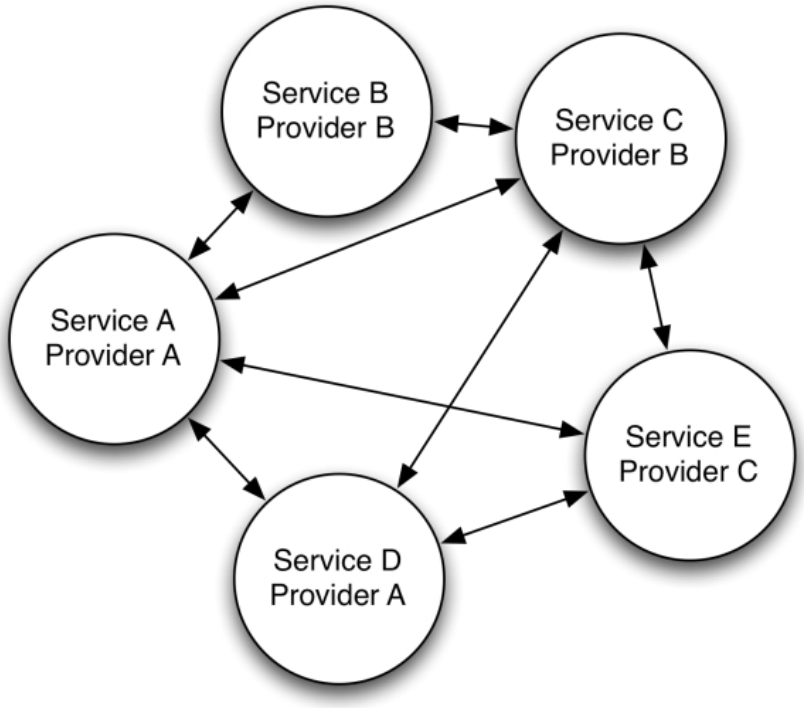

Figure III.2.2: Different services, by different providers, integrated in a common rights management solution

This approach has also the potential to enable the construction of mixed DRM solutions. These mixed rights management solutions result from the fact that the some specific and

proprietary services from one DRM provider could complement a DRM solution in which such specific service is missing. It is not hard to imagine a rights management system that provides specific services addressing the content production and protection, while other different system can provide the necessary rights definition and license production, necessary to govern the content [Serrão et al., 2006d].

The de-coupling of the vertical "all-in-one" rights management solutions into identified and open self-contained services is a step forward for the interoperability achievement in the rights management field. This service-oriented logic is made possible by this emerging service-oriented approach [Serrão et al., 2005c].

3 Rights Management Services as Web Services

The integration between different software elements provided by different vendors, using their own programming languages and deployment architectures, has always represented a large challenge for system integrators. A set of technologies has emerged to provide a common solution for the problem. These technologies are often referred as middle-ware.

Middle-ware is connectivity software that consists of a set of enabling services that allow multiple processes running on one or more machines to interact across a network. Middle-ware is essential to migrating mainframe applications to client/server applications and to providing for communication across heterogeneous platforms. This technology has evolved during the 1990s to provide for interoperability in support of the move to client/server architectures [Serrão et al., 2005c].

Middle-ware is computer software that connects other software components or applications. It is most often used to support the integration of large, complex and distributed applications or systems. Modern middle-ware is based on XML, SOAP, WS, and on service-oriented architectures [Cotroneo et al., 2004].

The same middle-ware characteristics that can make service oriented architectures helpful for enterprise application integration, dealing with some incompatibility architectures and formats, can also be used to address some of the DRM interoperability problems.

This chapter presents a contribution that extends the middle-ware solution for software and application interoperability problems to a similar scenario in rights management. Each of the DRM services identified previously in the generic DRM framework can be represented as standalone web-services (Figure III.2.3).

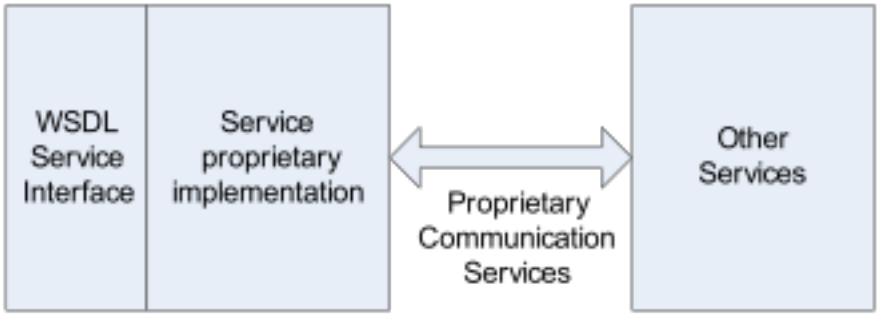

Figure III.2.3: Implementation of a generic DRM service

Each of these standalone web-services will have its own specific proprietary implementation – that is specific from the supplier of the DRM service – and will have a specific public description of the service available (WSDL) [Jamkhedkar et al., 2007]. It is envisaged that the middle-ware interoperability scenario in this case is MOM-based (Message Oriented Middle-ware). Therefore, the availability of a public web-services interface would provide the facility for services to exchange messages between them in a standardised format (using SOAP).

In more operational scenario it is possible that each of the DRM key-services providers to register themselves on a server repository (UDDI), providing their own description of their services in a normal standard format using WSDL. Each of the registered services is uniquely identified and a set of strong cryptographic credentials (X.509 certificates) are provided to each of these services. This mechanism, provides the necessary PKI mechanisms to allow different services to ensure trust each on each other. This is a critical issue to ensure both authentication and trust between the different rights management service providers [Cotroneo et al., 2004].

Scenarios in which a multiplicity of Users, Content Providers and License Issuers coexist in a common ecosystem are not very common nowadays (Figure III.2.4, Figure III.2.5). Currently, due to the vertical nature of DRM, the Users all need to share the same device, the Content Providers all need to supply the same type of content format and there can be only one specific License Server. The presented scenario is only possible, because there is a generic

WSDL interface provided by each of the License Servers, that is exposed publicly allowing first the Content Provider to identify which is the User's device to be able to supply the appropriate content type and for creating a specific license for that user and content on the appropriate License Server. Also, it will allow the device to contact the License Server to get the content license in a standard process. Although the example provided is very specific for License services it can be extended to all the DRM key-services presented before [Serrão et al., 2005c].

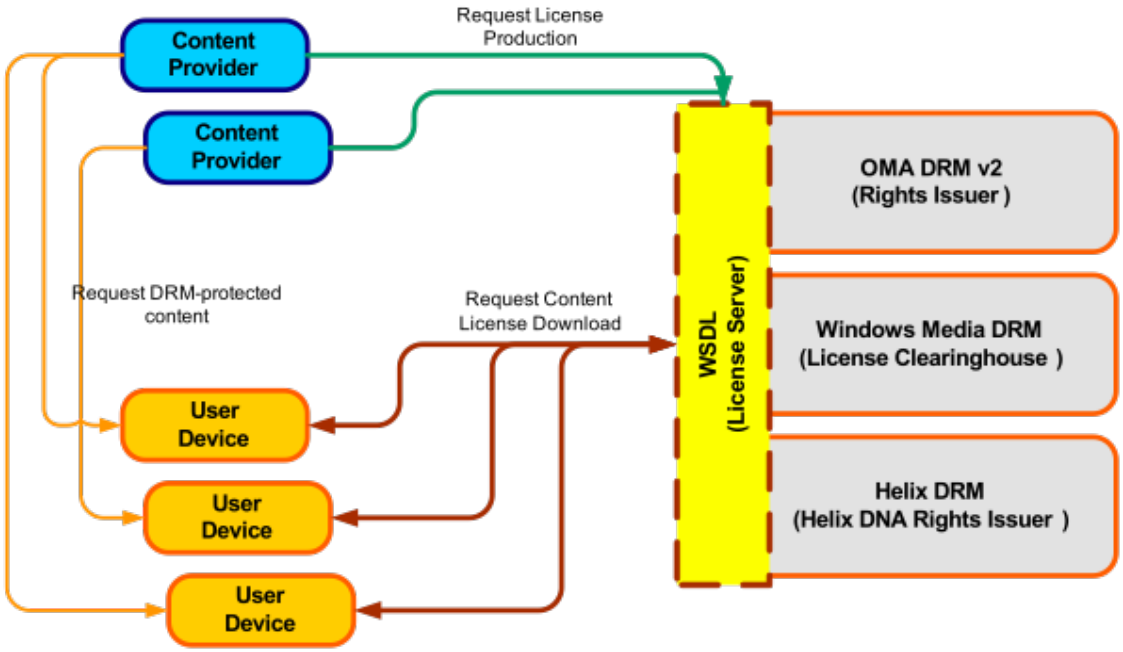

Figure III.2.4: Defining a generic WSDL interface, common to all proprietary DRM services is important because it allows the coexistence of several non-vertical applications

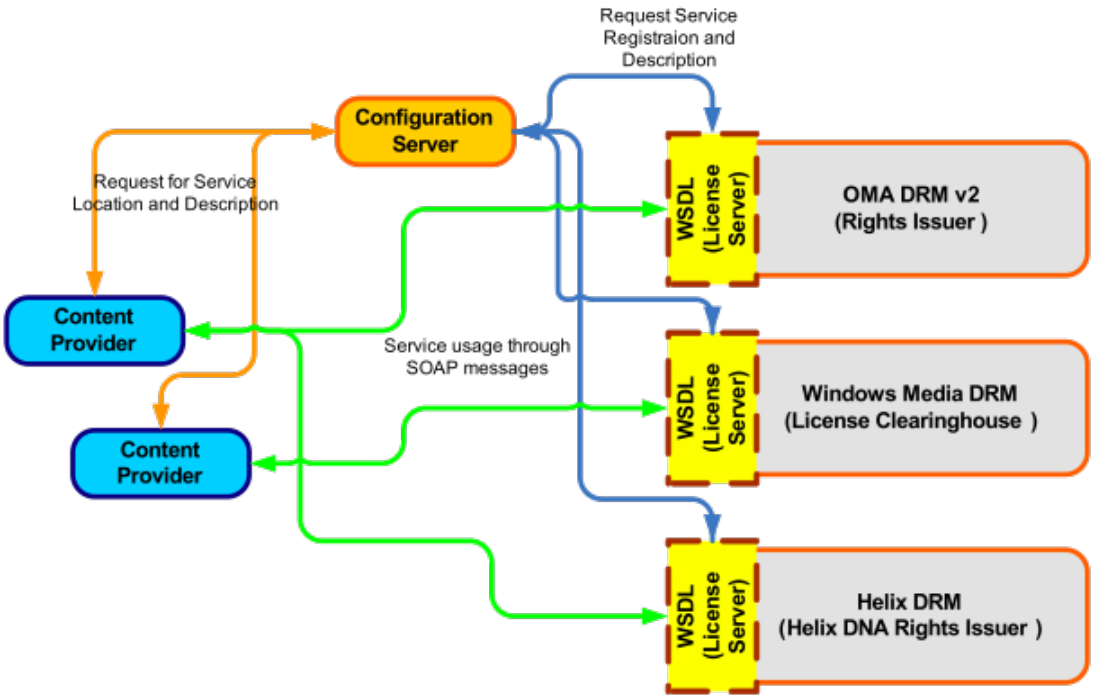

Figure III.2.5: DRM services registration and description on a registration server. When DRM services are requested, the registration server can provide to the client both the description of the service and the service credentials

This process will require that each of the DRM service suppliers to publish their own web-services description and make them available so that they can be found and used [Serrão et al., 2006d].

4 DRM interoperability using a service oriented architecture

The enterprise application integration approach through the usage of service-oriented architectures makes possible the achievement of some rights management interoperability levels [Marques et al., 2006a]. For this, the scenario where a user acquires DRM-governed content on a given content provider and wants to play it on its favourite media player will be considered. This is one of the most typical scenarios. In the current days, this operation is almost impossible to accomplish, due to the usage of different, incompatible and non-interoperable DRM systems. However, if a service-oriented approach is considered, like the one proposed on this chapter, it is possible to establish interoperability points between different DRM services and providers [Serrão et al., 2006c].

For this particular case, the following sequence of actions demonstrates how the service-oriented approach can be used to provide interoperable interfaces:

1. The media player, while trying to use the content provider DRM-governed content, analyses the content and checks the type of DRM-governance that has been applied;
2. The player contacts its local DRM broker (see Part III, Chapter 7) asking for permissions to render the DRM-governed content;
3. The DRM broker contacts an UDDI server to request information about the rights management services that are requested to render the DRM-governed content;
4. The UDDI server verifies the existence of such service and returns the necessary information to the client's DRM broker – in this information pack is the URI of the DRM service (and the WSDL of the DRM service to be used, if available);
5. The DRM broker contacts the DRM service and loads its public available WSDL. The WSDL contains the description of the functions provided by the service as well as the means to use such functionalities;
6. The DRM broker can now contact the DRM-service, using the appropriate DRM-service function (or functions) to get the necessary information to render the content (licenses and keys);
7. The DRM broker would then allow the media player to render the content (Figure III.2.6).

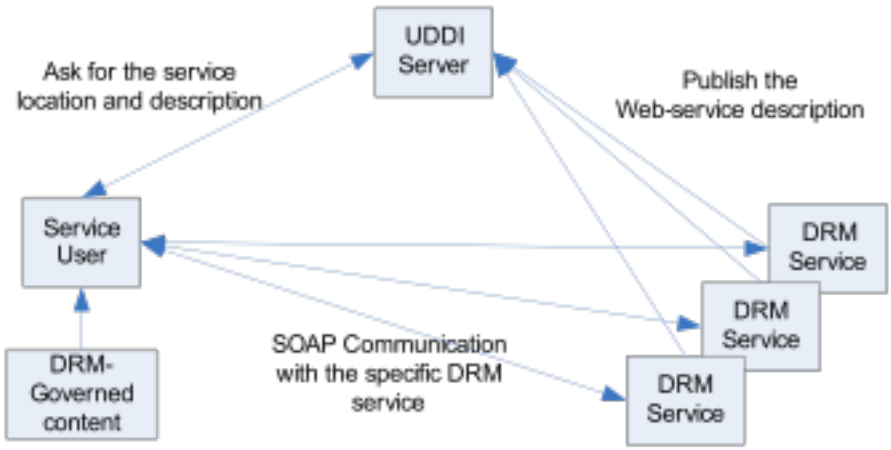

Figure III.2.6: A scenario with interoperable DRM services using the service-oriented approach (SOA) and Web-services

This is a simple scenario, where the presented approach can be used as an interoperability provider [Serrão et al., 2006d].

This SOA scenario (Figure III.2.7) has not only advantages for content end-users but also for content providers. Content providers could abstract from the DRM system that will govern their content, and therefore avoiding having to produce a multiplicity of different versions of the same content targeting different platforms and different devices [Marques et al., 2006b].

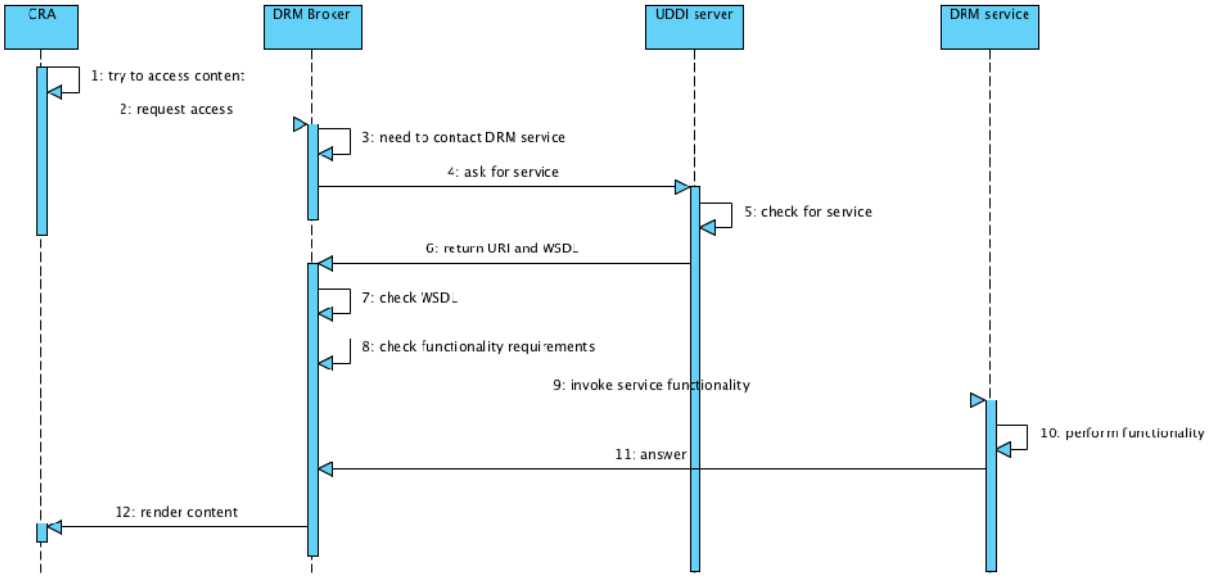

Figure III.2.7: A scenario with interoperable DRM services using the service-oriented approach (SOA) and Web-services

This is an high-level conceptual approach to the rights management interoperability problem. Many other issues need to be considered while handling this problem [Serrão et

al., 2006d]. However, the same SOA approach presented in this chapter could play an important role [Serrão et al., 2006c]. These issues can be resumed in the following topics:

- Content format: content format plays an important role in DRM. The player that renders the content should be able to read it and to understand its format in order to take the appropriate actions. In the presented scenario, the media player should be able to read specific tags or markers inside the content that signal how it is protected, its identification and probably the DRM system identification used [Ahamed and Ravinuthala, 2007];

- Security issues: from a security point of view, there are some issues that need to be taken into account. It is imperative that all the communication between the different actors of the system should occur in a secure and authenticated channel. It is important that the end user media player (user) can be authenticated by the specific DRM service in order to use the available service functions. Moreover, it may need extra authentication credentials to validate a user trying to download a license in order to use some content. Some of the used authentication mechanisms can be proprietary, or they may be provided by a different DRM system;

- Rights interpretation: rights can be expressed in many ways, and its expression may vary greatly. It is important that either the media player or the client-side DRM broker are able to interpret different rights expression languages or capable of translating between those languages. Therefore, there may exist services capable of providing either this interpretation languages or equivalent translation services;

- Content protection: different DRM-governed content may come with different content protection technologies (using different algorithms, for instance). The media player needs to know which specific algorithm has been used to protect the content, so that the inverse operation can be performed and the content used [Alberti et al., 2003]. An additional problem to this resides on the fact that the media player may not actually implement such algorithm and therefore may need to obtain it externally using a different DRM service.

It is clear that the growing vertical DRM World would benefit from a SOA approach in terms

of interoperability. This is the approach that is followed by the open rights management platform defined and specified on this work and presented on a latter chapter (Part III, Chapter 7).

5 Conclusions

The advent of Web Services has had a crucial impact on the Internet, pushing it forward to give birth to a network of services. In the Web 2.0 days, technologies such as Web Services and XML allow for an effective separation of content format and content meaning [Papazoglou and Heuvel, 2007]. This created a web scattered with micro-content islands, ready to be accessed by virtually anyone, anywhere and anytime. The appearance of the SOA paradigm was the natural evolution, opening the way to more specialised and independent digital service providers [Saunders et al., 2006].

The distributed programming field incurred also into significant changes: the ease of deployment, the simplicity of development, reusability and lightweight functionalities. To this effect normally cumbersome, proprietary and non-interoperable rights management systems can greatly benefit from these new paradigms in web application development [Saunders et al., 2006]. Not only due to those characteristics per se, but also to the fact that they could transform an overall handicapped user experience, which has been gradually degrading to the point where the Internet content consumer community has been questioning the effectiveness of DRM systems and even their core purpose. As a means of breaking out of this general wave of mistrust and activist like attitude towards DRM as a digital rights management solution, open, interoperable and standards compliant philosophy need to be created [Serrão et al., 2006d].

Using the described approach it is possible to achieve heterogeneous rights management scenarios where different rights management service providers can easily co-operate. This is only possible if the current DRM providers de-couple their tightly integrated and vertical solutions into self contained services that expose their functionalities through a well described public interface [Serrão et al., 2006c]. In the case of newly rights management service providers their integration task is simplified if they follow this approach.

All the different rights management providers can publish their services in UDDI repositories providing a description of their provided functionalities. This way, any content provider trying to make a digital governed content business model can select and integrate the services that best fit his needs.

This approach to DRM interoperability is relatively new and has been implemented and tested in several situations using an architecture that has been defined and described in the context of this thesis (see Part III, Chapter 6). It is an example of a DRM solution that follows the principles that are discussed in this chapter.

This new SOA approach can provide an effective contribution to the interoperability achieved in this system as well as to the easy and straightforward adaptability to different implementation scenarios, and thus to a possible increase of enjoyable user experience.

Chapter 3. Using Public-Key Infrastructures towards Rights Management interoperability

1 Introduction

Rights management solutions are composed of a chain of hardware, software and technology services that govern the authorised use of digital objects and manage any consequences of that use throughout the entire life cycle of the object [Rosenblatt et al., 2001].

From a security point of view, two major aspects need to be considered in any rights management solution. The first one refers to the digital object protection (most commonly known as copy-protection), in which the digital object is packaged in a specific container that is locked, trying to prevent non-authorised copies or modifications, making usage of strong cryptographic algorithms. The second important aspect refers to the establishment of a trustworthy environment that must exist between the different actors, devices and software components throughout the entire digital object life cycle. Therefore, rights management involves the establishment of trust environments between the different components and actors. This is not only important when considering a single self-contained rights management solution but it is fundamental when the interoperability of different rights management solutions is at stake.

When considering diverse rights management scenarios where the different components have to interoperate, trust relationships must be established between all of them. These

trust relationships will allow the different components and solutions to trust each other, through the use of strong authentication mechanisms.

Currently, the rights management panorama is mixed. There are some digital object rights management solutions that put a strong emphasis on the design of secure and trustworthy solutions, while others are more relaxed on this aspect [Serrão et al., 2006h]. Some solutions claim that the obscurities of their processes are their strongest security point, while others fully specify them. These processes are used to both protect the digital assets as well as to establish and manage the trust environment.

This chapter is focused on this second aspect. This is a very important and interesting topic for most of the rights management solutions, because they operate in open and non-trusted environments (such as the Internet) and they need to be able to establish the necessary mechanisms to "transform" and "adapt" these non-trusted environments in trusted ones. The establishment of this trusted environments is not an easy procedure. More, the process to establish such environments is sometimes proprietary, making most of the interoperability attempts useless [Erickson, 2003].

In this chapter, contributions made on the usage and applicability of standard Public-Key Infrastructure solutions (in particular PKIX) as a rights management interoperability driver are presented. In a first stage this chapter presents how PKI services can be helpful has an interoperability driver for different rights management services and components, in particular for the establishment of trusted environments [Adams et al., 2000]. Two different strategies are exploited. The first considers the existence of a single and central PKI service that it is responsible for providing a consolidated trust environment to all the rights management actors and components. The second strategy considers the existence of multiple PKI services that are particular to a specific rights management solution. In this second case, this thesis proposes the usage of a PKI services broker that is responsible for managing the interoperation between the different PKI services, therefore providing the establishment of a general trust environment [Serrão et al., 2006h].

This chapter also contains a contribution to the study of the authentication and trust mechanisms of two different open rights management platforms. This study involved the analysis and comparison of the two open rights management platforms in terms of actors

and components registration and authentication. The details of such trust establishment operations are presented and the most relevant commonalities and differences are identified [Serrão et al., 2006h].

Based on the study of both selected open DRM platforms, another contribution was presented on the usage of PKI brokering components [Serrão et al., 2007a] that will allow both open DRM platforms to interoperate in terms of actors and components registration and authentication, therefore establishing a common secure trust environment between them. Although this contribution is specific for the two selected open DRM platforms it can be extended to accommodate any other open DRM platform.

2 Rights Management and Trust

Rights management interoperability is important in three main aspects: rights management development, deployment and acceptance [Camp, 2003]. In the rights management development aspect it will allow different DRM platforms to evolve in an independent way, although providing mechanisms that will permit digital object governance by both in a seemliness manner. The second aspect of rights interoperability will allow digital content producers and providers to put in place their digital assets available through digital channels and ensuring that their creations, although deployed on a specific rights management platform, can also be used and governed by others. Finally, the third aspect is crucial for rights management solutions success. Without general user's acceptance, rights management cannot be simply imposed - users will develop and use circumvention mechanisms. Interoperability will have to offer end-users the possibility to use governed digital objects in a similar way they do with non-digital objects and therefore generate a consensus around the rights management technology (which currently does not exist).

DRM interoperability is conceptually simple, but hard to implement. There are many reasons for this, however the key management and trust establishment are amongst the most cited reasons.

A rights management system is a special purpose information system, which manages a different set of actors (content providers, content authors, distributors, license providers and

others), devices (computers, mobile devices, phones) and software components [Buyens et al., 2007]. In order to carry its functions, a secure and trustworthy environment has to be set-up to allow the establishment of secure and authentication channels between the different stake-holders of the DRM processes. This trust environment can be achieved through the usage of public-key mechanisms, allowing the issuance and management of digital credentials to the different actors, devices and software components. When these credentials are exchanged between them, they can be trustworthy because they share some common trust points [Serrão et al., 2007a].

In a self-contained rights management environment, in which only one DRM solution is used, trust environment establishment is facilitated due to an "absolute" control of the different elements on the system [Foroughi et al., 2002]. However, if this DRM environment is extended to support a multiplicity of other rights management solutions, the establishment of such environment is not easy to achieve. This has a direct impact on the digital objects rights management interoperability. This impact reflects on the following aspects of DRM interoperability [Serrão et al., 2007a]:

- **Security mechanisms implementation differs between different DRM**. The security mechanisms implemented by the several rights management solutions differ. Each DRM solution provider implements a set of their own specific security requirements and the necessary security algorithms and protocols. Since different DRM solutions have different specific requirements they only implement the basic security mechanisms to answer such requirements. When different DRM solutions try to interoperate, one has security mechanisms that the other cannot fulfil. Therefore, interoperability between them is not possible.
- **Security protocols differ**. The different DRM solutions implement their own specific protocols for handling most of the rights management processes, such as for registering new users on the system or to download licenses for digital object usage. These protocols have their own format, transport mechanisms and security implementation.
- **Cryptographic keys creation and management differs**. In a normal security

environment the different software components and devices generate key material. This key material is different from DRM solution to DRM solution, not only in terms of the key type and size, but also on the way this key is managed through its life-cycle (where are the keys stored, how to access them, how they can be revoked and others).

- **Digital credential issuance and management is different**. Digital credentials are used for authenticating actors, software components and devices. These credentials are usually issued by an entity within the rights management system itself, to establish trust relationships between the internal DRM services. Extending this model to a multiple DRM scenario requires some extra digital credentials management requirements (such as bridging credential issuance entities or creating cross-certification policies).
- **Security policies are different**. Behind any security implementation should be a security policy. Security policies help the definition of many key aspects of the security environment that go behind the mere technological details. Most of the current DRM solutions share a total absence of these policies.

There is currently a solution offering a multiplicity of security services, which can be used, by current and future digital object rights management solutions – Public-Key Infrastructures (PKI). PKI offers a set of basic/core services useful to DRM [Ellison, 2002]. These services can be resumed in the following:

- **Authentication services**: authentication is used to assure that an entity is truly the one that claims to be. It can be used in two major contexts: entity identification and data originality identification.
- **Integrity services**: these services offer the warranty that data has not been altered and allow knowing if and when data has been altered.
- **Confidentiality services**: this is a must in secure communications through open environments. It allows data to be read only by the entity that is the legitimate recipient of it. Confidentiality is a requirement when data is stored in a support from

which an authorised individual can easily read that and when data is copied to a device that might fall in the hands of a non-authorised individual.

These services are essential for rights management systems implementations to ensure an overall confidence level on the system, both at the content producers and distributors and the end-users side. Therefore, it will be possible for DRM solutions to make use of the available PKI services [Ellison, 2002] to implement their own security mechanisms, which include secure key management procedures.

PKI can support many of the current rights management functions on what concerns security and trust. DRM implementations can leave the burden of implementing complex security operations to underlying PKI solutions. For interoperability sake, PKI will also need to interoperate and to exchange information that allow trust environments establishment in a DRM transparent way. This can be an issue for DRM interoperability as well, but the current trend is that all the DRM suppliers can or may support the PKIX model. The impact of this is minimised on DRM interoperability [Serrão et al., 2006h].

3 PKI and DRM Interoperability

In order to achieve a trust environment within the same or different rights management solutions, two different approaches could be followed:

1. To use a single PKI service that is shared by all the different DRM platforms;
2. Each DRM platform uses their own internal PKI, and a brokering mechanism handles specific trust requests from one PKI to another.

Each of the proposed approaches have advantages and disadvantages [Serrão et al., 2006h]. They both offer a way for DRM architectures to share trust information in a transparent way and to establish trust relationships between the different DRM stake-holders to provide an interoperability layer between them.

In the first approach, a single PKI solution exists and is common to all the existing DRM solutions (Figure III.3.1). This would provide a single mechanism for DRM to engage their security and trust operations. Since all the DRM platforms depend on the same single PKI, it

would be possible to establish immediate trust between the different DRM actors, devices and software components. Using this approach the trust environment would be assured because the credentials used would always be trusted to all the different elements. But, although highly desirable, this is a very low probability scenario.

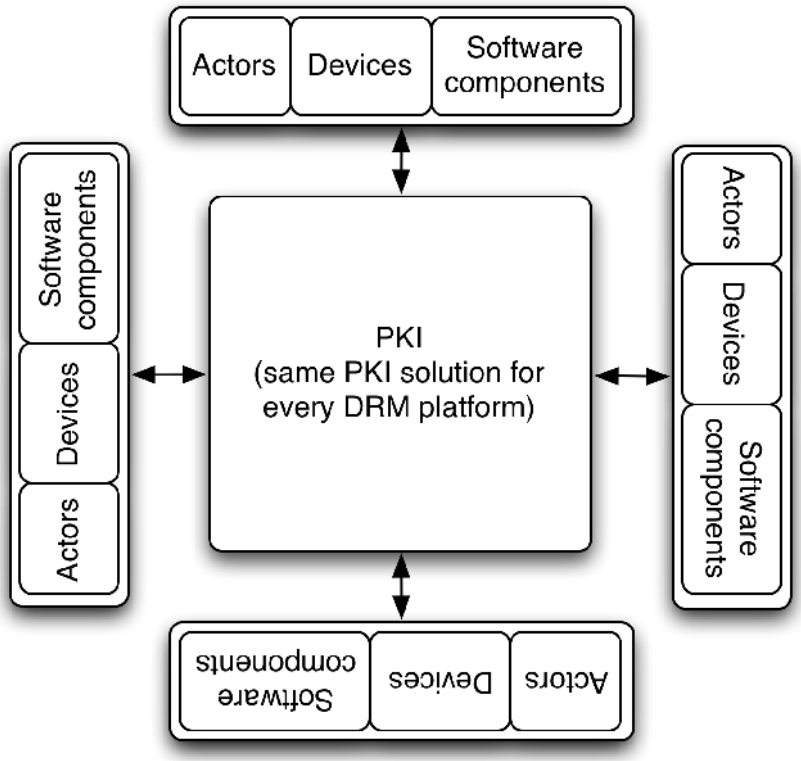

Figure III.3.1: A single PKI solution for multiple DRM platforms. A way to allow all the actors, devices and software components to rely on a single PKI solution to provide security and trust between all of them

It is improbable that a single PKI solution dominates the market in such a way that all the DRM providers will use the same PKI (this has never happened in the real World). It is also difficult that all the DRM providers will trust on the same PKI supplier and therefore they will prefer their own PKI solution. From a technical point of view it is also difficult, because the centralised PKI would have to handle with all the DRM actors, devices and software components on the World [Serrão et al., 2006h].

It is clear that the alternative is for each DRM implementation to choose their specific (local) PKI solution – as it already occurs today. Therefore, if interoperability is needed for both DRM and PKI the alternative is to have an intermediate broker that will be capable of interconnecting the different PKI solutions selected by the different DRM providers (Figure

III.3.2).

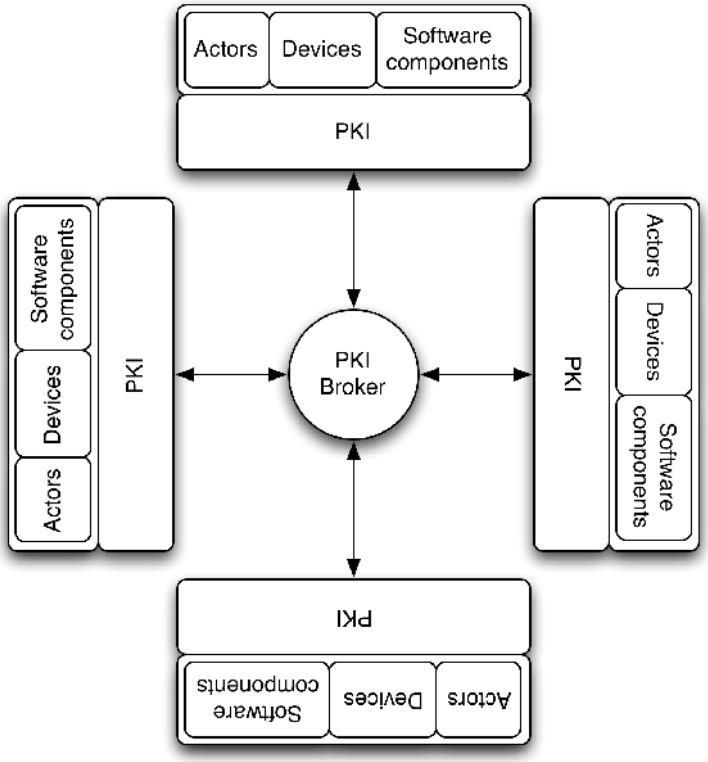

Figure III.3.2: DRM providers select their own PKI solution. An intermediary broker PKI will be necessary to handle trust requests from one DRM solution to another. This intermediary broker will have to know some details about the trust providing PKI in order to be able to carry its functions

"Locally" each of the DRM platforms would rely on their own PKI trust processes, offering registration and authentication mechanisms to all the DRM actors, devices and software components [Jonker and Mauw, 2004]. In a "local" DRM operation scenario, the different elements of the DRM platform rely on their own PKI for establishing trust between them. For instance, a device that downloads a governed digital object and wants to obtain a license from a license issuer will have to send its digital credentials to the license issuer. The license issuer receives the device credentials and sends it to the PKI sub-system for validation. The sub-system returns the validity of the credentials stating if the device is trustworthy or not. If it is, the license issuer generates the license for the device. This is a very simple and straightforward example of a DRM process. When elements from different rights management solutions need to establish trust between them, then the PKI brokerage system

will assert this trust by routing the authentication requests for each of the DRM PKIs. This will result in the trust bridge that is provided by the broker allowing different DRM elements, governed by different PKIs to be able to trust each other.

As an example, if a device acquires some digital object from other device, and if the digital object is not governed by the same DRM system then it will have to contact a different DRM license issuer that is not part of the device's trust environment. The device, in order to obtain the license, will contact the license issuer. Since the license issuer cannot trust the device directly because they do not belong to the same trust environment [Cooper and Martin, 2006a], the license issuer passes the device credentials to the PKI system. The PKI cannot trust these credentials and will have to contact the PKI broker, which then tries to map the issuer of the credentials with the location of the PKI. The PKI broker then routes a validation request to the appropriate PKI provider, which validates the credentials and responds to the PKI broker. The PKI broker, based on the devices' PKI answer can assert that the credentials are in fact valid and then answers to the license issuer PKI. The license issuer can now trust on the device and produce a valid license for the device to be able to access to the content [Serrão et al., 2006h].

This scenario brings open interoperability to the different rights management platforms at the PKI level. The following describes in a more formal way this approach (Figure III.3.3).

Assumptions:

1. DRM1 Device has a key pair: K_{pub}^{Device}, K_{priv}^{Device};
2. DRM2 License Issuer has a key pair: $K_{pub}^{LicIssuer}$, $K_{priv}^{LicIssuer}$;
3. DRM1 Device has a certificate issued by the DRM1 PKI: $Cert^{DRM1PKI}(K_{pub}^{Device})$;
4. DRM2 License Issuer has a certificate issued by the DRM2 PKI: $Cert^{DRM2PKI}(K_{pub}^{LicIssuer})$;
5. All the PKI are PKIX-based and use X.509 digital certificates;
6. PKI Broker has a key pair: $K_{pub}^{PKIBroker}$, $K_{priv}^{PKIBroker}$;
7. DRM1 PKI and DRM2 PKI are registered at the PKI Broker;

8. DRM1 PKI has to have a certificate from the PKI Broker: $Cert^{DRMBroker}(K_{pub}{}^{DRM1PKI})$;
9. DRM2 PKI has to have a certificate from the PKI Broker: $Cert^{DRMBroker}(K_{pub}{}^{DRM2PKI})$.

Protocol:

1. The DRM1 Device has acquired some digital object which is not governed by the same DRM;
2. DRM1 Device sends a message to DRM2 License Issuer to download the license for the digital object and their credentials: $licenseDownload(contentID, Cert^{DRM1PKI}(K_{pub}{}^{Device}))$;
3. DRM2 License Issuer sends the DRM1 Device credentials to the DRM2 PKI for validation;
4. DRM2 PKI has no way to validate the request, because the credential has been issued by other PKI. Therefore the DRM2 PKI asks to the DRM Broker to try to validate the credential: $validateCredentials(Cert^{DRMBroker}(K_{pub}{}^{DRM2PKI}), Cert^{DRM1PKI}(K_{pub}{}^{Device}))$;
5. The DRM Broker validates the requesting PKI credentials, and checks the credentials sent by the device, checking the issuer PKI. It resolves the location of this PKI (DRM1 PKI) and sends it a validation request: $validateRequest(Cert^{DRM1PKI}(K_{pub}{}^{Device}))$;
6. DRM1 PKI receives the request and then validates it, returning an answer to the PKI Broker;
7. PKI Broker receives the answer and sends the result to the requesting PKI (DRM2 PKI);
8. DRM2 PKI receives the answer from the PKI Broker asserting that DRM1 Device can be trusted;
9. DRM2 License Issuer generates the license and returns it to the DRM1 Device.

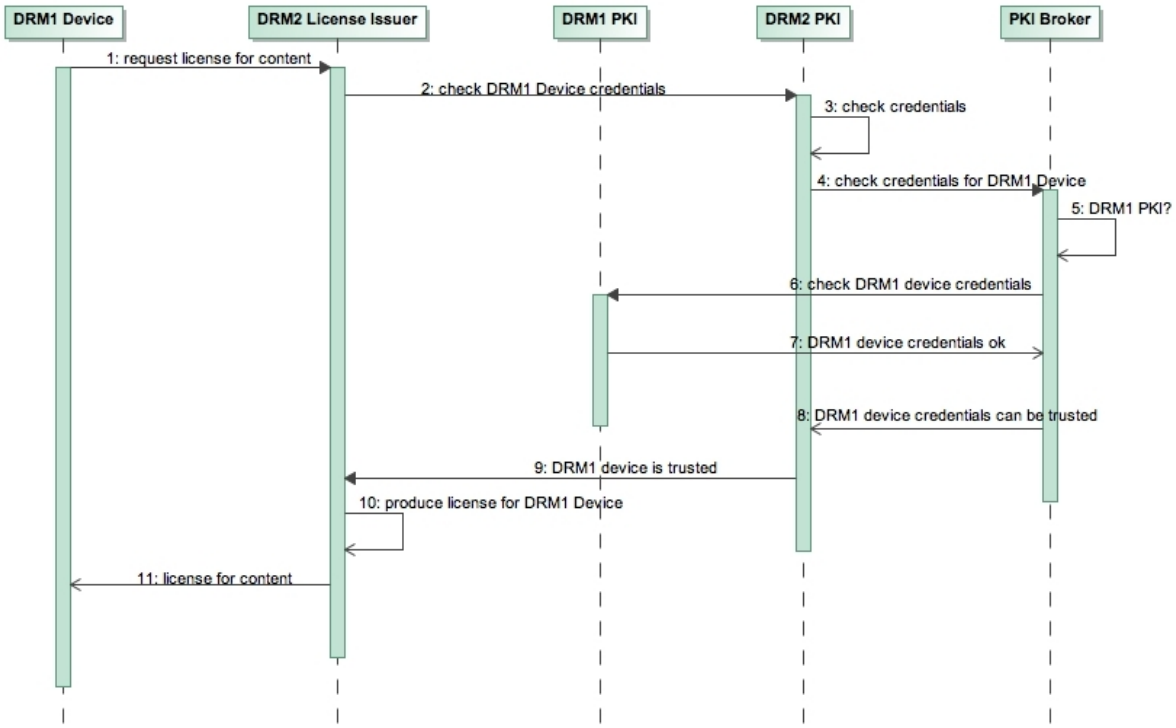

Figure III.3.3: Operation sequence while contacting the PKI broker to establish trust between two different elements governed by different DRM platforms. The PKI broker will be responsible for requesting the information needed to validate the trust from on PKI to the other

This protocol, although specific for the rights download function, can be extended to support further DRM operations. This protocol and the existence of the DRM PKI broker provide an open interoperability mechanism for rights management trust services through PKI [Cooper and Martin, 2006b].

This broker-based approach can be extremely useful in the establishment of common interoperable trust environments between the different components, devices and actors of the different rights management solutions. The following sections of this chapter, illustrate how this broker-based PKI solutions can bring registration and authentication interoperability to open rights management solutions [Serrão et al., 2006h].

4 DRM interoperability and PKI authentication mechanisms

This section focuses on the contribution to the development of interoperable PKI

authentication mechanisms for open DRM platforms (see Part III, Chapter 4). It will be focused both on open-specification and open-source DRM platforms to identify and describe the components and users registration and authentication mechanisms that are provided by the different components of DRM platforms.

In this contribution, one open-source implementation (OpenSDRM) and one open-specification DRM platform (DMAG/MIPAMS) were considered. OpenSDRM is the open-source rights management implementation [Serrão, 2004] that is one of the contributions of this thesis (see Part III, Chapter 6), while the DMAG/MIPAMS is the rights management solution that is being developed by the DMAG group (see Part II, Chapter 1) in which the author of this thesis is currently integrated.

4.1 OpenSDRM – Open and Secure Digital Rights Management

OpenSDRM is an open specification and open-source implementation of a DRM platform that is one of the contributions of this thesis (see Part III, Chapter 6). It relies on a distributed philosophy in which the different components are implemented in a self-contained way encapsulating a set of specific functionalities [Serrão, 2004]. These components are represented by web-services, with a public WSDL description, deployed either in the same or different hardware platforms remotely distributed. Messages exchanged between the different components are SOAP-based over SSL –secured and authenticated connections.

The OpenSDRM platform uses two important concepts that will be largely referenced afterwards: Actors and Components [Serrão et al., 2003b]. An **Actor** is a person or an organisation that uses a Component. A **Component** is a set of software and hardware tools co-operating to offer a set of specific DRM-related functionalities. More details about the OpenSDRM specification and design are provided in Part III, Chapter 6.

4.1.1 Components Registration

From a security point of view, OpenSDRM requires that each of the DRM platform components to be registered and certified as valid before interacting with any of the other

components of the platform [Serrão et al., 2003b]. Since each of the DRM components are installed on a web-server, each of these components needs to be certified. The certification process for each of the components involves the creation of a key pair $(K_{priv}^{Comp}, K_{pub}^{Comp})$ and the generation of a new X.509 digital certificate issued by a Certification Authority $(Cert^{CA}_{Comp})$. The registration process works in the following manner:

a) The DRM component generates a new key pair $(K_{priv}^{Comp}, K_{pub}^{Comp})$ and securely stores the K_{priv}^{Comp} protected with a password;

b) The DRM component generates a Certificate Signing Request (CSR) to be sent to a Certification Authority (CA). This CA can be an internal CA, or a publicly commercially available CA;

c) The request is sent to the CA;

d) The CA verifies the data included in the CSR and registers the new DRM component. A X.509 digital certificate is issued for the DRM component ($Cert^{CA}_{Comp}$);

e) The new certificate is sent to the DRM component;

f) The DRM component stores the certificate and installs it.

4.1.2 Components Mutual Authentication

A CA certifies each of the DRM components. Each of the DRM components contains a list of trustworthy CA, which will allow each component to trust each other. This mechanism is used to establish a mutually secure and authenticated communication channel (SAC) between the different DRM components. This is crucial to ensure a first secure communication layer [Serrão et al., 2007a]. All further communications within the DRM platform are handled over a mutually authenticated and secure SSL channel (Figure III.3.4). Whenever a function within a DRM component invokes another function on other DRM component this SAC process is repeated.

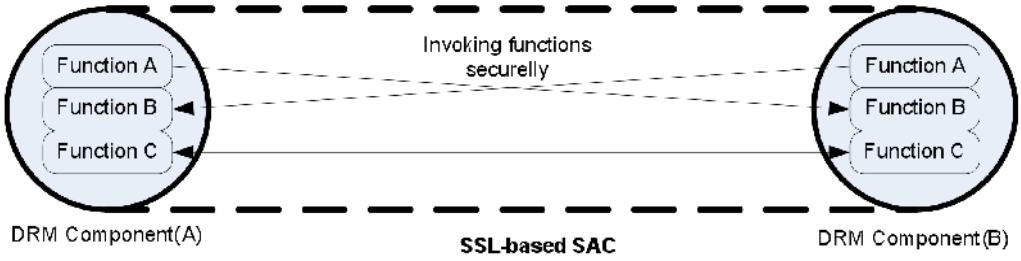

Figure III.3.4: Establishing a SAC between different DRM components

4.1.3 Actors and components registration

The secure and authenticated communication between the different DRM components is established in the way it was presented in the previous sections. Each of the different DRM components can contain and represent different functions in the overall DRM architecture. In the OpenSDRM case, the interaction between the different functions of different DRM components demands a new layer of authentication. This new layer is called application-level authentication. In this layer, the different actors that interact with the different functions of the DRM components need to be registered and authenticated [Serrão et al. 2007a].

In OpenSDRM, there is an architectural component called Authentication Server that is responsible for handling both the DRM component functions and the actors' registration and authentication.

The Authentication Server (AUS) is used to register and authenticate the Actors and the Components of the system. Every time a new component enters the system, it can only start interacting with other components after being properly registered. Also every actor needs to be registered with the system. The main functions of the component are: (a) to register other components in the system capable of providing different functionalities. This also includes functionalities to update and to delete/revoke components on the system; (b) allow the registration of the users that will interact with some of the components in the system. It is also used to update and delete/revoke users on the system; (c) verify if a user has or not a valid installed wallet on its system; and (d) verify and validate the available payment gateways (mechanisms) that are registered on the system.

At this level, OpenSDRM uses a proprietary format of digital certificates. It uses an X.509 certificate format mapped into a specific XML structure. This XML structure has the following

composition and structure [Serrão et al., 2006h].

```
<certificate>
     <issuer>
          <identifier/>
          <public-key>
               <n/><e/>
          </public-key>
     </issuer>
     <subject>
          <identifier/>
          <public-key>
               <n/><e/>
          </public-key>
     </subject>
     <validity>
          <not-before/>
          <not-after/>
     </validity>
     <signature/>
</certificate>
```

This <certificate> XML structure contains certain particularities:

i. <issuer> corresponds to the AUS entity that has signed and issued the certificate;

ii. <identifier/> is a unique identifier of the entity (either <issuer> or <subject>) and corresponds to the fingerprint of the public-key components;

iii. <n/> and <e/> are the components of the public-key represented in hexadecimal or Base64 format (modulus and public-key exponent);

iv. <not-before/> contains the date and time in UTC format, indicating the issuance date of the certificate;

v. <not-after/> contains the date and time in UTC Format, indicating the expire date of the certificate;

vi. Finally, <signature/> contains the digital signature generated by the <issuer> of all data within <issuer> and </validity>, in hexadecimal or Base64 format.

Also, OpenSDRM uses XML structures for representing both the public (<public-key><n/><e/></public-key>) and private keys (<private-

key><n/><e/><d/><p/><q/></private-key>). The private key contains the modulus, public-key exponent, private-key exponent, and the original prime numbers selected, and may be optionally ciphered, using a secret password

The process to register a DRM component functionality is the following (this information is exchanged using a previously established secure and authentication channel between the DRM component and the AUS):

i. DRM component creates a new key-pair (K_{priv}^{FComp}, K_{pub}^{FComp}), and stores the private key protected by a secret key (AES): $Sk[K_{priv}^{FComp}]$. This secret key is created using the fingerprint of the pair (login and password) used to setup the component;

ii. The DRM component generates a unique identifier, hashing (SHA1) the public-key components: <n/> and <e/> (fingerprint)

iii. The public-key and the unique identifier are sent to the AUS requesting the certification: K_{pub}^{FComp} and FCompId;

iv. AUS verifies the received data, stores it and generates a certificate (XML version) that contains the data that was previously identified and sends it back to the component: $Cert^{AUS}_{FCompA}$;

v. The DRM component receives and stores it.

The Actors registration on the OpenSDRM platform is mediated through a broker called Wallet (this will be exploited and described in Part III, Chapter 7). This broker is the software responsible for interacting with the other DRM components functions and in particular with the AUS [Serrão et al., 2005b]. The Actor registration process in the AUS is the following:

a) The Actor selects a login and a password for the Wallet broker. This login and password, together with some special information retrieved from the Actor device, are used to create a unique secret-key (128 bits MD5 hash value) that is used to create a secure storage database to hold private information: MD5(login, password, DeviceInfo) = Sk;

b) The Wallet broker creates a key-pair, in XML format: (K_{priv}^{Actor}, K_{pub}^{Actor}). The

K_{priv}^{Actor} is securely stored on the database: $Sk\ [K_{priv}^{Actor}]$.

c) The Wallet broker generates a unique identifier, hashing (SHA1) the public-key components: <n/> and <e/>;

d) The public-key and the Actor unique identifier is sent to the AUS requesting the certification: K_{pub}^{Actor} and ActorId;

e) AUS verifies the received data, stores it and generates a certificate (XML version) that contains the data that was previously identified and sends it back to the Wallet broker: $Cert^{AUS}_{Actor}$;

f) The Wallet broker receives and stores the certificate.

4.1.4 Components and Actors Authentication

In this section, it is described how OpenSDRM handles Components (in terms of its functionalities) and Actors authentication.

Whenever a function in a DRM component requires using another function in the same DRM component or on an external DRM component, it sends his AUS certificate as part of the message. This certificate is reviewed by the remote DRM component function and is checked at the AUS. This checking procedure is important to assure that the requesting DRM component certificate has not been previously revoked by AUS (Figure III.3.5) [Serrão et al., 2007a].

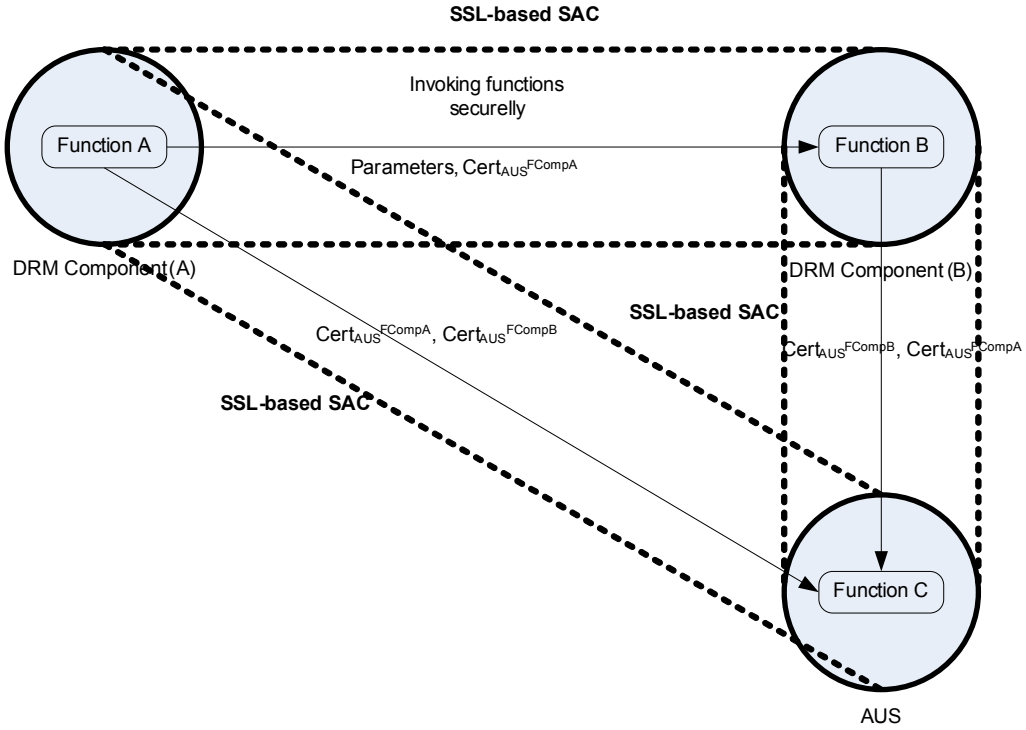

Figure III.3.5: Establishing a SAC between multiple different DRM components

Whenever a specific DRM component functions are invoked it is possible to request also the certificate from the remote component, to ascertain that it has been certified. To verify that the certificate is still valid and has not been revoked the DRM component function may also contact the AUS [Serrão et al., 2006h].

Actors also need to authenticate to DRM components when requesting some local or remote DRM functionalities. The authentication is performed through the AUS to ensure that the Actors credentials are not revoked – this is similar to what happens with the Online Certificate Status Protocol (OCSP) (Figure III.3.6). The process that OpenSDRM uses to authenticate users is the following:

i. The Actor, uses its Wallet broker to access his credentials: $\texttt{Cert}^{\texttt{AUS}}_{\texttt{Actor}}$;

ii. The Actor requests the authentication or any other operation on a DRM component using $\texttt{Cert}^{\texttt{AUS}}_{\texttt{Actor}}$;

iii. The DRM component receives the Actor certificate and connects to the AUS (that issued the Actor certificate) to validate it. In the process it sends its own certificate ($\texttt{Cert}^{\texttt{AUS}}_{\texttt{FComp}}$) and the Actor certificate ($\texttt{Cert}^{\texttt{AUS}}_{\texttt{Actor}}$);

iv. AUS validates both certificates: one to prove the DRM component identification and the other to check if the Actor certificate has not been revoked;

v. The result is returned to the DRM component and the operation is performed.

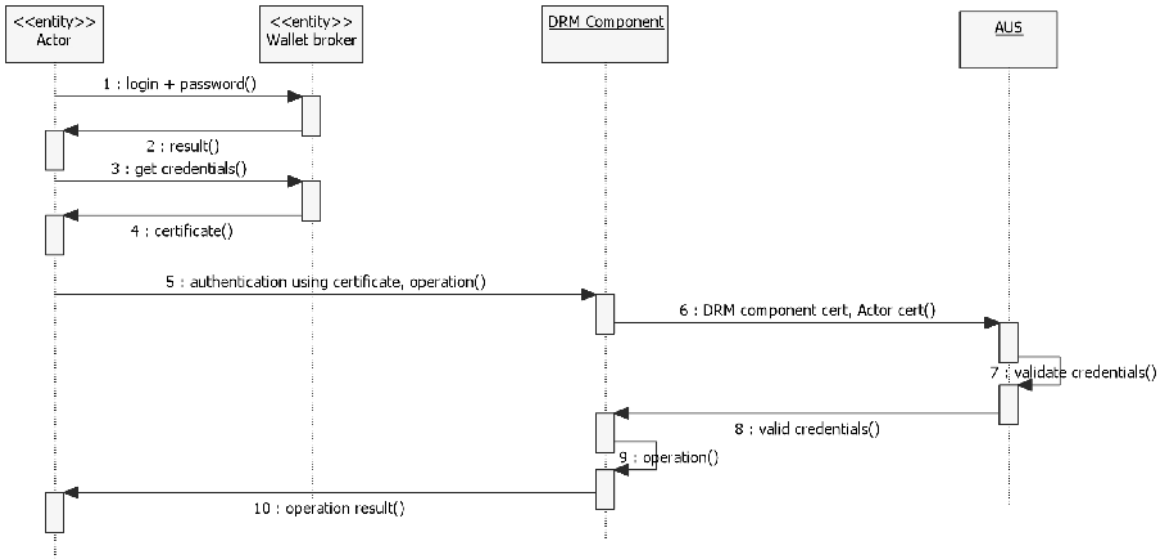

Figure III.3.6: Actor's authentication through the AUS

4.2 MIPAMS – Multimedia Information Protection and Management System

The MIPAMS architecture was already described in the Part II, Chapter 8 and is used to manage multimedia information taking into account digital rights management and protection [Torres et al., 2006]. The architecture consists of several modules or services, where each of them provides a subset of the whole system functionality needed for managing and protecting multimedia content. DMAG-MIPAMS is a service-oriented DRM platform and all its modules have been devised to be implemented using the web services approach, which provides flexibility and enables an easy deployment of the modules in a distributed environment, while keeping the functionality independent from the programming language and enabling interoperability. As in the OpenSDRM case, this rights management platform also deals with different Actors and components [Torres et al., 2006].

4.2.1 Component Certification

From a security point of view, the MIPAMS rights management solution requires each component to be verified and certified in the system before being able to deploy it [Torres et al., 2005]. Once certified, the component will hold a X.509 server digital certificate and a private key that will allow the establishment of secure communication with other components or actors in the system [Torres et al., 2006]. The certification process is performed in the following manner:

i. An Actor requests the certification of the component by submitting it to a verification process and selecting a password for the generation of a PKCS#12 package [PKCS-12, 1999];

ii. A verification process is performed over the component to check that it follows the system guidelines and acts as expected;

iii. A certificate for the component is requested to the Certification Authority (CA);

iv. The CA generates the component key pair and certificate and packs them together into a PKCS#12 package to be delivered to the component certification requester. The PKCS#12 is protected with the password chosen by the requester;

v. The component certificate and private key can be used to deploy the new component in the system.

In this way, each component is verified and owns a certificate issued by a common CA. A DRM component will use the component certificate to authenticate itself as a client or server when interacting with other components or actors in the system. For that purpose, each component owns a trust store file, which contains the CA certificates on which the component will trust when any client (component or actor) with a certificate signed by that CA tries to establish a communication towards it [Torres et al., 2006].

4.2.2 User registration and authentication

The Supervision Server is used to register and authenticate the Actors of the system. Any user must be registered in the system in order to be able to interact with the different

components. User information is stored in the Supervisor Server and is used for further verification purposes. Once an Actor is registered, the corresponding CA is requested a X.509 user certificate for the actor, which can be used to authenticate himself. The main functions of Supervisor Server component are: (a) authenticate Actors; (b) authenticate installed tools; (c) verify client tool installation attempts against registered tools features; (d) register new installed client tools (tool and device fingerprint); (e) request installed tool certificate to the Certification Authority; (f) receive and store action reports [Torres et al., 2006].

Every actor has an associated status in the Supervisor component that is used to determine whether it is blocked or not in the system when interacting with the server part. The user status can be modified if some critical operation attempt is detected.

Any actor using a tool will need to select his user certificate in the tool in order for the tool to know which Actor it is dealing with. The client certificate is used to extract the client information, as the user system unique identifier, which is then included in any request that goes from the client tool to the server part [Serrão et al., 2007a].

The communication between the client tool and the server part of the system is performed through the establishment of a secure channel established by means of a tool certificate and the server component certificate. The tool certificate is obtained during the first usage of the tool, after it is installed in the client device.

Server components will trust on client tools by trusting on the CA that signed their certificates. In order to authenticate Actors, the client application will send in the SOAP message the Actor user identifier, which is extracted from the Actor client certificate. In this way, Supervisor Server will authenticate the Actor in the system and verify its status.

The client certificate could be also used to provide security at the application level, something which is currently not present in MIPAMS architecture. By means of a digital signature on the transmitted information it is possible to provide a second security layer at the application level [Serrão et al., 2007a].

4.2.3 Tool Registration

All client Tools in the framework must be verified to accomplish a series of guidelines, which

are checked before registration is done. Once verified, each tool is registered for being potentially installed by Actors. During the registration phase, a fingerprint of the software tool is estimated so that its integrity can be checked later when the tool is installed and certified on a specific device.

4.2.4 Tool certification

The certification of an installed tool in MIPAMS is a necessary step for that tool to work. Before an Actor is able to run and use a tool, the tool must request the Supervisor Server to be certified as an "installed tool". Before installation, the tool integrity will be checked by comparing its fingerprint to the one stored during the tool registration process. Once installed, some information concerning the installation of the tool and the device (tool fingerprint) where it is installed is extracted [Serrão et al., 2007a].

Once an Actor successfully certifies a tool, any Actor in the system who owns a valid user certificate can use it. Blocked users cannot use tools in the system.

In order to have a secure communication for the certification request, the Actor client certificate is used.

The tool certification process (Figure III.3.7), is the following: (1) an actor or the tool itself requests the installed tool certification; (2) the tool computes a hardware and software fingerprint; (3) the tool uses the client certificate to request certification to Supervisor Server; (4) the Supervisor Server verifies the Actor credentials and tool software fingerprint against registered tool fingerprint; (5) the Supervisor Server generates a tool unique identifier; (6) the Supervisor Server requests a tool certificate to the CA by sending the tool identifier; (7) the CA generates the tool key pair and the certificate with the tool unique identifier as the CN; (8) the CA sends a PKCS#12 (tool certificate and private key) package to Supervisor server, which is protected with the user ID as the password; (9) the Supervisor stores the installed tool fingerprint for future verification purposes; (10) the Supervisor sends the PKCS#12 to the tool; (11) the tool receives the certificate and private key, stores them and activates itself; (12) the tool is finally certified.

As it was explained previously, the communication between the client tool and the server

part of the system is performed through a secure channel established using the tool certificate and the server component certificate [Serrão et al., 2007a].

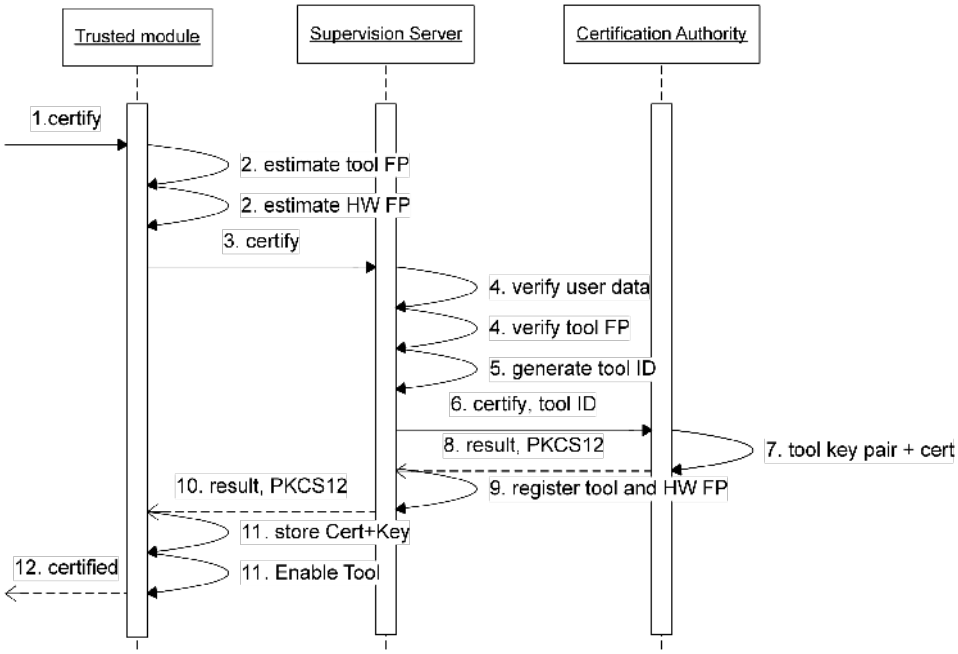

Figure III.3.7: MIPAMS client tool certification process

4.2.5 Actors and Client components authentication

Any actor in the system is authenticated in two manners: 1) through the selection of its user certificate in the client application; 2) through the user identifier, extracted by the application from the certificate.

Client components are authenticated in two ways: 1) through their tool certificate; 2) through the same manner as users, using their tool identifier.

This way, whenever a tool is blocked or revoked, it will not be able to operate in the system, as Supervisor will not authenticate it. The trusted client application module or the intermediary are responsible for centralising the communications with other server modules, so after a first authentication of the user against Supervisor component, the intermediary, when needed, will send the user identifier to other components, which will assume that it has been already authenticated and verified [Serrão et al., 2006h].

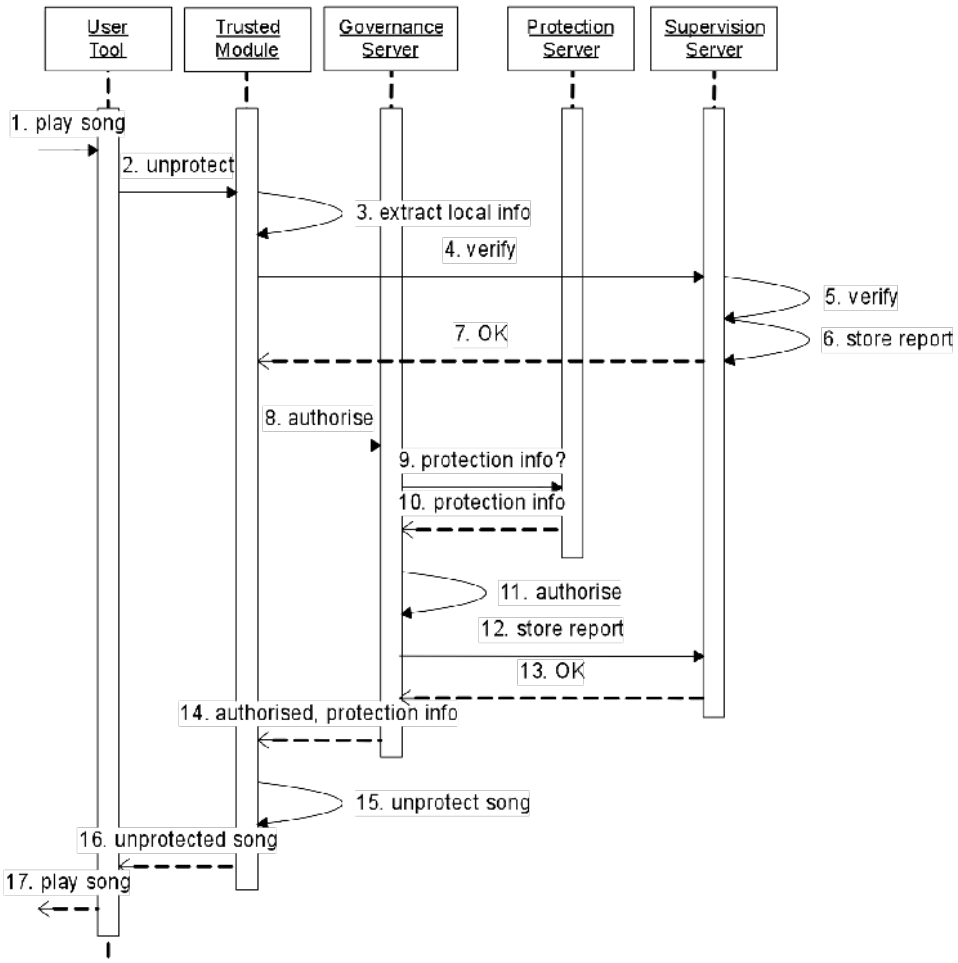

Figure III.3.8: MIPAMS Content consumption Use Case

In a typical content consumption scenario (Figure III.3.8), where a user wants to render protected content, the authorisation and authentication process are fundamental. At step 5, the Supervisor performs the authentication of the user and tool. If they were not authenticated, then the trusted module would not contact other components (steps 8 to 17) and the would not would not be unprotected and rendered.

The blocking of an Actor or client Component in the system supposes the modification of their associated status flag in the Supervisor component and also the revocation of their corresponding certificates in the appropriate CA.

4.3 Comparing both open Rights Management Systems

Some of the security features of two different open DRM systems were previously introduced and described. This new section will determine and present some common points and

differences between both and provide mechanisms to achieve interoperability at this level.

Regarding component registration and certification, both systems provide different mechanisms for the components registration in the system. While OpenSDRM enables a fully automatic registration and certification procedures, the MIPAMS platform uses a non-automatic registration but an automatic certification once the components are validated. Although different, the result of the process ends to be the same: a X.509 certificate for the component. In OpenSDRM, X.509 certificates are used to certify DRM components, while the different DRM components functionalities are certified by another different certificate. In this way, independently of the registration and certification processes, a component will own a X.509 digital certificate in both systems. To enable interoperability at this level, compatible certificates will have to be issued for having compatible component certificates [Serrão et al., 2007a].

Regarding the components mutual authentication processes, both systems perform a client-server mutual authentication based on their component digital certificates [Fan et al., 2006]. On one hand, both systems include a list of trustworthy Certification Authorities in their components. To ensure that the components of both systems will be trusted, it is necessary to assure that they are issued by a common Certification Authority or that the CA certificates used in both systems are included in all components. On the other hand, OpenSDRM enables any component to query the Authentication Server for retrieving the component credentials revocation status, whereas MIPAMS does not, assuming that the server components will be controlled. In order for MIPAMS components to be authenticated in OpenSDRM they would need to be registered in the Authentication Server. However MIPAMS client components are different since client components in MIPAMS. Client tools, are certified and registered in the Supervisor component, and authenticated in the same way of Actors, by using their unique identifier.

OpenSDRM partly uses the same process for both the Actor and Component authentication functions by querying the AUS to check for the revocation status of their credentials. However in the specific case of Actor authentication, there is software called Wallet broker (see Part III, Chapter 7) that is responsible for handling the Actor authentication processes. In this sense, MIPAMS acts in a different manner. MIPAMS Supervisor authenticates the user by

his identifier, which is extracted in the client application and sent in the SOAP messages over the secured channel. In this authentication mechanism, OpenSDRM always recurs to the full credentials, which are sent to ensure a strong authentication process [Serrão et al., 2003b].

In terms of credentials format, there is also some differences between OpenSDRM and the MIPAMS platform. OpenSDRM uses both X.509 certificates for DRM components certification and authentication and a special X.509 user certificates XML mapping for rights management functions and Actors certification and authentication, whereas MIPAMS uses only X.509 certificates. The differences in client authentication can be overcome by:

1. Extracting the user identifier of OpenSDRM XML certificates for authenticating users in MIPAMS;
2. Perform an authentication based on the user identifier instead of the whole XML certificate in OpenSDRM for MIPAMS clients;
3. Using an alternative authentication process based on SAML tokens in order to avoid multiple authentications for the same user, based on digital signatures.

There is a strong difference between both platforms in terms of security design. While in OpenSDRM two security layers coexist to ensure both transport-level and application-level security, MIPAMS depends only on one transport-level security. At the application-level, in the MIPAMS case, the different components assume that there is a secure channel established and the authentication processes are somehow shortcut.

5 PKI-based authentication interoperability between open DRM systems

In the previous sections, the different security related aspects of two open DRM architectures (OpenSDRM and MIPAMS) were described and identified. Across this description some common points and differences between the architectures were identified.

In this section, some directions in terms of interoperability between both DRM systems, on what concerns the registration and authentication aspects, are presented. Basically, what it will be accomplished is a mechanism, based on a brokerage architecture, which will be able

to handle the registration of components and actors from one DRM on another, and also to handle both the authentication mechanisms [Serrão et al., 2007a].

5.1 Registration Interoperability

Both the DRM systems require the registration of DRM Components and Actors. Additionally, OpenSDRM also requires that the DRM components functionalities to be registered as well. Both OpenSDRM and MIPAMS require the components to be registered through a Certification Authority, capable of issuing X.509 digital certificates.

In terms of interoperability, the solution to allow both DRM components to be registered and to obtain X.509 certificates is the usage of a common Certification Authority (CA), part of a global PKI. Nevertheless this may not be a possible scenario in the real World – many commercially available CA already exist and it is necessary to assure that DRM components registered and certified on one CA can trust on components registered by other CA. Currently, web-browsers solve this problem in a limited way, by having an internal trust CA database, that allow them to decide whether to trust or not on a presented X.509 certificate. This mechanism can also be used on both DRM systems, once all DRM components may also have an internal CA database, if the deployment scenario is more global. If the deployment, in terms of interoperability is restricted or local, an alternative solution may be used (Figure III.3.9).

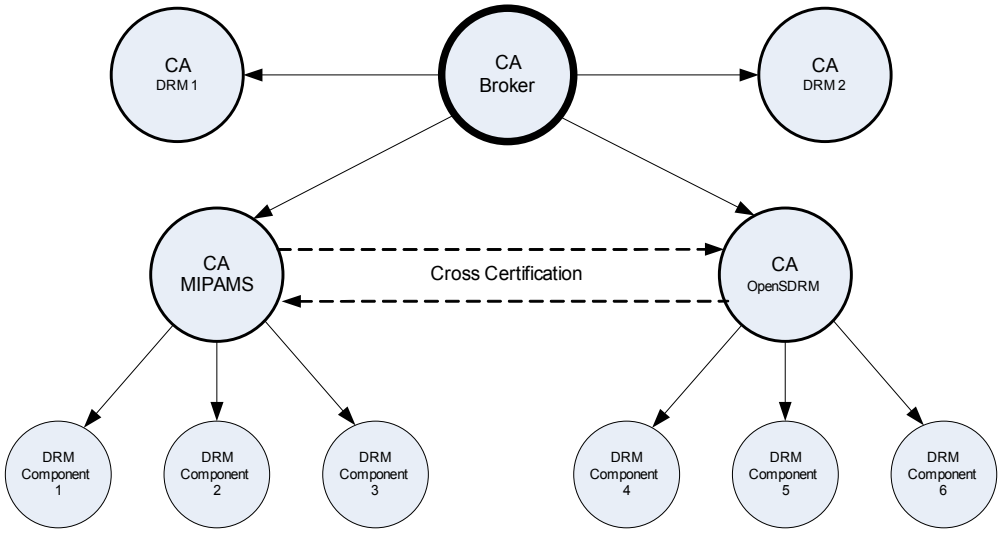

Figure III.3.9: Scenarios for CA registration interoperation

In a scenario where only MIPAMS and OpenSDRM need to interoperate, one possible option is to have a cross certification between the MIPAMS and OpenSDRM Certification Authorities. This way the certificates issued by both CA would contain their public-keys in the certification path: $Cert^{OpenSDRM}_{CA\text{-}MIPAMS}$ and $Cert^{MIPAMS}_{CA\text{-}OpenSDRM}$. Any certificate issued by any of the CA would always refer the other and allow trust relationships between components registered by one or the other CA: $Cert^{OpenSDRM}_{CA\text{-}MIPAMS} \Rightarrow Cert^{CA\text{-}MIPAMS}_{DRMComponent}$ and $Cert^{MIPAMS}_{CA\text{-}OpenSDRM} \Rightarrow Cert^{CA\text{-}OpenSDRM}_{DRMComponent}$.

While considering broader scenarios, the process presented in the previous scenario may not be very effective. As an alternative it is possible to establish a super CA (CA Broker, part of a specific PKI broker) that will issue certificates to each of the DRM CA. Any DRM component registered and certified on one CA would have a common trust point on the certificate certification path: $Cert^{CABroker}$. Since the certificates share a common CA, trust will be possible between different DRM components: $Cert^{CABroker}_{CA\text{-}MIPAMS} \Rightarrow Cert^{CA\text{-}MIPAMS}_{DRMComponent}$ and $Cert^{CABroker}_{CA\text{-}OpenSDRM} \Rightarrow Cert^{CA\text{-}OpenSDRM}_{DRMComponent}$.

In the specific case of OpenSDRM, the rights management functions inside the different DRM components need also to be registered and have a credential. This digital credential uses an XML specific format that maps a X.509 certificate in XML. Every time a rights management function inside a DRM component is invoked this XML credential is passed to authenticate the caller function. In MIPAMS this does not exist, and therefore to achieve interoperability at this level, the MIPAMS DRM components need to obtain such certificate. There are two possible ways for this:

i. DRM components perform their registration on the OpenSDRM AUS component, and obtain an XML certificate that must be used every time a MIPAMS component wishes to invoke an OpenSDRM function;

ii. To use an external component (Certificate broker) that will allow the translation between the MIPAMS X.509 certificates and the XML format required by OpenSDRM. This component would automatically register the different MIPAMS components on OpenSDRM AUS, as well.

The same occurs in the case of Actors registration. While in MIPAMS they are assigned with

an X.509 certificate, in OpenSDRM they use an XML certificate. This allows that the same mechanisms presented before, could also be used in the Actors registration case, either by requesting the MIPAMS actors to register in AUS as well or to use a mapping/translating mechanism between X.509 and XML.

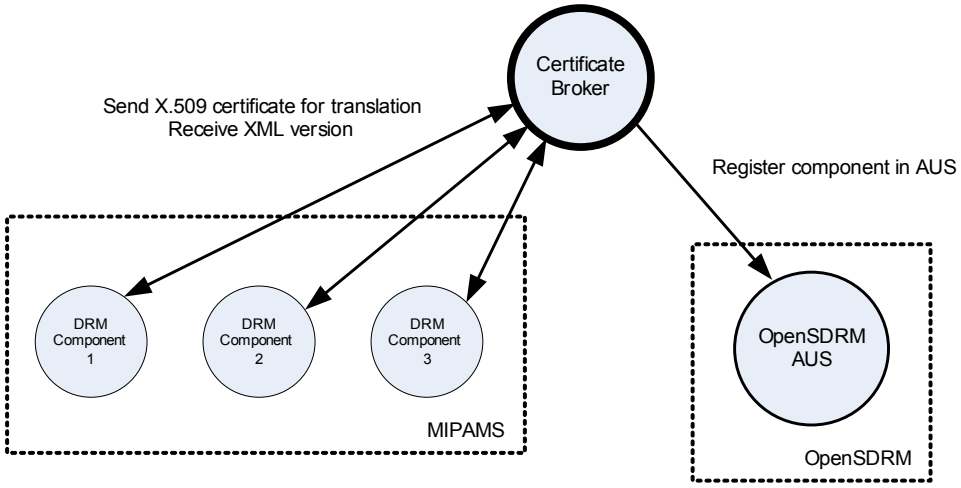

Figure III.3.10: Bridge between the X.509 and XML certificates

The processes described here in this section cover the main different registration and certification aspects for both open DRM platforms. In the next section the aspects related with the authentication interoperability will be handled.

5.2 Authentication Interoperability

In terms of DRM components authentication, there is the need to have a common authentication point between both X.509 certificates that identify the different DRM components (Figure III.3.10). Three possibilities are presented:

1. Both open DRM platforms components have an internal database of the trusted CA, so that $Cert^{TrustedCA}_{MIPAMS\text{-}DRMComponent}$ and $Cert^{TrustedCA}_{OpenSDRM\text{-}DRMComponent}$ can be compared with that internal database to ensure trust;

2. There is a limited number of open DRM in the interoperability scenario and cross certificates can be used between CA. This would allow trust establishment between $Cert^{OpenSDRM}_{CA\text{-}MIPAMS} \Rightarrow Cert^{CA\text{-}MIPAMS}_{DRMComponent}$ and $Cert^{MIPAMS}_{CA\text{-}OpenSDRM} \Rightarrow Cert^{CA\text{-}OpenSDRM}_{DRMComponent}$ because they both share the same CA;

3. A third option would use a super CA that will act as a broker between the different DRM CA seeking interoperability. This would allow trust to be established between $Cert^{CABroker}_{CA\text{-}MIPAMS} \Rightarrow Cert^{CA\text{-}MIPAMS}_{DRMComponent}$ and $Cert^{CABroker}_{CA\text{-}OpenSDRM} \Rightarrow Cert^{CA\text{-}OpenSDRM}_{DRMComponent}$.

An important aspect in the DRM component authentication is the certificate revocation process (this may occur if the private key is compromised, for instance). There are many methods that can be used to ensure that a certificate is still valid and has not been revoked. Certificate revocation lists (CRLs) are usually used to accomplish this function; however, on such dynamic systems as DRM platforms, it is best to recur to the Online Certificate Status Protocol (OCSP). This protocol will allow authenticating DRM components to verify online with the CA if the presented X.509 certificate has not been revoked.

Other issue where authentication interoperability should be discussed is on the Actors authentication on the DRM platforms. OpenSDRM and MIPAMS differ on this aspect – both in terms of processing and credentials format. In terms of format, in the MIPAMS platform case, the actors are authenticated through an X.509 certificate, handled by a global Supervisor that acts on the Actor behalf. An XML certificate is used in the OpenSDRM case. In terms of process, they also differ. While in MIPAMS there is a DRM component (Supervisor) that globally handles the registration and authentication processes for Actors, in the case of OpenSDRM, the Actors authentication is handled through a Wallet broker that is individually located at each Actor.

The best way to provide interoperability between the different authentication processes (both on the two analysed open DRM platforms and on others) is to have an external entity – a Certificate broker – that will bridge both of the authentication mechanisms and also will carry the necessary translations between the different certificate formats (Figure III.3.11).

To achieve Actors authentication interoperability for DRM components an external Certificate broker is established. Here are two of the many interoperability scenarios: a) OpenSDRM Actor authenticates to a MIPAMS DRM component and b) MIPAMS Actor authenticates to an OpenSDRM DRM component:

a) The Actor authenticates using a login and a password to the Wallet that reads from

the secure repository the Actor's XML certificate ($Cert^{AUS}_{Actor}$). The certificate is sent to the DRM component where the authentication is to be performed – the DRM component sends the Actor's certificate to the MIPAMS Supervisor to check its status. The Supervisor checks that this is an OpenSDRM XML certificate and therefore sends it to the Certificate Broker to obtain its X.509 version. After this, the authentication can be completed;

b) The actor authenticates to the DRM components using its unique identifier and the Supervisior acknowledges the DRM component of the authentication result. In this case the Supervisor component contacts the Certificate broker to obtain a XML version of the X.509 certificate. This XML certificate is then used to contact the OpenSDRM AUS to perform the authentication.

With this, it is possible to authenticate Actors from one open DRM platform on the other platform and vice-versa.

Other authentication aspect to consider is the authentication of the DRM functions in the OpenSDRM case. MIPAMS does not require this. Therefore the Certificate broker can be used to obtain an XML version of the X.509 certificate to be used by MIPAMS while invoking OpenSDRM DRM functions [Serrão et al., 2007a].

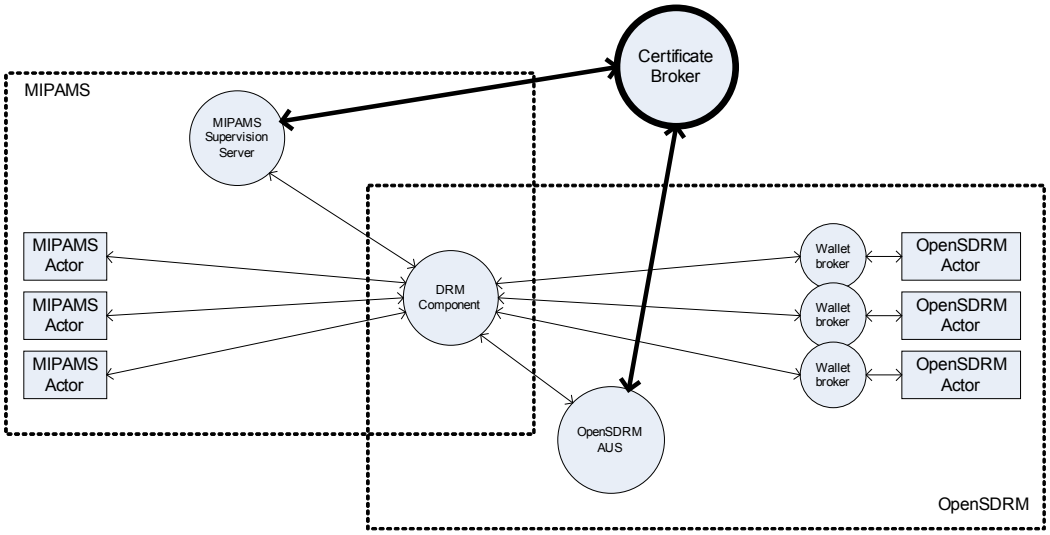

Figure III.3.11: Interoperability in the Actors authentication

6 Conclusions

The establishment of trust environments is a crucial aspect for any rights management solution [Bradbury, 2007]. The effort to efficiently manage rights on digital objects would fall apart without the appropriate security mechanisms, and above all without the capabilities to establish trust relationships between the different elements that compose digital rights management solutions – actors, devices and software components [Buyens et al., 2007].

Currently, most of the DRM solutions do not rely on already existing PKI services or vendors for that operation. Rather they implement their own security services, making DRM implementations more complex, and sometimes not controlling if security is at stake or not. DRM implementations take the burden of performing PKI complex and time-consuming operations rather than on concentrating on their real core - rights management [Serrão et al., 2006h].

This chapter presented different novel contributions that handle the rights management interoperability issues from a trust establishment point of view. These contributions aim at the achievement of common trust environments between different non-interoperable solutions, making usage of existing PKI mechanisms. Different security services implementation and management makes interoperability mechanisms hard to implement between DRM solutions. Two different approaches for interoperability through PKI services were presented. One is based on the existence of a single PKI solution for all the available DRM while the other is based on the existence of a PKI broker capable of managing trust between different DRM PKI implementations.

This chapter also presents an approach to the DRM interoperability between two different open DRM platforms that were considered to analyse how the rights management interoperable trust establishment could be achieved. The registration and authentication processes in both platforms were described and analysed, and commonalities and differences between them were identified. Based on this analysis and on the previous chapter contribution on trust interoperability through PKI, an interoperable architecture was proposed to accommodate both rights management platform's security and trust requirements.

Chapter 4. Open Rights Management platforms towards interoperability

1 Introduction

One of the most perfect examples of interoperability in the IT World is the Internet. The Internet gathers in a heterogeneous environment, different hardware, architectures and systems; however, every hardware device or application can exchange information in a common established way. This is only possible because there is a single standard communication protocol on the network bounding everything together (IP - Internet Protocol). Any system or application willing to use the Internet has to implement the mechanisms to comply with the IP specifications.

This is a straightforward way of providing interoperability. In this case, everyone agrees to follow a single and common standard [Rump, 2004]. This is an approach that has worked perfectly in Internet case but it cannot be applied to all the situations. Multimedia digital content is one of these situations. The multimedia World presents a panorama where almost everything is proprietary (content formats, media players, multimedia content protection mechanisms and multimedia rights management) and where no single standard exists that is strictly followed and implemented by everyone (in particular on what concerns rights management aspects) [Rump, 2004]. Therefore, interoperability in multimedia DRM is far more complex to handle than in the Internet scenario.

But the other reason (and perhaps the biggest), why the Internet has became such a hit in

terms of interoperability is its openness – specifications openness, platform openness and even (in some cases) source-code openness. It is hard to imagine the World Wide Web as it is today, if Tim Berners Lee had not made his CERN project publicly available for free. In this chapter the same approach will be uphold has a way to address the DRM interoperability issues. It is proposed, a methodology that could be used to analyse how the different open DRM could interoperate [Seki and Kameyama, 2003].

The introduction of different open-source software (OSS) development models has revolutionised the way software is designed, produced, distributed and managed. The main idea behind most of the OSS development is knowledge sharing, in which the developed knowledge and software is shared in the hope that another developer could add more value to the previous developments and continue this share process (there are exceptions and variations to this, but in essence they share the same goal) [Serrão et al., 2003c]. This model enhances the creation of win-win relationships through the availability of free software and source-code while incorporating at the same time new third party developments. The OSS development model is completely open where source-code changes can be introduced by anyone and shared to anyone [Serrão and Siegert, 200b].

The main contribution of this chapter is a vision of how the open nature of rights management could be used to provide capable interoperable systems. During this chapter, a short analysis of a set of particular open rights management technologies is made, and some directions for interoperability are indicated.

2 Openness and Interoperability

It has been discussed throughout this document, that interoperability is a serious and hot topic in the rights management community. Interoperability can be addressed from different perspectives.

Considering the end-user side perspective, one of the most important interoperability aspects is the possibility of using the same legally owned content in multiple devices. This simple and basic usage scenario is not yet possible today [Serrão et al., 2006e].

From the content provider perspective, interoperability is also desirable. First, because by

using interoperable content a broader range of users can be reached. Second, because in the production side it is cheaper and efficient to produce content in a common way than to produce specific versions for each available end-user device and to apply different specific protection mechanisms. This way, content providers can focus on their core business and on what they do best, enhancing of the content provided to the final user [Serrão et al., 2006e].

On the retailer side, the present model will give place to a more generic one. With this new model, digital content retailers will be capable of selling and supplying content to a broader range of platforms, increasing their potential base of clients. On the other hand, this will also reduce market entry barriers to new competitors, therefore increasing competition on the market. Indirectly, this would also benefit consumers, because offer will increase, lowering the prices [Serrão et al., 2006e].

In today's market, there are two dominant commercial rights management solutions: the Windows Media DRM (WMDRM) (see Part II, Chapter 1) and Apple Fairplay (see Part II, Chapter 1). These two rights management solutions are non-interoperable. They use different file format containers, protection mechanisms, protocols, security systems, and are targeted to different devices. Content acquired on one platform will not work on the other. However, they share something in common: they both are closed rights management systems. These systems do not have public specifications available, do not provide connectivity interfaces for other DRM systems and their source-code is not available [Serrão et al., 2006e].

WMDRM is the Microsoft DRM for Windows Media content. It is a closed DRM system, although some documentation and a Software Development Kit (SDK) are provided in exchange for a non-disclosure agreement signing [Prunela, 2001]. However, the development of rights management systems based on this SDK is firmly bound to the Windows architecture and to the Windows operating systems.

Fairplay, from Apple, is even more restricted than WMDRM. There is neither documentation nor SDK available for this rights management system. It is very difficult to create a Fairplay compatible service (there are some cases of reverse engineering[11]) or device. Apple enforces a very tight control over the content that enters the iTunes system, it's protection

11 http://www.doubletwist.com

mechanism, the store, and finally the rendering application (iTunes and QuickTime) or device (the iPod series).

At the current state, there is no chance for WMDRM and Fairplay to interoperate (or any other solution that follows the same approach). Also, no single one will overcome all the others and completely dominate the market. The lack of interoperability between different DRM systems can lead to disappointing user experiences as well as jeopardising the digital media concept [Serrão et al., 2008a]. For example, it may happen that someone who purchases a new protected digital item is unable to reproduce it on their player due to the lack of interoperability between the DRM systems that have been used. Therefore, the only viable alternative is the open model uphold in this chapter.

3 Open DRM platforms

In the previous sections of this thesis, the rights management problems were presented and described, and references were made to different approaches to address this problem. An integrated and viable approach to deal with this problem is to place intermediary brokers between the different DRM components and functions and also to extend these components through the download of new functionalities. In order for any of the two approaches to be possible, one of these two requirements must be met [Pimenta and Serrão, 2006]:

1. DRM components and functions should be openly specified, so that the present functionalities could be extended and new ones could be easily integrated;
2. The open and available implementation of such DRM components should exist in an open source-code regime.

The ideal scenario is the one where both requirements are satisfied. However, such scenario is not common in the DRM field. Most of the currently available DRM platforms are neither open-specification or open-source [Serrão et al., 2006e].

In the State of the Art (see Part II) different DRM solutions have been identified and described. The following part will focus on the presentation of the most relevant open rights management platforms to identify and compare the different initiatives.

3.1 Open-source DRM platforms

In this section, the different existing open-source based rights management will be identified and described. There are currently a set of Open-Source Software (OSS) rights management initiatives that make available a set of software and the correspondent source-code. The most relevant initiatives are listed on the following sections and they reflect the number of OSS projects that are currently addressing rights management issues and some of them are also addressing interoperability problems. The following open-source rights management solutions have been considered: Media-S [Media-S, 2003], OpenIPMP [OpenIPMP, 2003], DreaM [SunLabs, 2006], DMP Chillout [Chiariglione and Liao, 2006] and OpenSDRM [Serrão et al., 2003b].

3.1.1 Media-S

Media-S is an open-source DRM solution developed by SideSpace Solutions[12]. Although Media-S claims to be format-independent, the available release only supports the Ogg Vorbis open-source audio codec. In terms of architecture, Media-S is divided into three different components – a packager, a client and a license server (Figure III.4.1) [Media-S, 2003]. Media-S is based on XML (for the creation of licenses and data in licenses), and an open-source version of SSL (OpenSSL). For security, the Media-S DRM architecture separates licenses from content, and uses a combination of user and device authentication. The lack of a globally unique identity mechanism means that it could be easy to fake the license from one DRM protected item to another augmenting the potential to confuse the DRM client [Chiariglione et al., 2005]. Another problem with Media-S is the use of non-standard license Rights Expression Language (REL). Media-S uses their own REL specifications, even with the availability of two well documented REL standards (such as ODRL and XrML [Wang et al., 2002]). These two facts contribute to major difficulties for other DRM systems to interoperate with Media-S [Media-S, 2003].

12 http://www.sidespace.com/

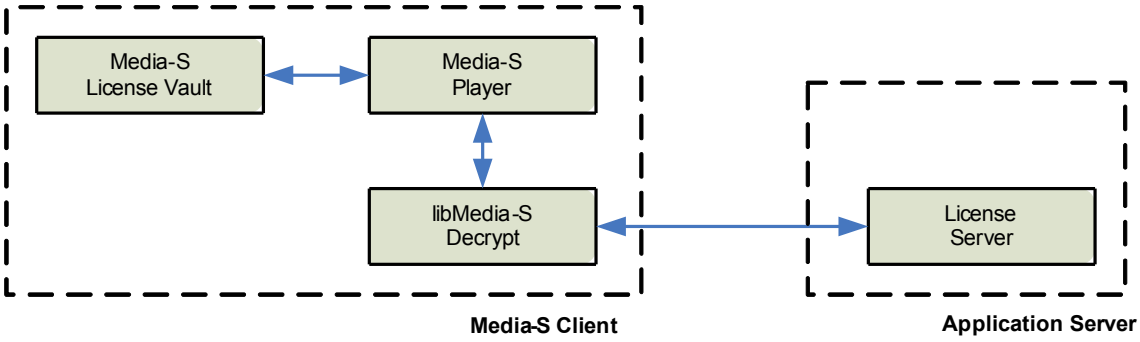

Figure III.4.1: Media-S architectural elements

Media-S DRM was released later on 2003 and it seems to have had a low impact on the DRM industry. As an open-source project, Media-S has released their last version of the source-code in 2003.

3.1.2 OpenIPMP

The OpenIPMP DRM was developed by Objectlab[13], and is based on open standards [OpenIPMP, 2003], including MPEG-4 [Pereira and Ebrahimi, 2002] and implements one of the leading MPEG-21 proposed content identification schemes to solve the challenge of uniquely identifying digital assets. The OpenIPMP DRM includes an implementation of the Intellectual Property Management and Protection (IPMP) [Lacy et al., 1998] specification for MPEG-4. OpenIPMP allows for the expression of licenses using a choice of the leading rights expression languages, such as MPEG-REL and ODRL. Within the OpenIPMP framework, each MPEG-REL [MPEG-21-REL, 2004] or ODRL XML license [ODRL, 2002] is signed by the OpenIPMP Certificate Authority for authenticity and is cryptographically specific to its intended recipient. OpenIPMP has already been described in the State of the Art chapter of this thesis (see Part II, Chapter 1).

13 http://www.objectlab.com/

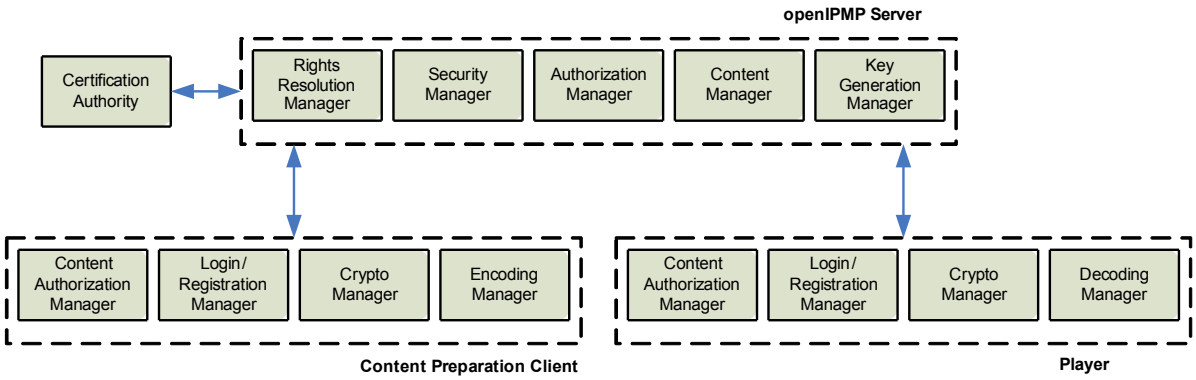

Figure III.4.2: OpenIPMP architecture

OpenIPMP is a framework (Figure III.4.2) composed by a collection of tools and services capable of delivering a robust, scalable, and adaptive infrastructure to support the management and secure delivery of multimedia content. The OpenIPMP DRM conforms to industry standards enabling the interaction between various hardware and software products not locking the user to proprietary solutions. To manage rights and enforce role based rules and permissions, the framework needs to be able to uniquely identify every user of the system. The OpenIPMP DRM issues each user a Digital Certificate (or Digital Id) when they register with the system and enables secure, confidential communications with the OpenIPMP server components ensuring that sensitive data is not compromised during transit.

This open-source DRM is an active project that has recently released the second version of the OpenIPMP platform adding support for OMA, ISMAcrypt [ISMA-DRM, 2005] and MPEG-2. OpenIPMP due to its extensive support and to its distributed architecture has the potential to interoperate with other DRM solutions.

3.1.3 DReaM

DReaM is a Sun Microsystems initiative to develop a rights management solution focusing on open-standards [SunLabs, 2006]. According to DReaM own information, whenever the market requires proprietary solutions DReaM will be capable of integrating with these solutions providing openness and interoperability that meets customer requirements [Fernando et al., 2005]. DReaM is an initiative to leverage the methodology of Service Oriented Architectures (SOA) and introduce rights management services that leverage open

standards and support cross-service capabilities [Fernando et al., 2005]. This solution has already been presented and described on the State of the Art chapter (see Part II, Chapter 1).

The DReaM architecture supports the separation between the rights management components (through the de-coupling of authentication, licensing, rights management and protection systems). This disintermediation enables the choice and selection of these technologies independent of each other without any compromise for the overall solution. There are two key elements for disintermediation in DReaM:

- Separation of rights management from the content protection systems;
- Separation of identity and authentication services from individual hardware devices.

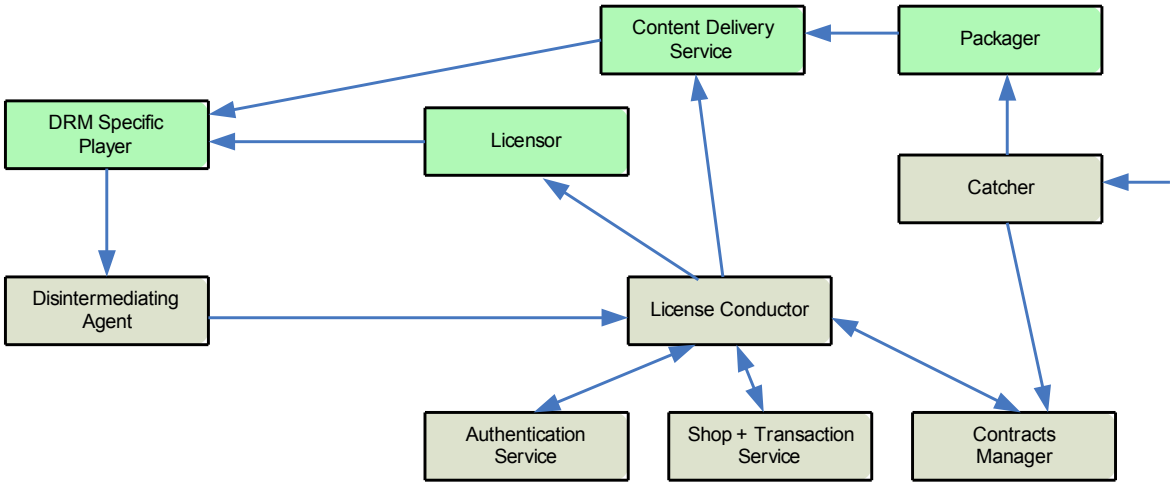

Figure III.4.3: Sun DReaM architecture

DReaM has a central objective towards the creation of an interoperable rights management, offering the capability to inter-operate directly with other content protection technologies and supporting services that enable both Conditional Access System (CAS) and DRM. A key-concept in the DReaM platform is the disintermediation concept. This concept enables multiple instances of these components to exist in a DRM/CAS system [SunLabs, 2007]. The DReaM disintermediation system (Figure III.4.3) enables the coexistence of multiple instances of content protection specific components (Player, Licensor and Packager) and components that are not content protection specific (disintermediation agent, conductor, catcher, licensing conductor, contracts manager, authentication service, shop and transaction system and content delivery system).

3.1.4 DMP Chillout

Chillout [Chiariglione and Liao, 2006] is the name of the Interoperable DRM Platform (IDP) Reference Software [DMP, 2007]. The IDP is composed by a set of standardised DRM tools based on "primitive functions" derived from existing digital media systems, derived through the analysis of several Digital Media Project (DMP) selected use cases (Figure III.4.4) [DMP, 2007]. A sizeable part of the IDP-2 technologies are based on MPEG-21 and only those functionalities that are not supported by MPEG-21 (e.g. domain management) are DMP native. Unlike MPEG-21, the technologies have already been integrated.

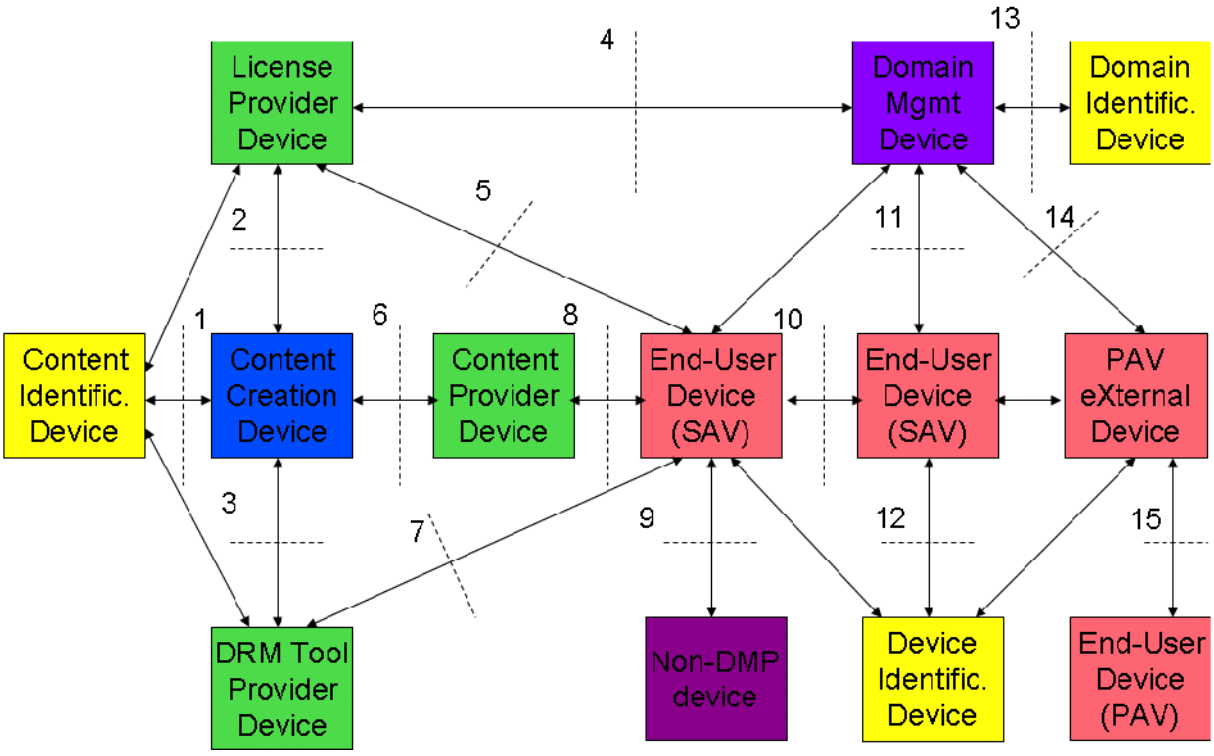

Figure III.4.4: DMP Chillout reference architecture

Chillout provides a set of Java-based libraries that implement the DRM functions and Java applications that are built on top of the Chillout libraries. Chillout is released as Open Source Software under the Mozilla Public Licence 1.1. This reference implementation can be used to test conformance of independent implementations of the DMP specifications and to set up interoperable value chains for use by independent users.

DMP and Chillout have already been described on the State of the Art part (see Part II, Chapter 2).

3.1.5 OpenSDRM

OpenSDRM (Open and Secure Digital Rights Management) was the selected name for the implementation of a new open-source rights management platform developed in the context of this thesis, composed by a set of server-side software components that implement a wide range of distributed key services. OpenSDRM is an open-source project and the source code is available under a LGPL license.

The OpenSDRM infrastructure (that will be further developed and described on Part III, Chapter 6) was designed to be adaptable and applicable to all types of content, business models and distribution channels (download, super-distribution, streaming or even broadcasting) [Serrão et al., 2003b]. This way, it is possible to enable digital multimedia content business models through the enforcement of licensing agreements for content use and offering business opportunities to the content rights owner and content provider. The OpenSDRM design has followed an horizontal approach and the implementation has followed the guidelines and specifications of open standards. This way it is possible to address rights management interoperability through the definition of common interfaces, tools and mechanisms that different rights management solutions can use. OpenSDRM is based on the Service Oriented Architecture (SoA) paradigm as proposed by the W3C consortium [Serrão et al., 2005c], providing a distributed nature to each of the major DRM functions and also a clear interface defined in WSDL for external services to integrate and a communication mechanism based on SOAP. OpenSDRM uses primarily ODRL [Serrão et al., 2005a][Serrão and Siegert, 2004a] to express rights, a language for rights expression derived from an international effort aimed at developing and promoting an open standard for the DRM expression language which has been adopted by OMA. Nevertheless the platform can be easily expanded to use other RELs due to a license template mechanism. This mechanism is also explained and detailed on a specific chapter of this thesis (see Part III, Chapter 7).

The OpenSDRM development is targeted primarily for xAMP environments and makes use of open-source tools such as PHP and MySQL (Windows, Linux and Mac platforms). Due to its distributed and open nature, OpenSDRM is a completely scalable platform that has to potential for integration with other rights management systems.

3.1.6 Comparing different Open-Source DRM platforms

This section defines a set of criteria for the comparison of the different open-source software DRM platforms that were described and analysed on the previous sections.

The objective of this section is not to establish a ranking between the existing different OSS rights management, but to perform an objective comparison between them, using a set of selected criteria [Serrão et al., 2008a].

For this comparison the following criteria have been selected:

- **Organisation**: this is simply a descriptive criteria. This is useful to understand what is the organisation support behind the DRM OSS development initiative;
- **License**: in this criteria the open-source license model for the DRM project is indicated;
- **Activity**: indicates if the given project is still active or not, and the level of activity of the project;
- **Base components**: this criteria indicates the different architectural components of the DRM project. This is important to understand the level of completeness of the solution;
- **Development status**: indicates what is the development status of the project, and the project maturity;
- **Deployment**: indicated how deployed is the project, that is, if the solution is being used in many situations or it is not used at all;
- **Number of Developers**: this criteria is important in OSS. It indicates the number of developers that are involved and actively contributing to the development of the project;
- **Fields of Applicability**: this indicates the different fields of application of the solution and, possibly, the different types of applications where the solution has been used;

- **REL Support**: rights expression is an important component of most rights management applications, therefore it is important to analyse how well is the REL support implemented on this type of OSS solutions;
- **Content Support**: this indicates which are the types of content that are supported by the DRM platform.

The following table resumes the main characteristics according to this selected criteria and for each of the choose OSS rights management implementations.

	Media-S	OpenIPMP	Dream	Chillout	OpenSDRM
Organization	SideSpace solutions	ObjectLab	Sun Microsystems	Digital Media Project (composed by several partners, including private companies and Universities)	MOSES (European Project), Adetti, the author
License	GNU Public License (GPLv2)	Mozilla Public License (MPL)	Common Development and Distribution Language (CDDL)	Mozilla Public License (MPL)	Lesser GNU Public License (LGPL)
Activity	Project is dead	Project is active (although with low activity)	Under development (low activity)	Under development (high activity)	Under development (low activity)
Base Components	License Vault, Player, License Server	Media Encoding Tool, Media Player Tool, User Registration Service, Content Management Service, Rights Authorisation Service, License Management Service, Administration Tool	Media Player, Licensor, Content Delivery Server, Packager, Disentermediating agent, License Conductor, Catcher, Authentication Service, Transaction Service, Contracts Manager	License Provider Device, Domain Management Device, Role Verification Device, Content Identification device, Content Creation Device, Content Provider device, End-User Device External Device, DRM Tool Provider Device, Device Identification Device	Commerce Service, Content Production Service, Media Distribution Service, Registration Service, Payment Gateway Service, Protection Tools Service, License Service, Authentication Service, Configuration Service, Wallet RM Interoperability Middleware
Development status	Beta stage, but the project seems to be ended. Last updates date from April, 2003.	Stable/Production	Under development (Alpha stage)	Under development (Beta stage)	Under development (Beta stage)
Deployment	N/A	Under	Still under	Just some internal	Mostly used and

		development (two major releases)	development. Not yet deployed.	demonstrations. Some larger demonstrations are being prepared.	deployed on R&D projects
Number of Developers	1	2	N/A	5	1/2
Fields of Applicability	Digital Audio	Audio and Video Content	Audio and Video Content	Video Content	Audio, Video and Still Images
REL Support	No REL support	ODRL/ XRML/ MPEG-21 REL	Use an alternative to REL. Uses the DReaM-MMI (Mother May I)	MPEG-21 REL	REL independent (supports ODRL, MPEG-21 and others)
Content Support	Ogg-Vorbis	MPEG-2, MPEG-4	Independent of the content format/type	MPEG-21	Independent of the content format/type. Different types of content: tested with MP3, MPEG-4, JPEG2000, Motion JPEG2000

From this table analysis is possible to draw some conclusions on what the concerns the actual rights management OSS panorama, dividing such conclusions in relation to the different selected criteria.

Organisation

In this criteria it is important to note that the panorama between the different OSS DRM solutions is mixed. There are some initiatives which are promoted by single entities or companies, while others are promoted by a large number of entities or individuals. This reveals the support that the solution/implementation is receiving from the community. On one side, solutions such as Media-S and Sun DReaM are mostly private company initiatives, and therefore they have little support from the developers community (consequently they may have a lower change to influence the market and to enhance on further developments).

On the other side, some other initiatives have a larger support from developers and are not entirely tied to just a single organisation, seem to have more activity and their development progresses fast. However, there are some questions if they can influence the rights management solutions market. This is for instance the case of DMP Chillout.

License

The software license model (more precisely the source) has a direct impact on the software distribution model and on the associated revenue. There are dozens of OSS license (approved by the Open Source Initiative (OSI)[14]) each of them with their particularities.

They all share something in common. The licenses control the distribution of the software and the source-code. However, some of the OSI license disfavour completely the software lock-in on a commercial product while others do not.

From the analysis, two major licenses are being used in DRM OSS projects: the Mozilla Public License (MPL) and the GNU Public License.

Activity

In this criteria, the activity of the different DRM OSS projects was analysed. From this analysis it was possible to conclude that almost all the projects are still an "on-going" work. This is a common situation on OSS projects due to the number of contributions and improvements made by software developers. This causes the impression that the project is "never" a final product.

There are some cases where new developments have completely ceased and therefore the project is considered dead. This is the Media-S case, in which project the activity seems to be stopped since 2003.

Base Architectural Components

This criteria is helpful to analyse the number of functionalities that are provided by the OSS DRM solution. From the analysis it is possible to conclude that most of the analysed solutions offer a set of components and functionalities that allow the implementation of end-to-end governed digital content business models.

The majority of the analysed solutions offer rights management functionalities that cover the entire governed digital content value-chain, from the content provider side until the end-user side. Some of them also offer solutions for the distribution of content and for the establishment of revenue models.

14 http://www.opensource.org

An exception to this scenario is Media-S. This solution offers only a very limited set of functionalities that are restricted to license issuance and control.

Development Status

The number of different OSS DRM platforms is impressive, considering that it is such a specific technology. However, most of the projects are still under heavy development and their maturity is low.

This can be interpreted in two different ways. On one hand the different solutions may have important security problems that might compromise their purpose, and on the other the solution is being scrutinised and improved by the developers community to correct and solve potential problems.

Deployment

This is perhaps the "Achilles heel" of the analysed OSS DRM projects. The different analysed solutions have just been deployed on very limited scenarios. Some of the solutions have not been deployed at all.

Most of these OSS DRM solutions are being deployed on some particular R&D projects and on very limited and controlled trials and experiences. None of these systems have been deployed on an open commercial system.

Number of Developers

The number of developers involved on these type of initiatives is quite limited. The average number of developers on these analysed OSS DRM projects is two or three developers.

This number reveal two important aspects. First, there is little interest from of the open-source developers community to participate in these type of projects and second, this reveals one of the reasons for the slow development and low activity on these projects.

An increase in the number of active developers would imply a faster development and the creation of an higher awareness to this type of OSS DRM solutions.

Fields of applicability

Most the OSS DRM solutions analysed address directly different types of multimedia content.

There are some solutions that are targeted to some specific types of digital content (like Media-S) while others are completely independent from the digital content type (like the OpenSDRM, DMP Chillout or Sun DReaM platforms).

The independence from the content type will allow one particular OSS DRM solution to be applicable in different digital content scenarios, and increase its deployment.

REL Support

As one of the most common characteristics of any rights management solution, the ability to handle rights expression languages is also present in most of the analysed OSS DRM platforms.

It is therefore no surprise that most of the analysed solutions support some kind of rights expression mechanisms and support a particular rights expression language. It is also interesting to notice that some of the platforms support more than one way of rights expressions while others implement mechanisms that support any rights expression mechanism.

Content Support

This criteria is closed related with the “Fields of applicability” one. From the analysis a mixed panorama was found. There are some solutions that are being developed to target just a very specific type of content, like the case of Media-S that is targeting Ogg-Vorbis digital audio, while others are pursuing the content format independence goal.

Therefore, it is possible to conclude that OSS DRM solutions have some potential for growing in the future, however they would need to find their proper place in the open-source World as well as to motivate and captivate the developers community, to further contribute to its development.

3.2 Open-specification DRM platforms

Another category of open rights management platforms are those who make available their specifications and that make their interfaces publicly available so other parties could develop ways to interoperate with that platforms.

In this section, the following open specification rights management platforms have been considered: MIPAMS [Torres et al., 2006] and OMA-DRM [Pimenta and Serrão, 2006].

3.2.1 MIPAMS (Multimedia Information Protection and Management System)

MIPAMS (Multimedia Information Protection and Management System) [Torres et al., 2006] [Rodriguez and Delgado, 2007b] manages multimedia information using DRM and content protection. The architecture consists of several modules and services, which provide a subset of the whole system functionality needed for managing and protecting multimedia content. MIPAMS is a service-oriented rights management platform and all its modules use the web services flexible approach.

MIPAMS encompasses an important part of the content value chain, from content creation and distribution to its consumption by final users (Figure III.4.5).

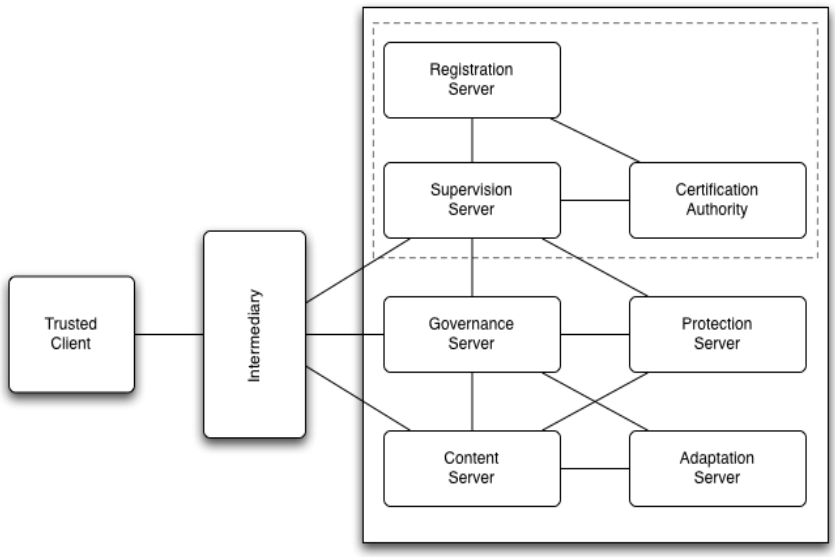

Figure III.4.5: MIPAMS generic architecture

The following servers and modules form part of this architecture:

- Content server: Offers the needed functionality to deal with content, including search, downloading, encoding and registration of new content.
- Adaptation server: Adapts content and their metadata, depending on the constraints (transmission, storage, consumption, etc.).

- Protection Server: Provides protection functionality, including object protection, key production and storage and description of protection tools, among others.
- Governance server: Provides licensing functionality, including generation and storage of licenses, both for distribution and end-user, authorise user actions and license adaptation / translation.
- Certification server: Provides different functionality that is decomposed into Certification Authority, Registration Server and Supervision Server.
- Certification Authority: Issues X.509 certificates for different Components and Actors in the system.
- Registration Server: Registers actors and tools. In case of user registration, a user certificate is issued.
- Supervision Server: Authenticates and supervises actors and system components. It also verifies users and tools by checking the actions done by them.
- Trusted client: Module for interacting with the client application in order to enforce DRM.
- Intermediary: It is usually part of the trusted client, but it could be also located on the server if the client device has limited resources (for instance, a mobile device).
- Client application: Multimedia player or edition tool that deals with protected content.

A more detailed description of the MIPAMS platform is provided on the State of the Art (see Part II, Chapter 1).

3.2.2 OMA-DRM (Open Mobile Alliance DRM)

The OMA-DRM (Open Mobile Alliance DRM) platform, currently on version 2.0 [OMA, 2006a] has been developed to enable the controlled consumption of digital media objects by allowing content providers the ability, for example, to manage previews of DRM Content, to enable super distribution of DRM Content, and to enable transfer of content between DRM

Agents.

Similarly to other previous analysed DRM architectures, OMA-DRM defines a set of Actors and Components in its reference architecture. The most relevant are the DRM Agent, Content Issuer, Rights Issuer, User and Off-Device Storage.

The DRM Agent (DRM-A) represents a trusted entity in a device. This entity enforces permissions and constraints associated with DRM content, controlling the access and usage of DRM content.

The Content Issuer (CI) delivers DRM content. OMA DRM defines the DRM content format to be delivered to DRM-A, and also defines the way the content can be transported from a CI to a DRM-A using alternative transport mechanisms. The DRM content packaging may be handled directly by the CI or it may receive it from an external source.

The Rights Issuer (RI) is the OMA DRM entity that assigns permissions and constraints to DRM content, and generates Rights Objects (RO). The RO is represented in XML and expresses permissions and constraints associated with DRM content.

A User is a human user of DRM content, which can only access DRM content through a trusted DRM Agent.

The Off-Device Storage allows DRM content to be stored off-device, for backup purposes, to free memory on the device. RO with stateless permissions can also be off-device stored.

The OMA DRM system enables CI to distribute Protected Content and RI to issue Rights Objects for the Protected Content. The DRM system is independent of media object formats, operating systems, and runtime environments. For User consumption of the Content, Users acquire Permissions to Protected Content by contacting RI. RI grants appropriate Permissions for the Protected Content to User Devices. The Content is cryptographically protected when distributed; hence, Protected Content will not be usable without an associated Rights Object issued and cryptographically bound to the User's Device.

The Protected Content can be delivered to the Device by any means. But the Rights Objects are tightly controlled and distributed by the RI in a controlled manner. The Protected Content and Rights Objects can be delivered to the Device together, or separately. The system does

not imply any order or “bundling” of these two objects. It is not within the scope of the DRM system to address the specific payment methods employed by the RI.

More details about the OMA-DRM platform can be found on the State of the Art chapter (see Part II, Chapter 1).

4 Open DRM platforms compared

In the previous sections of this Chapter, different open rights management systems have been identified and described. From these, three of them have been selected for comparison (OpenSDRM, MIPAMS and OMA-DRM). They all share some commonalities: either the specifications are public, or they provide public documented interfaces, or their source-code is publicly available.

OMA-DRM development has been supported by a wide community of companies and it is widely implemented and deployed in the marked by the major mobile phone suppliers (Nokia, Sony-Ericsson and others). OMA has already launched two versions of its DRM recommendations – the later one (version 2.0) reached its maturity in middle 2006 – and it is not yet widely implemented in mobile terminals. Both MIPAMS and OpenSDRM were born in the heart of the scientific research community, supported by public financing (National and European), and are mostly used by academic communities. OpenSDRM was developed on the context of this PhD thesis (Part III, Chapter 6).

Although OMA DRM has been developed to address a very specific vertical sector – the mobile phone industry – it can be used, up to a certain extent, in other different usage scenarios as well. OMA DRM mandates a specific file format container – the DRM Content Format (DCF) – to hold the DRM-governed items. This is a key-point in the interoperability of the different devices developed by different manufactures and operated by different mobile Telecom providers. On the other hand, content, rights, business models and platform independence were the three major design goals of OpenSDRM. Therefore, OpenSDRM can be easily adapted to several types of content and to different content business models – in fact this has already been done since OpenSDRM has been used in several research projects, dealing, for example with MPEG-4 [Serrão, 2005] and JPEG2000 [Serrão et al., 2006a].

MIPAMS content format is based on MPEG-21 Digital Items, thus providing enough flexibility to handle many different content formats. Licenses are expressed in MPEG-21 REL, although MIPAMS provides mechanisms for the conversion from and to OMA DRM version 2.0 REL .

Both OMA-DRM and OpenSDRM define DRM architectures that deal directly with the distribution of rights to the final consumer. This means that they have mechanisms (the DRM Agent in OMA and the Wallet in OpenSDRM) that handle all the authorisation clearance and rights parsing at the end-user side, having specific protocols for getting the licenses from rights issuance entities (Rights Issuer in OMA and the License Server in OpenSDRM). OMA DRM implements a protocol called ROAP (Remote Object Access Protocol), while OpenSDRM uses a specific web-services call entry in the license issuer. MIPAMS extends this model by adding the possibility that the licenses issued are not final user licenses. MIPAMS supports final user as well as distribution licenses. This supposes that a content distributor will need to own a distribution license in order to be able to derive final user licenses from it, something that will be controlled when issuing the end-user license. On the other hand, MIPAMS also supports the usage of license-like documents that can be defined by content creators or content owners, which are useful to state the rights that a distributor could exploit over the content and the related conditions. Any distribution license, when created, must fit the rights and conditions stated in those license-like documents. The authorisation algorithm performed for final users against end-user licenses can optionally involve the verification of the distribution license from which they derive. In this way, the whole content value chain can be controlled [Serrão et al., 2008a].

From an architectural point of view, both MIPAMS and OpenSDRM platform are richer than OMA-DRM. From a functional point of view, there are functionalities that are shared between all of the three platforms. However, some extended functionality offered by MIPAMS and not considered in OMA-DRM is the following:

- Rights enforcement: MIPAMS enables local or remote enforcement of rights, whereas OMA-DRM performs it at the point of consumption.
- Supervision: MIPAMS Supervision Server enables post-usage payment and provides statistical, tracking and control functionalities, which are not considered in OMA-DRM.

- Certification and trust on clients: In OMA-DRM, the trust on DRM Agents is based on the possession of digital certificates. However, the periodic verification of the client modules integrity is not considered.

In the same direction, OpenSDRM is quite comparable with the MIPAMS DRM platform. In fact, OpenSDRM offers also some extensions when compared to the OMA-DRM platform, such as:

- The license template mechanism present in OpenSDRM allows an easy extension of the Rights Expression Language (REL) supported and OpenSDRM is therefore not particularly tied to any specific REL;
- The Wallet, which has a similar behaviour as OMA's DRM Agent, is capable of not only handle the registration and authentication processes, license download, enforcement and authorisation, but also with the download of new protection tools;
- OpenSDRM offers the connection to payment functionalities, whereas in OMA-DRM no payment functionalities are mentioned;
- OpenSDRM rights management mechanisms are independent of the type of governed content, while OMA-DRM specifies its own OMA-DRM DCF format.

OMA-DRM also presents some advantages over the OpenSDRM and MIPAMS platforms:

- It is widely tested, supported and implemented in most mobile phones (in particular OMA-DRM version 1);
- It uses a common format, which in part simplifies the overall DRM mechanisms.

A key aspect of rights management is security. In this particular aspect, MIPAMS, OpenSDRM and OMA-DRM provide a secure environment for the DRM operations, making extensive usage of symmetric and asymmetric cryptography, digital credentials and secure protocols [Serrão et al., 2008a].

5 Open DRM solutions SWOT analysis

One of the biggest questions that is still on the air today, is the applicability of the open-

source software concept on the development of rights management solutions. Several documents and articles have been published and discussed on this topic and the opinions are split. Some argue that it is impossible to have a rights management solution in open-source software regime, while others defend this type of solution.

An important misconception related with open-source software that is particularly relevant for DRM systems concerns open-source software security [Herzog, 2006]. In special this misconception is related to the fact that the software is less secure because everyone can look at the source-code to look for flaws and vulnerabilities. The fact is that the availability of source code has in fact several security advantages. First, it facilitates the scrutiny by many people to find and flush out weaknesses in the design and coding processes. Second, when a vulnerability is found, fixes and patches can be made available without waiting for the code authors if the vulnerability is critical [Serrão et al., 2003c][Raymond, 2004]. Third, the fixes made can also be studied and scrutinised to ensure that they are made correctly. Finally, independent code checks and audits can only be made if the source code is available. Another important advantage of source-code availability is that in the event that the hardware platform on which the application is being used is made obsolete by the vendor, it is still possible to continue using the application safely on that obsolete platform or on other downloading the updated source and re-compiling it. For closed-source applications the only choice is to upgrade to the newer supported platforms.

An important fact is that the unavailability of source code does not mean vulnerabilities cannot be discovered. Modern debugging and software development tools can be used to examine, disassemble and reverse-engineer the close-source executables to look for design flaws and vulnerabilities. The most obvious illustration of this are the never-ending vulnerabilities discovered on closed software . Another important concept and analogy was long time ago proposed by Kerckhoffs [Kerckhoffs, 1883] which defended that a crypto-system should be secure even if everything about the system, except the key, is public knowledge. Later on this concept was extended by Eric Raymond [Raymond, 2004] to open-source as well, saying that any security software design that doesn't assume the enemy possesses the source-code is already untrustworthy. It seems clear that security through obscurity will never work and that open-source software can be as secure as closed-source

software. This is not and will never be an impediment for the development of DRM systems.

There are two different dimensions while considering the open rights management solutions and the interoperability problems. One is directly connected to the DRM complexity, meaning that it is necessary to handle protection (encryption, decryption, watermarking, key distribution, etc.), authorisation based on licenses (rights expressions, verification, license distribution, etc.), metadata, enforcement, governance, authorities and others.

A second dimension is related to how interoperability can be achieved, and how the different DRM interoperability levels can be defined. Therefore the interoperability of DRM systems can be classified in the following levels:

- Proprietary systems;
- Standards and architectures;
- Software framework based;
- Open Source.

At the first level, the solution for achieving interoperability is through the enforcement to all entities to use a specific proprietary system. This approach may work only if a system monopolises the entire market.

The “Standards and architectures” level is a solution for interoperability, to be approached from three different points of view: full format, connected and configuration-driven interoperability. Different standardisation organisations and industry and user fora are working in this direction, trying to specify standards at different levels, from REL to full interoperability architectures. The problem with this approach is the selection of the “neutral” format/protocol/architecture that should be used as a gateway to interoperate with the “others”. Since the number of these formats/protocols/architectures is growing, a solution in this direction is getting harder to achieve. However, this does not mean that standards are not useful; on the contrary, standards for expressing rights expressions, for expressing the semantics of rights and conditions or for specifying an algorithm for authorising access to a protected content, as simple examples, are definitely necessary.

There is an intermediate level before moving to the OSS level , named “software framework

based". This approach is based on the development of a set of tools ("software framework") that is publicly available so that DRM systems builders could use to implemented their required rights management system. The idea is that these tools offered in the framework are guaranteed by someone and although the OSS model could be used, a trusted organisation is necessary to certify all tools implementations (this would make a complete OSS rather complex to manage, although not impossible). The AXMEDIS European Project [AXMEDIS, 2007] is developing one of these frameworks that will contain tools not only for DRM, but also for multimedia content creation, distribution, consumption and management. The final business model to use is still under discussion, but the specifications of the tools are already available. It is finally worth mentioning that standards are followed when available and interoperability between standards, mainly through mappings, is also foreseen.

The openness of OSS with the availability of source is an additional level to facilitate DRM interoperability. However this may be not enough. It is also important to have a good distribution/separation of DRM functionalities and a clear interface between those functions and possible external systems.

There is a problem in modern days with open-source projects – open-source may not necessarily mean that it is free. In particular, in the DRM case, most of the technologies and standards applicable are encumbered by patents. This means that the source-code and the resulting implementation may be free, but the usage of such software on a commercial exploitation may be not. One good example of such is the case of ODRL and OMA. ODRL is a REL that is open [Guth et al., 2003]; however, there are registered patents that claim to cover the possibility of expressing rights using XML, so in fact all the REL implementation that use XML to express rights would need to pay royalties to the patent owner if they would request so and a Court would not declare the patent as not applicable. Several researchers are trying to demonstrate that the concepts covered by those patents are older than the patents themselves [Guth and Ianella, 2005]. However this is an important obstacle to the OSS DRM deployment. This has an impact on DRM interoperability as well since patents will be an obstacle to the deployment of OSS DRM platforms

On the other hand there is also the issue of GPL version 3. GNU Public License (GPL) is the license that governs most of the OSS projects in the World. Its principles are quite simple –

someone who uses GPL-ed source-code in its software development must also release the resulting source-code under a GPL license. GPL ensures that no one could actually take a piece of source-code (or even an entire software) developed by someone else and close it, brand it and sell it as it was his own.

In essence, the new GPL license will prevent the use of GPLv3 governed source-code in the development of DRM applications. As a free software license, GPLv3 intrinsically disfavours technical attempts to restrict users' freedom to copy, modify, and share copyrighted works. Each of its provisions shall be interpreted in light of this specific declaration of the licensor's intent. Regardless of any other provision of GPLv3, no permission is given to distribute covered works that illegally invade users' privacy, nor for modes of distribution that deny users that run covered works the full exercise of the legal rights granted by GPLv3 [Lamonica, 2006].

These two last factors may have a deep impact in the DRM OSS projects, and consequently on DRM interoperability as well. The first is not only directly applicable to OSS but the second one is very restrictive for OSS DRM implementations. Therefore DRM interoperability, although very desirable and achievable in OSS DRM gets seriously compromised by this. However the doors to DRM interoperability are always open due to the availability of either the source-code or by very well and documented interfaces.

On the other hand the lack of interoperable open-source DRM solutions will always create opportunities for closed non-interoperable solutions to get circumvented – it is curious to notice that most of the attacks that iTunes and the FairPlay system have suffered in the past have to do precisely with their unavailability on Linux platforms.

A more analytical approach is necessary to reach some type of conclusion. Therefore in this section of this chapter a SWOT (Strengths, Weaknesses, Opportunities and Threats) analysis will be conducted towards determining how successful an open rights management solutions (in particular, those who follow the open-source software models) could be.

	Helpful to achieving the objective	Harmful to achieving the objective
Internal Origin (contributes of the organization)	**Strengths** System is highly scrutinized Potential problems are easily discovered and corrected Allows the development of interoperable third-party solutions Permits the continuity of the solution development beyhond the initial developer frontiers	**Weaknesses** Public access to internal mechanisms (impossible to practice obscure security) System can be easily modified and circumvented Impossible to lock-in the solution
External Origin (contributes of the environment)	**Opportunities** Wide availability of source-code Interoperability Less design and implementation security problems Wide involvement of the development community and end-users Open Standards EU pressure	**Threats** The new GPL v3 license System can be circumvented System is open to potential attackers or crackers Software patents

Figure III.4.6: Open DRM systems SWOT analysis

5.1.1 Open Rights Management solutions Strengths

In this section, the major strengths of the open rights management solutions are presented. These strengths represent strong points in favour of the open rights management solutions and their deployment as a potential mean for interoperability between different rights management solutions.

One of the strengths of open rights management solutions is the fact that they are highly scrutinised. Due to their open nature, these type of open rights management solutions are available to be analysed and scrutinised by any third party. The continued scrutiny helps solving security errors and also helps creating a public awareness that the solution is reliable and secure.

The increased security level is also a strong point on this type of open rights management

solutions. Powered by the availability of the source-code, public specifications or even public interfaces, they do not base their security on the secrecy and obscurity of their processes and therefore they base their security on a strong design and implementation. Its open nature allows the rapid discovery of problems and the rapid correction of such problems.

Another strength of this type of solutions is the ability for third parties to develop new solutions that might be integrated in a easy way to the open rights management solutions. This is an approach that will push the development of new and interoperable solutions and therefore contribute to help solving one of the biggest issues of the current rights management panorama.

Finally another important strength of these open rights management solutions , in particular what concerns open-source rights management, is the fact the the solution could grow and continue to be developed beyond the original developer or development team. Therefore, an open-source rights management project can continue to evolve with the contribution of third party developers, or it can even fork and give origin to many different derived projects or solutions.

5.1.2 Open Rights Management solutions Weaknesses

One of the weaknesses that can be point out, especially to open-source rights management solutions is the fact that since everyone has access to the source-code and can learn from the internal details of the solution (even related to the security procedures), it is not possible to hide obscure tricks to enforce security.

Due to the fact the solutions are not closed and that the particular less well-designed security processes are exposed to everyone, less well intentioned persons could circumvent these systems and therefore compromise the entire solution and the governed digital objects.

This is a weakness that affects particularly systems that are entirely based on open-source. If all the components are open-sourced, including key-storage and handling mechanisms, it will be possible to develop and include new mechanisms that will shortcut the basic security procedures and retrieve cryptographic keys. However, this will not happen if the open-source

solution uses hardware tokens to handle and store cryptographic keys, such as smartcards, for instance.

Although this is something that is a particular weakness of open-source implementations, the very same open-source distribution model is helpful to eliminate such weakness. The modifications on the source-code to introduce vulnerabilities may occur on a local base, but will never work on a global base.

Another weakness that applies in particular to open-source rights management systems is the fact that most of the open-source licenses (at least the more common, such as GPL) prevent someone from locking the source-code, and built a closed-solution.

5.1.3 Open Rights Management solutions Opportunities

One of the biggest opportunities for the development of open-source rights management systems is wide availability of source-code. This is truly the basis for the open-source development movement: the ability to re-use the source-code that was already developed for other projects, contributes to faster development and increased code quality (do not reinvent the wheel). Moreover, using existing source-code has advantages from the security point of view as well. These source-code blocks have been developed, evaluated and corrected many times, achieving a maturity level, that will contribute to the stability, reliability and security of the open-source rights management solution where they are incorporated.

The open nature of these solutions will contribute to the establishment of solutions that can interoperate among each others. This a crucial aspect for rights management and by adopting and open approach the interoperability between the components of different rights management solutions could be possible.

Another opportunity for open rights management platforms is the fact that due to its open model, the solution could be analysed, revised and improved by third parties, and therefore is less prune to design and implementation security problems. This is a fact that contributes to more secure rights management systems, with the potential to resist to more sophisticated security attacks.

An interesting opportunity for open rights management solutions is the wide involvement of the developer community and end-users. This is an important because it could facilitate the adoption of a specific open rights management platform, since the involvement of the end-users and the developers would result on a solution that has into account the design requirements that will fulfil the end-users needs.

The involvement of the developer community would facilitate the proliferation of the development of new rights management solutions that will have on top of their requirements the need to interoperate between them. This will benefit end-users and even content providers, providing a larger choice.

An open standard is a standard that is publicly available and has various rights to use associated with it. Open standards are an opportunity for open rights management systems, in the sense that the existence of open standards allows the development of open solutions and that enables the free access to the standards so that new interoperable solutions could be developed as well.

An important opportunity for the open rights management is the actual European Union pressure over closed and monopolist rights management systems, such as Apple iTunes Fairplay and Microsoft Windows Media DRM. This pressure results in a clear opportunity for the development of a multiplicity of newer, open and interoperable systems that will allow the free competition on the market and the free choice of end-user consumers. This is an opportunity that has a geographic impact (in Europe) but that could be extended to other parts of the World.

The development of an open-source rights management solution would result on a clearer and transparent system, that could help on a clarification of the rights management processes and create more interoperable and open systems.

All of this would result on a faster adoption and wider acceptance of open and interoperable rights management.

5.1.4 Open Rights Management solutions Threats

One of the biggest threats to the development of open rights management solutions in the

new GPL version 3 license. This new version of the most well-known open-source license introduces protection against the "tivoization" and trusted computing. The Tivo contains GPLv2 source-code, and complies to the GPLv2 terms, allowing the end-user to access the source and modify it. However, if the user tries to run the modified version on the Tivo, it will not work.

So, in essence, GPL version 3 prevents the usage of source-code in any type of software solution that limits the users freedom, in the terms defined by the GPL version. Since any generic rights management solution is used to govern digital objects, this may be considered as serious limitation of the user freedom. This is precisely the interpretation of the Free Software Foundation (FSF)[15]. Therefore the implementation of open-source rights management systems is threatened by this licensing model.

The open nature of the rights management system can be the target of potential attackers and crackers that can try to circumvented the system. This is a potential threat to the open source-code implementations, in particular if the solution is entirely software based.

An important threat to open rights management platforms are software patents. This threat affects also open-source implementations. Even if the rights management solution implementation if free and the source-code is available, it is most likely that it may also violate any patent. Therefore rights management open-source implementations are threatened by software patents, and this is a serious obstacle to the deployment, proliferation and interoperation of these platforms.

6 Interoperability Solution for Open Systems

Rights management interoperability is a challenging technical task and involves dealing with complex problems [Serrão et al., 2006h]. Some point the direction of the International Standardisation to deal with the DRM interoperability problem [Koenen et al., 2004][OpenIPMP, 2003][Serrão et al., 2006e], indicating three different strategies for its achievement.

The first strategy is about full-format interoperability. This strategy requires that all the

15 http://www.fsf.org

protected digital objects need to conform with some globally standardised and accepted format.

Connected interoperability is the second rights management interoperability strategy. In this strategy, third party translation entities would need to be used to translate operations from one DRM regime to another. In this approach a peer-to-peer architecture is established in which each node allow an interface to its peers, and if it cannot satisfy a direct request then redirects the search to other peers. Another approach is the "intermediated digital rights management" or brokerage mechanism, where are identified different tasks to be carried by the intermediary in transferring digital object in the format used by the content provider to the format required by the end-user. Rights management tasks are executed by a third party server (the intermediary) on behalf of the content scripts and end-users.

The open rights management model, is a model in which any rights management solution willing to provide interoperability would have at least a set of public and documented interfaces, implemented in such a way that they would allow any other application or rights management solution to use such interfaces. This public interfaces would expose internal rights management functions, that would produce some type of result when invoked with the appropriate parameters. Considering a typical example: a user has acquired a music on a Windows Media store and wants to play it on its iPod. The iTunes software would invoke the public interface of the Windows Media DRM License Server to download a license for content. Once the license is obtained the music can be played on iPod. This is a very simple scenario that can only be achieved if an open rights management model is adopted.

The availability of the source-code for this model is not a essential, however it is desirable. If the source-code is available, the integration of elements is facilitated by several reasons. One is the possibility of checking how an interface works internally and make the invoking functions act in a more reliable way. Second, anyone could define public interfaces for rights management solutions, providing wider choices for interoperability.

While considering open systems and open-source, usually a concern emerges – security. On one side, open-source systems are more prone to scrutiny and therefore can be potentially shortcut. On the other so can closed systems. In fact, the more scrutinised a system is the securer it tends to get, and studies reveal that one of the major motivations of the attackers

of these systems is just the fact that they are closed.

As in any other computing system that wishes to exchange information with others, there are two major approaches for this: either a specific private interface is defined to allow the communication between two single entities (1-to-1 relationships) or, in alternative, a generic open interface is defined to allow the communication with a generic third party that will route the information for the target system (n-to-n relationships). The second approach, more generic, can be depicted as a rights management middle-ware, composed by a supra-set of rights management functionalities, capable of brokering and mapping each of the specific individual rights management functions into other specific similar rights management functions.

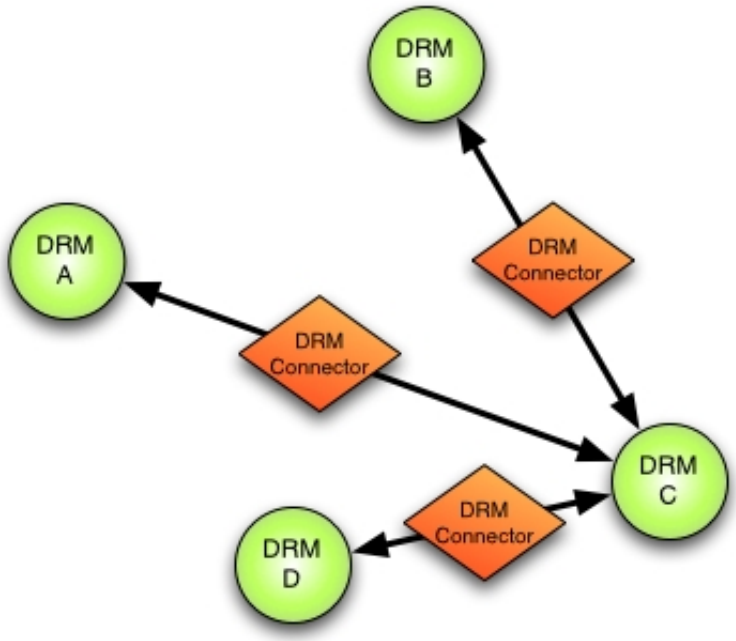

Figure III.4.7: Establish connectors between the different rights management systems to promote interoperability

The third and final strategy is configuration driven interoperability. Through the download of adequate tools any DRM system can get the ability to process governed digital objects on end users devices.

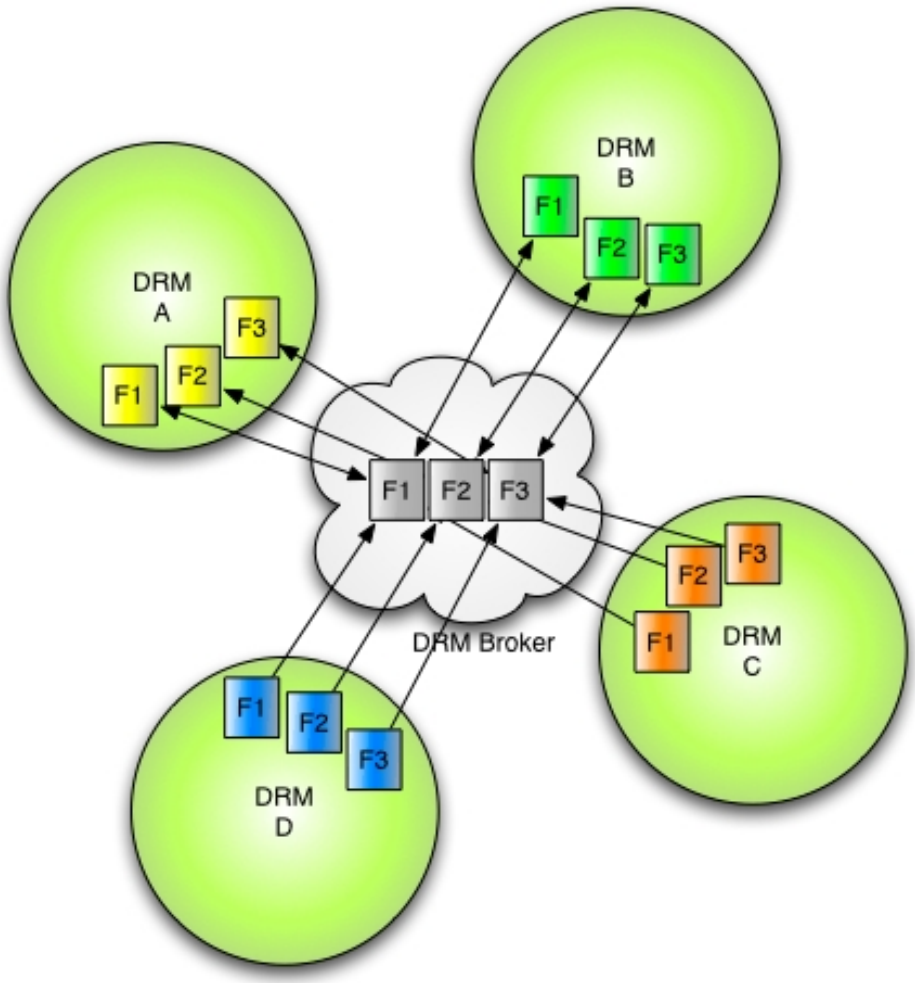

Figure III.4.8: Interoperability mechanism for rights management systems using a broker

One of the most viable strategies for addressing the rights management interoperability problem is to adopt connected interoperability. This type can assume two dimensions: a peer-to-peer connection (Figure III.4.7) between DRM solutions, or a generic broker mediated connection (Figure III.4.8) between different DRM solutions.

There are advantages and disadvantages in adopting one or the other approach. However, the broker-based rights management interoperability (Figure III.4.8) is a more generic approach to rights management interoperability, but it raises a lot more technical issues and complexities.

A generic methodology for interoperability between different open rights management solutions can be established. This method can be used to extend the interoperability of other available open rights management solutions and can be summarised in the following steps:

- Identify the open rights management solutions that need to interoperate: this is the starting point of the proposed method where is necessary to identify if the interoperability requirements are met, to enable the identification of the core rights management functionalities;

- Identify which are the major rights management functions that each open rights management solution offers and the corresponding interfaces: this is an important step because it will enable the identification of the specific functionalities, the data formats, the input and output parameters and their behaviour. If possible these interfaces should be described using a self descriptive language, such as WSDL;

- Define a super-set of functionalities that any open rights management solution may require: this super-set of functionalities will be internally implemented in the DRM as a routing mechanism to the real open rights management solution;

- Implement a set of public and generic interfaces that map to the previously identified functionalities: the complete set of generic interfaces will be part of the rights management middle-ware that can be implemented using a Service Oriented Architecture (SoA) approach;

- Create a location mechanism to allow different open rights management systems to interoperate with each other through the middle-ware.

The development of this rights management middle-ware solution aims at having, in the end, a global super-set of rights management functionalities that could be accessed through generic interfaces so that any DRM implementation could make use of any of them to interact and interoperate with other rights management regimes.

As an example of the methodology defined here it will be considered the case of OMA-DRM and OpenSDRM case (Figure III.4.9). Imagining that a user has acquired content that is governed by OMA-DRM and is trying to use it on a OpenSDRM based player, the proposed method would have to allow the implementation of this scenario. OpenSDRM knows that the content is governed by another DRM solution and would have to contact the rights management Middle-ware to get the appropriate authorisation to render the content. The rights management Middle-ware knows how to map the generic request into the specific

open interface provided by OMA-DRM, routing the authorisation through the appropriate OMA-DRM rights issuer (this request may include the necessary credentials for user or device authentication required by OMA-DRM). OMA-DRM returns an answer containing the Rights Object to the rights management Middle-ware. This answer is parsed into the generic format used by the rights management Middle-ware and sent back to the OpenSDRM. If the OpenSDRM does not know how to handle the rights contained in the Rights Object, the rights management Middle-ware can also provide generic services for conversion between different rights expression languages, or handle the authorisations at the server-side.

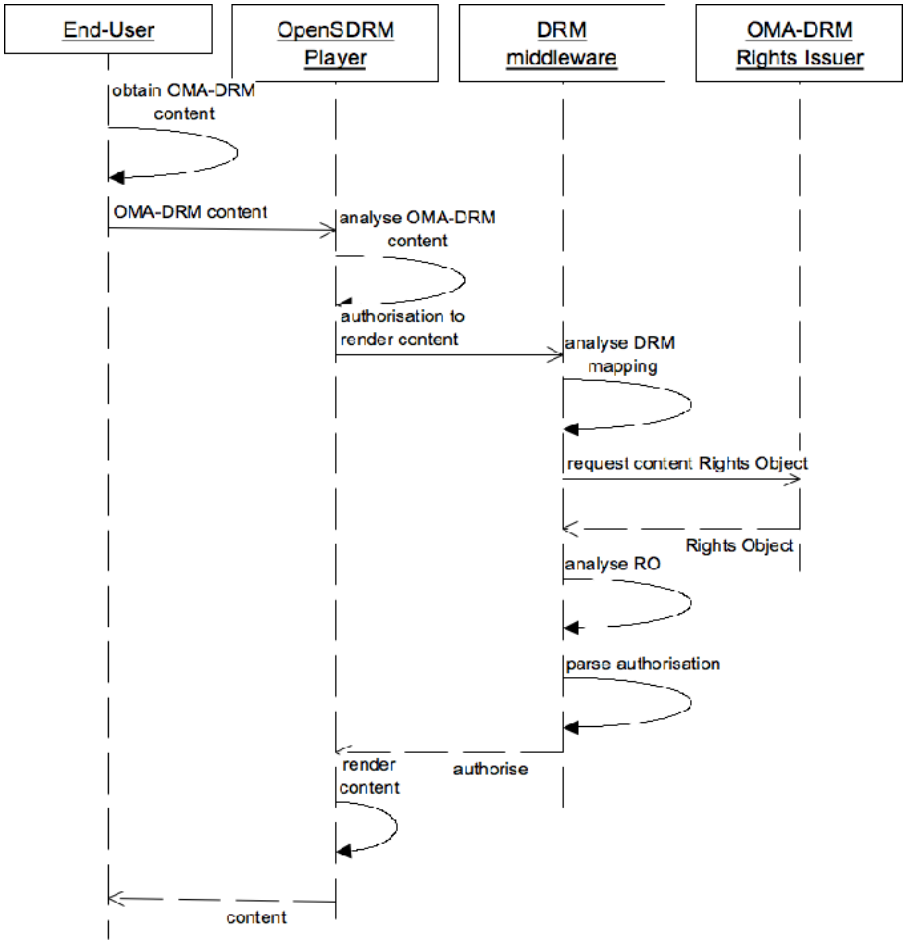

Figure III.4.9: Interoperability example between different open rights management solutions

From the interoperability and integration perspective, the openness is a "*sine qua non*" condition for the achievement of rights management interoperability. This type of interoperability will bring to the digital content consumers a better user experience and increase the potential revenue of content authors and distributors.

7 Conclusions

The current digital content market has been dominated by both Apple iTunes Fairplay and Microsoft Windows Media Rights Management. The Apple iTunes Fairplay represents the implementation of a market strategy vertically integrated and strongly dominated by Apple, that controls the full digital content value-chain, from the content producer side, until the end-user side, where governed content can only be played on Apple certified content rendering devices, the iPods. In the case of Windows Media Rights Management, the model is not so restrictive. The WMRM can be licensed to third parties so that content producers can control their own governed content production, licensing and distribution. This governed digital content can be rendered on any device containing the Windows Media Player software or in any software/device that is compatible with the PlaysForSure logo. These are the top dominant players in the digital governed-content market. However this two systems do not provide interoperability between each other.

On an unprecedented move, digital music providers and even authors are starting to refute DRM as solution for upholding their rights. Partly this is because these two market dominators are still back to back, and they did not find the way to make their solutions interoperate.

Most of the commercial DRM solutions existing today follow a completely vertical strategy, in which they are incompatible between them. These solutions are closed and do not offer any opportunity for interoperability – in fact, as part of their business strategy, interoperability is seen as a bad thing and as a menace to their own sustainability. On the other hand there are the content producers that wish to increase their digital content distribution and controlled usage and the end-users that want to use digital content as similar as possible to their analogue counterparts. To achieve these two goals, DRM Interoperability is crucial.

The existence of closed rights management solutions are one of the main reasons preventing interoperability and the alternative to these solutions are rights management solutions that follow an open model.

In this chapter, the most significant open rights management solutions were presented and described. These open rights management solutions include both open-source based rights

management solutions as well as solutions that do not endorse the open-source software models but provide open specifications and or open interfaces.

It is clear that closed rights management systems make harder interoperability and a new open model should emerge to tackle this issue. Only this model offers the necessary conditions to achieve full interoperability between the different DRM solutions. In the vision uphold on this chapter, this interoperability will be achieved through the definition of a DRM middle-ware that would map each of the specific rights management solutions functionalities into a super-set of generic functionalities, which could be provided by a Service Oriented Architecture.

This chapter also introduces some strategies needed to achieve DRM interoperability. These strategies can only be accomplished if the current and any future rights management solutions either provide open-specifications and/or source-code to allow this interoperability. At least, the most crucial DRM functions of each of the DRM systems must be publicly specified and source-code libraries (free of charge) should be made available to allow others to build the translation or bridge mechanisms between the different DRM solutions. Therefore, it is clear that OSS can play an important role in DRM interoperability and that traditional proprietary solutions will have to adapt to a World where governed digital content should have no barriers and its usage should be done without any technical restrictions.

The major conclusion to retain from this chapter is that a crucial approach has to be followed in order to obtain truly interoperable rights management systems. Rights management systems should follow an open model to create the necessary mechanisms to provide interoperability points. From a simple SWOT analysis is easy to conclude that the open rights management model, although challenging, specially from the security point of view, presents many advantages over the traditional and actual closed model.

Chapter 5. Secure Key and License Management for open DRM platforms

1 Introduction

Typically, a Rights Management System involves the description, layering, analysis, valuation, trading and monitoring of the rights over an individual or organisation assets, in digital format [Rosenblatt et al., 2001]. Managing the way users/actors can interact with digital objects is one of the major functions of a digital rights management solution [Serrão et al., 2005d]. Copy-protection and unauthorised usage prevention are two of the most relevant functionalities of DRM and are usually tightly bound, although they might exist in an independent manner. Modern DRM solutions allow the definition of a set of conditions and rights, under which a specific actor can use a governed digital object. These conditions are expressed using a special purpose language, based on XML [Prados et al., 2005] and are designated as Rights Expression Language (REL) [Craig and Graham, 2003]. These REL can syntactically bound a digital object identifier, an actor identifier, a content encryption key and set of conditions, together [Delgado et al., 2005]. The goals and purpose of these RELs can be characterised as the (a) expression of copyright, the (b) expression of contract or license agreements and (c) the control over access and/or use.

An important aspect of any of the rights management solutions presented in this thesis is the security. To implement such security mechanisms, rights management solutions depend on the effective use of cryptographic processes.

While designing and implementing such cryptographic processes, rights management systems have to deal with a relevant amount of cryptographic keying material. Such keying material, if not properly managed may lead to important security breaches [Cooper and Martin, 2006a].

This chapter presents contributions on the design of a generic mechanism to securely manage the licenses and the keys associated to the digital content governance. In this contribution, an analysis of the current open rights management platforms, about how the key management life-cycle is implemented, was also conducted.

Also in this chapter, an analysis of the relationship between licenses, rights expressions languages and content encryption keys is presented.

2 Licenses and Rights Expression Languages

As it was referred before, a rights expression language can be used to bound a digital object, with an actor, cryptographic material, and a group of conditions that establish the governance of the digital object. The major goal and purpose of rights expression is the definition of copyright, the expression of contract or license agreements and control over access and/or use.

Two of the used REL on today's rights management panorama are ISO MPEG-21 REL [MPEG-21-REL, 2004] (a derivative from XrML [Wang et al., 2002]) and ODRL (Open Digital Rights Language) [Wang et al., 2005].

Although RELs are a very powerful mechanism for rights expression and licenses they have little use if a system is not capable of interpret them nor capable of imposing the restrictions (if any) presented on the license [Safavi-Naini and Yung, 2006].

Two main processes occur while dealing with digital objects licenses: the creation of licenses and the usage and enforcing of the licenses. The license creation process involves the enumeration and specification of the conditions that will have to be enforced over the digital object and the necessary digital object encryption keys necessary to access the digital object [Zhang et al., 2005]. The second process is the license usage and enforcement and is

responsible for the compliance of digital object usage to the rights declared within the license, which can be enforced by the existence of content encryption or scrambling keys that might exist or not inside the license. It is therefore important for the license to be protected and managed properly. It is central to the digital object rights management that licenses, which express the way digital objects, can be used need to be managed throughout its entire life cycle [Sander, 2002].

3 The generic license management life cycle

In the curse of the digital objects rights management development, the existence of a mechanism to specify how that digital object could be used has always been important. Today's digital rights expression languages have a lot to thank to the early work developed by Xerox, giving origin to DPRL (Digital Property Rights Language) that was posterior evolved to XrML by ContentGuard [ContentGuard, 2001].

3.1 Rights management typology

Digital object rights management systems highly depend on these functionalities to express how digital objects can be used. These functionalities are present in DRM systems in many different ways [Gooch, 2003], through combinations between the existence of a formal rights expression and the content encryption keys (CEK) [Serrão et al., 2007b].

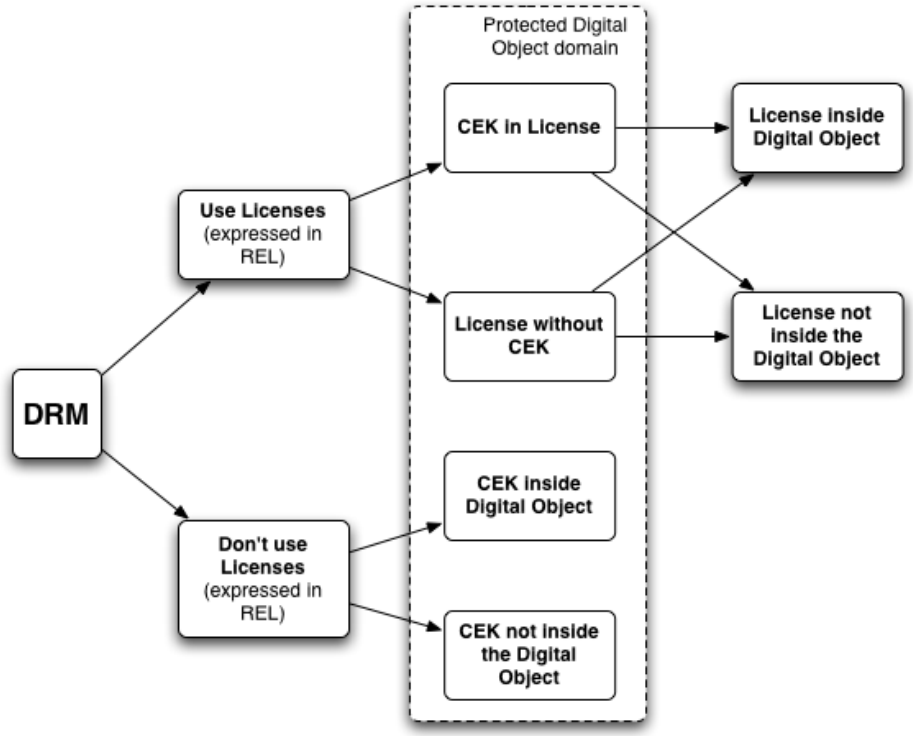

Figure III.5.1: The typology of the different types of license expression

Several different combinations and scenarios (Figure III.5.1) result in the following generic typology identification [Serrão et al., 2007c] that relates the license expression with the usage in the DRM specific instance:

1. Using licenses expressed using REL, CEK inside the license, and the license contained inside the digital object: in this case, a formal REL is used to express the digital object rights, and the content encryption key is placed inside the REL as part of the license. The license is also placed inside the digital object and is part of it – the license is obtained at the same time than the digital object;

2. Using license expressed in REL, CEK inside the license, and the license is not contained inside the digital object: in this case, a formal REL is used to express the digital object rights, and the content encryption key is placed inside the REL as part of the license. In this case the license is not part of the digital object and therefore it may be obtained at a different moment than the digital object;

3. Using licenses expressed in REL, the CEK is outside the license, however the CEK is inside the digital object: in this specific case a REL is used and the content encryption key is not part of the license, however this key is part of the digital object;

4. Using licenses expressed in REL, the CEK is outside the license, and the CEK is outside the digital object: in this case a REL is also used, however the content encryption key is not part of the license or the digital object. This means that both the license and the CEK need to be obtained in different moments apart from the digital object;

5. Not using license, but the CEK is inside the digital object: in this case a REL is not used but the content encryption key is place inside the digital object;

6. Not using license, and the CEK is outside the digital object: in this final case, no REL is used and the content encryption key needs to be obtained to access the digital object.

All the different combinations of the presented cases can be implemented in DRM solutions and scenarios. Nevertheless, the most active and representative are those who use a REL to represent licenses, and that contain the content encryption key, like for instance the Windows Media DRM. There are different cases in which the license is contained or not inside the digital object himself [Serrão et al., 2007b]. There are also some cases where a REL is not used to express the content access rights (license) and everything is left to the rendering software or device – this is for instance the Apple iTunes FairPlay DRM [Serrão et al., 2006c].

3.2 License management life cycle

The creation and usage of REL based licenses can be represented in a cycle that goes from the digital object creation to the digital object usage. There is also the case in which the final user is also a re-distributor and therefore the license rights may be changed. The enumeration and description of such cycle is quite important because it provides the mean to identify which are the basic procedures/processes in the cycle and which are the crucial security mechanisms [Ku and Chi, 2004] that need to be implemented to make the system robust (Figure III.5.2). The identification of such processes is also crucial to the identification of interoperability mechanisms in the license management life cycle between the different DRM systems [Wyant, 2002].

The license management life cycle starts with the object capture and its encoding into a

digital form (although if the object might already exist in digital format this step may be skipped or it may involve some re-coding or transcoding operation), giving origin to a digital object [Hassen et al., 2007]. This process involves the choice of the appropriate encoding mechanisms taking into account where and how the digital object is going to be used. For instance, the digital object will be encoded differently if the target is a mobile device or if it is a PC [Serrão et al., 2005d]. At this stage, the appropriate protection and packaging mechanisms may also be selected and the content encryption key (or keys) is selected and applied during the protection of the digital object [Nutzel and Beyer-b, 2006]. Depending on the digital object protection strategy, a single encryption key can be used to protect entirely the object, or several keys may be used to cipher different parts of that same object. During this phase the digital object can also be uniquely registered and assigned with a proper standardised identifier that will be used to identify uniquely the digital object in the digital world. This is a necessary step to accomplish the association of digital objects with the protection measures and to conduct tasks such as event reporting or tracing digital object usage [Serrão et al., 2007b].

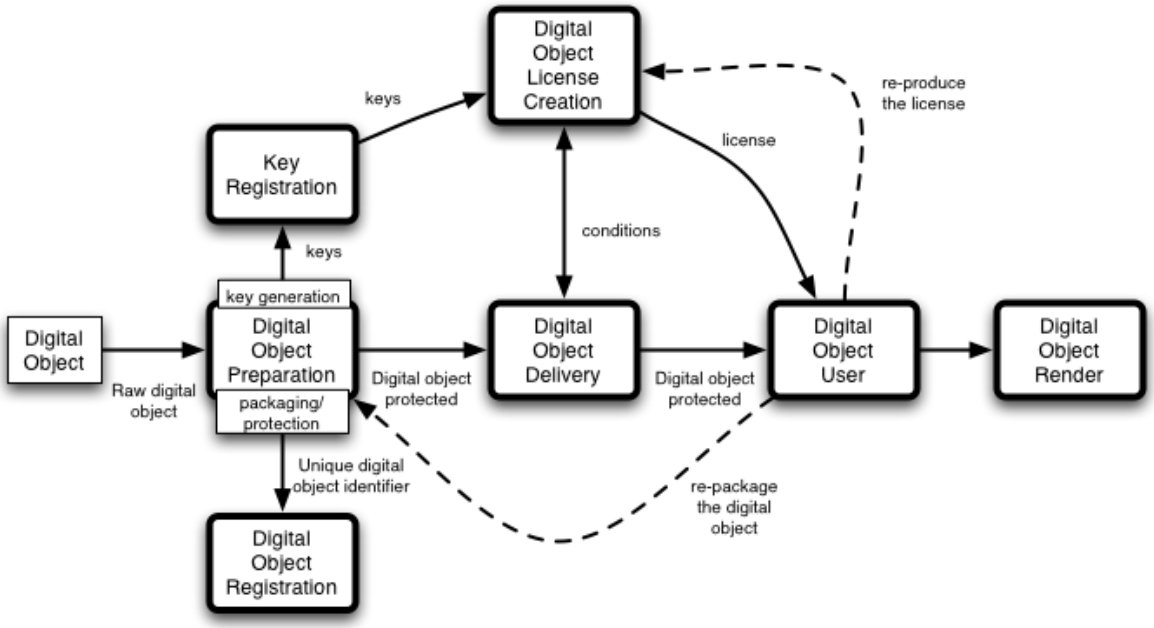

Figure III.5.2: The digital object license life cycle. This life cycle starts with the creation of the digital object and the specification of the conditions under which the digital object can be used

The keys used to cipher the content are registered on the system, and assigned to the unique digital object identifier generated on the previous step of the life cycle [Hassen et al., 2007].

These keys should be stored on a secure digital container and will be used on the future to allow the access to the digital object. This concludes the digital object preparation stage and the different digital assets are made available to final users using several distribution channels that depend on the specific business model implemented to achieve this distribution. At the end of this stage the digital object is protected through a strong cryptographic or scrambling process [Kim et al., 2006].

During the previous stage, the digital object authors, the legal copyright owners or its representatives established the conditions under which the digital objects can be licensed and used. In most cases, this involves the signature of legal contracts between the copyright holders and the digital object distributors or retailers. In principle, the conditions enforced by the copyright holders will have also to be enforced by the digital object distributors, unless otherwise stated. More, if the digital object is to be re-used and re-distributed in some way, the original licensing conditions should also apply [Serrão et al., 2007 c].

When distributing the digital object to final users the conditions for its distribution are setup and an optional negotiation process with the end user may actually occur. This is a parallel process that is outside the scope of this contribution. These conditions can be expressed in several ways, however the most common mechanism consists in using an XML-based REL. These REL can be used to define a license that bounds the digital object unique identifier, the digital object usage conditions, the content encryption key and the user or device identifier. In the previous section, the different types of relations between the REL, the content encryption keys and the digital objects were identified. In this, the most significant scenario is considered in which the license contains the content encryption keys, the license is not part of the digital object and it is obtained in a different moment than the digital object itself.

After the user receives the digital object and the license, the content encryption key and the inherent usage conditions need to be extracted from the license and used together with the user digital object handling system to allow governed access to it, upholding the conditions defined by the copyright holders and the specific conditions negotiated by the users [Serrão et al., 2007c].

4 Security in the license management life cycle

In order to fulfil its purpose, the license management life cycle has to employ the necessary security mechanisms to achieve the its goals/requirements which include the following [Serrão et al., 2007b]:

- The security of the keys generated to protect the digital objects, and securing the appropriate storage and ensuring the future availability of such keys to authenticated clients;
- The security and integrity of the content encryption key creation service, to ensure that the entity responsible for generating the content encryption keys is trustworthy;
- The security and integrity of the content encryption keys storage service. It is necessary to ensure that the service that is responsible for storing sensitive material, such as content encryption keys, is secure and trustworthy as well;
- The security and integrity of the license creation service. Creating a license in a DRM system is a serious and critical step. The service that is responsible for handling this should be strongly authenticated and trustworthy;
- The security and integrity of the content registration service. This service must also be trustworthy since it will be responsible for the unique identification of digital objects and for building mapping associations between digital objects and licenses;
- The integrity of the license structure to prevent its unauthorized modification. Unless otherwise stated by the license itself, the content of a license should be immutable. The way to achieve this immutability property is through the usage of a digital signature that is generated by the license issuer;
- The confidentiality of the content encryption key inside the license. If the content encryption keys are present inside the license they must be protected, otherwise they can be used without any restriction, retrieved from the license context and freely distributed. Therefore these keys need to be encrypted in such a way that only authorised clients can access to it in a secure way;

- The secure storage of the license on the end-user device (on a PC or non-PC). This is a critical requirement for avoiding attacks against the licenses that are stored on the end-user device, if the device has the need to store information locally. This secure storage should ensure that it is tamper proof and resistant against basic attacks, such as license replication or modification;

- The security and integrity of the license handling application, to ensure that it a trustworthy application, produced by a trustworthy developer and that it will not circumvent the rights expressed on the license. To achieve this requirement some strong authentication procedures must be conducted between the license production service and the license handling service, before the rights are transferred to the application;

- The security of the digital object rendering process, ensuring that the digital object is rendered according to the rights expressed in the license, in such a way that it does not compromises either the license, the content encryption keys or the digital object itself (if possible).

All of these processes goals/requirements need to be achieved in the specific simplified scenario identified previously. Next, the specific security procedures to address each of the goals/requirements identified, are introduced and described [Serrão et al., 2007b].

The process for generate cryptographic keys is always complex. The generation of keying material should follow a set of standards to have the necessary random data to be generated securely. Therefore the keys ($CEK_{[1]}$, $CEK_{[2]}$,… $CEK_{[n]}$) used to cipher the digital object need to be generated, stored and handled in a secure and reliable manner. The generated keying material is dependent of the type of object and type of the protection that is going to be used on the digital object (Figure III.5.3).

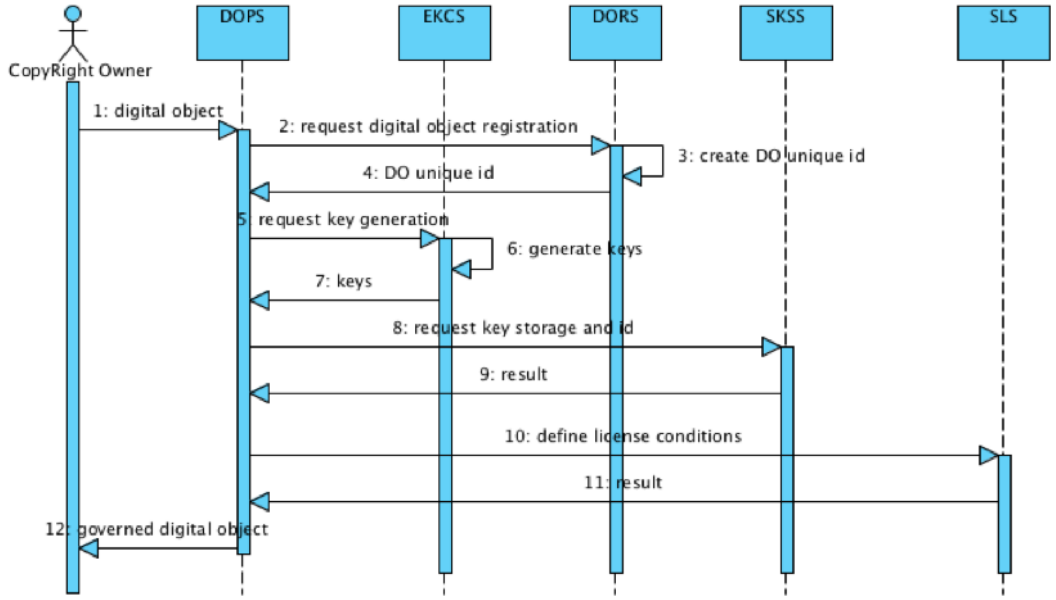

Figure III.5.3: Process to generate the license for governed digital items

These digital object protection keys need to be stored securely to be used in the future. This is necessary to avoid the compromise of these keys and to allow its posterior secure usage on the production of licenses. The keying material generation mechanism (provided by the Encryption Key Creation Service - EKCS) should own a key-pair ($EKCS_{pubk}$, $EKCS_{prvk}$) and a digital certificate issued by a trusted entity (a certification authority, which may be internal or external to the DRM system) ($Cert^{CA}_{EKCS}$).

During this stage the digital object is also registered on the Digital Object Registration Service (DORS). Also the DORS holds a key-pair ($DORS_{pubk}$, $DORS_{prvk}$) and a certificate issued by a trusted entity ($Cert^{CA}_{DORS}$). The DORS issues a Digital Object Unique Identifier (DOUI) that is digitally signed to prevent further alterations during the digital object lifetime: $DORS_{prvk}[DOUI]$. The format of the DOUI could follow any of the available standards. If the EKCS is not responsible for the long-term storage of digital object keys, then a Secure Key Storage Service (SKSS) needs to be used. This SKSS holds a key pair ($SKSS_{pubk}$, $SKSS_{prvk}$) and a certificate issued by a trusted entity ($Cert^{CA}_{SKSS}$). The EKCS after creating the keys sends them encrypted to the SKSS ciphering this information with SKSS public-key, contained in $Cert^{CA}_{SKSS}$. Additionally this information is combined with the DOUI: $SKSS_{pubk}\{ CEK_{[1]}, CEK_{[2]}, \ldots CEK_{[n]}, DORS_{prvk}[DOUI]\}$. During this stage the copyright owner (CO) also defines the license conditions for the digital object. Depending on the DRM system, the

definition of such license conditions may be performed by the instantiation of pre-defined templates, or building them from the scratch. The CO holds also a key pair (CO_{pubk}, CO_{prvk}) and a digital certificate ($Cert^{CA}_{CO}$), and will digitally sign the license conditions, expressed using a REL under which the digital object can be used: CO_{prvk}[conditions]. These conditions will be securely stored by the Secure License Service (SLS), and will be indexed by the DOUI. This means that the license conditions are tightly connected with the digital object.

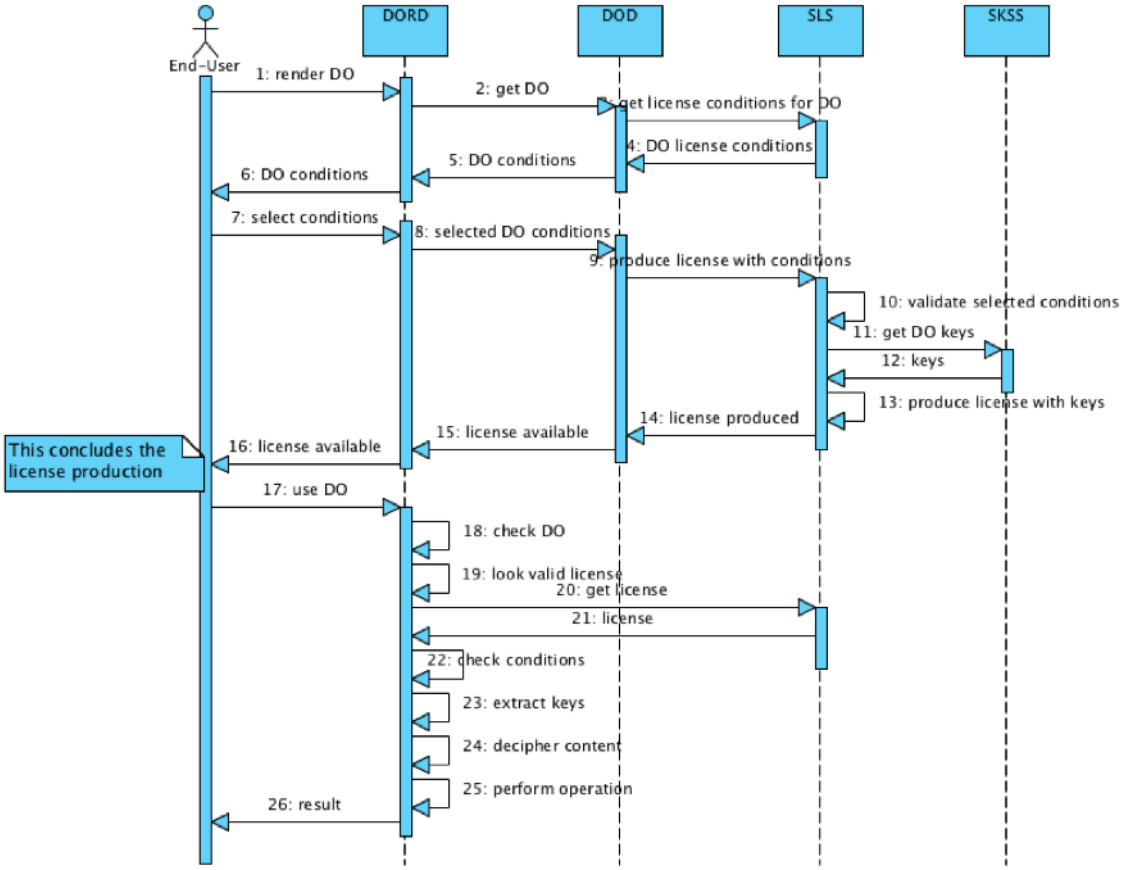

Figure III.5.4: The digital object license license acquisition and usage to enforce rights on the digital object at the end-user side

Whenever an end-user (U) desires to obtain a protected digital object on a Digital Object Provider Service (DOPS) it has to be authenticated by the rights management system that governs the digital object. To accomplish this, the user has a key pair (U_{pubk}, U_{prvk}) and a digital credential ($Cert^{CA}_{U}$) issued by a trustworthy authority recognised by the system. The U will use this credential to authenticate to the DRM system and start the digital object acquisition process. The U uses its credential to establish a secure session with the Digital Object Distributor (DOD) and the SLS, through the Digital Object Rendering Device (DORD).

The DORD, in this scenario represents a general-purpose device (hardware or software based) capable of performing some functions over a digital object. This DORD also uses it's own credentials to authenticate to the system [Serrão et al., 2006h].

During this process, the DOD loads the specific licensing conditions for the digital object that were previously established by the copyright owner. This step can be two folded: either the copyright owner allows the negotiation of some specific conditions inside the license (examples may include an expiration date, or the number of times the object can be rendered), or the license is closed and final, not allowing any kind of modifications. If the second situation occurs the end-user will have only the same type of usage defined for that specific governed digital object (Figure III.5.4). After the definition of the specific licensing conditions between the U and the DOD, the DOD instructs the SLS to produce a license for a given digital object (using the signed DOUI ($DORS_{prvk}[DOUI]$)), the U identification, represented by a specific mean, and bounded by a digital certificate ($Cert^{CA}_{U}$) and the conditions signed by the DOD ($DOD_{prvk}[conditions]$)). The SLS receives this information and validates this by the verification of the digital signature of the U, the DORS and DOD. This provides the warranty to the SLS that the request has not been tampered by some external and malicious entity. After these validations have been performed, the SLS verifies if the conditions present on the DOD request are valid and allowable by comparing them with the previous agreement with the CO. If they are, the license can be produced and the keys necessary to access the digital item can be provided to the U. With this information the SLS can obtain the CEK from the SKSS. The SLS authenticates to the SKSS using $Cert^{CA}_{SKSS}$, and uses the DOUI ($DORS_{prvk}[DOUI]$) to retrieve the appropriate keys to the digital object. These keys are ciphered with the SLS public key to avoid its compromise: $SLS_{pubk}\{CEK_{[1]}, CEK_{[2]}, \ldots CEK_{[n]}\}$. These keys will be placed inside the license for further usage, and will be given to the entity trying to access the digital object.

During the license production phase it is extremely important that the key (or keys) contained are protected from peeking eyes. Therefore the SLS ciphers these keys with the receptor (user or device) public keys, avoiding therefore that some unauthorised entity may obtain the keys and use the digital object: $U_{pubk}\{CEK_{[1]}, CEK_{[2]}, \ldots CEK_{[n]}\}$. These keys are placed together with the end-user entity identity, the digital object unique identifier, the

licensing conditions and a time validity: $U_{pubk}\{CEK_{[1]}, CEK_{[2]}, \ldots CEK_{[n]}\}$, $DORS_{prvk}[DOUI]$, conditions, timedate}. This bundle is digitally signed by the SLS to avoid unauthorised modifications of it: $SLS_{prvk}[U_{pubk}\{CEK_{[1]}, CEK_{[2]}, \ldots CEK_{[n]}\}$, $DORS_{prvk}[DOUI]$, conditions, timedate}]. After the license (LIC) is produced, a notification is returned to the DOD and optionally to the digital object-rendering device (DORD). Whenever the DORD tries to access the digital object, it verifies if it is protected or not. If so, the DORD checks for the existence of a valid license on the system (locally where the DORD is deployed and running). If the license can be found, it is verified to check if the DORD is allowed to perform the requested action over the digital object or not. This validation process involves not only the verification of the license on the system, but also the verification of the license content, such as the acquired conditions and the time validity (if present). The DORD should implement also some security mechanisms that will allow not only the secure storage of the licenses but also the mechanisms to handle the secure persistent storage of state information (such as render counters and others). This is still one of the major security breaches on software-only based DRM systems [Shapiro and Vingralek, 2002].

If a license for the digital object is not available on the system, the DORD contacts the SLS (the information about the proper SLS to contact can be placed inside the digital object, or in a more interoperable manner DORD can contact any SLS without any concern if the requested license was or not produced by the same SLS), requesting a license for that digital object (using the DOUI) and the U identification: $DORS_{prvk}[DOUI]$, $Cert^{CA}_{U}$. The information contained on this request will be signed by the U: $U_{prvk}[DORS_{prvk}[DOUI]$, $Cert^{CA}_{U}]$. This will avoid the modification of the request by some man in the middle attack. This information is received by the SLS, verified and checked against the information on the data storage, and if a matching license is found it is returned to the DORD. The license conforms to the format previously described and is signed by the SLS. When the license is on the DORD, it is securely stored and any state information (such as play counters) will have to be instantiated. While accessing the digital object, the DORD securely retrieves the digital object protection keys from the license, and uses them to securely render the deciphered content and allow the execution of the requested operation over the digital object. The described process corresponds only to one of the many possible scenarios that were

identified, in which the combination of REL, content encryption keys and digital objects may coexist [Zeng et al., 2006].

5 Open DRM Secure key management

As it was referred throughout this thesis, DRM interoperability is a key technological aspect for the DRM success [Schmidt et al., 2004]. Rights management non-interoperability causes notorious setbacks to users, which have their rights restricted in a way that will prevent them from being able to use its legal DRM-governed content wherever and whenever they like. Currently there is no interoperability mechanism capable of allowing the usage of different DRM-governed digital objects over different DRM platforms.

Solving this interoperability issue is not an easy task. It involves many different aspects that cannot be solved out of the box. Moreover, most of the times, interoperability issues are driven by the business model than by technology itself. Nevertheless, the focus will be on technology issues, and to that matter aspects like device incompatibilities, different language expression, different content protection algorithms, different protocols, different key management, among other aspects [Serrão et al., 2005d].

In this section, not all the technical aspects of interoperability are considered. This section will focus mainly in the key management aspects of the different rights management solutions, where an analysis will be conducted on how the different open rights management solutions (see Part III, Chapter 4) handle this key management life cycle and a comparison between them will be established. The objective of this is to identify which are the aspects of key management which are considered and which ones are left behind.

5.1 Key management in open rights management systems

Current initiatives that define rights management systems or the elements that are part of rights management systems, are specifying content formats, protection information, license creation and authorisation mechanisms based on licenses, authentication infrastructures, mechanisms for notifying the events within the system, among others. However, key management is not completely specified in most of the rights systems and sometimes is

forgotten or avoided, as for example in the MPEG-21 standard specification [Bormans et al., 2003] (it is out of the scope) [Serrão et al., 2007b].

On the other hand, most of the rights management systems specify or chose an existing REL for declaring the rights and conditions that govern the use of the digital content. Nevertheless, they do not specify how sensible data within licenses can be protected or how licenses can be securely delivered to users.

Some of these DRM architectures are closed environments, only for issuing licenses and delivering content using internal servers, others are middle-ware interoperable systems which support relating each others in order to make better the user experience and easier business relations.

Such rights management architectures have been developed in order to work in a particular environment, using different controlled/non-controlled hardware devices so it can be possible to compare what is being done in terms of security in our environment.

5.2 Key management and key management life cycle

Key Management refers to the set of techniques and procedures supporting the establishment and maintenance of keying relationships between authorised parties [Menezes, 1996]. Key Management encompasses a set of techniques and procedures for supporting:

- The initialisation of users within a specific given domain;
- Generation, distribution and installation of keying material;
- Controlling the usage of keying material;
- Update, revocation and destruction of keying material;
- Storage, backup/recovery and archival of keying material.

The main objective of Key Management is to maintain keying relationships and keying material in a manner which counters relevant threats, such as:

- The compromise of secret keys;

- The compromise of private and public keys;
- The unauthorised use of secret or public keys.

One of the main aspects of key management is the definition of a security policy. The security policy implicitly or explicitly defines the threats a system is intended to address [Nutzel and Beyer-b, 2006].

Security policies usually specify:

1. The practices and procedures to be followed in carrying out technical and administrative aspects of key management, both automated or manual;
2. The responsibilities and accountability of each of the party involved;
3. The types of records to be kept, to support subsequent reports or reviews of security-related events.

The sequence of states which keying material progresses through over its lifetime is called the Key Management Life Cycle (Figure III.5.5) [Menezes, 1996]. There are three different keys stages within the life cycle:

- **Pre-operational**: The key is not yet available for formal cryptographic operations;
- **Operational**: The key is available and in normal use;
- **Post-operational**: The key is no longer in normal use, but off-line access to it is possible for special purposes.

The different key management life cycle stages may include:

- **User registration**: An entity becomes an authorised member of a security domain. This involves acquisition, or creation and exchange, of initial keying material such as shared passwords or PINs by a secure, one-time technique.
- **System and User initialisation**: System initialisation involves setting up/ configuring a system for secure operation. User initialisation involves the initialisation of its cryptographic application.

- **Key generation**: Generation of cryptographic material should include measures to ensure appropriate properties for the intended application or algorithm and randomness in the sense of being predictable with negligible probability. An entity may generate its own keys or acquire keys from a trusted system component.
- **Key installation**: Keying material is installed for operational use when the software, hardware, system, application, crypto module, or device is initially set up, when new keying material is added to the existing keying material or when existing keying material is replaced. The test keying material must be replaced prior to operational use.
- **Key registration**: Keying material is bound to information or attributes associated with a particular entity. This information may include the identity of the entity associated with the keying material and the authorisation information or specify a level of trust. This step is typically performed when the entity is a participant in a key management infrastructure.
- **Normal usage**: The objective of the key management life cycle is to facilitate the operational availability of keying material for standard cryptography purposes. Under normal circumstances, a key remains operational until the end of the crypto-period.
- **Key backup**: the backup of keying material in independent, secure storage media provides a data source for key recovery. The backup refers to short-term storage during operational use.
- **Key update**: Prior to crypto-period expire, the operational keying material is replaced by new material. This may involve some combination of key generation, key derivation, execution of two-party key establishment protocols, or communications with a trusted third party. For public keys, update and registration of new keys typically involves secure communication protocols with certification authorities.
- **Archival**: Keying material no longer in normal use may be archived to provide a source for key retrieval under special circumstances. Archival refers to off-line long-term storage of post-operational keys;

- **Key de-registration and destruction**: Once there are no further requirements for the value of a key or maintaining its association with an entity, the key is de-registered, and all copies of the key are destroyed. In the case of secret keys, all the traces are securely erased.
- **Key recovery**: If keying material is lost in a manner free of compromise, it may be possible to restore the material from a secure backup copy.
- **Key revocation**: It may be necessary to remove keys from operational use prior to their originally schedule expire, for reasons including key compromise. For public keys distributed by certificates, this involves revoking the certificates.

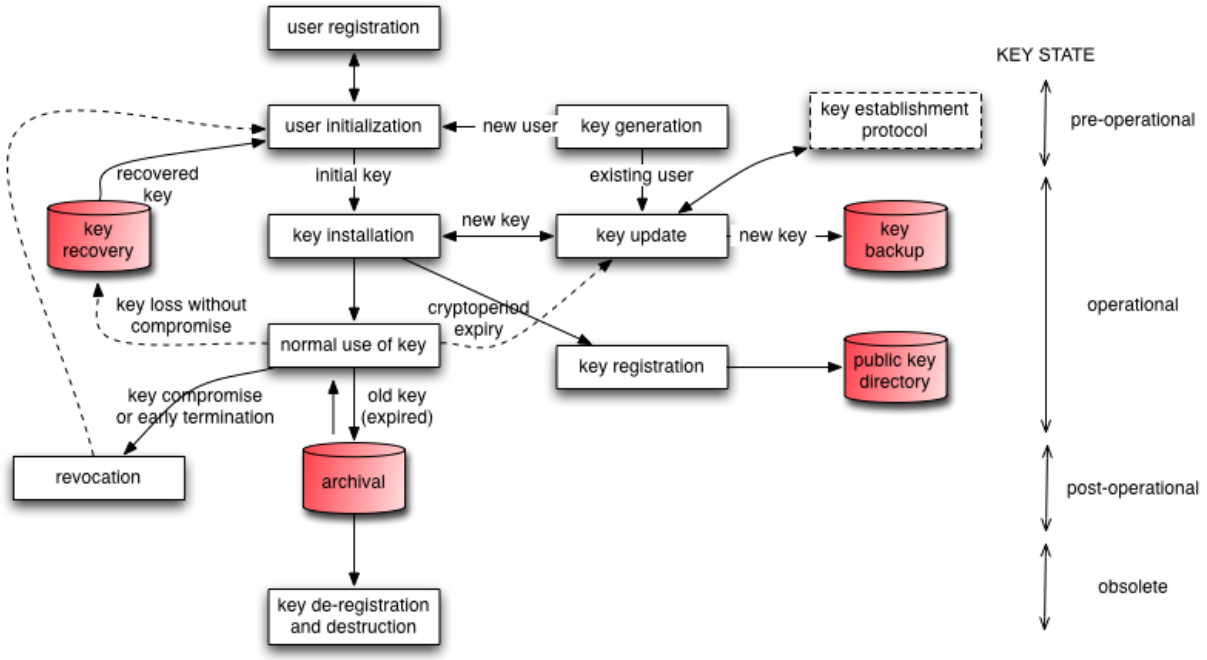

Figure III.5.5: Key management life cycle

5.3 Analysis of key management life cycle on open rights management platforms

In the previous section a description of the key management and key management life cycle was provided. In this section an analysis on how key management and key management life cycle is handled by different selected open rights management solutions is presented.

In this analysis, the open rights management solutions, that were considered on the previous

chapter (see Part III, Chapter 4), were selected. Therefore, the following rights management systems were selected for this analysis: OpenSDRM, OpenIPMP, DMAG MIPAMS, AXMEDIS, DMP Chillout, OMA DRM and Sun DReaM. The following platforms were selected due to the availability of clear and open specifications or source code implementations of such specifications [Serrão et al., 2007c].

OpenSDRM is a complete DRM platform developed in the context of this PhD thesis and for a former European IST FP5 project called MOSES [MOSES, 2007]. It has been developed to offer end-to-end support to the production, distribution and consumption of protected digital objects. OpenSDRM source-code is available to the open-source community to stimulate its future development. Therefore, OpenSDRM has both open-source and specifications available.

OpenIPMP [OpenIPMP, 2003], is a collection of tools and services capable of delivering a robust, scalable, and adaptive infrastructure to support the management and secure delivery of media assets through each step in the asset life cycle. The value proposition is fundamental but unique in its approach: asset security and control utilising an open framework. The “open” portion of OpenIPMP refers to its stated goal of being developed as an Open System adhering to Open Standards. Open Systems conform to industry standards enabling interaction between various hardware and software products. OpenIPMP is an open-source product and the source-code can be obtained.

DMAG MIPAMS [Torres et al., 2005] (Multimedia Information Protection and Management System) manages multimedia information using DRM and content protection [Alberti et al., 2003]. The architecture consists of several modules and services, which provide a subset of the whole system functionality needed for managing and protecting multimedia content. MIPAMS is a service-oriented DRM platform and all its modules use the web services flexible approach. MIPAMS encompasses an important part of the content value chain, from content creation and distribution to its consumption by final users. MIPAMS is not open-source, but it has open specifications.

AXMEDIS is an European Project, aiming to create and exploit innovative technological framework for automatic production and distribution of cross-media contents over a number of different distribution channels (e.g., networked PC, PDA, kiosk, mobile phone, i-TV, etc)

with DRM. For that purpose, AXMEDIS has designed and implemented a DRM architecture that consists on several independent modules that interact as web services or directly. AXMEDIS is not open-source, but it possesses open specifications.

DMP Chillout [DMP, 2007] is the name of the Interoperable DRM Platform (IDP) Reference Software, released as open-source software by the Digital Media Project initiative. Chillout currently provides a set of open-source Java libraries implementing DRM functions and Java applications built on top of the Chillout libraries. Chillout can be used to test conformance of independent implementations of the DMP specifications and to set up interoperable value chains for use by independent users. Chillout is the result of DMP which was established in 2003. The IDP is a toolkit, i.e. a set of standardised DRM tools based on "primitive functions" derived from existing digital media systems by investigating several selected use cases.

OMA DRM (Open Mobile Alliance DRM) [OMA, 2006a] has been developed to enable the controlled consumption of digital media objects by allowing content providers the ability, for example, to manage previews of DRM Content, to enable super distribution of DRM Content, and to enable transfer of content between DRM Agents. OMA DRM has a complete set of open specifications.

Sun DReaM [Fernando et al., 2005], is an initiative to develop an open DRM solution for multiple domains (media, documents, enterprise, personal, etc.). DReaM open source project develops an end-to-end reference implementation for the DReaM Specifications. This is to enable a quick-start for DRM solutions. The DReaM project is open-source and open specifications based.

In these open DRM solutions, four are open-source projects (the source code is available) while the other three have open-specifications available.

During the analysis process conducted to the different DRM solutions, the first step consisted on the identification of the key management aspects and on a second step, the specific key management operations were analysed.

OpenSDRM is not very well documented in terms of specifications. Some generic specifications are available, but some internal details are missing. However, from the source code analysis it is possible to extract several findings. OpenSDRM does not cover the full key

management life cycle as it has been presented before. It only offers support for the user and system registration and initialisation phases and also normal operation. All the subsequent phases of the key management life cycle are disregarded, especially those related to revocation and destruction of obsolete keying material. OpenSDRM is a distributed architecture with multiple components. Each of the components uses key-pairs and digital certificates to exchange securely information between them. OpenSDRM is based on both X.509 certificates and a specific certificate format based on XML. In OpenSDRM, X.509 certificates are used to authenticate the different components at the network level, using the SSL protocol, while the XML based certificates are used to establish trust relationships at the application level [Serrão et al., 2007c].

OpenIPMP is entirely based on X.509. OpenIPMP offers coverage of the same coverage of the key management life cycle than OpenSDRM, however it also includes certificate revocation, and therefore key revocation. OpenIPMP key management implementation is based on Enterprise Java Beans Certification Authority (EJBCA), a reliable Java-based J2EE Certification Authority that has the possibility not only to issue X.509 certificates but also to process certificate revocation. However, OpenIPMP does not offer support for key backup, update, archival, recovery or destruction. Therefore, similarly to other open DRM solutions, OpenIPMP key management life cycle support is not fully complete.

In the case of DMP's Chillout, the implementation of the DMP specifications are available in open-source format. The IDP specifications are quite complete and descriptive. The implementation is quite extensive, however it is not yet complete, since it is work in progress. In terms of key management and key management life cycle, both the specifications and the implementation provide very few details. From the analysis conducted to the specification and source-code it is possible to conclude that the key management life cycle is poorly covered both by the IDP specifications and by the Chillout implementation. In this implementation it is possible to observe that (like other open DRM solutions) aspects related to key archival, backup, recovery, destruction and revocation are not or poorly handled.

AXMEDIS is also based on X.509 certificates. AXMEDIS offers the same coverage of the key management life cycle than OpenIPMP, including certificate and key revocation in some

specific cases, as when a client tool is detected to be corrupted. AXMEDIS also supports automatic key update when an AXMEDIS tool detects that the user or tool certificate have expired. In the case of the user certificate, the application redirects the user to the registration portal. In the case of the tool certificate, the tool is automatically re-certified after checking its integrity. AXMEDIS, like OpenIPMP, also uses EJBCA. However, AXMEDIS does not offer support for key backup, update, archival, recovery or destruction. Therefore, similarly to other open DRM solutions, AXMEDIS key management life cycle support is not complete [Torres et al., 2006].

OMA DRM does not have an open-source implementation available. However, the OMA DRM specifications and freely available and offer a detailed overview of the OMA DRM system and the way it should operate. From the key management point of view, the OMA DRM specifications are one of the most detailed (from the ones analysed), covering many aspects which are usually left out on other DRM solutions. These aspects includes issues like the type of cryptographic algorithms that need to be used, the format of keys and the detailed description of the protocols. OMA DRM dedicates a specific section of the overall specification to key management, where it covers aspects such as cryptographic components description, key transport mechanisms and key distribution. However, as in other specifications, a lot of assumptions are made in terms of key management and very few is said about both the key initialisation and the key revocation, archival, backup and destruction [Torres et al., 2006].

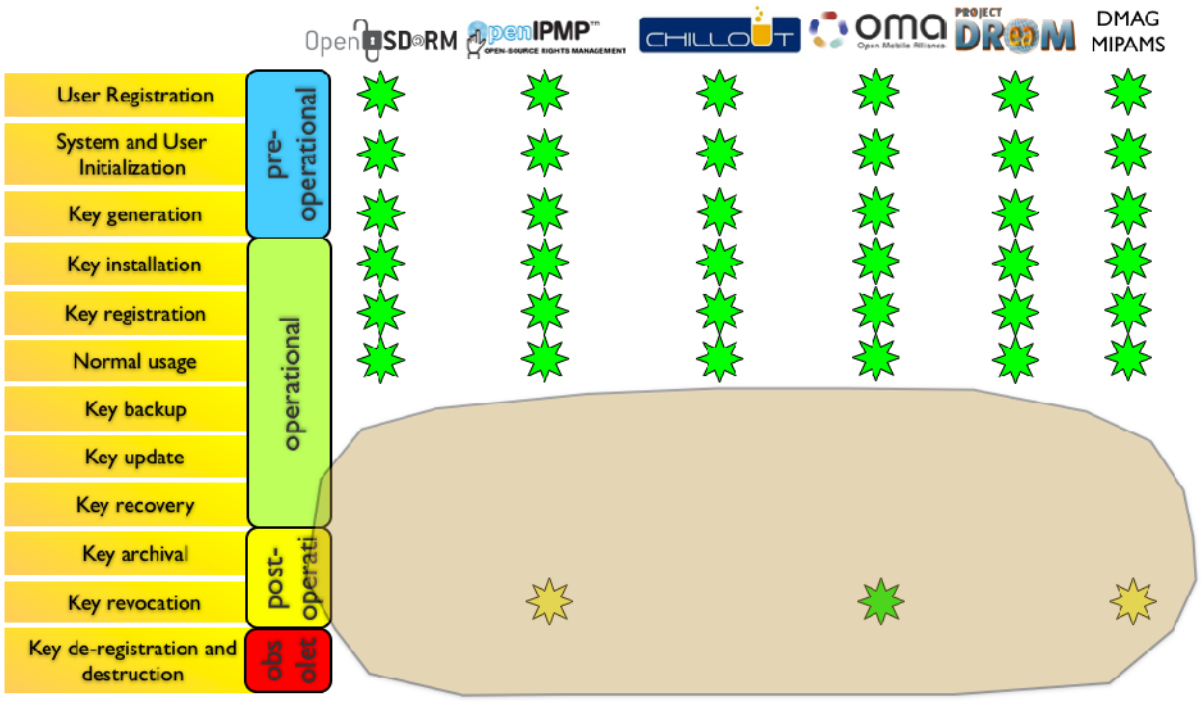

Figure III.5.6: Comparison between open DRM platforms in key management aspects (X - fully covered, * - partly covered)

In the Sun DReaM case, both the source-code and the specifications are available. However, they are both in a very early stage of development. On what concerns specifications they are reduced to two documents that describe the Conditional Access System (CAS) and the Mother-May-I (MMI) protocol. On what concerns key management, this is an issue that is treated in a superficial manner on the specifications. They concentrate on the content encryption key protection specification, however they relegate other key management aspects to the existing of an underlying PKI solution [Serrão et al., 2007 b].

DReaM is still in a very early stage of development on what concerns source-code and therefore it is hard to evaluate how much the key management life cycle is going to be implemented by Sun's DReaM final solution.

From the analysis of these selected open rights management solutions (Figure III.5.6) it is possible to observe that key management and key management life cycle are still two aspects which are not very well considered in the design and deployment of current rights management solutions.

In particular, considering the different stages of the key management life cycle as they were

presented previously, most of the analysed open rights management solutions only offer specifications and/or implementation for the pre-operational and part of the operational stages, which include User Registration, System and User initialisation, key generation, key installation, key installation, key registration, and normal usage [Serrão et al., 2007c].

On what concerns the other part of the operational, post-operational and obsolete stages (including key backup, key update, archival, key de-registration and destruction, key recovery and key revocation), they are disregarded by most of the studied open DRM solutions [Safavi-Naini and Yung, 2006].

6 Conclusions

The main focus of this chapter was on the way most digital object rights management solutions handle with the management of the digital representation of rights. In this chapter it was also identified and briefly described a set of different scenarios, about the usage of digital rights expression languages for expressing digital object licenses, the presence of the content encryption key inside the licenses and the presence of such licenses inside the digital objects. This identification has resulted in six different scenarios, and the most relevant one (implemented in the most significant rights management solutions today) has been selected and the license management life cycle was described. After the identification and description of the major processes in the selected scenario of license management life cycle model, it were identified the basic security procedures that make the license management processes effective on the digital objects rights management. Crucial aspects such as confidentiality, integrity and authentication are of extreme importance and therefore need to be used with care to offer trust across the entire license management life cycle [Serrão et al., 2007b].

The other fundamental topic of this chapter was key management. Key management plays a fundamental role in cryptography as the basis for securing cryptographic techniques providing confidentiality, entity authentication, data origin authentication, data integrity, and digital signatures. The goal of a good cryptographic design is to reduce more complex problems to the proper management and safe-keeping of a small number of cryptographic keys, ultimately secured through trust in hardware or software by physical isolation or procedural controls [Sun et al., 2007].

Systems providing cryptographic services, such as rights management solutions, require techniques for initialisation and key distribution as well as protocols to support on-line update of keying material, key backup/recovery, revocation, and for managing certificates in certificate-based systems.

Key management have a significant role on rights management systems [Pinkas, 2004]. Such systems deal with a large amount of cryptographic material that needs to be properly managed to avoid security breaches. This is a serious situation and modern rights management solutions should be able to handle it [Serrão et al., 2007c].

In order to evaluate how modern rights solutions were dealing with key management aspects and how they implement the key management life cycle, different open rights management solutions were selected and an analysis was conducted. From this analysis it was possible to conclude that although all of these open rights management systems had some kind of key management mechanism considered or implemented, this mechanisms did not implemented the full key management life cycle. In most of the cases analysed only the pre-operational and part of the operational stages were considered [Serrão et al., 2007b].

The lack of an appropriate key management scheme in rights management solutions could lead to some serious security problems, such as:

- the compromise of confidentiality of secret keys;
- compromise of authenticity of private or public keys, and;
- the unauthorised usage of private or public keys.

It is therefore important, that rights management solutions consider the different aspects of key management as a way to reduce potential flaws and security risks. This aspect should be considered on the design of such rights management solutions.

Chapter 6. The OpenSDRM open DRM architecture

1 Introduction

This part of the document provides information and details the contribution made in the study, design and development of an open rights management service-based architecture, called OpenSDRM [Serrão et al., 2003b]. OpenSDRM was one of the first breakthrough open rights management architectures to be designed from scratch considering the emerging service-oriented paradigm and designed to be open-source and open standard-based [Serrão and Siegert, 2004a] (see Part III, Chapter 2).

This chapter details not only the different components of the open DRM architecture, but also present the different services which are part of OpenSDRM. This architecture, developed in the context of this PhD thesis, has been used in several and different digital content business scenarios, with proper adaptations and has proved to be a good tool for experiences in the electronic commerce of governed digital content field.

The chapter starts by making an introduction to some of the basic concepts behind the OpenSDRM architecture design and introduces some of the interoperability mechanisms implemented in the architecture.

After this introduction, the chapter follows a top-down approach to detail the OpenSDRM architecture. First, the different components and actors, which interact with the OpenSDRM platform, to build end-to-end digital content business scenarios, are identified, described and the interactions between such and OpenSDRM are presented. Second, the internal

OpenSDRM services are presented and described, and the interactions between the different services are also presented.

After the presentation and description of the OpenSDRM architecture, this chapter details some of the security mechanisms used within the architecture, and a description of some of the protocols used is presented. Some of the security mechanisms will be more detailed in other chapters of this document (see Part III, Chapter 7).

Both the architecture design and the implementation of OpenSDRM have been contributed to the open-source community and are now available for everyone to test and further development. OpenSDRM can also be used as a test-bed for governed digital content business models testing and evaluation.

Moreover, the OpenSDRM architecture has been the base for some contributions made for some standardisation initiatives, such as JPEG2000 (JPSEC) [Serrão et al., 2002], MPEG-21 IPMP [Serrão et al., 2004a][Sheppard, 2007] and the Digital Media Project (DMP) [Serrão et al., 2004b].

Finally, at the end of the chapter some conclusions are presented.

2 OpenSDRM Architecture

The OpenSDRM architecture was developed to address the possibility of adaptation to several business models and different types of digital content, aiming at enabling business involving multimedia content to function, by enforcing licensing agreements for content use and offering business opportunities to the content rights owners and content providers [Serrão and Siegert, 2004a]. At the time of development, various players and approaches in the rights management arena were already in place, some considering interoperability aspects while others did not. One of such approaches addressed the rights management interoperability problem between different actors of the value chain by defining one unique media format and one type of technical solution used by every device. The format may be proprietary or open. The Open Mobile Alliance (OMA) [OMA, 2007] has targeted the DRM support for mobile phones and appliances, using an open specification. For the broadcast area and the home network Digital Video Broadcast DVB-CP/CPT [Hibbert, 2005] is an on-

going step towards the standardisation of a Content Protection and Copy Management system, within the home network. Other vendors [Microsoft, 2004][Lenzi et al., 2003][RealNetworks, 2007], developed their own vertical strategies, seeking and competing to establish a “de facto” standard in the rights management arena. Contrary to these efforts, during the OpenSDRM DRM design, specification and implementation, a horizontal approach was followed (in opposition to the vertical one described before). In this approach, the implementation has followed the guidelines and specifications of open standards, which address the DRM interoperability by defining common interfaces, tools and mechanisms that different DRM solutions should comply [Serrão et al., 2003b].

One of the major design goals of OpenSDRM architecture was to be adaptive, in the sense that that it could be configured to use with several digital content business models and to support different types of content. OpenSDRM deploys a rights management solution for content rights protection and can be applied for publishing and trading of digital content. Additionally, the security architecture design has took in consideration and reflects the work done by the OPIMA international specifications [OPIMA, 2000][Kudumakis, 2000a][Koenen, 2002], MPEG-4 IPMP Hooks [Lacy et al., 1999] and Extensions [Kudumakis et al., 2000b][King et al., 2001] and has contributed and taken in consideration the work done by MPEG-21 IPMP Components [Serrão et al., 2004a][Bormans and Hill, 2002], DVB [Kudumakis et al., 2001], ISMA [ISMA-DRM, 2005] and DMP [Chiariglione et al., 2004][Serrão et al., 2004a].

The design and development of OpenSDRM started within the EC IST FP5 MOSES project [Kudumakis, 2003]. MOSES was an EC project joining some companies over Europe that is implementing the new MPEG-4 IPMP Extensions framework and at the same time developing business models and applications for secure content exchange between embedded devices [Lacy et al., 1999], where the author has participated.

OpenSDRM design considered end-to-end digital content business model scenarios and therefore in the architecture designed tried to cover all the different steps of the digital content value-chain (Figure III.6.1). The solution offers support for content rights establishment and validation, content registration and repository, content trading and distribution, rights definition and license issuance, content payment and rights management control [Serrão et al., 2003b].

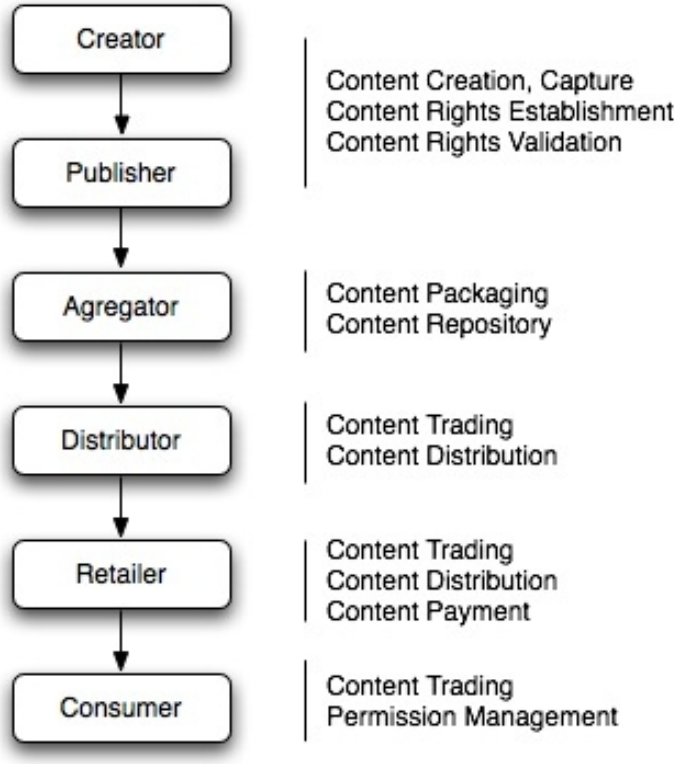

Figure III.6.1: Digital content value chain – from digital creation to consumption. Protection and management of rights through this chain is one of the major functions of DRM

3 Introduction to OpenSDRM

OpenSDRM was designed having into account some basic principles to promote as much as possible interoperability. There are two basic levels of interoperability that OpenSDRM supports.

The first refers to the support that OpenSDRM offers for different implementations (even closed and proprietary) to be used within the global OpenSDRM architecture, through a well-defined service-oriented approach, in which public interfaces, described using Web Services Description Language (WSDL) (see Annex A), allow the integration of these services with the remaining OpenSDRM services (Figure III.6.2).

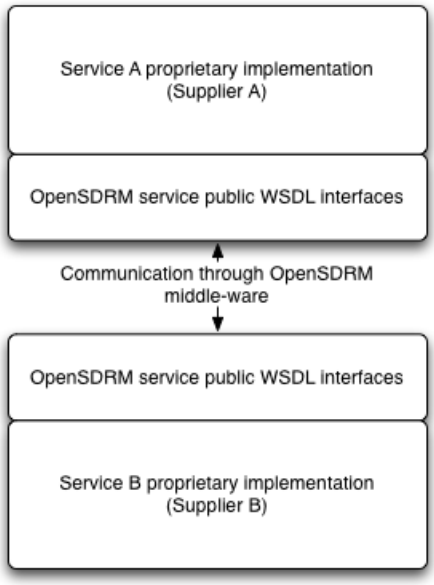

Figure III.6.2: Different service providers using a common OpenSDRM architecture and middle-ware functionalities

In this first approach, the different service providers can implement their own closed versions of any of the services in the OpenSDRM architecture. These closed implementations share the public OpenSDRM interfaces, that allow these services to be OpenSDRM-compatible and therefore interoperable between each other.

The second consists in the fact that OpenSDRM was designed having in mind the possibility that there could exist different service providers, providing the same service, coexisting in the same OpenSDRM architecture and environment (Figure III.6.3) [Serrão et al., 2006b].

With this second approach multiple different suppliers can implement the same service, and interoperate with the other services implemented by different suppliers. Again, the OpenSDRM open WSDL interfaces help to glue everything together (see Annex A).

An example of such is the case of the Payment Gateway. There may exist different Payment Gateways implementations depending on the payment mechanism that is supported, but they both will be able to offer their services, through an open interface to the remaining OpenSDRM services [Serrão et al., 2003a].

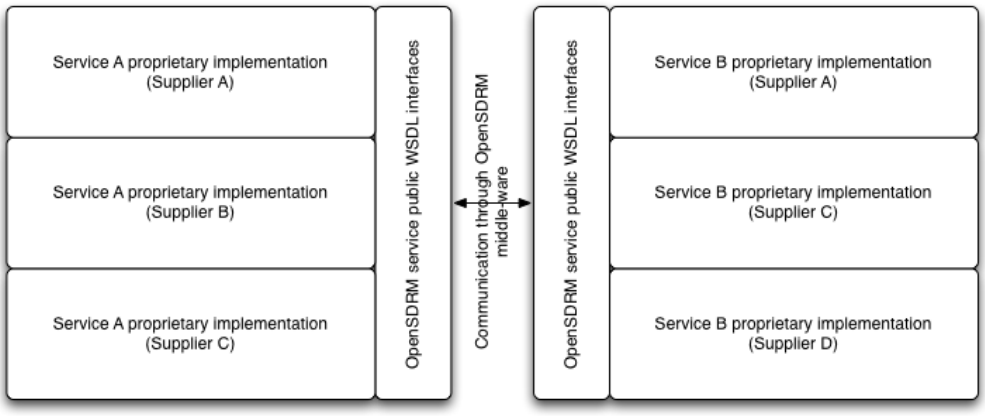

Figure III.6.3: Different service providers, providing the same service, using OpenSDRM

The DRM solution is composed of several optional elements and services covering the digital content value-chain, from content production (content author or producer) to content usage (final user). It covers several major aspects of the content distribution and trading: content production, preparation and registration (Content Preparation service, Registration service), content protection (Registration service, License service, Protection Tools service and Authentication service), interactive content distribution (Media Delivery service), content negotiation and acquisition (Commerce service, Payment Gateway service), strong actors and users authentication (Authentication service) and conditional visualisation/playback (Media Player, Protection Tools service, License service) [Serrão et al., 2003b].

Among the external actors and components are the Certification Authority, the Protection Tools Provider, the Content Provider, the Payment Infrastructure and the final User. These external components and actors are described in the next section .

4 External components and actors

This part will present and detail the components and actors that interact externally with the OpenSDRM architecture: User, IPMP Tools Provider, Content Provider, Payment Infrastructure and Certification Authority.

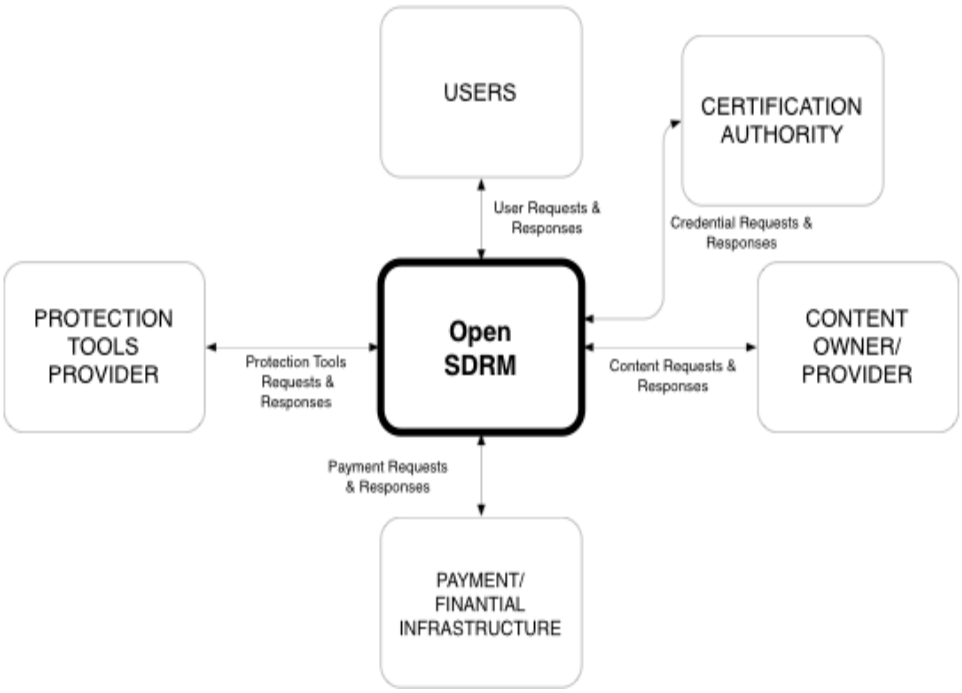

Figure III.6.4: Interaction between OpenSDRM and external actors and systems

As it was referred previously, the OpenSDRM design took into account the possibility to support an end-to-end solution, covering the major steps of the digital content value-chain, and therefore a set of different external entities and actors have been involved to establish a global framework (Figure III.6.4). The following parts detail these entities and actors and the way they interact with the OpenSDRM architecture [Serrão et al., 2003b].

4.1 User

The **User** represents a person who wishes to operate a way of enjoying some piece of digital content (this content may or may not be protected, however the way to access and display such content may require the use of protected devices, software and licenses). The user will make requests to OpenSDRM in order to: provide some identification information, request digital content and licenses and render the digital content (Figure III.6.5). These requests are produced through an appropriated interface (such as a web browser or any other specific application (EPG[16], Media Player)).

16 Electronic Program Guide

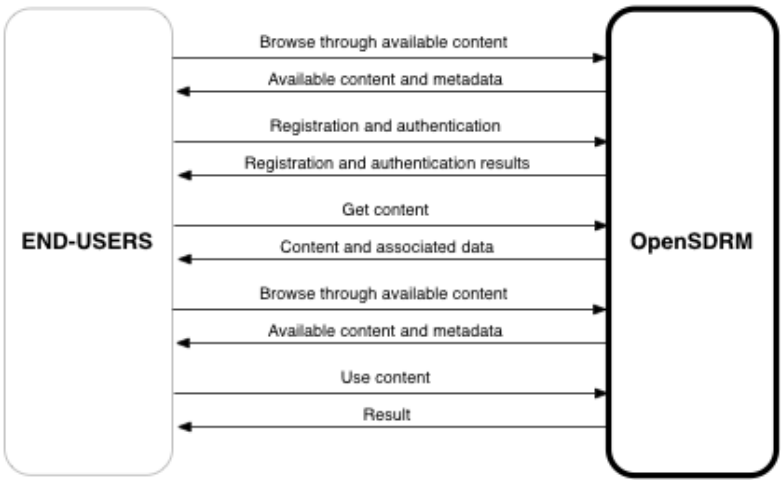

Figure III.6.5: Interface between the End-User and OpenSDRM

OpenSDRM will provide an appropriate response to the User, for example, by playing the multimedia content. Resuming, the User interaction with the OpenSDRM architecture will always result in one of two actions: either the User can render the digital content and enjoy it or he cannot; being then informed of the reason for such prevention. In this context, the render action can be not only the playback or display of content, but in general the ability to perform one operation (use, move, copy, modify, and others) over the content [Serrão et al., 2005d].

4.2 Protection Tools Provider

One of the security concepts behind OpenSDRM as a secure and open rights management platform is renewability. This is an important concept, because if one of the secure components of the architecture (in particular those who are related with digital content protection or digital rights protection) gets compromised in some way, it will be possible to enforce its replacement. This replacement will occur by blacklisting all the compromised components and force the download and installation of new ones, in order to be able to continue to use the content.

The Protection Tools Provider represents in an abstract form any individual, entity or organisation that produces software tools for encryption, scrambling, watermarking and others that can be applied to content protection. These tools will be made available to OpenSDRM for use in content data and rights protection (Figure III.6.6).

Figure III.6.6: Interface between the Protection Tools Provider and OpenSDRM

A registered Protection Tools Provider can request to the OpenSDRM platform the registration and deposit of a list of protection tools that will be made available to third parties to implement their protection mechanisms over content and rights. Any Protection Tools Provider needs to be registered on the OpenSDRM platform to be able to upload the protection tools. These tools will need to comply with some rules and to ensure that these tools are not compromised by default a security audit may have to be performed (this is however out of the scope of this work). The protection tools will need to be accompanied by a detailed description of how they can be used and in which situations. The tools should also have a public API description to allow the usage by third parties. In most of the cases, a business relation may already exist between a given Content Provider and a Protection Tools Provider, since a given content producer and/or distributor may want to choose which type of protection the content will have and which tools can be applied to the content and from which supplier. This situation is also covered by the OpenSDRM architecture [Serrão et al., 2005d].

4.3 Content Provider

One of the objectives of OpenSDRM, as a rights management platform, is to remain as much as possible digital content agnostic. This means that the platform, with minor adaptations, would be able to support almost all different types of digital content.

The Content Provider is any digital content supplier that feeds OpenSDRM with content and/or metadata. Any registered and authenticated Content Provider can upload digital content to the platform (Figure III.6.7). An alternative way to use it is that the Content Provider can push a link to the content, but not the content itself – this is the case for streaming, media casting or broadcasting models. The Content Provider is also responsible

for the provision of the appropriate metadata about the provided digital content that will be presented to end-users. Digital content can be any type of complex multimedia content that is ready for distribution, or simple content, that can be edited and combined with other content [Serrão, 2004].

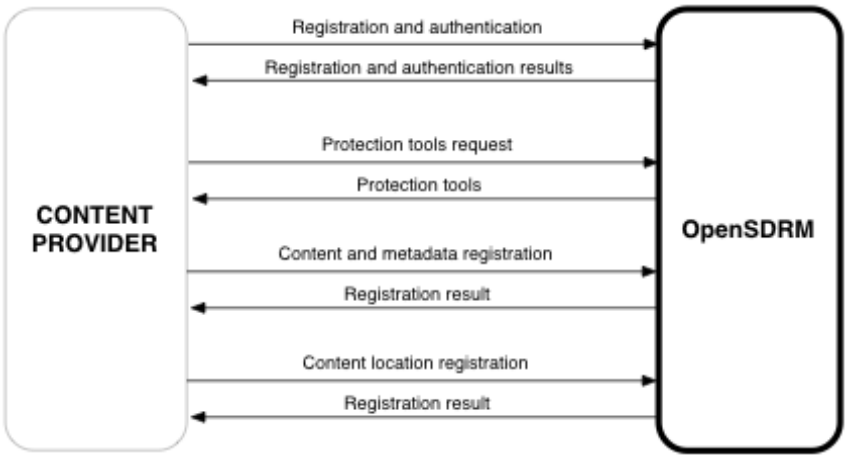

Figure III.6.7: Interface between the Content Provider and OpenSDRM

A Content Provider must be registered on the OpenSDRM platform in order to be able to use it. This may require the existence of some business relation between the actual author of content and the content distributor (for instance, songwriters or bands and editing companies) [Serrão, 2004]. In a more abstract way, OpenSDRM may lead to the disintermediation in the digital content value-chain, since it may enable a direct link between the content author (taking the role of Content Provider) and the end-user [Serrão, 2002].

4.4 Payment/Financial Infrastructure

In an effort to create an end-to-end architecture, one of the issues that OpenSDRM had to deal is digital content payments. The Payment/Financial Infrastructure facilitates OpenSDRM e-commerce features by providing services for handling electronic payments. The Payment/Financial Infrastructure will be any infrastructure capable of handling payments that is properly registered and identified by OpenSDRM (Figure III.6.8).

The OpenSDRM platform can issue requests to the Payment/Financial Infrastructure requesting authorisation for payments to be performed. The Payment/Financial Infrastructure replies to OpenSDRM indicating if the payments are authorised or refused

[Serrão et al., 2003a].

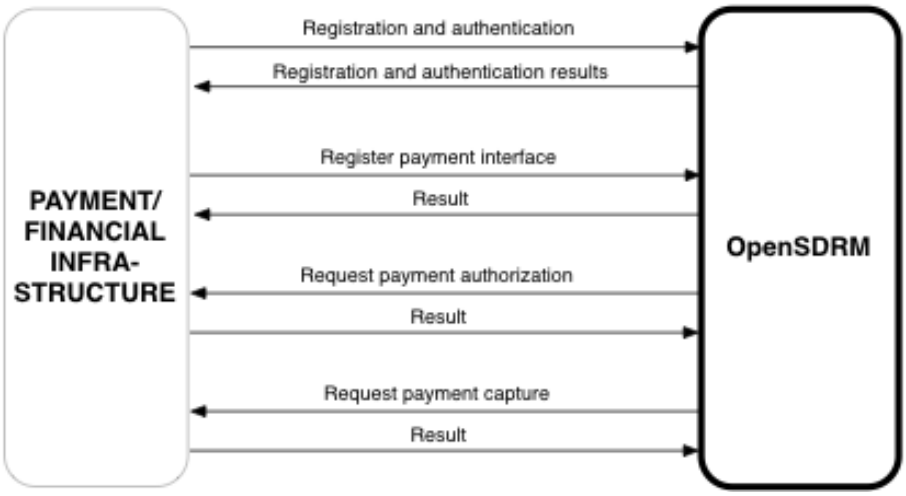

Figure III.6.8: Interface between the Payment/Financial Infrastructure and OpenSDRM

The interface between a specific payment mechanism and OpenSDRM is proprietary of the payment mechanism (Paypal[17], VISA and others), while internally OpenSDRM has a generic payment interface that can be used by OpenSDRM-enabled services. Due to its modular and service oriented design, it is extremely easy to extend this interface to support more payment options. Moreover, and depending on the digital content business model, micro-payment mechanisms may be adopted by the content provider, and therefore OpenSDRM is also capable of handling them.

4.5 Certification Authority

The Certification Authority is responsible for receiving requests for the issuance of digital credentials to entities, individuals or services (Figure III.6.9). These credentials will be used by them to authenticate themselves to each other, allowing the establishment of secure and authenticated communication channels between them. All the components in the Open SDRM architecture communicate using the channel security provided by the SSL/TLS protocol [Rescorla, 2000][Thomas, 2000].

17 http://www.paypal.com

Figure III.6.9: Interface between the Certification Authority and OpenSDRM

This Certification Authority may be external or internal to OpenSDRM, and therefore entirely managed by some entity, or it may be an external commercial Certification Authority such as Verisign[18] or Thawte[19]. Either way this will not affect the performance or security of the architecture. However, from a Public-Key Infrastructure point of view, the usage or global certification services, would allow to broaden trust relationships among a wider number of entities.

5 Internal Services & Interfaces

In the previous sections, the actors and systems that are external to the OpenSDRM architecture were presented. However, most of these actors and/or systems interact with the OpenSDRM internal services through well-defined interfaces.

In this part of the chapter, the OpenSDRM internal services and the corresponding interfaces are presented. These services include:

- The **Content Rendering Application** and the **Wallet Rights Management Interoperability Middle-ware** (see Part III, Chapter 7), as part of the End-User System;
- The **E-Commerce service**, **Content Production service**, **Media Distribution service** and **Registration service**, as part of the Content Management System;
- The **Payment Gateway service**, as part of the Payment System;
- The **Protection Tools service**, as part of the Protection Tools System;

18 http://www.verysign.com
19 http://www.thawte.com

- The **License service**, as part of the License Management System;
- The **Authentication service** and **Configuration service**, as part of the Authentication and Accounting System.

The different services are aggregated in different systems in a logical manner. The main reason for this is that the different systems may operate independently of each other; whilst in some cases some services may have specific workflow dependencies from other services within the same system.

The OpenSDRM architecture offers the necessary services to design an end-to-end digital content business model, offering the full coverage of the digital content value-chain (Figure III.6.1), from the content creator through the end-user, while offering rights management at the end-user site [Serrão et al., 2003b]. These services are integrated in a virtual secure environment while the functionalities needed by a specific digital content business model instantiation (Figure III.6.10).

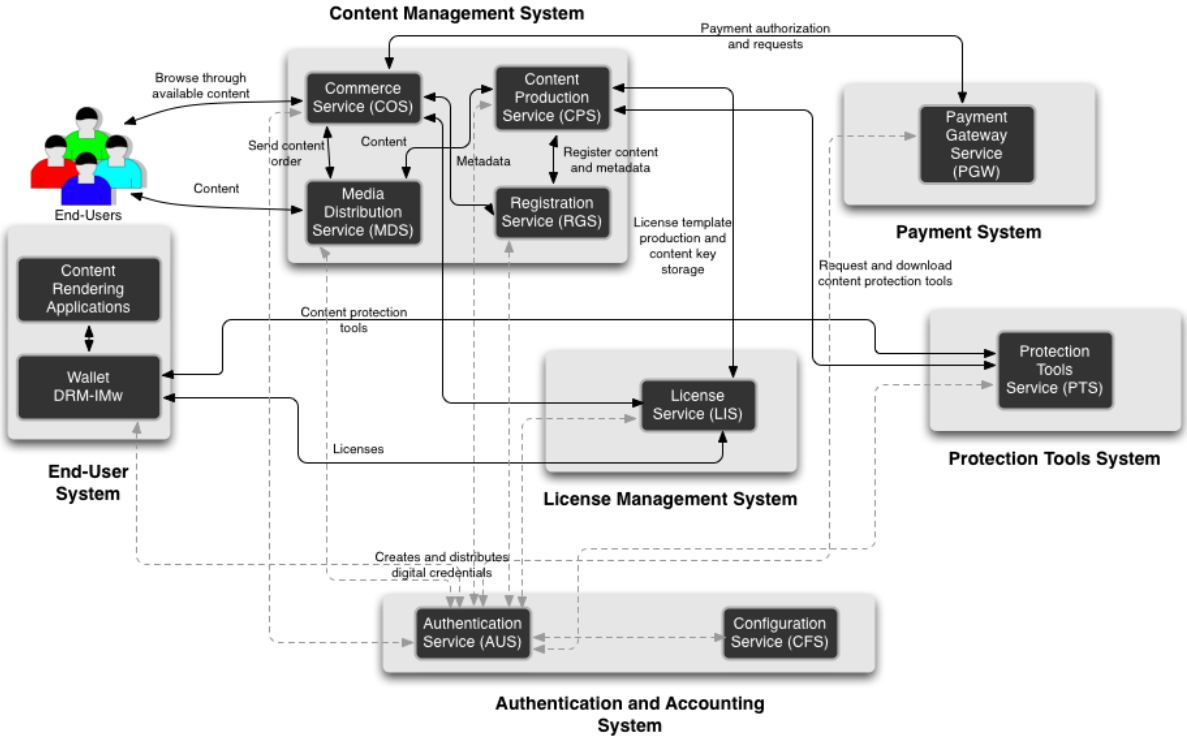

Figure III.6.10: OpenSDRM detailed architecture

The following will provide detailed information about each of the different OpenSDRM services and provide some details about each service interface.

5.1 Content Rendering Application

Rendering content is the ultimate goal of any end-user. In this context, it is considered rendering any action that an end-user will wish to perform over some digital content (either it is governed or not). These actions may include, not only digital content playback or display, but also copy, move, modify, pass, and any other action that will make digital content to pass from one state to other on a given moment. In OpenSDRM, digital content rendering is performed by the Content Rendering Application (CRA) [Serrão et al., 2006b]. More details about this are provided in Part III, Chapter 7.

This component represents the software that will be used to render the content. This can be a generic component with the particularity of being able to display/playback the appropriate digital content for which the necessary audio/video codec is available (if this codec is not available it may be downloaded from a remote secure server). This player may require, for its secure operations, the installation of one or several Protection Tools in order to control how the content is accessed by a particular entity. This is a component that works on the client side of the overall OpenSDRM architecture, however it plays an important role in the rights management functions provided [Serrão et al., 2003b].

Although this component is part of the OpenSDRM architecture, it may not be provided directly by OpenSDRM, but by third parties. This is for instance the case, when a specific player already exists to handle a particular type of content, and the Content Provider wishes to use it. In this case, specific player minor adaptations will have to be performed for it to take advantage of the OpenSDRM functionalities (Figure III.6.11).

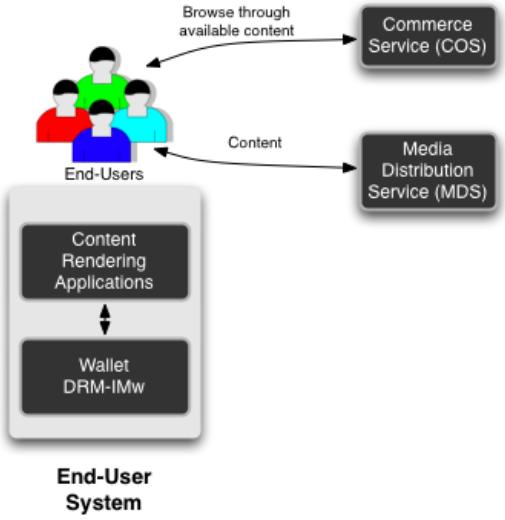

Figure III.6.11: Interface between the CRA and OpenSDRM services

This CRA can interface with the following OpenSDRM services:

1. Media Delivery service: this interface is one of the many ways the CRA may obtain the digital content (it should be able to work in a download, streaming or broadcast manner). This interface is optional, since the user may acquire content through another medium and afterwards just use the CRA to render the content;

2. E-Commerce service: if available, this interface will provide directly inside the CRA the possibility for the end-user to access the available content on an digital content store (something similar to what iTunes does), and to select this content and acquire it. This interface is also optional, since the user may use alternative methods to browse a digital content store (such as a web browser) and acquire content;

3. Wallet Rights Management Interoperability Middle-ware: this mechanism will allow the CRA to obtain the necessary mechanisms to access to DRM-governed content. This interface will provide to the CRA the necessary means to obtain the appropriate protection tools, and the rights to access governed digital content. This interface is mandatory, if the CRA wishes to be able to access and render protected content.

The specific interface between the CRA and the Wallet – Rights Management Interoperability Middle-ware are further described in Part III, Chapter 7.

5.2 Wallet Rights Management Interoperability Middle-ware

The panorama today for CRA is to have all the rights management functionalities integrated directly inside the CRA. Although this provides a more tight integration between the rendering functions and the rights management operations, a more loose-coupled integration could actually benefit the solution interoperability.

It is hard (if not impossible) to have a single and common CRA for all the different types of content, different platforms, different devices and different digital business models, and although some [Lenzi et al., 2003][Microsoft, 2004a][RealNetworks, 2007]are following this path, this is not the case of OpenSDRM.

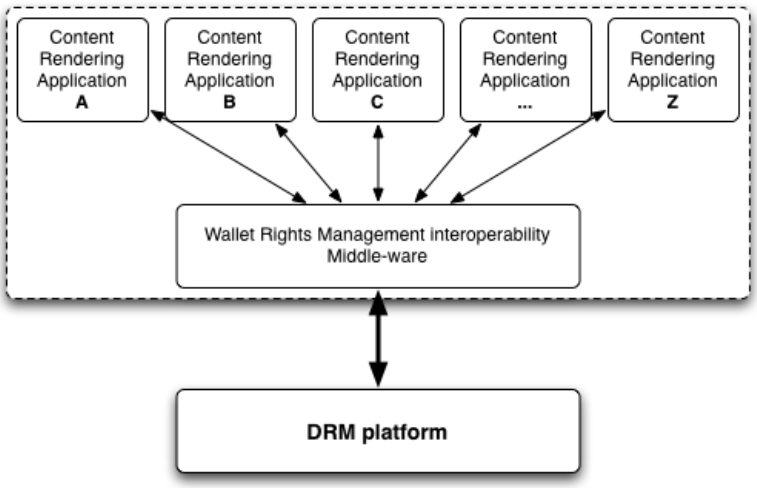

Figure III.6.12: Different CRA interacting with the Wallet Rights Management Interoperability Middle-ware

The design choice for OpenSDRM was to provide a middle-ware mechanism that would allow many different CRA in the same system, to use a single unique rights management layer (Figure III.6.12). This would provide independence to the CRA from the complex rights management tasks, and in fact free the CRA from any DRM-platform specific dependence. However, this has a cost. The CRA must be able to communicate with this middle-ware.

The Wallet Rights Management Interoperability Middle-ware (WRMiM) is the OpenSDRM mechanism, placed at the end-user side system, to allow the different CRA to be able to render DRM-governed content. In fact, the WRMiM will mediate the interactions of the CRA

and the rights management platform (OpenSDRM or others), without having to know too many details about them. The WRMiM is described and specified with more detail on Part III, Chapter 7.

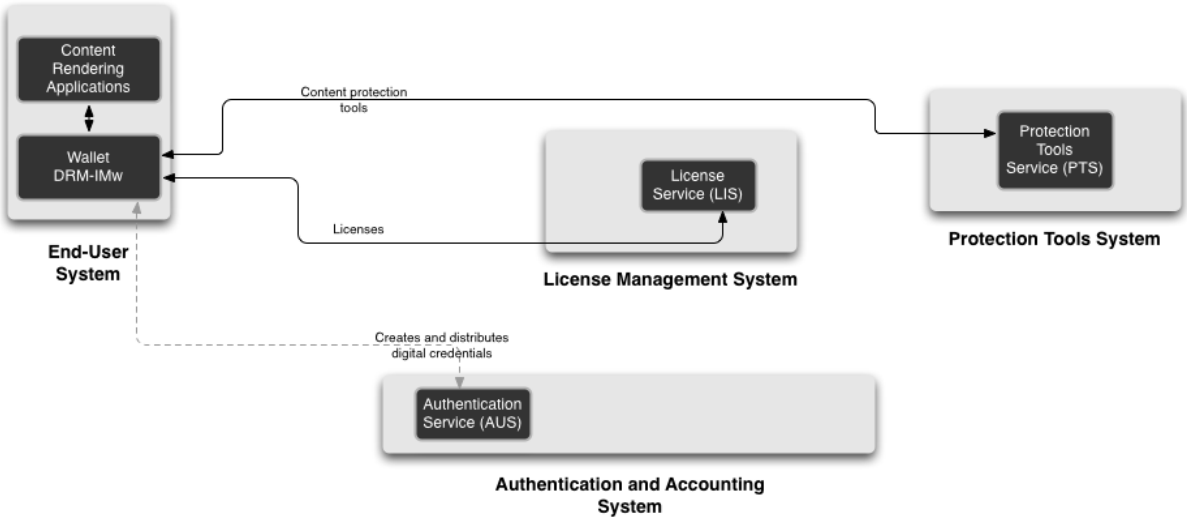

Figure III.6.13: Interface between the WRMiM and OpenSDRM

The WRMiM interfaces with different OpenSDRM systems and services (Figure III.6.13). These interfaces are identified bellow:

(a) Protection Tools service: through this interface the WRMiM is able to obtain new protection tools, from the Protection Tools service. This interface is also used to renew the existing protection tools in case the existing ones get compromised, or if the governed digital content requires so;

(b) License service: through this interface, the WRMiM is able to get the appropriate licenses that contain the rights than entitle the end-user system to render the content. These rights are enforced at the end-user side by mean of the WRMiM component that interactively communicates with the CRA. This interface with the License service may also be used to obtain the appropriate cryptographic keying material to access protected content being handled by the CRA;

(c) Authentication service: through this interface, the WRMiM will obtain the necessary digital credentials to be used while performing any transaction within the remaining services of OpenSDRM platform. It will also be used to certify the end-user and the end-user system, and to manage the different CRA that may form a specific domain.

Moreover, it will also through his interface that payments will be handle within the OpenSDRM architecture.

5.3 E-Commerce service

OpenSDRM was conceived to provide digital content electronic commerce functionalities, depending of the digital content business model that is going to be implemented. Although OpenSDRM itself does not provide any e-commerce store, it offers the services needed to integrated the store front-end and back-end with the rights management platform.

The E-Commerce service (COS) is a service, part of the OpenSDRM architecture, responsible for providing the needed business relationship establishment facilities between the Content Provider and the End-User. In fact, the term “commerce” should be interpreted in a more broader sense, because the COS represents, in abstract, any mechanism put in place by the Content Provider to allow the user to browse and, most probably, acquire digital content. In a traditional digital content commerce operation, the content can be chosen through a browsing mechanism, some very generic metadata may be consulted (and metadata drill-down can also be obtained), information about the digital content price is also available, and conditions for the digital content usage are also presented to the end-user (Figure III.6.14).

All the necessary authentication mechanisms, required by the COS are handled through the end-user's WRMiM and the OpenSDRM Authentication service. It is of most importance that the user and the CRA gets authenticated to this component through the Authentication service, so that the digital content licenses are produced based on this user and CRA authentication and the conditions previously established [Serrão et al., 2003b].

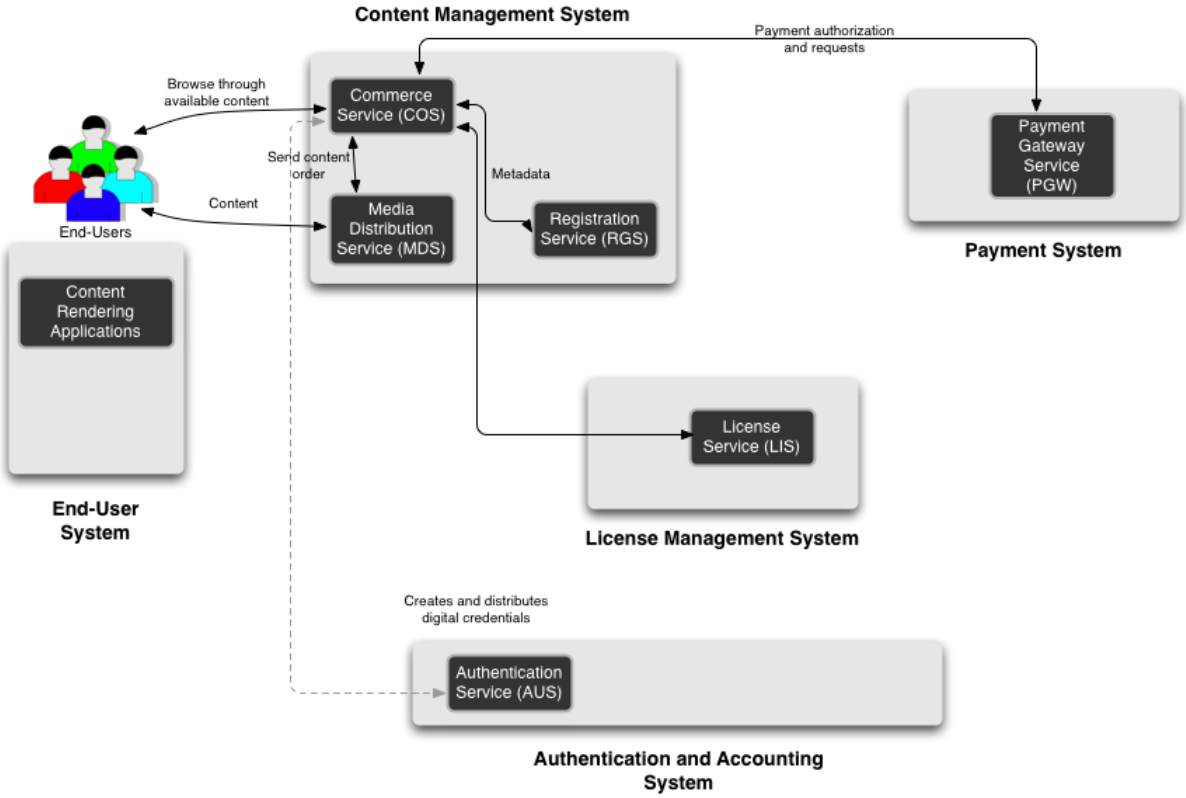

Figure III.6.14: Interfaces between the COS and the OpenSDRM

The E-Commerce service interfaces with different other OpenSDRM services. A list and a description of such interactions follow:

- Content Rendering Application: if the Content Provider provides the e-Commerce site directly, it is possible to use the CRA to navigate through the content, view related metadata, negotiate content conditions, and order it. Any other mechanism provided by the Content Provider may also be used;
- Media Distribution service: this interface is used to notify the Media Distribution that a specific digital content should be sent to a specific end-user, or to a group of end-user systems (also known as domain);
- Registration service: this interface is used to obtain information (identification and meta-information) from the Registration service. This information is used to provide an enhanced browsing experience to the end-user;
- Payment Gateway service: this interface is used for the e-Commerce service to send payment processing claims to the payment gateway, on behalf of an end-user,

through the authorisation provided to the Authentication service. OpenSDRM uses a mechanism that hides both the end-user identification and the payment instrument used by the end-user, from the Content Provider. This is a measure to protect the user privacy. The payment information will be provided by the Authentication service. This mechanism will be further detailed in a following section;

- License service: this interface is used to perform two different tasks. The first is to import the appropriate rights template that is associated with the digital content that is going to be negotiated. The second tasks is related to the production of a specific license according to the digital content, the end-user and the conditions established during the negotiation phase;

- Authentication service: this interface is used to perform several different operations. First, to perform the registration of the different e-Commerce services and to issue digital credentials that will be used for service authentication. Second, to request the end-users authentication when the start the digital content acquisition process. Third, to request a payment authentication envelop, that will be latter on sent to the Payment Gateway service.

5.4 Content Production service

This is a service that may or may not be provided by OpenSDRM. The objective of this service is to receive, prepare, encode, and protect content in such a way that it may be governed by the rights management solution. It may receive raw content from a specified source or sources, encode it on a specified format, add metadata and protect it according to a set of protection parameters. Depending on the type of digital content, there may be different mechanisms for preparing and protect it, but they can all be extended from the generic Content Preparation service (CPS) framework [Serrão et al., 2003b].

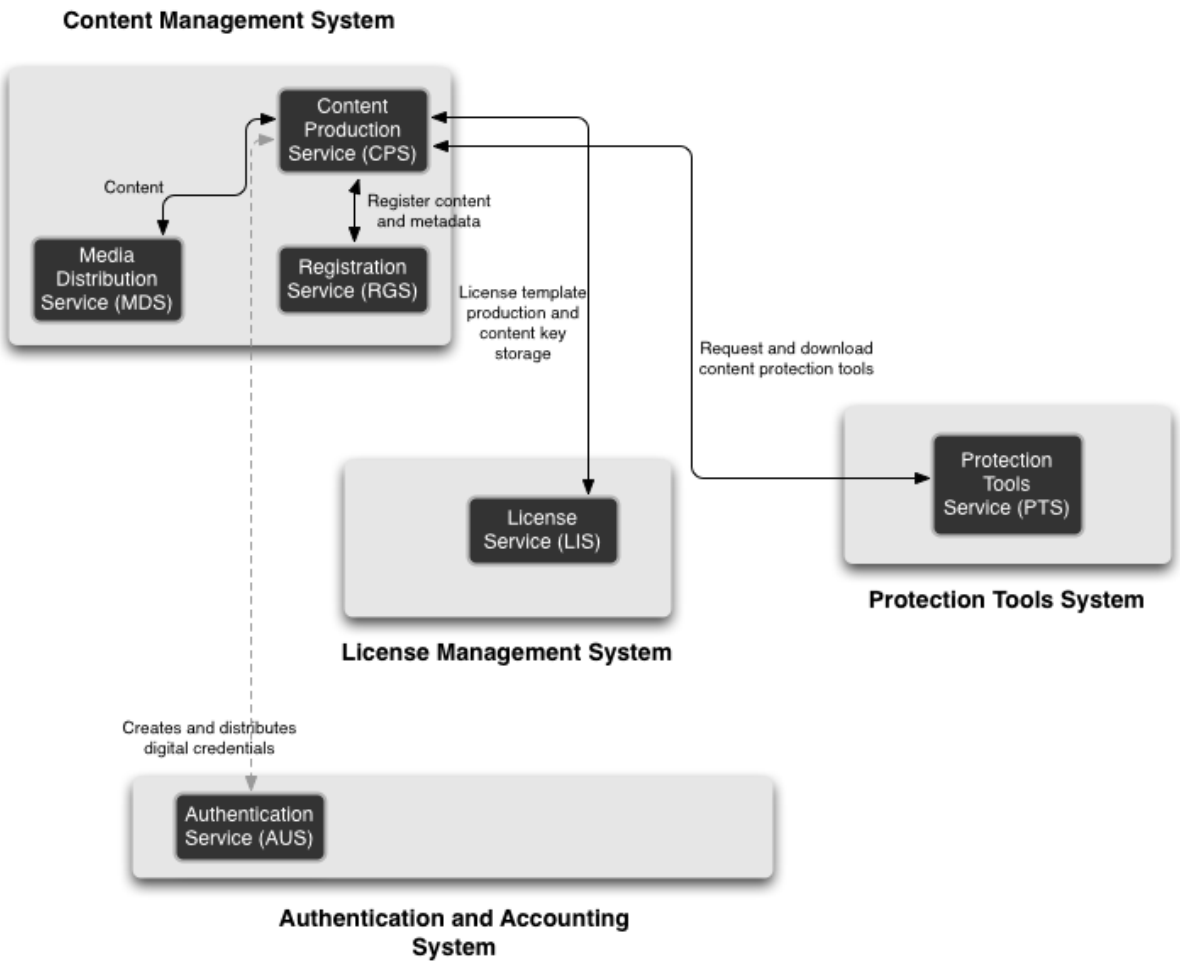

Figure III.6.15: Interfaces between the CPS and the OpenSDRM services

The Content Preparation service provides the necessary tools for a Content Provider to be able to produce governed digital content, and make it available through the remaining services of OpenSDRM (Figure III.6.15).

This Content Preparation service interfaces with different other OpenSDRM services. A list and a description of such interactions is presented bellow:

- Media Distribution service: this interface will be used to store the content (or the content location) to be delivered to Users in the Media Delivery service. The actual digital content server (or the content delivery mechanism) may be outside the OpenSDRM boundaries, however the Media Delivery service, always stores the location of the content (the content is identified to an URI);
- Registration service: this interface is used to perform two basic operations. First, to require from the Registration service, a unique digital content identifier that will

identity the digital content throughout its entire life cycle. Second, this interface is used to feed the Registration service with the digital content associated metadata;

- License service: the Content Production service will use this interface to register the cryptographic keys that were used to protect the content during the production phase. The amount of keys, their length and their type are dependent of the protection tool that has been used to protect the digital content. This interface is also be used to establish a set of pre-conditions (or usage/negotiation rules) that will establish a template licensing mechanism for the future negotiation of this content. The rights template is associated with the unique content identifier and Content Provider identifier, and will be used in the future to establish the negotiation mechanism between the end-user and the e-Commerce service.
- Protection Tools service: when protecting the digital content, the Content Provider will have to use one or more Protection Tools that implement the protection mechanism. These tools may be obtained from a Protection Tools Provider using the Protection Tools service. Therefore, this will be used to list the available tools and the associated information, and to allow the Content Provider obtain the appropriate tool(s);
- Authentication service: this interface will allow the registration and authentication of both the Content Provider and the Content Production service. This authentication will be used to establish virtual private and authenticated channels with the remaining services in the system.

5.5 Media Distribution service

The objective of the Media Distribution service (MDS) is to make the digital content available for the end user. While designing the OpenSDRM platform, it was assumed that the end-user might obtain rights-management digital content from many sources, such as:

- Directly at a digital content e-Commerce store;
- From a friend, or any other user;

- From a P2P network;
- Or, from any other source.

Therefore, the MDS is an optional component in the overall architecture. It may or not be used, and it depends on the business model put in place. However, the role of this service is not to push or pull the content to the end-user, but rather act as a centralised repository of information about the digital content location. This choice was made, taking into account the large number of different digital content delivery mechanisms (and the ones that are still to come) and the independence of the actual delivery mechanisms to have a more open and interoperable scenario [Serrão et al., 2003b].

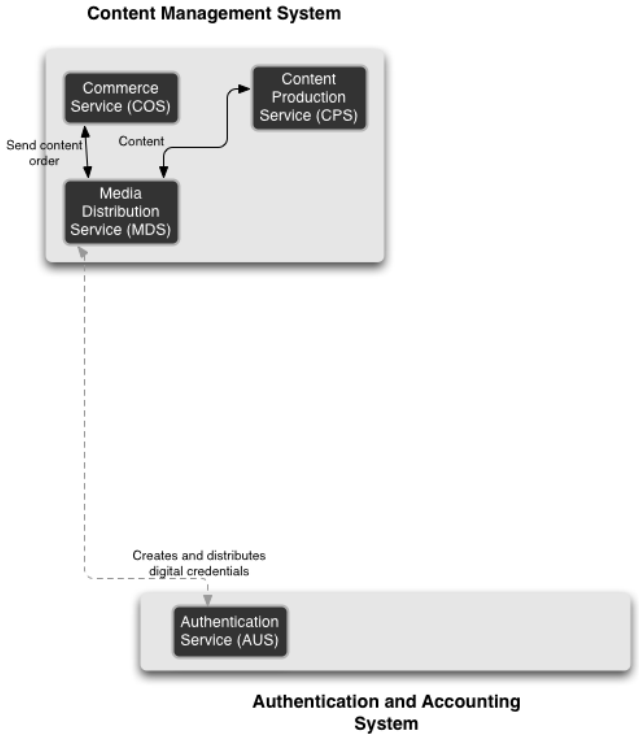

Figure III.6.16: Interfaces between the MDS and the OpenSDRM services

This service interacts with the following OpenSDRM services (Figure III.6.16) using the following interfaces:

- E-Commerce service: through this interface the COS will be able to notify the MDS that a specific digital content will be sent to the end-user, or that the end-user will use some specific mean to obtain it. The MDS should afterwards make available the digital content location;

- Content Production service: through this interface the CPS will be able to notify the MDS that some DRM-governed digital content will be made available in a specific location;
- Authentication service: this interface will allow the registration and authentication of both the Media Distribution service. This authentication will be used to establish virtual private and authenticated channels with the remaining services in the system.

5.6 Registration service

The OpenSDRM architecture assumes that every DRM-governed digital content uses some kind of unambiguously identification mechanism. The role of the Registration service (RGS) is to provide this identification.

The RGS assigns unique identifiers to content and registers metadata information associated for that specific content. OpenSDRM follows the MPEG-21 directives about Digital Item Identification (DII) [Bormans and Hill, 2002], using a reduced version of the MPEG-21 DII Digital Object Identifiers (DOI) [Dalziel, 2004].

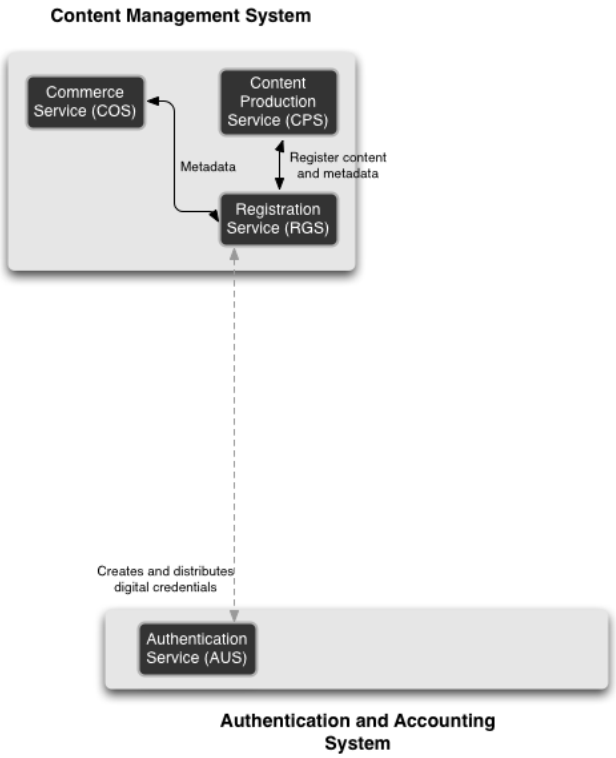

Figure III.6.17: RGS interfaces with the remaining OpenSDRM services

RGS interfaces with following OpenSDRM services (Figure III.6.17):

- Content Production service: this interface is used by the CPS to register new DRM-governed digital content obtaining a unique identifier and to register metadata associated with the content;
- e-Commerce service: through this, the COS can obtain information about the digital content available, and their associated metadata. It will be possible for the COS to pass through the interface complex queries to present content and associated information, as requested by the end-user;
- Authentication service: through this interface, the RGS will be able to register the service and obtain the necessary credentials that can be used to establish secure and authenticated channels between the different OpenSDRM services.

5.7 Payment Gateway service

The OpenSDRM architecture offers payment support to the digital content trading operations. This is provided by the Payment Gateway service (PGW), implementing an abstraction layer between the OpenSDRM environment and the real payment infrastructure that will be used at the other end.

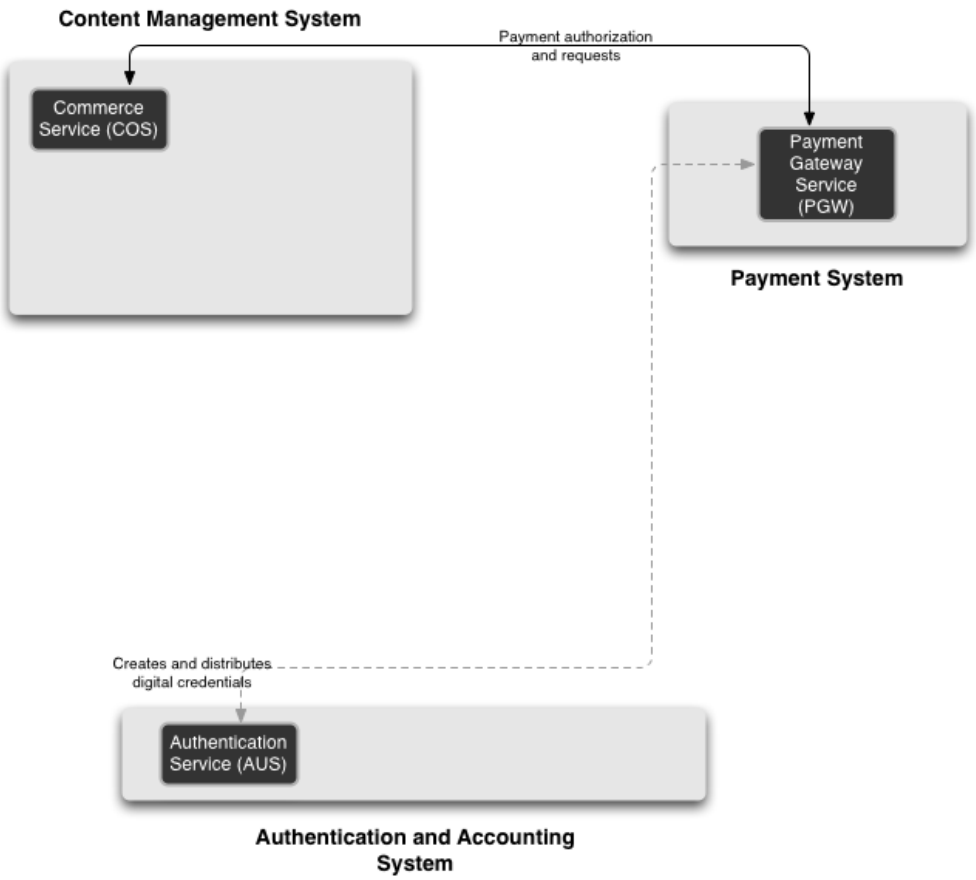

Figure III.6.18: Interfaces between the PGW and the different OpenSDRM services

It was implemented in the payment operations an anonymising mechanism that prevents E-Commerce service and the Content Provider from knowing the details of the end-user and the real payment mechanism and that prevents the PGW and the upper financial infrastructure from knowing what was acquired by the end-user [Serrão et al., 2003b].

It was also implemented a mechanism that facilitates the payment by the end-user, preventing the user from filling payment details every time a financial transaction is performed. This not only improves the end-user experience, but also at the same time reduces the fraud risks.

The PGW interfaces directly with the following OpenSDRM services (Figure III.6.18):

- E-Commerce service: this service interfaces with the PGW to perform the following operations. First, to check with the PGW if the payment mechanism presented by the end-user is valid or not. Second, at the end of the transaction, to capture the payment;

- Authentication service: this interface is used by PGW to register in the Authentication service and obtain the necessary credentials to establish trust relationships with the other OpenSDRM services.

5.8 Authentication service

This service is an important service provider for the overall security of the OpenSDRM system. All services, entities and devices, which interact with such services, need to be properly registered and authenticated at the Authentication service (AUS).

Whenever a new service is added the stack pile of OpenSDRM services, it must require to be registered on a valid AUS, before performing any operation with any other OpenSDRM service. All the transactions within OpenSDRM services are performed over secure and authenticated channels and require a valid credential to be effective. This credential is mandatory and can only be obtained during AUS registration phase [Serrão et al., 2003b].

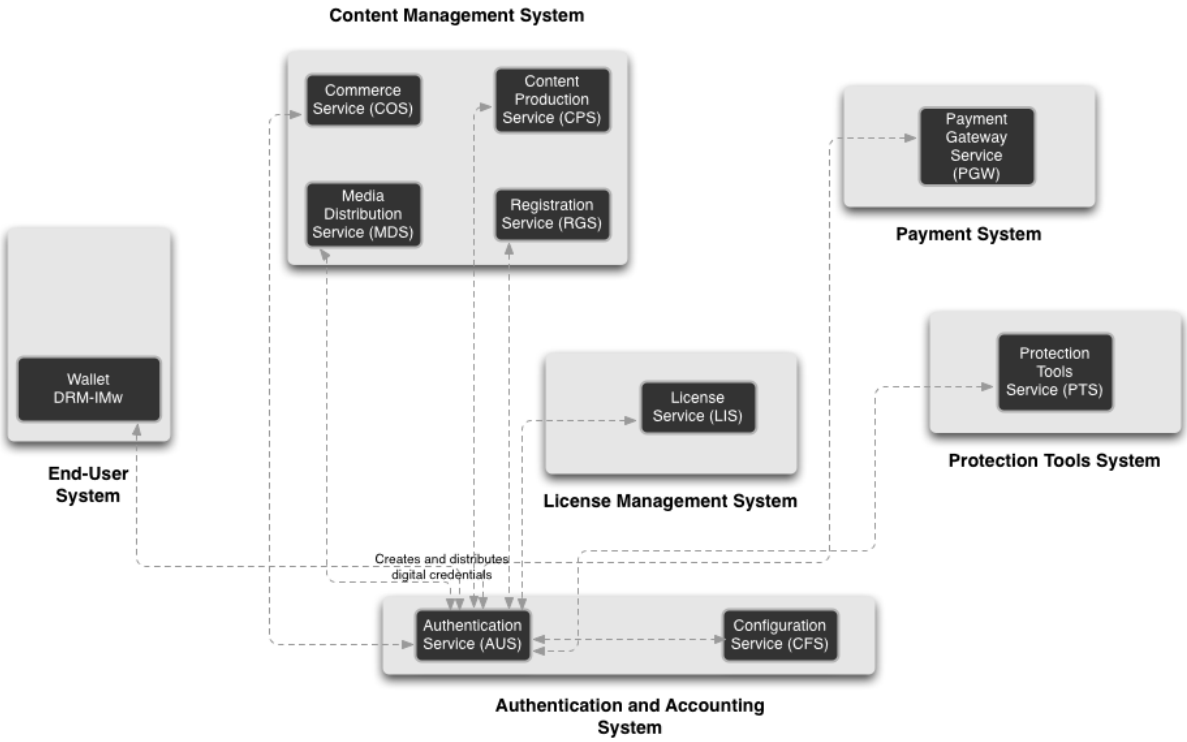

Figure III.6.19: The AUS interfaces with the remaining OpenSDRM services

AUS acts as a single-sign-on point for the entire system, registering and managing components and users on the system. It uses cryptographic XML credentials to authenticate both components and users in order to authenticate the transactions exchanged between

them. Further details about these mechanisms will be provided in other sections of this chapter.

This service interfaces with all the other OpenSDRM services (Figure III.6.19) and it is used to perform the registration and authentication of the different services and entities that use such services (see Annex A).

OpenSDRM has also the ability to handle domain management. AUS acts has a domain management service on the OpenSDRM platform. AUS is able to handle the creation of new domains, the management of this domains (the entrance and leaving of users and devices) and the deletion and extinction of domains.

The user can create new domains and add new devices (devices with the WRMiM installed). Also, a user device can be part of more than one domain. Only the user that has create the domain is able to remove it. It is possible also to issue a license for one or multiple domains, instead of issuing licenses for specific devices (Figure III.6.20) [Serrão et al., 2003b].

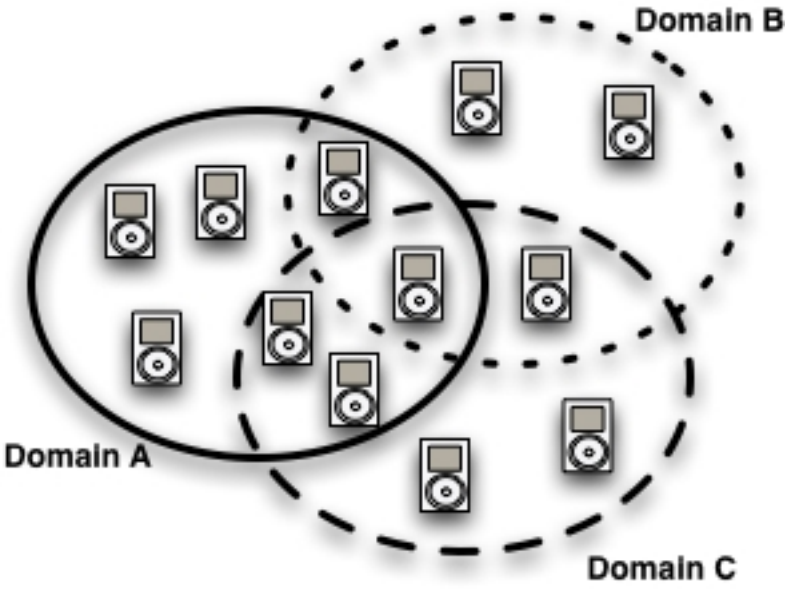

Figure III.6.20: Devices that belong to single or multiple domains

The authentication server (AUS) is used to manage domains. It will possible for the user to create domains, to add or remove devices from one or multiple domains, and also remove domains (only the domain creator can do such).

5.9 License service

OpenSDRM is above all an integrated and open architecture for rights management, and this is one of its major design goals. Part of this goal fulfilment is provided by the License service (LIS). The LIS is responsible managing the rights associated to digital content, for the establishment of rights management templates and for creating licenses that associate a user, the content and his/her corresponding access rights.

This service will accept connections from authenticated client WRMiM for downloading of licenses, which will be applied to the DRM-governed and protected content through an appropriate protection tool. OpenSDRM uses a template mechanism that provides license format independence to the platform. Therefore, the two major XML-formatted rights expression languages, Open Digital Rights Language (ODRL) [ODRL, 2002] and MPEG-21 Rights Expression Language (REL) [MPEG-21-REL, 2003], are supported.

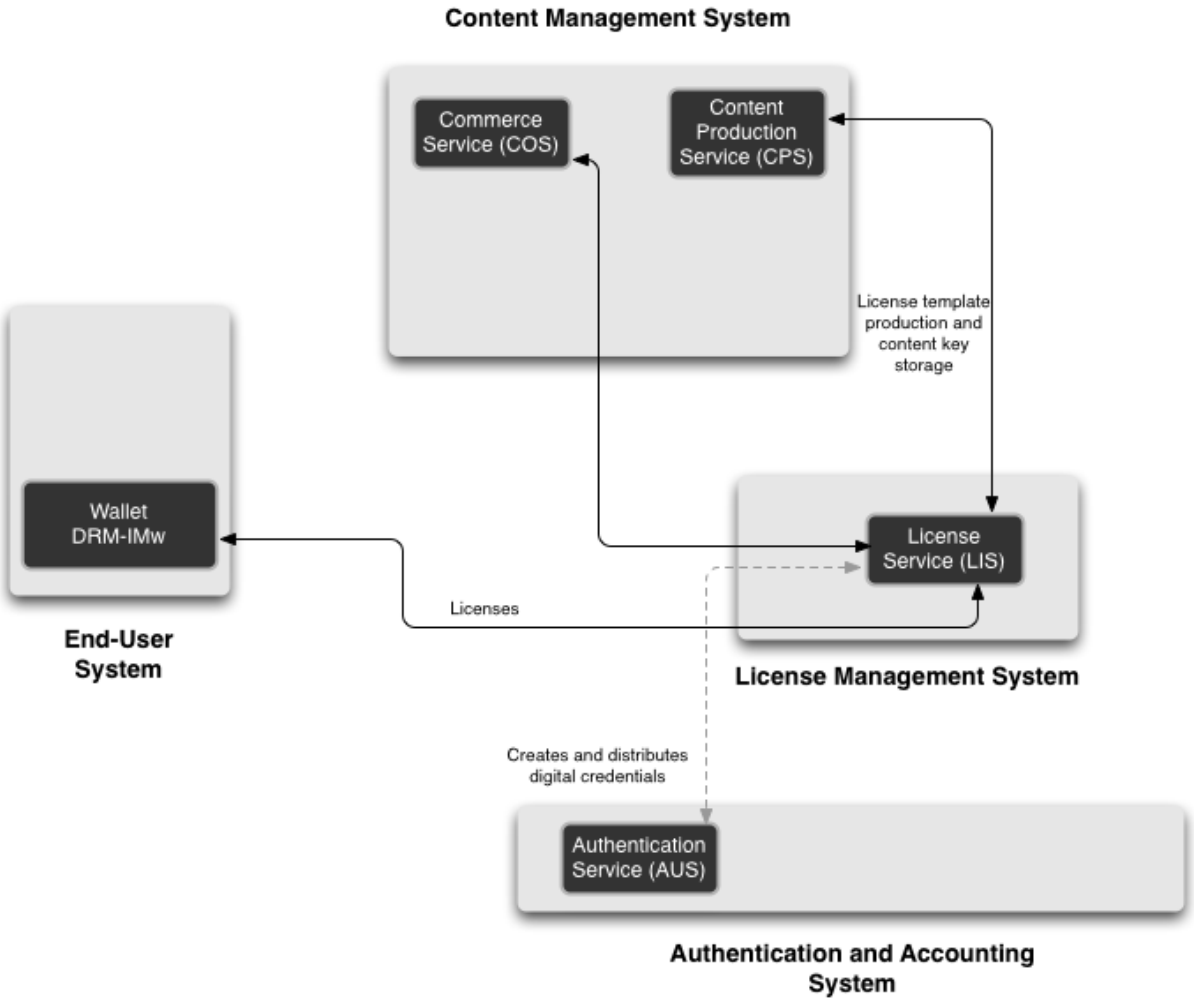

Figure III.6.21: Interaction between LIS and the other OpenSDRM services

The LIS interfaces primarily with the following OpenSDRM services (Figure III.6.21):

- Content Production service: through this interface it will be possible for the Content Provider via the CPS, to establish the appropriate conditions that will be available for a specific digital content. In such a way a rights template is created, representing the micro-business model for that specific piece of content. This can be done for each of the different digital content provided by a Content Provider, or to all. If considering the iTunes case, the conditions and the even the price are always the same for the same set of digital content – in this case a single template is enough. This interface is also used for the LIS to store the keying material that has been used to protect the digital content;
- E-Commerce service: this interface will be used for the COS to notify the LIS that a new license must be created. This license will be created for a specific end-user, a

specific end-user system, or domain, and will contain the identification of the digital content and an instantiation of a specific rights template;

- Wallet Rights Management Interoperability Middle-ware: through this the WRMiM will be able to obtain licenses from the license server and to perform several operations with that licenses (such as pass those licenses to other devices or to other users);
- Authentication service: this is used for providing the credentials necessary to LIS authenticate to other OpenSDRM services.

5.10 Protection Tools service

The Protection Tools service (PTS) is responsible for the registration of new protection tools and for receiving authenticated client WRMiM requests for the downloading specific protection tools. It is also responsible for making protection tools available to the CPS to allow the protection of content.

This mechanism allows the independence of both the CRA and the CPS from the digital content type that is used and from the content protection mechanism used. This is an important aspect in the OpenSDRM architecture scalability, allowing adding and changing some functionality with low impact in the overall architecture. Moreover, from a security point of view, this will allow the renewability of the protection tools, whenever some of them are compromised. At the end-user system, these tools are downloaded by the WRMiM and installed on the CRA allowing the process to be almost transparent for the end-user [Serrão et al., 2003b].

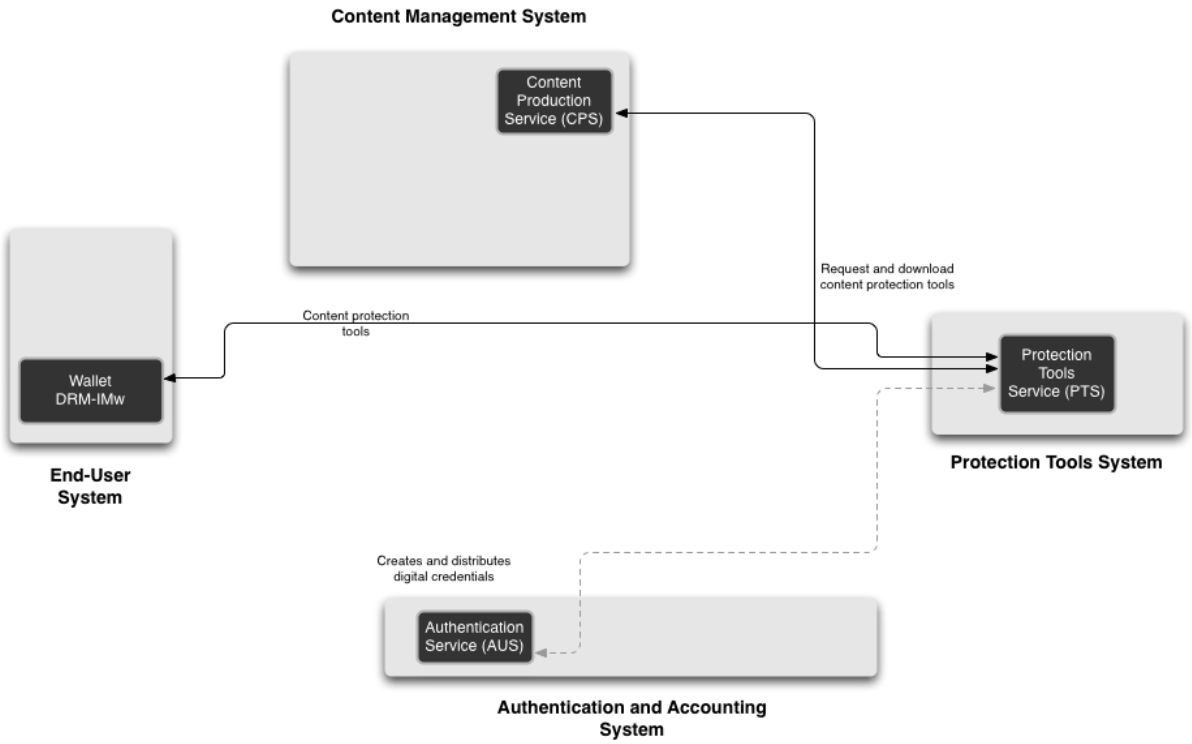

Figure III.6.22: Interfaces between the PTS and the other OpenSDRM services

The Protection Tools service interacts with the following OpenSDRM services (Figure III.6.22):

- Content Production service: through this interface the CPS can perform two different operations. First, the CPS can interrogate the PTS to learn which are the protection tools available that can be used to implement its own digital content business model. Second, it will request a specific protection tool from the PTS;
- Wallet Rights Management Interoperability Middle-ware: this interface is used by the WRMiM to request specific protection tools which are not yet available on the end-user system;
- Authentication service: used to provide the necessary security and authentication credentials to allow authentication and secure communication with the other OpenSDRM services, and to allow the signature and certification of the protection tools.

5.11 Configuration service

In a highly flexible and extensible environment, such as the one presented in OpenSDRM,

there should exist mechanisms to allow an easy and simple configuration of the different OpenSDRM services. This configuration and customisation functionalities are provided by the Configuration service (CFS).

6 OpenSDRM specific security mechanisms

As in any other rights management solution, also in OpenSDRM, the security aspect is quite crucial. Designing security systems is always a major challenge, and it is even more challenging when protecting digital assets protection and personal and private information is at stake.

OpenSDRM is a distributed architecture. The communication between the single components will usually take place within insecure networks. Furthermore the components communicate with a text-based protocol. This introduces special needs regarding the security of this communication [Serrão et al., 2003b].

An underlying concept behind the OpenSDRM platform is the existence of two secure communication layers (Figure III.6.23). A first security layer is established at the communication level, which will provide the necessary secure and authenticated communication channel to components to communicate with each other. A second layer is established at the application level, ensuring the security, integrity, authentication and non-repudiation mechanisms needed by the different components.

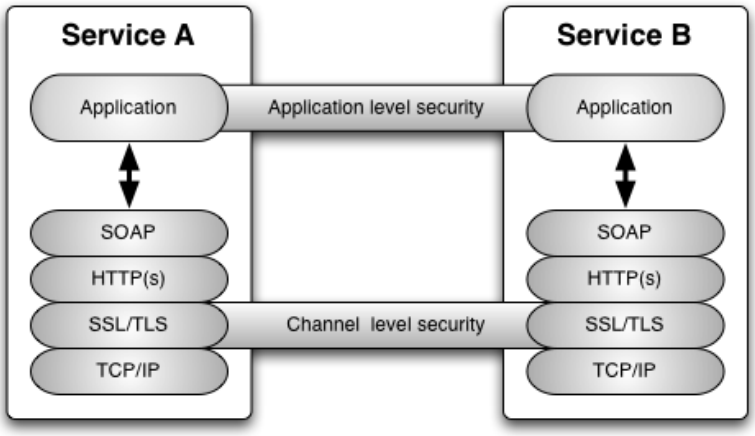

Figure III.6.23: Two security levels in OpenSDRM

OpenSDRM uses public and secret-key cryptography to provide the necessary security to the

architecture and to implement the above security layers. In the following sections, a description of the different security mechanisms used on the OpenSDRM architecture are further detailed.

The following section will start by detailing the process for establishing the initialisation of the OpenSDRM security.

6.1 OpenSDRM security bootstrapping/initialisation

In order to establish the secure channel layer (Figure III.6.23), the software server components of the OpenSDRM architecture, use the SSL/TLS protocol [Rescorla, 2000][Thomas, 2000] to ensure such functionality. Each of the servers, on which the OpenSDRM services are installed, need to have a valid X.509 certificate issued by a Certification Authority (CA). If more than one of the OpenSDRM services is installed on the same server then they will share such certificate to create secure and authenticated channels. The CA can be operated either internally by OpenSDRM itself or can be an external and commercial one.

In this way, OpenSDRM can establish an underlying secure and authenticated transport channel that will allow the messages to flow from component to component securely. The process that initialises these software server components is the following (Figure III.6.24):

- Each component computes a key pair (public and private), K_{pub}^{Server}, K_{priv}^{Server}, using a public-key algorithm (such as RSA) and create Certificate Signing Request (CSR) using its public key and some additional information sending it after to the CA;
- The CA verifies the CSR validity and issues the X.509 SSL certificate to the appropriate component, $CertX.509_{Server}$;
- The X.509 SSL certificate is installed and the components can use SSL/TLS to communicate, establishing therefore the secure transport layer.

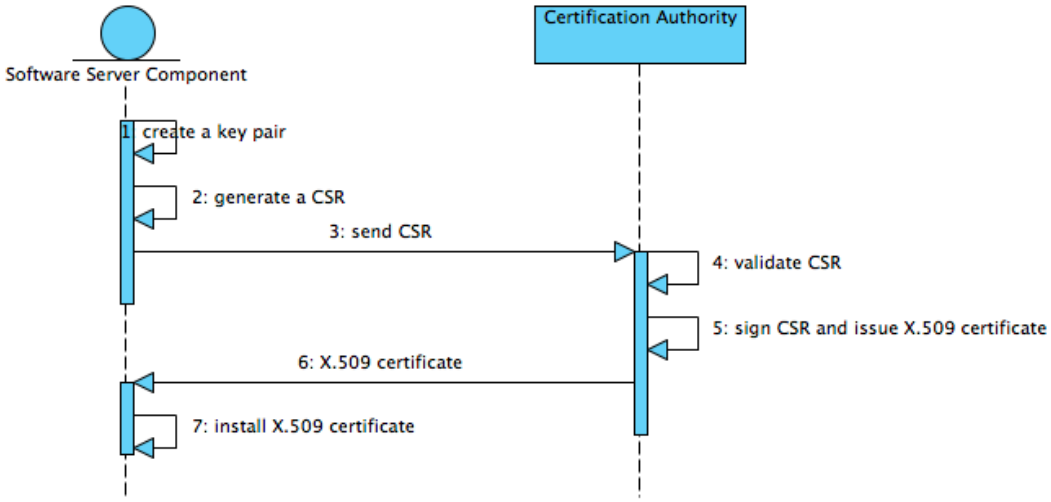

Figure III.6.24: Sequence to register a software server component

6.2 Services and systems registration on OpenSDRM

The architecture requires that both services and Users on the OpenSDRM architecture to be registered, in order to establish the Application/Transaction level security.

Concerning the OpenSDRM services (COS, PGW, RGS, LIS, PTS, CFS, MDS, CPS) those are registered on OpenSDRM AUS. In order to complete this process the following steps are necessary, during the installation of each of the components:

Each component computes a key-pair (currently OpenSDRM at least 1024 bit length RSA keys, to be as robust as possible): $K_{pub}^{Service}$, $K_{priv}^{Service}$ (respectively the public and private keys);

The service administrator selects a login and a password, and ciphers the service private key ($K_{priv}^{Service}$), using AES, with the key (K_{AES}) computed from the hash of the concatenation of the login and password selected: $K_{AES} := MD5(login+password)$. The ciphered service private key gets then protected from unauthorised usage: $K_{AES}[K_{priv}^{Service}]$.

The service then connects to the AUS and sends some registration information together with the $K_{pub}^{Service}$.

AUS verifies the information sent by the service, validates and registers it, and issues a certificate for the service: $Cert^{AUS}_{Service}$. This certificate contains, among other information,

a unique identifier of the service and its public key (Figure III.6.25). This certificate is return to the service.

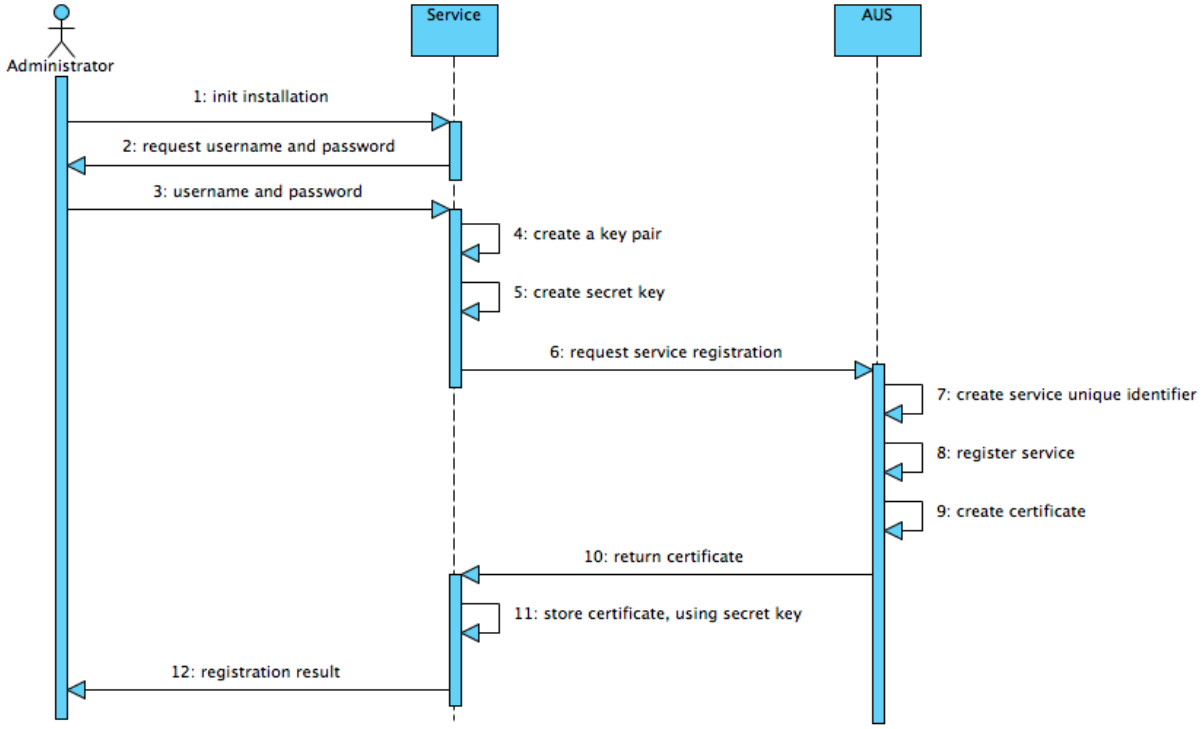

Figure III.6.25: Services registration in AUS

With these service certificates, each of the services will be able to establish trust relationships among them and sign and authenticate all the transactions – this establishes the Application Level security.

6.3 Users registration on OpenSDRM

In the OpenSDRM architecture three services interact directly with external users/entities – CRA, CPS and PTS. These users, respectively End-Users, Content Providers and Protection Tools Providers are registered on the platform, through the AUS.

Content Providers and Protection Tools Providers subscribe respectively on the CPS and PTS, relying on the registration and authentication functionalities of the AUS (Figure III.6.26). Therefore, when a new user subscribes, it provides some personal information, a login and password and requests the registration. The following processes can be described like this:

- The services (PTS and CPS) gather the new registrant information (Info) and request

the registration of a new user on the AUS;

- The services build a new message: $K_{priv}^{Service}\{Cert^{AUS}_{Service}, Info\}$. This message is send to AUS;
- AUS verifies and validates the message, registering the new User and returning a digital certificate that identifies the user: $Cert^{AUS}_{Service}(USER)$.

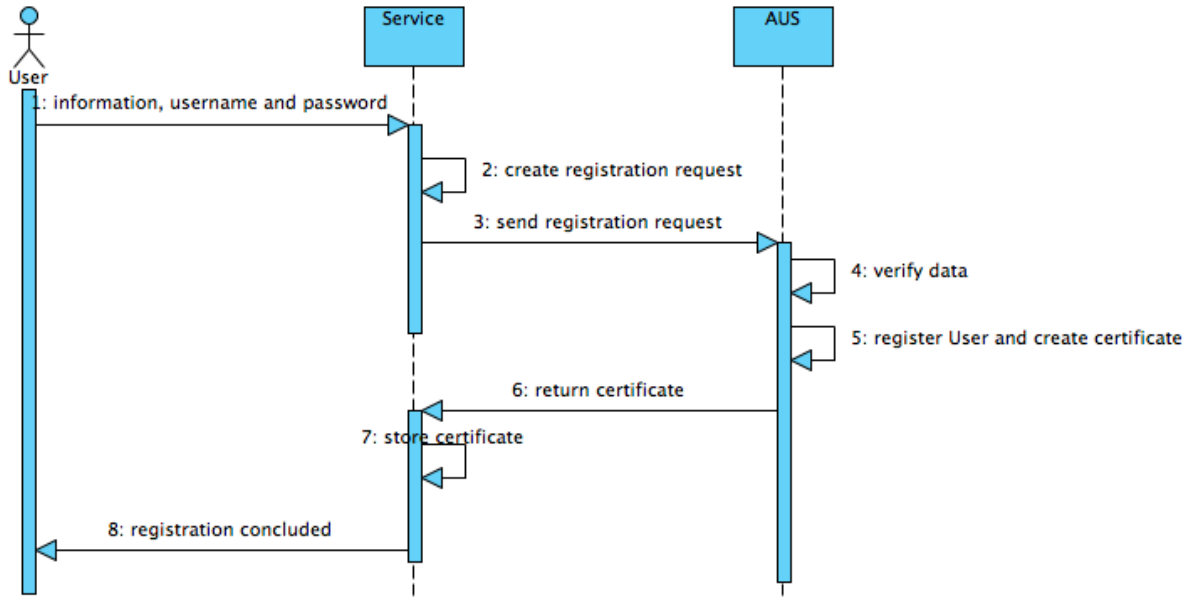

Figure III.6.26: User registration on the services, through AUS

Registering End-Users in the OpenSDRM architecture is a more complex process. This is due to the fact that while both Content Providers and Protection Tool Providers have their information stored on remote servers, End-Users rely on their own platforms to store their data (although OpenSDRM may support different models). In order to provide some additional security level, OpenSDRM architecture uses the WRMiM, capable of storing sensitive information such as cryptographic data and licenses in a secure way. The process to register new End-Users can be described in the following steps:

- When the End-User runs the WRMiM for the first time, it creates the User RSA key pair (K_{priv}^{User}, K_{pub}^{User}) and asks the user to enter a login and a password;
- Moreover, the WRMiM creates also the Device key-pair (K_{priv}^{Device}, K_{pub}^{Device});
- Using the entered login and password, it creates the secure repository master key: K_{AES}=MD5(login+password), and stores sensitive information (Info) on it:

$K_{AES}[Info]$;

- The wallet asks the user to enter some personal data (PersonData) and also some payment data (PayData) used to charge the user for any commercial content usage;
- The wallet requests the AUS to register a new User, sending all the information ciphered with the AUS K_{pub}^{AUS}: $K_{pub}^{AUS}[PersonData, PayData, K_{pub}^{User}, K_{pub}^{Device}]$;
- AUS receives the data, deciphers it and registers the User. AUS responds to the Wallet with two new certificates generated for the User and the Device: $Cert^{AUS}_{User}$, containing among other information the unique identifier of the User, its public key, the identification of the AUS and its signature, and $Cert^{AUS}_{Device}$, containing the $Cert^{AUS}_{User}$, the Device public-key and the AUS signature;
- The wallet stores all the relevant information on the secure repository: $K_{AES}[Cert^{AUS}_{User}, Cert^{AUS}_{Device}]$.

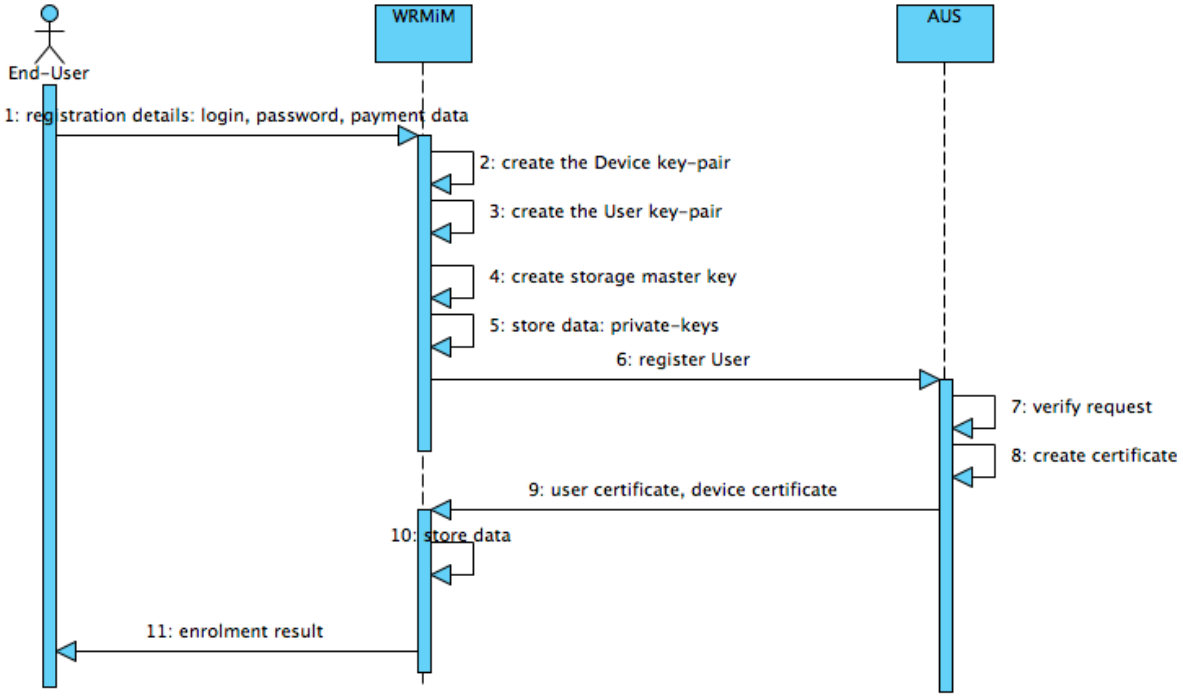

Figure III.6.27: End-User registration through the WRMiM

After the conclusion of this process, both the End-User and the End-User system (Device) are properly registered in the OpenSDRM system (Figure III.6.27).

6.4 Domain Management

An important OpenSDRM functionality is domain management. In rights management context, a domain may be described as a group of devices or users that share digital content access and licenses.

This means that the User can create and remove domains, and add devices to one or multiple domains. If a domain license exists, it can be shared by all the devices and users that are part of that domain.

The first important process is domain creation (Figure III.6.28):

- The domain management operations are handled by the WRMiM. When a domain is created, automatically the device (represented by WRMiM) is added to the new domain;
- The user can select a name for the new domain. Then WRMiM loads the AUS User certificate ($Cert^{AUS}_{User}$) and the AUS device certificate ($Cert^{AUS}_{Device}$);
- WRMiM sends a signed message to AUS requesting the creation of the domain: $K_{priv}^{User}\{DomainName, Cert^{AUS}_{User}, Cert^{AUS}_{Device}\}$;
- AUS receives the information and validates it;
- AUS sends a credential that certifies the creation of the new domain ($Cert^{AUS}_{Domain}$). This certificate will be send to all the devices that join the same domain;
- WRMiM receives this information and stores it.

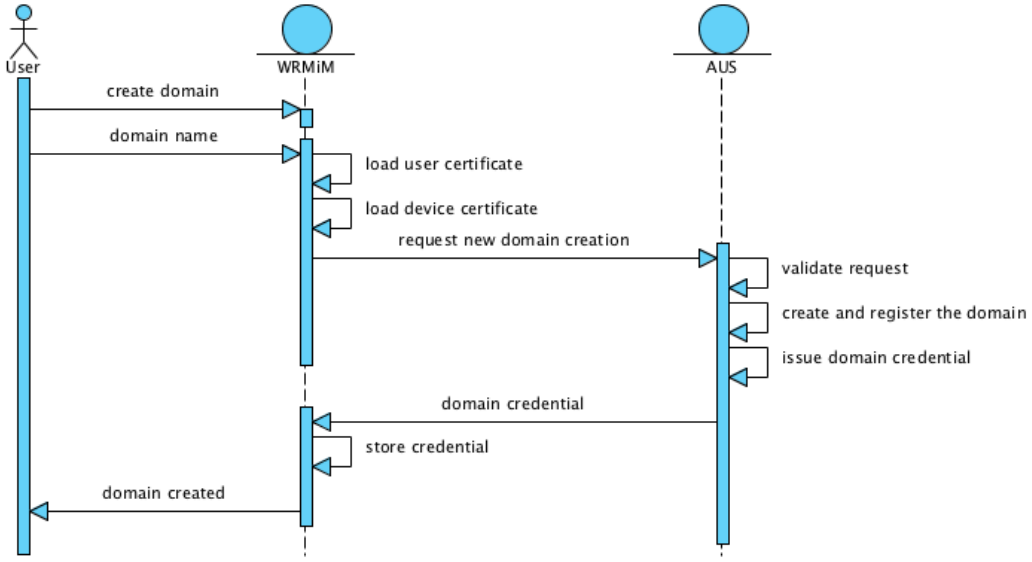

Figure III.6.28: Domain creation on OpenSDRM

Another important operation in OpenSDRM domain management is adding a new device to an existing domain. This can happen in two different ways. Either the domain was created by the User adding the device to the domain, or the User is trying to add a device to a domain not owned by the User.

The first is a simple scenario. The operation starts by request of the End-User to add a new device to a domain (Figure III.6.29):

- The end User requests the device to be added to an existing domain;
- WRMiM sends a message to the AUS requesting a list of the User domains;
- To do this, WRMiM gets the User AUS certificate ($Cert^{AUS}_{User}$), and sends a signed message to the AUS requesting the domain listing: $K_{priv}^{User}\{listUserDomains, Cert^{AUS}_{User}\}$;
- AUS receives the request, validates it and checks for the User created domains;
- AUS returns the list of available domains to the WRMiM;
- WRMiM displays the list of available domains to the User;
- The User selects the domain. Then WRMiM loads the AUS User certificate ($Cert^{AUS}_{User}$) and the AUS device certificate ($Cert^{AUS}_{Device}$);

- WRMiM sends a signed message to AUS requesting the device to join the domain: $K_{priv}^{User}\{DomainName, Cert^{AUS}_{User}, Cert^{AUS}_{Device}\}$;
- AUS receives the information and validates it;
- AUS sends a credential that certifies that the device has joined the domain ($Cert^{AUS}_{Domain}$);
- WRMiM receives this information and stores it.

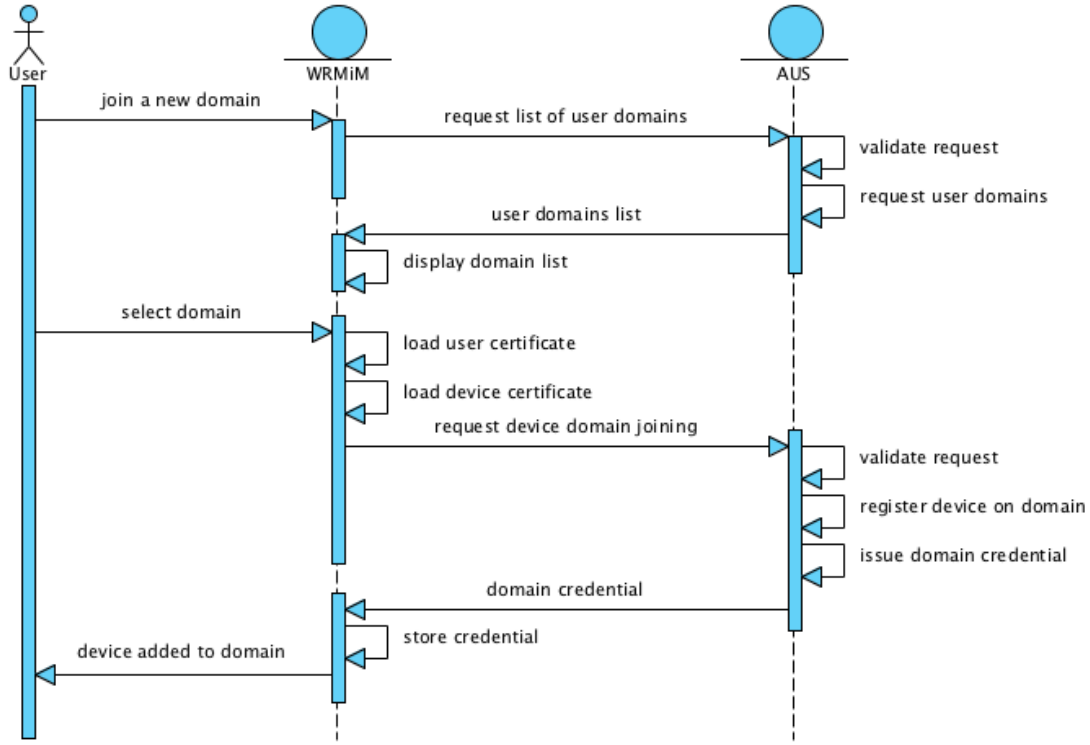

Figure III.6.29: Adding a new device to a User created domain

The second scenario is a little more complex. In this scenario a User is invited to join a domain that was created by other User. The process is the following (Figure III.6.30):

- User A, using its WRMiM-A, sends an invitation for User B to join a specific domain (created by User A);
- WRMiM-A, sends an invitation message, through the AUS, to a User-B;
- WRMiM-A loads the User-A ($Cert^{AUS}_{User\text{-}A}$) and Device-A ($Cert^{AUS}_{Device\text{-}A}$) credentials and builds an invitation request, sending it to the AUS. This invitation

contains also the invited User-B identification, either his email address or the User-B credential ($Cert^{AUS}_{User-B}$), obtained in some way. Finally, the request contains also the domain ($Cert^{AUS}_{Domain}$) to which the invited user will be added: $K_{priv}^{User-A}\{invitateUser, Cert^{AUS}_{User-A}, Cert^{AUS}_{Device-A}, Cert^{AUS}_{User-B}, Cert^{AUS}_{Domain}\}$;

- AUS receives the request and validates it;
- If the request is valid and the destination User-B is known and authenticated, the invitation is stored on the AUS;
- When User-B starts its device, if a connection is available, the WRMiM-B connects to the AUS and checks for domain invitations: $K_{priv}^{User-B}\{checkDomainInvitations, Cert^{AUS}_{User-B}\}$;
- AUS checks the request and validates it. Afterwards, AUS checks if domains invitations for that particular user exist or not. If invitations exist on AUS, they are sent back to WRMiM-B;
- WRMiM-B lists the invitations to the User- B, and the User-B may accept, reject or ignore them. If the invitation is rejected it is deleted from AUS. If the User ignores the invitation, the request remains on AUS. This is the case of the User selecting another device to join the domain and not the one that he is currently using. If the User accepts the request, the process continues and the request is deleted from AUS at the end of the process;
- User-B accepts the invitation;
- WRMiM-B loads the AUS User-B certificate ($Cert^{AUS}_{User-B}$) and the AUS Device-B certificate ($Cert^{AUS}_{Device-B}$);
- WRMiM-B sends a signed message to AUS requesting the Device-B to join the domain: $K_{priv}^{User-B}\{joinInvitedDomain, DomainName, Cert^{AUS}_{User-B}, Cert^{AUS}_{Device-B}\}$;
- AUS receives the information and validates it;

- AUS validates also that User B has received an invitation to join the domain;
- AUS sends a credential that certifies that the device B has joined the domain ($Cert^{AUS}_{Domain}$);
- WRMiM-B receives this information and stores it.

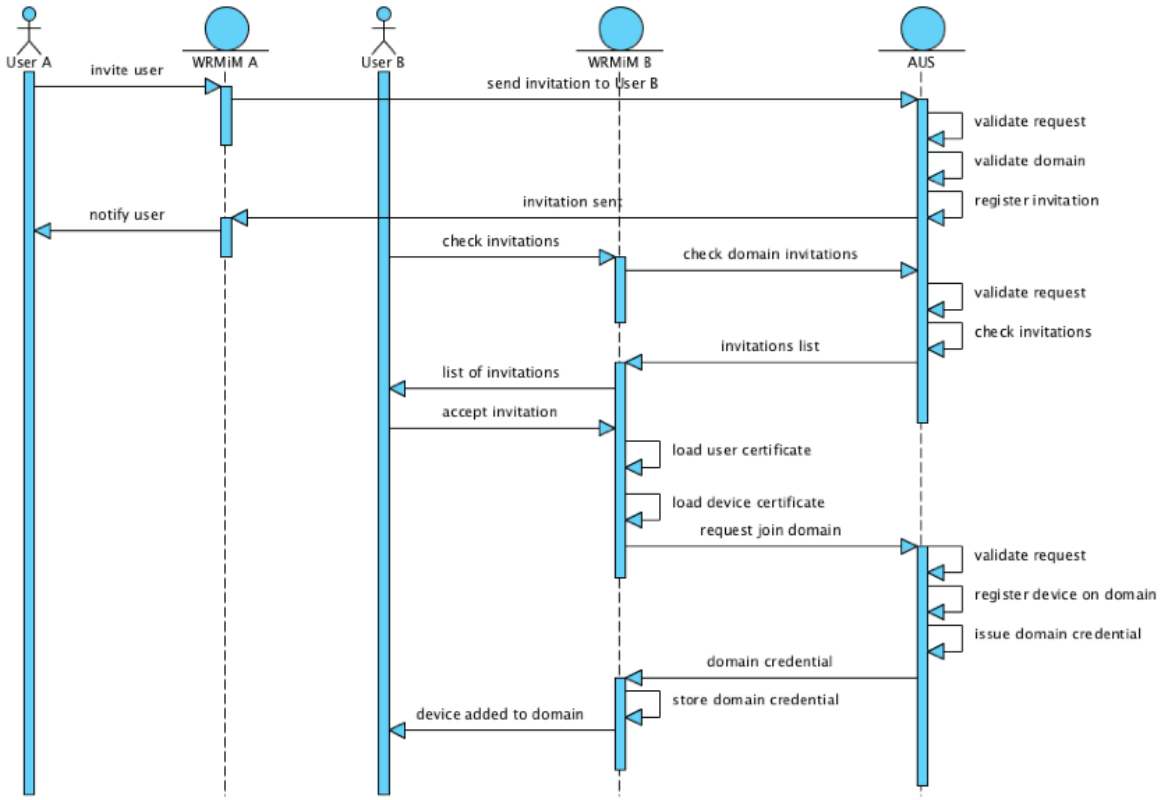

Figure III.6.30: Adding a device to a domain by invitation

6.5 Services message exchange

The process for the OpenSDRM services to exchange messages and to verify the authenticity and validity of such messages is composed of the following steps:

- The sender Service (SSender) composes a message using the following syntax: $K_{priv}^{SSender}\{Payload, Cert^{AUS}_{SSender}\}$;
- The receiver Service (SReceiver) receives the message and verifies the trust on the message. This trust is assured in the following way:
    1. SReceiver gets $Cert^{AUS}_{SSender}$ certificate and checks if it was issued by an AUS

in which SReceiver trusts.

2. This verification can be conducted if SReceiver has also a certificate issued by AUS: $Cert^{AUS}_{SReceiver}$.

- After the trust is established, the message signature can be verified and validated and SReceiver can trust its contents, and also in the service who has originally sent this message;
- SReceiver can then process the message payload and return its results for the SSender;
- SReceiver returns the following message to SSender: $K_{priv}^{SReceiver}\{Results, Cert^{AUS}_{CReceiver}\}$.

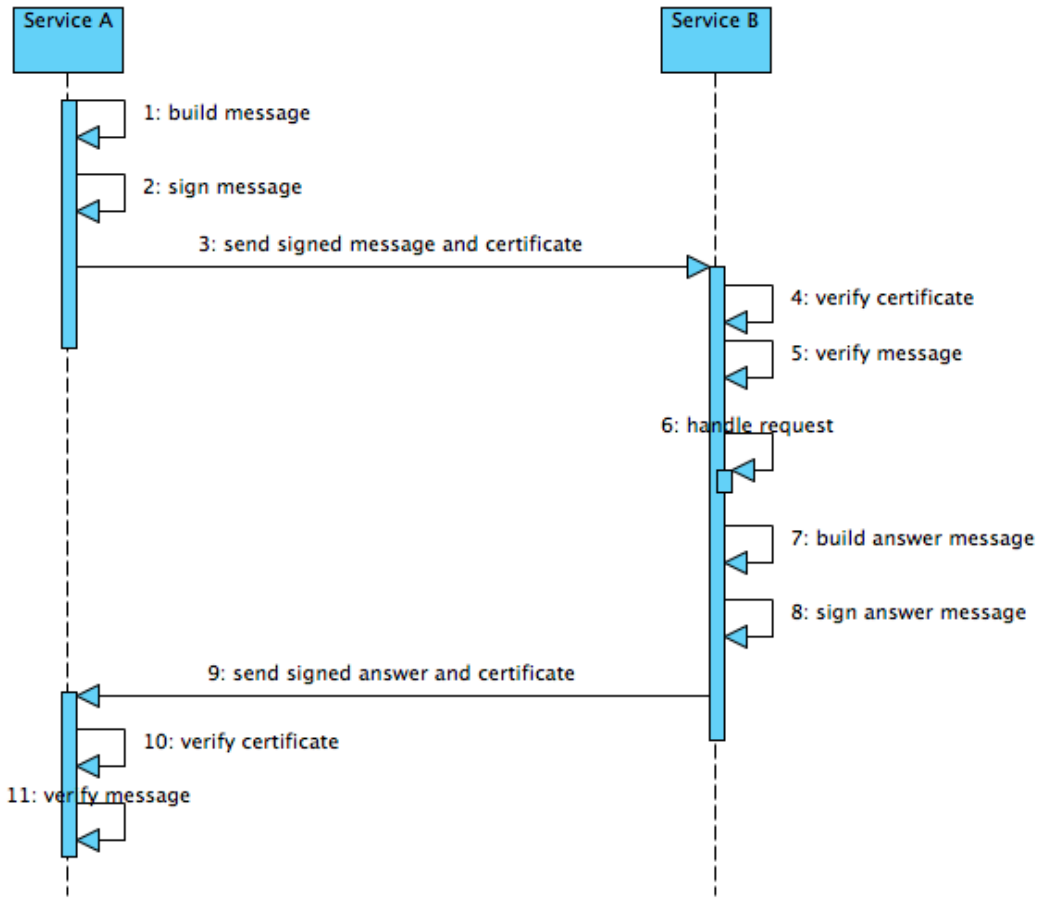

Figure III.6.31: Exchange of messages between services

With this mechanism, all the messages exchanged between the different OpenSDRM services are authenticated and digitally signed to prevent their modification.

6.6 Payment Information

Has it was previously mentioned, the OpenSDRM architecture supports the possibility to pay directly by the digital content and incorporates mechanisms for payment, although the payment method is outside the scope of the OpenSDRM architecture itself.

To provide this functionality a direct trust relationship must be established between the COS and the PGW. Therefore the COS (that relies on the payment functionalities) needs to subscribe a PGW. The process to subscribe a PGW can be described as the following:

1. The COS connects to the AUS and asks the AUS which are the PGW available on the system. COS sends $K_{priv}^{COS}\{Cert^{AUS}_{COS}, RequestAvailablePGWs\}$ to AUS;
2. AUS verifies the message, and returns an answer to the COS: $K_{priv}^{AUS}\{<ListOfAvailablePGWs, Cert^{AUS}_{PGW}>\}$;
3. The COS selects one available PGW and sends to it a subscription request: $K_{priv}^{COS}\{SubscribePGW, Cert^{AUS}_{COS}\}$;
4. PGW receives the request from the COS, validates its request and subscribes the COS. Therefore, this PGW will be used to validate and process payments used by a given User.

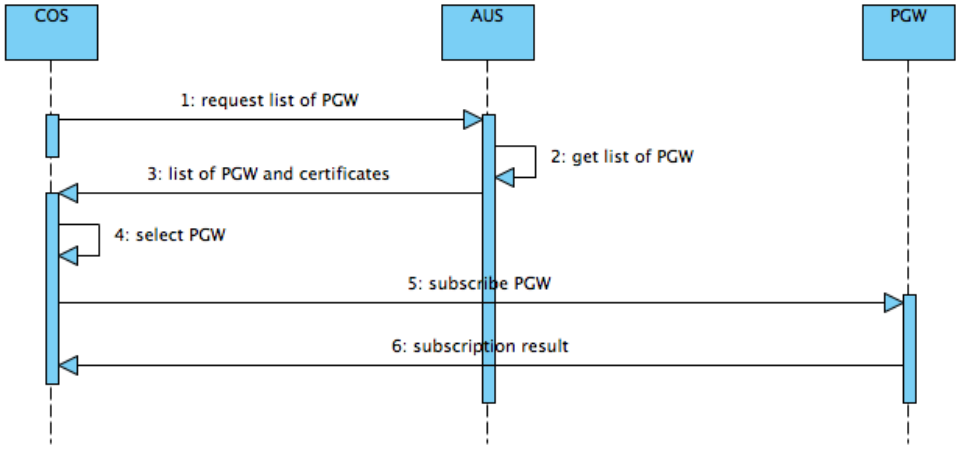

Figure III.6.32: PGW subscription

After this process concludes, the COS will be able to send payment information to an available PGW, and to process the different payment mechanisms supplied by the End-Users

to the AUS and supported by the PGW.

The following section will provide further information on how the payment is processed by the OpenSDRM architecture.

6.7 Paying services with OpenSDRM

Using the payment service provided in OpenSDRM involves two steps: validating the payment instrument and capturing the payment.

Validating the payment instrument is an important step on the system, since it will allow the COS to be sure that the payment method supplied by the user is authentic and valid, and that the transaction can be conducted without problems. Validating the payment involves the following steps:

1. The COS sends information about the payment details, namely information about the End-User order and the price to pay for it, to AUS: $K_{priv}^{COS}\{Cert^{AUS}_{COS}, Cert^{AUS}_{U}, Cert^{AUS}_{PGW}, PayData\}$;
2. AUS verifies and validates the COS request and checks the $Cert^{AUS}_{U}$ in order to retrieve the appropriate payment method choose by the User upon registration on the AUS. This data is ciphered with the public key of the PGW: $K_{pub}^{PGW}[PaymentClearence_{U}]$;
3. The AUS returns this information for the COS, signing it: $K_{priv}^{AUS}\{K_{pub}^{PGW}[PaymentClearence_{U}]\}$;
4. This information is then passed by the COS to the PGW, requesting it to validate the payment transaction: $K_{priv}^{COS}\{Cert^{AUS}_{COS}, K_{pub}^{PGW}[PaymentClearence_{U}]\}$;
5. PGW validates the message and deciphers the User payment clearance, using this information to communicate to the corresponding Payment Infrastructure, validating it. After, the PGW returns the result of the payment validation to the COS: $K_{priv}^{COS}\{Cert^{AUS}_{PGW}, Transaction_{ID}\}$;
6. This concludes the payment method validation on the PGW. This process assures the

COS that the services he is supplying to the User will be in fact charged.

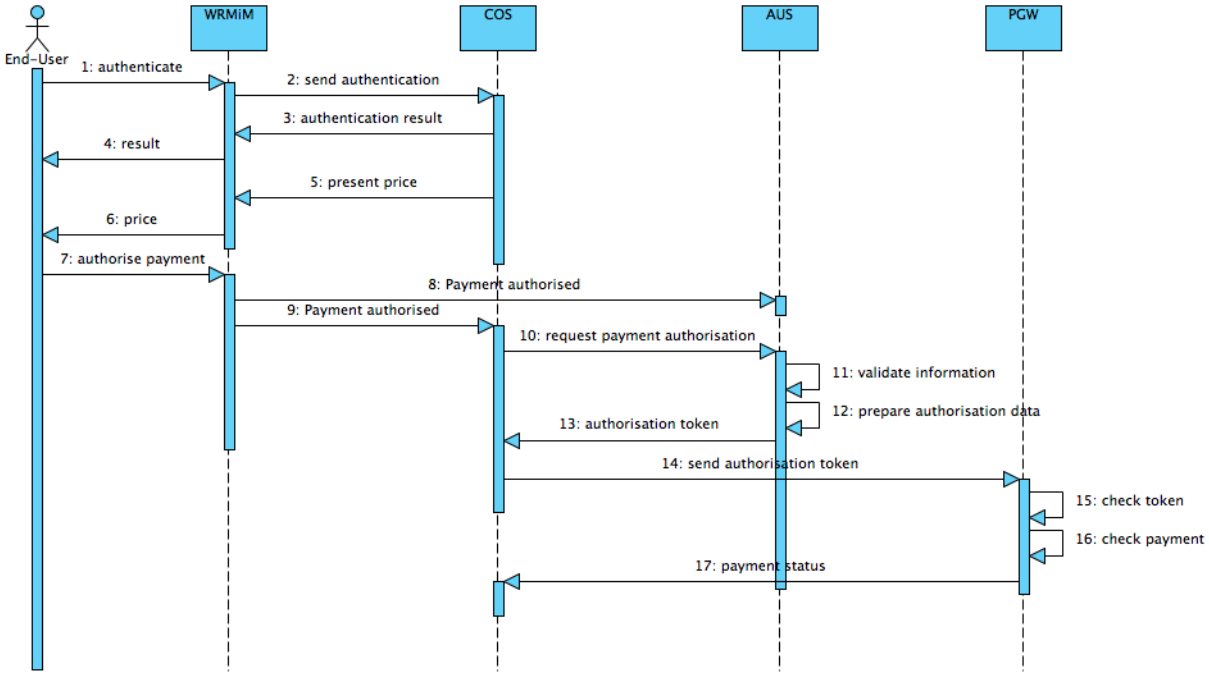

Figure III.6.33: Payment of digital content using OpenSDRM services

The second step in the payment procedure involves the payment capture. This process requires that first a payment capture has occurred and second that the COS possess a valid $\texttt{Transaction}_{ID}$. The capture process can be described in the following:

1. COS sends a message to PGW: ${K_{priv}}^{COS}\{{\texttt{Cert}^{AUS}}_{COS}, \texttt{Transaction}_{ID}\}$;
2. PGW validates the message and verifies the $\texttt{Transaction}_{ID}$, in order to evaluate if that transaction is in fact pending, and processes the payment;
3. PGW returns and a result status to the COS: ${K_{priv}}^{PGW}\{{\texttt{Cert}^{AUS}}_{PGW}, \texttt{Transaction}_{ID}, \texttt{Result}\}$.

This concludes the payment process in the OpenSDRM platform, and the COS will receive the indication from the PGW that the payment has been made.

6.8 License Production

One of the major functionalities of the OpenSDRM platform resides on the fact that it can be used to enforce control mechanisms on the way the Users access and use the content

protected by the platform. This process is ensured by the production of licenses. These are later applied on the content of the user on the Content Rendering Application by the appropriate set of Protection Tools. These licenses are produced and stored securely by the LIS, according to the choices made by the End-User and after the payment has been performed. This mechanism is further detailed on Part III, Chapter 5.

The process can be described in the following steps:

1. The User selects a set of available conditions, that allow him to define the usage conditions (rights) of the content the User wants to access;

2. COS sends a message to the LIS, requesting the production of a new license, for a specific content, and for a given User (it may also be specific for a Device or a Domain): $K_{priv}^{COS}\{Cert^{AUS}_{U}, Cert^{AUS}_{Device}, Content_{ID}, LicenseConditions, Cert^{AUS}_{COS}\}$;

3. LIS receives the request, verifies it and validates it. LIS generates the license using the appropriate REL language and parameters, contacting after the AUS for ciphering the license data for the User: $K_{priv}^{LIS}\{License\}$;

4. AUS receives the data, retrieves the K_{pub}^{U} and ciphers the received data: $K_{pub}^{U}[License]$, returning it afterwards to the LIS: $K_{priv}^{AUS}\{K_{pub}^{U}[License]\}$;

5. LIS receives and stores $K_{pub}^{U}[License]$.

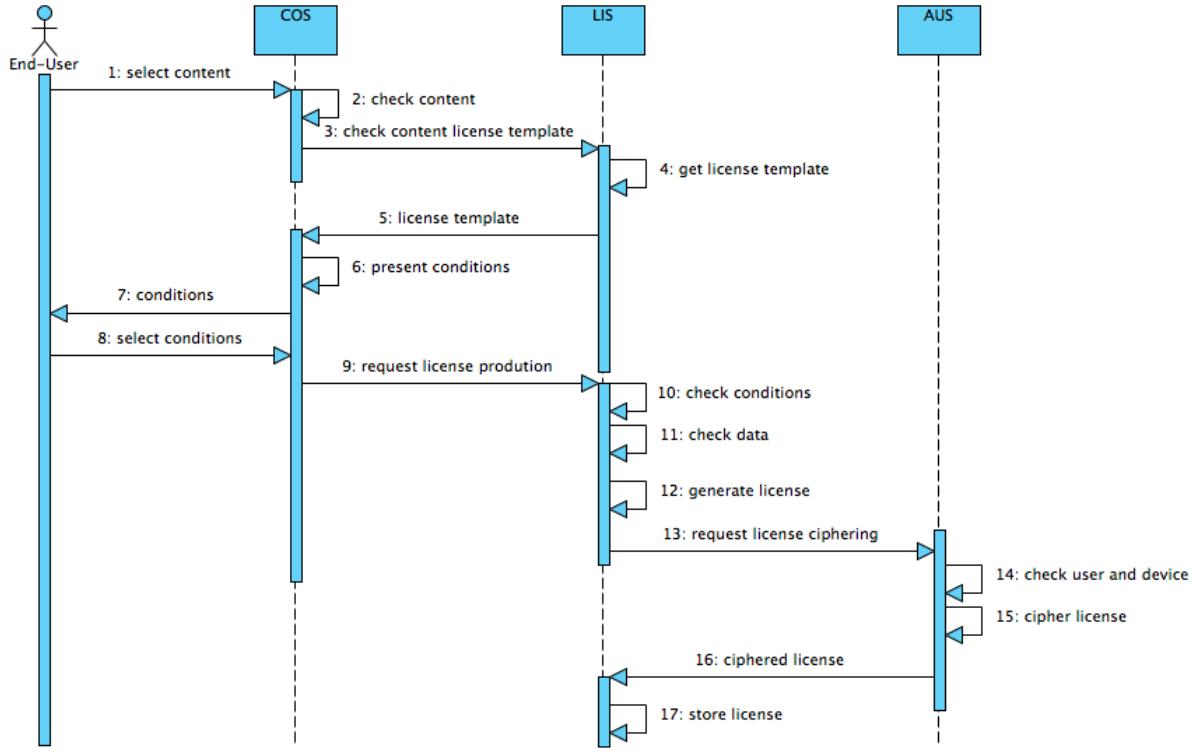

Figure III.6.34: Creating a license on OpenSDRM

6.9 License Download

When the End-User tries to access the content on the client side User system the CRA verifies that a license is needed to access the content. The CRA contacts the WRMiM to try to obtain the required licenses and corresponding keys to access the content. This mechanism is further detailed on Part III, Chapter 5.

This process can be described in the following steps:

1. The CRA contacts the wallet to obtain the license for the $Content_{ID}$ and $User_{ID}$;
2. The WRMiM checks on its secure repository if a license for that specific $Content_{ID}$ is already there. If that is true than this license is returned for the CRA in order for the content to be deciphered and accessed, controlled by a set of Protection Tools. If the WRMiM does not contain the license, it will request it from the LIS: $K_{priv}{}^{U}\{Cert^{AUS}{}_{U}, Cert^{AUS}{}_{Device}, Content_{ID}\}$;
3. LIS receives the data, validates it and retrieves the license from the database,

returning it to the WRMiM: $K_{priv}^{LIS}\{K_{pub}^{Device}[K_{pub}^{U}[License]], Cert^{AUS}_{LIS}\}$;

4. The WRMiM receives the data from the LIS, validates the message and deciphers the license that is passed to the CRA. Also the license is stored on the WRMiM secure repository for future accesses.

The downloaded license is kept in the LIS for later crash recovery in an event of failure and later expiration checks.

The following diagram (Figure III.6.35) illustrates the access to a license when the license is already at the end-user side, and is still valid.

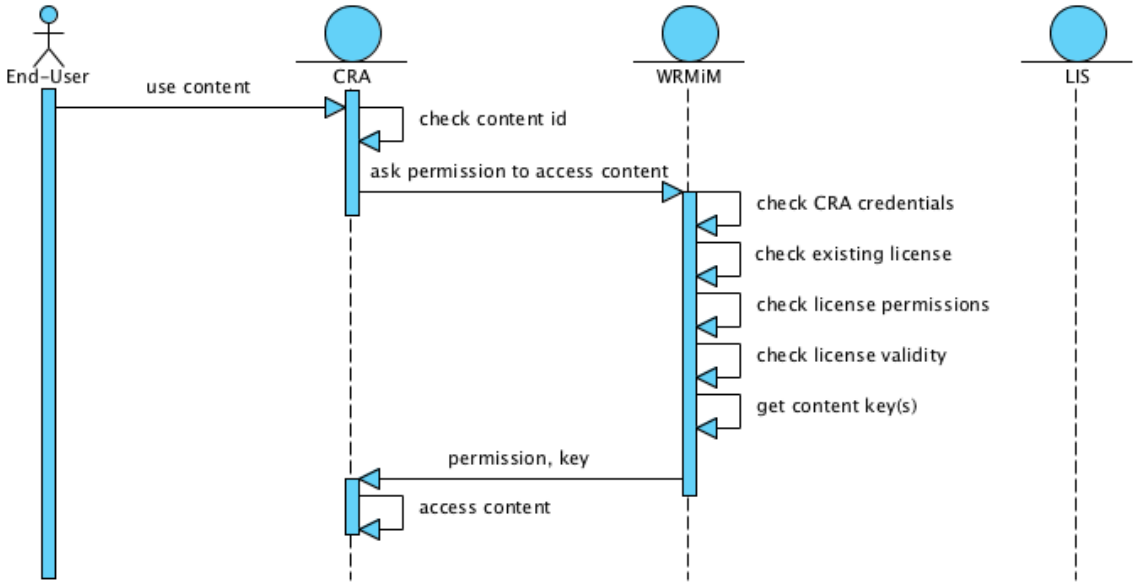

Figure III.6.35: Access to license, when license is already in the end-user side

If the license is not at the end-user side or if the license has already expired, it has to be downloaded from the License Server (Figure III.6.36).

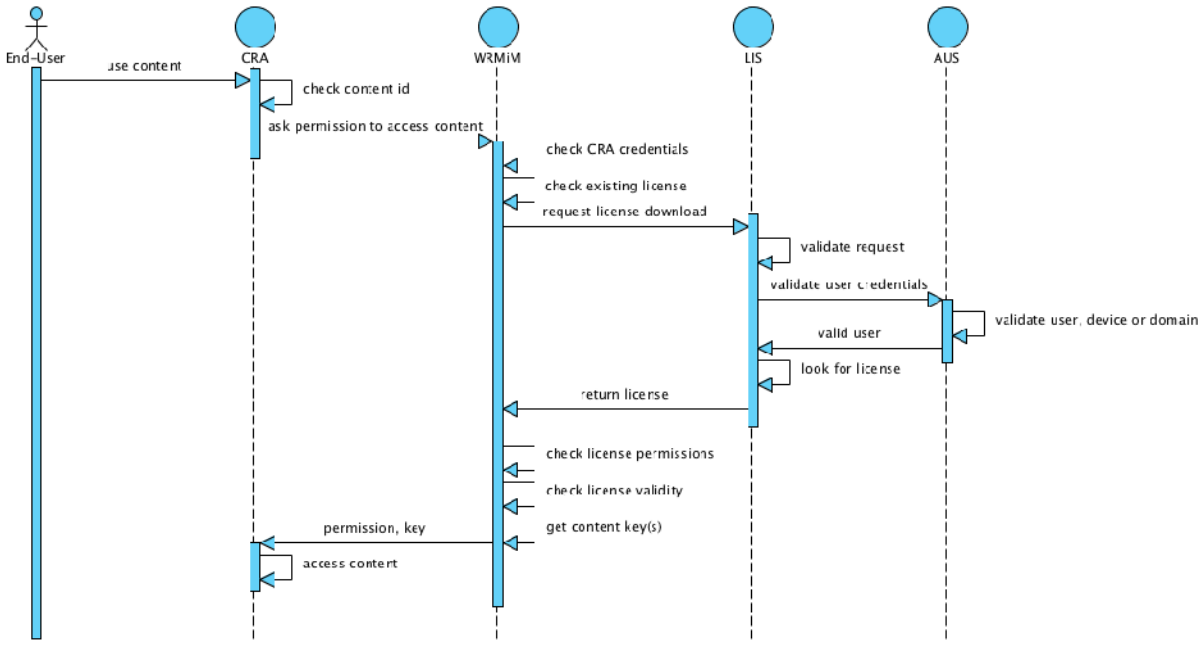

Figure III.6.36: Download a license from a remote License Server

6.10 License Expiry

Depending on the rights specified, a license will eventually expire. Rights such as a play counters or a date/time validity period may restrict the access to content to a certain number of times or to a certain time frame. The state of the license is maintained within the WRMiM that will from time to time validate with the LIS. Upon expiration, for example when a play count reaches zero, the WRMiM automatically checks at the LIS for a new license for that particular content. If there is no license available and the End-User wants to continue with the consumption of the digital content he has to purchase a new License as described before. The LIS also applies an internal checking algorithm to manage the state of its licenses. Licenses that expired are moved from the LIS to a backup table, to maintain the license history.

7 Conclusions

This chapter presented a contribution in terms of an open DRM architecture, based on a service-oriented approach to promote the integration and interoperability with other services. This open DRM architecture, implements an end-to-end model that can be used to

test and implement governed digital content business models, and its integration with other open DRM architectures [Serrão et al., 2003b].

This contribution has been made available as an open-specification and open-source implementation to the general community, and can be obtained at the Sourceforge.Net web-site (see Annex A).

During this chapter, the architecture of the OpenSDRM platform was presented and the main entities, actors and services were described and their interactions detailed. Each of these services provide an open interface based on a service-oriented approach to allow their internal and external integration. This approach allows the interoperability between the different OpenSDRM service providers, in order to build end-to-end digital content business models.

Finally, some of the main OpenSDRM architecture security mechanisms are presented and detailed as well as some aspects of the security protocols that need to be handled by the OpenSDRM inner processes.

Chapter 7. Wallet RM interoperability Middle-ware and License Templates

1 Introduction

The digital content distribution mediums are extremely powerful but also highly challenging from the intellectual property point of view. This presents strong barriers to those who wish to plan to use this distribution channel – in particular for authors and content owners. To overcome some of the problems created by unauthorised digital content distribution and usage, some technologic measures have to be put in place [Owens and Akalu, 2004]. Digital Rights Management (DRM) and copy protection technologies have to be put in place.

In particular, DRM can uphold specific conditions to specific groups of users, devices or domains in terms of digital content operation. This is in line with the content owners, content providers and author's expectations, on what concerns piracy prevention and IPR upholding, but creates obtrusiveness on the end-user side. One of the most important requirements for end-users is that DRM technology does not alter its digital content user experience when compared with non-DRM governed content [Serrão et al., 2005a]. Some of the reasons for this end-user side obtrusiveness are derived from the fact that most of the end-user requirements on what concerns the digital content experience are completely disregarded. Currently, only content owners, content providers and authors are considered in the design and implementation of the actual DRM-solutions [Serrão et al., 2005a]. There is a gap between the content owners, content providers and authors interests and final user's

expectations. These results in a bad end-user experience with DRM governed and protected digital content, and most of the times, users are forced to find alternative ways to enhance such experience – obtaining digital circumvented content on a massive file-sharing site, for instance.

One of the most important sources for this user's obtrusiveness results from the fact that most of the existing DRM solutions are proprietary, closed and vertical, following a 1-to-1 strategy on what concerns DRM. This means that they have their own protected-content formats, specific protection tools; specific rights expression mechanisms, specific rights enforcement mechanisms and even specific pre-defined business models [Serrão, 2004].

This DRM interoperability problem can be addressed from different perspectives. If digital multimedia content interoperability is considered, it may be addressed in terms of content format interoperability, content protection methods interoperability, rights expression interoperability and many others. In what concerns the object of this chapter, interoperability refers to rights management interoperability, although some of the concepts may also be applicable to other types of interoperability such as content protection, for instance. The approach followed in this chapter to address this rights management interoperability will consist in two different aspects:

1. At the server-side, at the DRM platform, using a set of pre-defined rights templates that will enable digital content providers to align their own business model with the DRM enforcing layer. These templates are defined by the content provider using a specific Rights Expression Language (REL) that defines the business rules, however the details of such rights language is hidden from the content provider [Serrão et al., 2005a];

2. At the client-side, the establishment of a DRM middle-ware layer, between the content rendering applications and the rights definition and management, allowing any content rendering application to integrate into this middle-ware layer, and to request permissions for conducting different operations over the digital content [Serrão et al., 2006b].

The main objective of this chapter is to present and detail the contributions made in terms of

the design and implementation of a client-side rights management middle-ware mechanism, called Wallet Rights Management interoperability Middle-ware (WRMiM). This mechanism implements a client-side interoperability mechanism, allowing the abstraction of the Content Rendering Applications (CRA) from the rights management complexities required by the governed digital content rendering [Serrão et al., 2006b].

Another contribution presented in this chapter refers to the establishment of a rights expression language independent template mechanism that allows rights templates to be instantiated by the content providers without having to deal with the complexity of the many existing REL. This way, the Content Provider, can define the conditions and terms that best suit their own digital content business model, without having to make any assumption about the specific rights expression mechanism that will be used.

This chapter will start by introducing the interoperability problem at the client-side and how this problem affects the end-user experience with end-user content.

After this introductory part, the Wallet Rights Management interoperability Middle-ware (WRMiM) is introduced, and a description of its major functionalities and characteristics is given. In this section the different modules that compose WRMiM are presented and its operation is also described. Another important aspect that is discussed in this section refers to the content rendering applications registration and the authorisations requests [Serrão et al., 2006b].

Next, a mechanism for creating and defining rights is presented. This mechanism is based on the concept of license templates, and allows the definition of a specific template, using a specific rights expression mechanism, that can be used by content providers to define their specific business models without having to deal with the complexities of rights expression mechanisms.

Finally, at the end of this Chapter, some conclusions from this work are presented and some future work considerations are made.

2 Digital Content Rendering Applications DRM interoperability

In this chapter introduction it was referred that interoperability is a key issue in DRM and it is extremely hard to achieve. Most of the DRM approaches existing nowadays assume a completely vertical strategy. Throughout the exploitation of this strategy they assume that the digital content value chain will always use their DRM solution:

- The content owners will use the DRM solution to package and protect the content;
- The content providers will use the DRM solution to establish the rights and conditions;
- The content end-users will use a specific content rendering application that uses the same DRM solution.

This is the approach that has been followed by two of the major DRM providers today. Both Windows Media Rights Management and Apple iTunes FairPlay assume that this is the most correct digital content business model.

Although this strategy does not bring too much hassle for both the content owner and content providers, end-user consumers are completely blown away by it [Serrão et al., 2006b]. This verticalisation strategy causes to end-users extreme difficulties in their digital content usage experience. Here is an example list of some of the difficulties that users have to face:

- Digital governed content using a specific DRM solution can only be rendered on a DRM-compatible player, or on a DRM-compatible device;
- DRM governed content cannot be rendered on traditional user's devices, nor converted into a format that is readable by them;
- DRM governed content cannot be lend or given to third persons;
- And many others.

For the sake of this chapter, only content rendering applications interoperability will be

considered, an aspect that will allow any content rendering application to be abstracted from the underlying rights management layer. Currently these applications include their own closed right management mechanisms making them dependent of a specific rights management technology.

This aspect creates serious interoperability problems between different content rendering application and devices. This chapter presents a contribution for partly solving this client-side DRM interoperability issues, based on the establishment of a generic transparent rights management interoperability middle-ware layer at the client-side. The objective of this interoperability layer is to free the content rendering applications from the burden of having to support multiple rights expression languages processing and different authorisation modules implementation. This interoperability layer provides such functionalities to all registered content rendering applications. Until now, all content rendering applications integrate the rights management mechanisms, making interoperability much difficult [Serrão et al., 2006b].

In order to achieve such interoperability level, a DRM middle-ware layer is defined and built at the client-side. Although this interoperability layer can be integrated directly in the operating system level, the approach followed in this work, was strictly to create such layer at the application level. The major goal of such interoperability layer is to allow that different content rendering applications can be abstracted from the inner DRM mechanisms that will uphold the content provider and content owner user rights at the end-user side and at the same time free the end-user from some intricacies presented by some more traditional DRM mechanisms. However, this does not reduce the need for content rendering applications to support the necessary cryptographic mechanisms that will be needed to access to the protected and governed content, and such mechanisms may be provided in the form of Protection Tools, as it was previously presented in Part III, Chapter 6.

3 Wallet Rights Management interoperability Middle-ware

This DRM middle-ware layer is capable of mediating the access to protected and governed

content by the different Content Rendering Applications (CRA) installed on the end-user system (Figure III.7.1). This DRM middle-ware layer will be referred as Wallet Rights Management interoperability Middle-ware (WRMiM) [Serrão et al., 2006b].

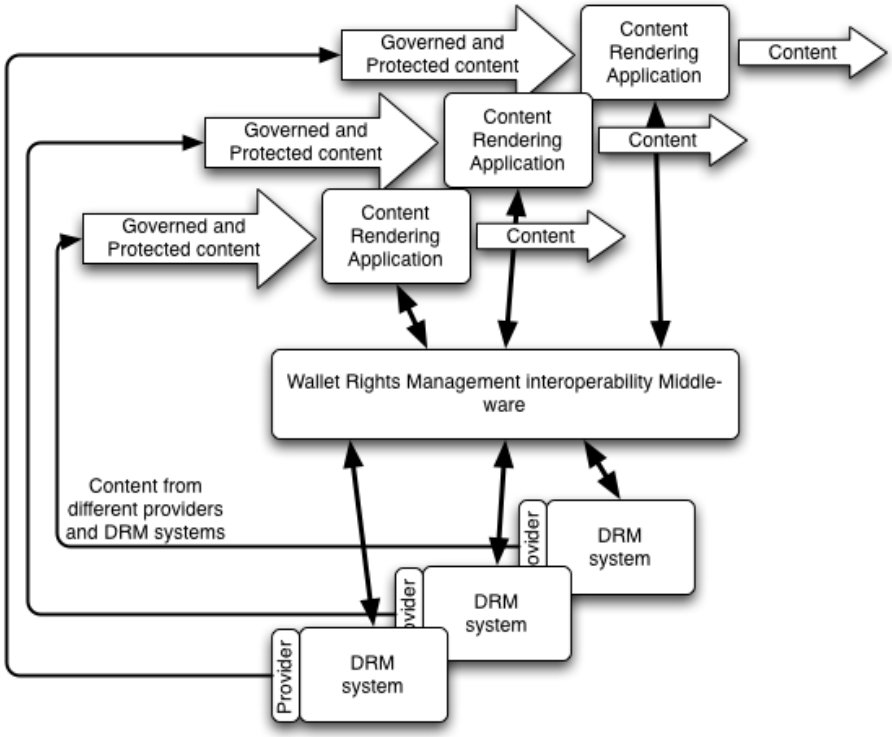

Figure III.7.1: Different types of protected and governed content used in multi-CRA environment

The WRMiM acts a broker infrastructure between the different CRA present in the End-User system and the many DRM systems that govern the content. WRMiM removes from the different CRA several rights management tasks, improving the interoperability between the different CRA.

Moreover the WRMiM is more than just the responsible for the establishment of an abstraction rights management layer. It was designed and developed to provide a different set of functionalities such as information secure storage, and a replacement for a payment mechanism, integrated with the remaining platform.

4 WRMiM modules

The WRMiM is a software-based application layer that sits on top of the operative system of

the device providing the necessary functionalities both to the different CRAs and End-Users.

This DRM middle-ware software acts as an intermediary between the different DRM platforms and the different CRAs that are installed on the user system, and that are registered in the WRMiM. This middle-ware layer is composed by a set of different functional modules. The most important modules are presented in the following sections (Figure III.7.2).

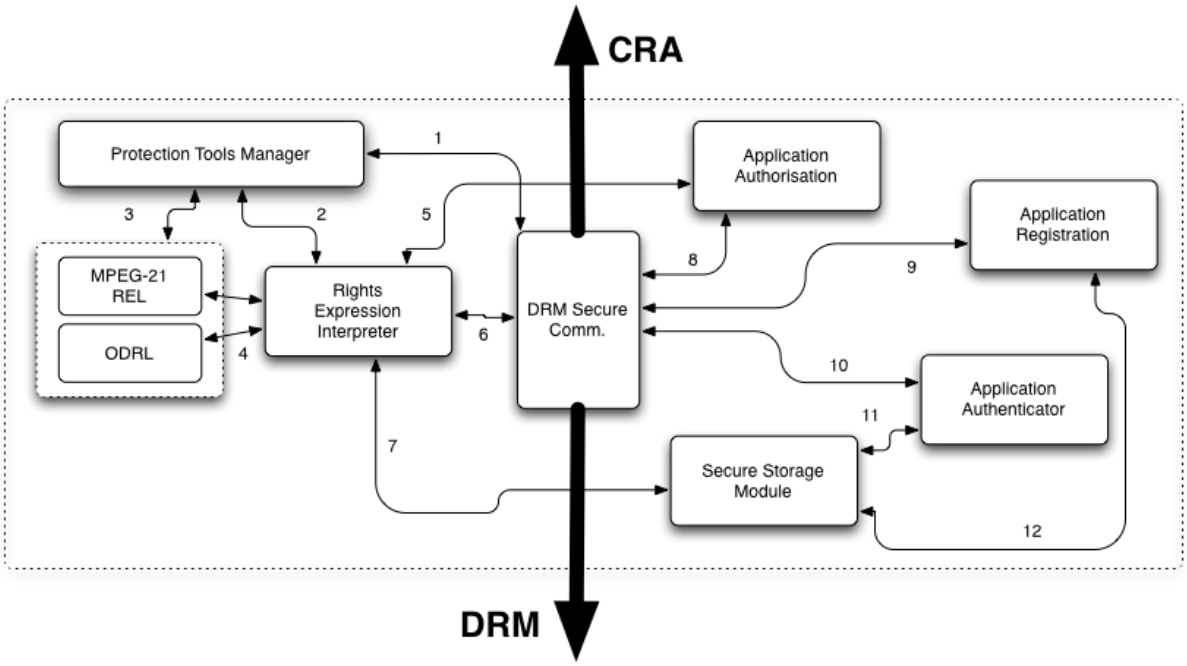

Figure III.7.2: Internal modules that compose the WRMiM

A set of interactions is established between the different modules. These interactions (Figure III.7.2) are described here:

1. The Protection Tools Manager uses the DRM Secure Communication module to obtain the appropriate protection tools that are need to perform some DRM governed operations;

2. The Protection Tools Manager instructs the Rights Expression interpreter about the different protection tools that are available on the system, and how they can be used;

3. The Protection Tools Manager installs and creates running instances of the different protection tools available on the system. Also, if some of the protection tools are not

used, they can be shutdown or removed from the system;

4. In this case, the Rights Expression interpreter uses one of the available protection tools to interpret the rights in the license that are expressed in a particular REL format;
5. The Application Authorisation module passes the parses the authorisation requests with the help of the Rights Expression interpreter;
6. The Rights Expression interpreter may request to the DRM Secure Communication module to download a new rights licenses form the DRM platform;
7. The Rights Expression interpreter may use the Secure Storage module to read or write information to a secure repository;
8. The DRM Secure Communications module is used to perform application authorisation;
9. During the application registration the DRM Secure Communications module is used to communicate with the requesting application;
10. To perform the application authentication, the DRM Secure communications module will have to be used;
11. The Application Authenticator may require to access information stored on a secure storage device to process the authentication requests from a CRA;
12. While performing the application registration, the Application Registration module can access the secure repository for both reading and writing of sensitive information on it.

Other software modules that are in charge of managing different aspects of the application compose WRMiM. However, the lists of modules that will be presented in the following sections are the most important and relevant for the DRM-related processes.

4.1 Secure Storage

One of the most important operations of WRMiM is the possibility to securely store

information in the device. In a way, this functionality allows that anon trusted device to provide some trusted operations, such as storage.

In essence, this module is the responsible for storing information at the End-User side in a secure manner. This secure storage uses cryptographic mechanisms to cipher information on the device file system, based on the AES cipher algorithm [FIPS, 2001] (or any other strong cryptographic algorithm), on information provided by the user and information collected from the device system. This module stores information about the user, the CRA, the licenses that are associated to content and its current state.

Although the WRMiM has been primarily designed to support the secure storage of information on the End-User side, it can also be used with other storage information strategies (Figure III.7.3). For instance, it can be used to support the remote storage of information, information storage on external devices (such as smart-cards [Lee et al., 2004], security tokens or others), or even some mixed strategies that combine any of the previous [Serrão et al., 2006b].

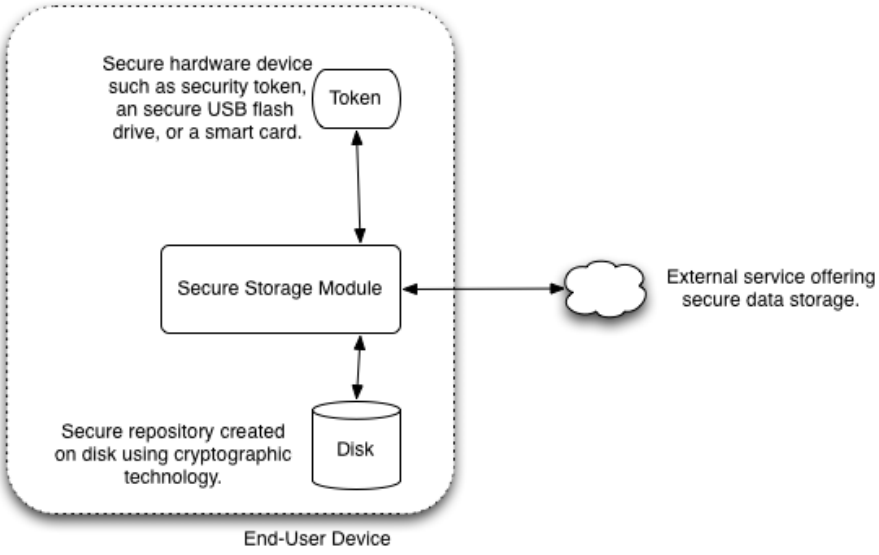

Figure III.7.3: Different interfaces with different storage sources

4.2 Application Registration

The middle-ware rights management software is available on the system to be used by the different CRA that need to handle with governed content. In order to use this middle-ware layer, each of the different CRA needs to be registered, in order for a trust relationship to be

established between both.

This module handles the CRA registration requests. It is responsible for receiving requests from client-side content rendering applications that will be able to handle DRM-protected content. This module registers the application generating cryptographic credentials that will be used latter for application validation.

WRMiM, through this module, maintains a registration of all the CRA installed on the End-User devices that are capable of handling governed digital content. The registration process can be limited to digitally signed CRAs, or open for all CRAs on the system. The content provider can however specific if its content can be delivered to a signed registered CRA or not.

4.3 Application Authorisation

Authorisation is an important mechanism in the rights management process. It is the mechanism that allows or disallows that a specific action is executed by a CRA over the governed content. Different actions are provided by the user to the CRA through some graphical user interface (GUI) mechanism. The CRA translates such actions into a normalised representation, and they are sent to the WRMiM.

This is a module that will receive requests from CRA to perform actions over a DRM-governed item and that verifies if the CRA is authorised (or not) to perform such action over the item. This module checks the license stored in the system returning the clearance information to the CRA. If the DRM-governed item is also protected, the appropriate content encryption key (or keys) is also returned to the CRA. The complete authorisation process can be resumed in the following:

1. The user performs some operation over the DRM-governed digital content, using the CRA;
2. The CRA, translates this operation, into a specific request;
3. This request is sent to the WRMiM;
4. The WRMiM, through the authorisation module, verifies if the User has the rights to

perform such action;

5. If the User has the right to perform such action, the authorisation module, checks if the rights are still valid;
6. If they are not, the authorisation module tries to obtain new rights (a new license) from the DRM platform;
7. The authorisation module returns to the CRA the result of the authorisation process and if needed the key (or list of keys) needed to render the content.

These previous steps are further specified and detailed in Part III, Chapter 6.

4.4 Application Authenticator

The main task of this module is to authenticate a client-side CRA that is requesting access to a DRM-protected content item. This authentication is based on the credentials that CRA supplies (and that were issued previously by the Application Registration Module). This module verifies the credentials and a secure and authenticated channel is established between the WRMiM and the CRA.

This process is important to avoid that external and non-authorised applications may interfere in the authorisation process and to try to circumvent the process and obtain the content protection keys in a non-authorised manner.

All the CRA need to be previously registered on the WRMiM prior to these authentication mechanisms take place.

4.5 DRM secure communications

One of the major functions of the WRMiM is to provide an abstraction layer between the CRA and the actual DRM mechanisms that are implemented to govern the content. Some of these mechanisms are dependent on DRM platform components which are located remotely and that need to be contacted for these mechanisms to complete successfully.

The communication between the End-User device WRMiM and the remote DRM

components needs to be performed securely. Moreover, before this secure communication takes place the WRMiM must be registered and recognised by the different DRM architectures that can be used.

This is the software module, part of the WRMiM, that handles all the secure communications performed between the DRM middle-ware layer and the CRA and between the DRM middle-ware layer and the server-side DRM platform components.

4.6 User registration and validation

As it was already presented on a previous chapter (see Part III, Chapter 6), one of the components of the DRM architecture depicted is used to represent the End-User. This component ensures that he user is properly registered on the system, and that he is entitled to receive the rights to use DRM-governed content.

This role is assigned to this WRMiM module whose purpose is to handle the end-user registration at the DRM server-side components, establishing the basis for the creation of the secure storage and for validating the users that try to access to the DRM middle-ware layer.

4.7 Rights Expression Interpreter

The ultimate goal for this WRMiM software is to separate the CRA for the complexity of having to support the processing of different rights expression languages (REL) or any other way of rights expression. The interpretation of any form of rights is conducted directly by this module.

Therefore this module is capable of performing the interpretation of any XML formatted license expressed using a REL, and provide meaningful information to uphold the user content rights over the different CRA requesting access to DRM-protected items.

This module uses an instance of the appropriate protection tool that carries the work of interpreting the specific REL (or any other rights representation mechanism) and that is obtained remotely by the Protection Tools Manager module (Figure III.7.4).

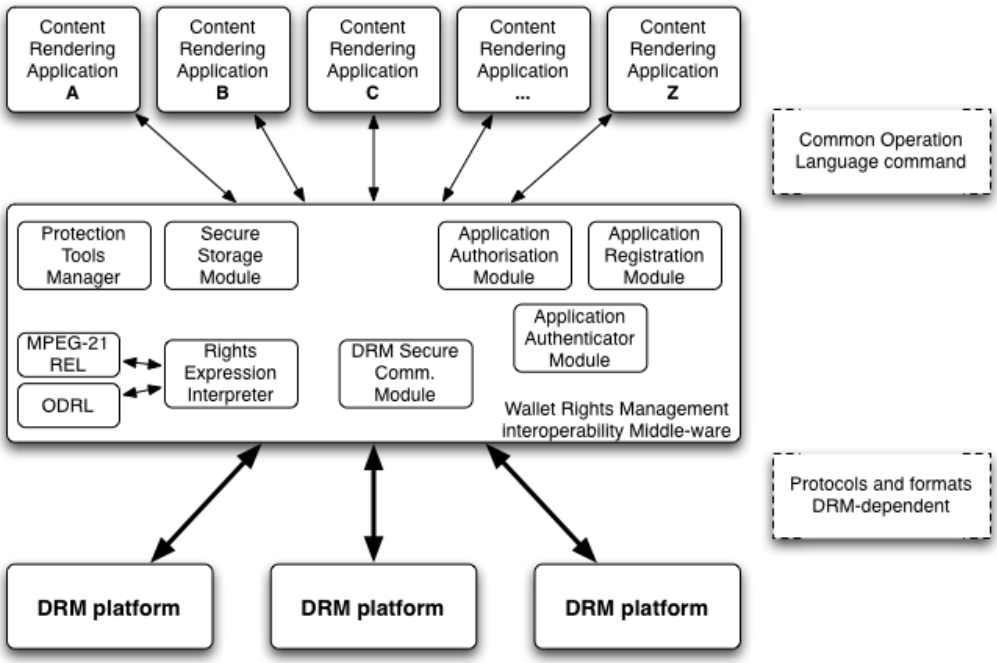

Figure III.7.4: The WRMiM using the Rights Expression Interpreter module

This module is capable of not only interpret the user rights but also to integrate with the Application Authorisation module and to match between the Common Command Operation Language commands (CCOL) and the rights expressed in the licenses (this is further detailed in Section 7.1).

4.8 Protection Tools manager

In order to ensure the scalability and extensibility of the platform, a mechanism based on the download and installation of remote protection tools have been used.

These protection tools are downloaded remotely by the WRMiM and installed and started on the end-user device. The governed content contains information that allows the selection of the necessary protection tools, its download, installation and start-up.

This module also verifies the integrity of such protection tools, if the content provider requests that. Also, it verifies if new versions of these tools exist and installs them.

This extensibility mechanism allows that WRMiM grows in terms of functionalities supporting new ways of rights expression and new ways of communicating with the different DRM platforms.

5 Establishing a DRM middle-ware layer

To establish such DRM middle-ware layer at the end-user side require that several steps are concluded with success. One of the most important steps that will need to be achieved is the end-user registration at the DRM platform. This DRM registration process occurs the first time the DRM middle-ware is boot up and uses the SSL/TLS protocol to establish a secure and authenticated channel with the DRM platform servers (Figure III.7.5). The following steps compose the process:

1. The DRM middle-ware layer software computes a key-pair (K_{pub}^{WRMiM}, K_{priv}^{WRMiM});

2. The K_{priv}^{WRMiM} is stored on a secure repository (internally, externally or on an hardware token). In the case an internal secure repository is used, a key ($SSkey_{AES}$) is computed based on the information hashing of the user-name and password pair, choose by the end-user, plus some additional generic information collected from the device – $SSkey_{AES}[K_{priv}^{WRMiM}]$. In the case an external remote repository or an hardware token is used, the responsibility of establishing the secure information storage methods is of such entities;

3. The user will introduce some more information to the DRM middle-ware interface and then this information is sent to the DRM platform. If the user wishes, the WRMiM can also act as a payment mechanism. In order for this functionality the user will have to provide some payment mechanisms such as a credit card number or a Paypal account;

4. The DRM platform registers the WRMiM and returns back a certificate that will validate this DRM middle-ware installation ($Cert^{DRM}_{WRMiM}$). This certificate contains the K_{pub}^{WRMiM} and the K_{pub}^{DRM}, signed by the DRM platform;

5. This certificate is received by the DRM middle-ware and is also stored in the secure storage – $SSkey_{AES}[Cert^{DRM}_{WRMiM}]$. This concludes the user registration process and the establishment of the WRMiM. Every time the DRM middle-ware boots up, the user is requested to authenticate – the WRMiM supports more than one user, meaning that each of the users has to individually register to DRM, repeating these

five steps.

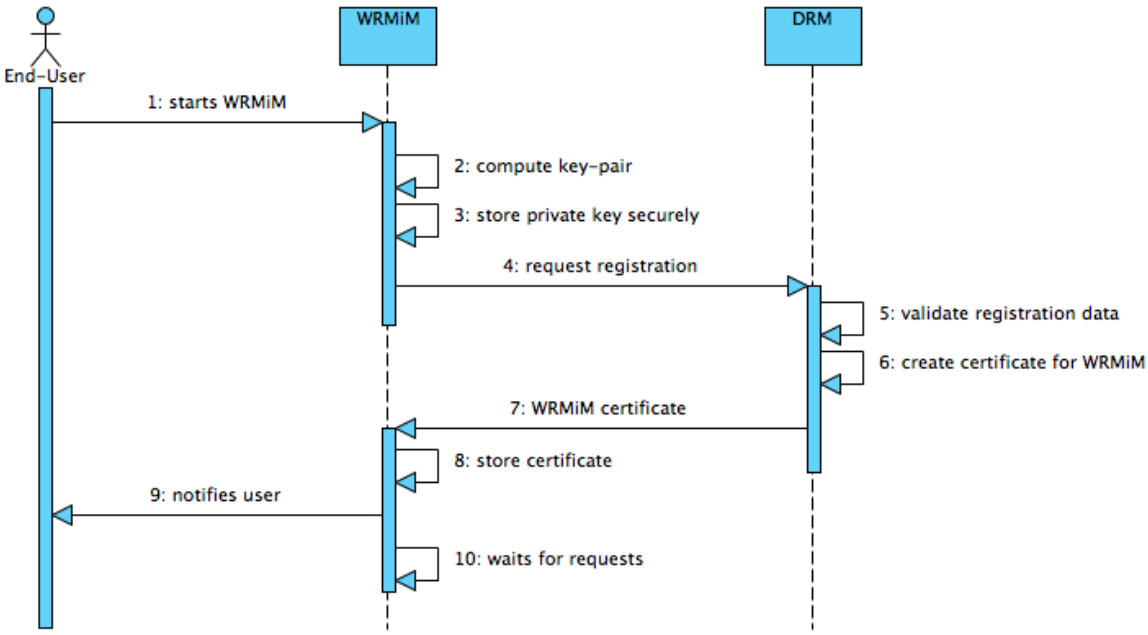

Figure III.7.5: Initial boot-up (first time run) sequence for the WRMiM

After a successful boot up the WRMiM, the end-user device is prepared to receive requests from the different applications installed on the system and that will wish to handle DRM-governed content (Figure III.7.6).

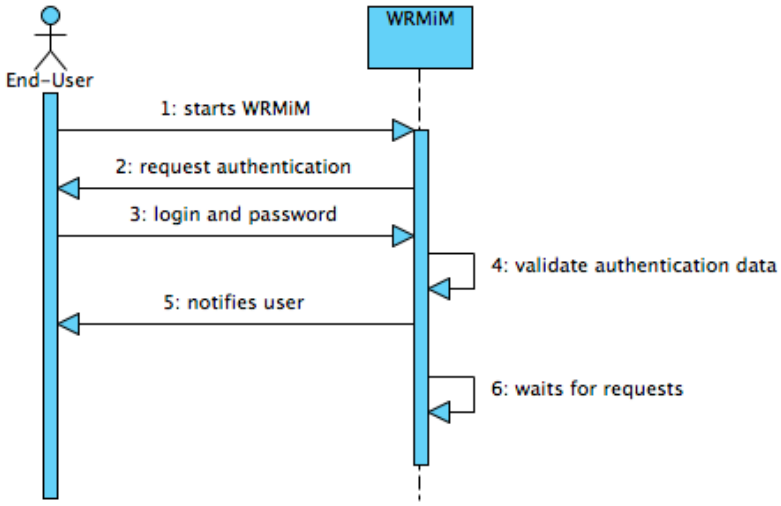

Figure III.7.6: WRMiM initialisation

6 Registering content rendering applications on WRMiM

After the initialisation of the WRMiM, it is prepared to start receiving requests from content rendering applications. Before the WRMiM accepts any request from any CRA on the system, this CRA has to enrol with WRMiM.

This is a necessary step to ensure that only registered applications are allowed to use the platform. Only registered and authenticated CRA can request content operations to the DRM middle-ware (this may include receiving content deciphering keys provided in the rights expressions). Any of the CRA that wishes to use this system will need to know how to execute the following two processes:

1. Enrol to and request authentication to the DRM middle-ware, exchanging a set of credentials with it, to enable the CRA authentication and the establishment of a secure channel between the application and the WRMiM;
2. Request a specific operation over the DRM governed content using a formalised language (CCOL) and receive or not the clearance to perform such operation.

The first process is important to establish the trust between the CRA and the WRMiM. The CRA registration process can take two different paths, depending on the fact that the CRA already has a certificate or not:

- If the CRA has been signed by its provider, it already has a digital certificate issued by a trustworthy CA ($\mathtt{Cert}^{CA}_{CRA}$) containing the CRA public-key ($K_{pub}{}^{CRA}$), and therefore the trust between the WRMiM and the CRA could be established with an higher degree of trust. This is important because some content providers could require supplying its content only to this type of CRA (Figure III.7.7). In this case, the set of operations necessary to register the CRA on the WRMiM can be resumed in the following:
    1. CRA sends $\mathtt{Cert}^{CA}_{CRA}$ to WRMiM as part of the registration request;
    2. WRMiM verifies $\mathtt{Cert}^{CA}_{CRA}$, registers the CRA and generates a new certificate

for the CRA ($Cert^{WRMiM}_{CRA}$). This certificate contains the CRA public-key that was retrieved from the original certificate;

3. The CRA receives the new certificate and can now start communicating with WRMiM in a secure and authenticated way.

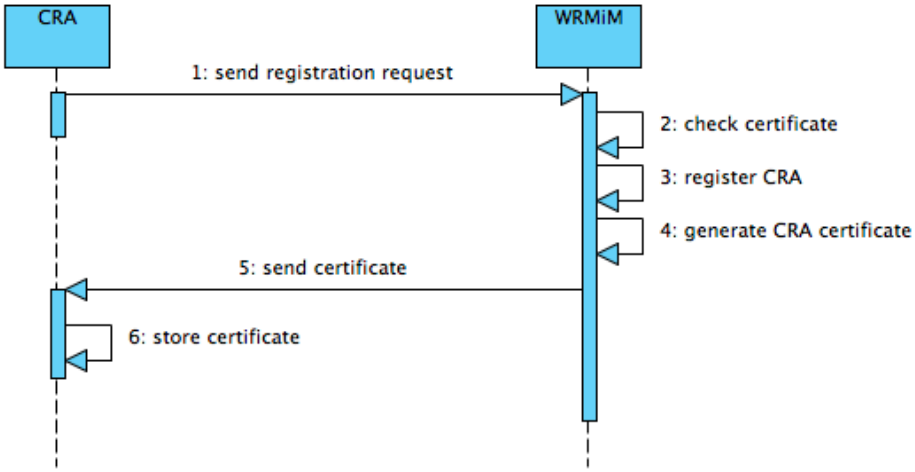

Figure III.7.7: CRA registration using a certificate

- The other case, the CRA will need to create a new key pair and stores securely the private key in one place (this is the responsibility of the CRA). In this case the trust level between the CRA and the WRMiM is lower (Figure III.7.8). In this case, the set of operations to conclude the CRA registration is the following:

1. The first operation that the CRA needs to perform is to compute a key pair (K_{pub}^{CRA}, K_{priv}^{CRA});

2. The K_{priv}^{CRA} should be stored securely by the CRA (this is a decision of the CRA itself). The CRA sends K_{pub}^{CRA} to the WRMiM, as part of the registration request;

3. WRMiM registers the K_{pub}^{CRA}, and generates a certificate to be returned for the CRA ($Cert^{WRMiM}_{CRA}$) – this certificate contains the K_{pub}^{WRMiM} and is signed by WRMiM;

4. The certificate is received by the CRA and stored. The registration process is concluded with success and WRMiM can establish secure and authenticated

communication channels among each other.

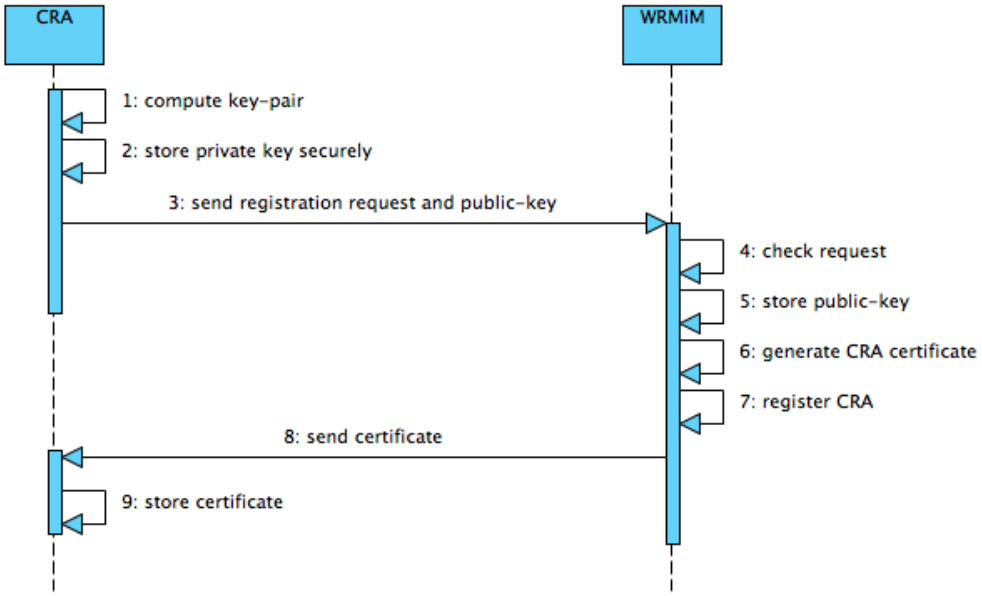

Figure III.7.8: CRA registration without having a previous certificate

At the end of both of these processes, the CRAs installed on the end-user device are certified to request authorisations to perform some type of operations over the governed digital content. In the following section, the authorisation process is explained.

7 Requesting CRA authorisations from the WRMiM

After the CRA is properly registered on the WRMiM it can use it to request access clearance to perform DRM-protected content operations. This process starts with an authentication between the CRA and the DRM middle-ware to establish a common secret key to create a secure channel between them. This secure channel prevents some malicious applications on the end-user device to listen to the information exchanged between the CRA and the WRMiM.

This process depends on two different but important aspects: a) that there is a common language between the CRA and WRMiM and b) that the DRM-governed content contains some metadata information, readable by the CRA, to instruct the WRMiM how to proceed [Serrão et al., 2005a].

7.1 Common Command Operation Language

The first aspect refers to the definition of a commonly accepted and understood language between the CRA and the WRMiM, that is used to indicate what is the operation behind performed over the DRM-governed content. This language, here refereed as Common Command Operation Language (CCOL), is a simplified representation of the operation that is trying to be performed by the CRA over the DRM-governed content.

The function of the WRMiM is to match the request expressed in CCOL and the rights actually owned to allow or disallow the operation. The need to comply with such CCOL is one of the requirements that CRA would have to comply. The CRA must also be previously registered on the WRMiM. The proposed DRM-governed content life cycle (Figure III.7.9) is composed of the following steps:

1. User orders CRA to perform operation;
2. CRA verifies parameters in content (if they exist) and build request;
3. CRA sends permission to WRMiM;
4. WRMiM matches the request with rights and builds the answer;
5. WRMiM authorises or denies;
6. CRA verifies the answer and performs or not the operation (considering that the applications do not miss-behave);
7. Finally the result is displayed or not to the end-user.

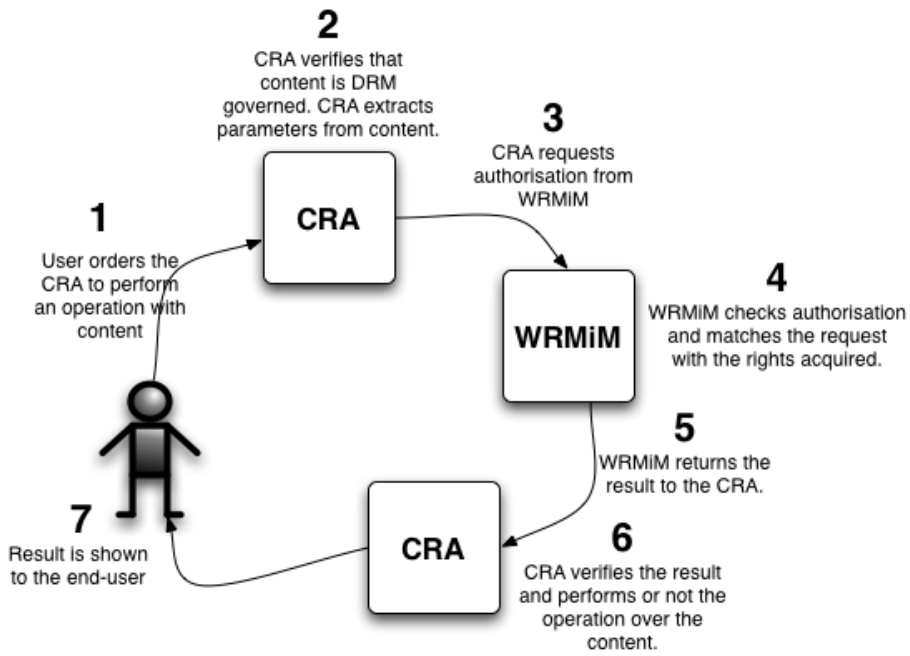

Figure III.7.9: DRM-governed content life cycle

This cycle is repeated each time the user requests the CRA to conduct any operation over the DRM governed digital content.

As it was referred previously, CCOL is a simplistic language that is used to send CRA command requests to the WRMiM. It is XML-based and basically represents the following structure: `CCOL_command := {operation, operation details, content identifier, content relevant meta-information}`. Here is what each of the element of the message contains:

- **operation**: this is a mandatory field that indicates what is the operation that will be carried over the digital object. Examples of this operation are: play, copy, move, save, and others;
- **operation details**: this is an optional field that enhances the previous field with some more details about the operation being performed. For instance it may add details about a particular part of the content that is going to be played, or about a specific saving operation that is being attempted to an external device, and many others;
- **content identifier**: the content identifier is a mandatory field that allows the identification of the digital content. It is important for performing rights management

operations such as the rights acquisition or usage tracking;

- **content relevant meta-information**: this is an optional field contained in the content, that has been placed by the content provider, to provide both the CRA and the WRMiM information about additional requirements needed to use the content. This field contains information about the location of the rights holder or information about the protection tools.

Since CCOL is an XML-based language, the full schema is presented bellow.

```
<?xml version="1.0" encoding="UTF-8"?>
<schemaxmlns="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://www.example.org/ccol/"
targetNamespace="http://www.example.org/ccol/">
<complexType name="command">
      <sequence>
            <element name="operation" maxOccurs="1" minOccurs="1">
                  <simpleType>
                        <restriction base="string">
                              <enumeration value="play"></enumeration>
                              <enumeration value="display"></enumeration>
                              <enumeration value="print"></enumeration>
                              <enumeration value="execute"></enumeration>
                              <enumeration value="modify"></enumeration>
                              <enumeration value="excerpt"></enumeration>
                              <enumeration value="annotate"></enumeration>
                              <enumeration value="aggregate"></enumeration>
                              <enumeration value="sell"></enumeration>
                              <enumeration value="lend"></enumeration>
                              <enumeration value="give"></enumeration>
                              <enumeration value="lease"></enumeration>
                              <enumeration value="move"></enumeration>
                              <enumeration value="duplicate"></enumeration>
                              <enumeration value="backup"></enumeration>
                              <enumeration value="install"></enumeration>
                              <enumeration value="delete"></enumeration>
                              <enumeration value="verify"></enumeration>
                              <enumeration value="restore"></enumeration>
                              <enumeration value="uninstall"></enumeration>
                              <enumeration value="save"></enumeration>
                        </restriction>
                  </simpleType>
            </element>
            <element name="details" type="string" maxOccurs="1"
                  minOccurs="0">
            </element>
            <element name="cid" type="string" maxOccurs="1"
                  minOccurs="1">
            </element>
            <element name="meta" type="tns:meta"></element>
      </sequence>
</complexType>

<complexType name="meta">
```

```
      <sequence>
            <element name="license_location" type="string" maxOccurs="1"
                  minOccurs="1">
            </element>
            <element name="ptools" type="tns:ptools" maxOccurs="unbounded"
minOccurs="0"></element>
      </sequence>
</complexType>

<complexType name="ptools">
      <sequence>
            <element name="toolid" type="string"></element>
            <element name="toolurl" type="string"></element>
      </sequence>
</complexType>
</schema>
```

Please note that although CCOL is currently based on the operations defined by ODRL [Guth et al., 2005], it might be extended in the future. Next is also presented a simple example of how the CCOL can be used to express a simple authorisation processing. Imagine that a user has a DRM-governed music and wants to play it.

```
CCOL_command    :=    {"play",    "myMusicIdentifier",    "url_of_license,
url_of_protection tool"}
<?xml version="1.0" encoding="UTF-8"?>
<ccol:command>
      <ccol:operation>play</ccol:operation>
      <ccol:cid>myMusicID</ccol:cid>
      <ccol:meta>
            <ccol:license_location>URLLicenseServer</ccol:license_location>
            <ccol:toolid>CommParser</ccol:toolid>
            <ccol:toolurl>URLCommParserToolsLocation</ccol:toolurl>
            <ccol:toolid>RELParser</ccol:toolid>
            <ccol:toolurl>URLRELParserToolsLocation</ccol:toolurl>
      </ccol:meta>
</ccol:command>
```

This command is interpreted by the WRMiM that verifies if some of the protection tools needs to be downloaded or not, if the user has already any license for this DRM-governed content or if it needs to get one. Finally, WRMiM verifies and tries to match the request to any of the rights granted by the license. An answer is returned to the CRA, that executes or not the requested action.

7.2 Authorisation Request protocol

The authorisation process is a very delicate aspect of the rights management at the end-user

device. It is important that both the CRA and the WRMiM trust each other in such a way that the WRMiM is not receiving fake requests to disclose information, or the CRA is not sending and receiving sensitive information by some malicious software that mimics the WRMiM behaviour.

Therefore a secure and authenticated channel between the CRA and WRMiM is established and an Authorisation Request protocol is used to carry the authorisation processes sent by the CRA and mediated by the WRMiM. The building blocks to establish the trust between CRA and WRMiM have already been discussed previously in this Chapter (in Section 6). The following paragraphs present the details of the authorisation request (Figure III.7.10):

- The CRA initiates the Authorisation Request protocol by sending its own credentials (obtained in a previous step) to the WRMiM: $Cert^{WRMiM}_{CRA}$;
- WRMiM validates these credentials, assuring that the CRA has been in fact registered previously, and computes a secret session key ($SessKey_{AES}$). This session key is ciphered with the CRA public key (K_{pub}^{CRA}) and returned to the CRA: $K_{pub}^{CRA}[SessKey_{AES}]$;
- The CRA receives and deciphers this message, obtaining the session key ($SessKey_{AES}$), that will be used on the subsequent steps;
- Next, the CRA analyses the content and extracts some specific meta-information, related to rights governing, that will allow building the appropriate request (using the CCOL notation);
- The CRA builds the authorisation request message (using the CCOL_command structure) that will be send encrypted to the WRMiM: $SessKey_{AES}[CCOL\_command]$;
- This request is received by WRMiM that verifies if there are on the system licenses for that match the request the CRA has made. If not, WRMiM will have to connect to the DRM platform (indicated by the request) and check if there is any license available for that request. If this license exists, it is downloaded by WRMiM and securely stored;
- WRMiM locally interprets the license. WRMiM uses some specific protection tools

that might already have been downloaded, and the authorisation requested by the CRA is checked against the rights expressed in the downloaded license. If the requested authorisation is a valid action over the content, an answer is sent to the CRA authorising the action. If the content is also protected (ciphered or scrambled, for instance) the Content Encryption/Scrambling Key (CEK) is read from the license and returned to the CRA securely - $\texttt{SessKey}_{\texttt{AES}}\texttt{[CEK]}$;

- The CRA receives the $\texttt{SessKey}_{\texttt{AES}}\texttt{[CEK]}$ and deciphers the CEK. This CEK is then use to perform the operation over the content. This authorisation request is performed each time the CRA when an operation is conducted over the content.

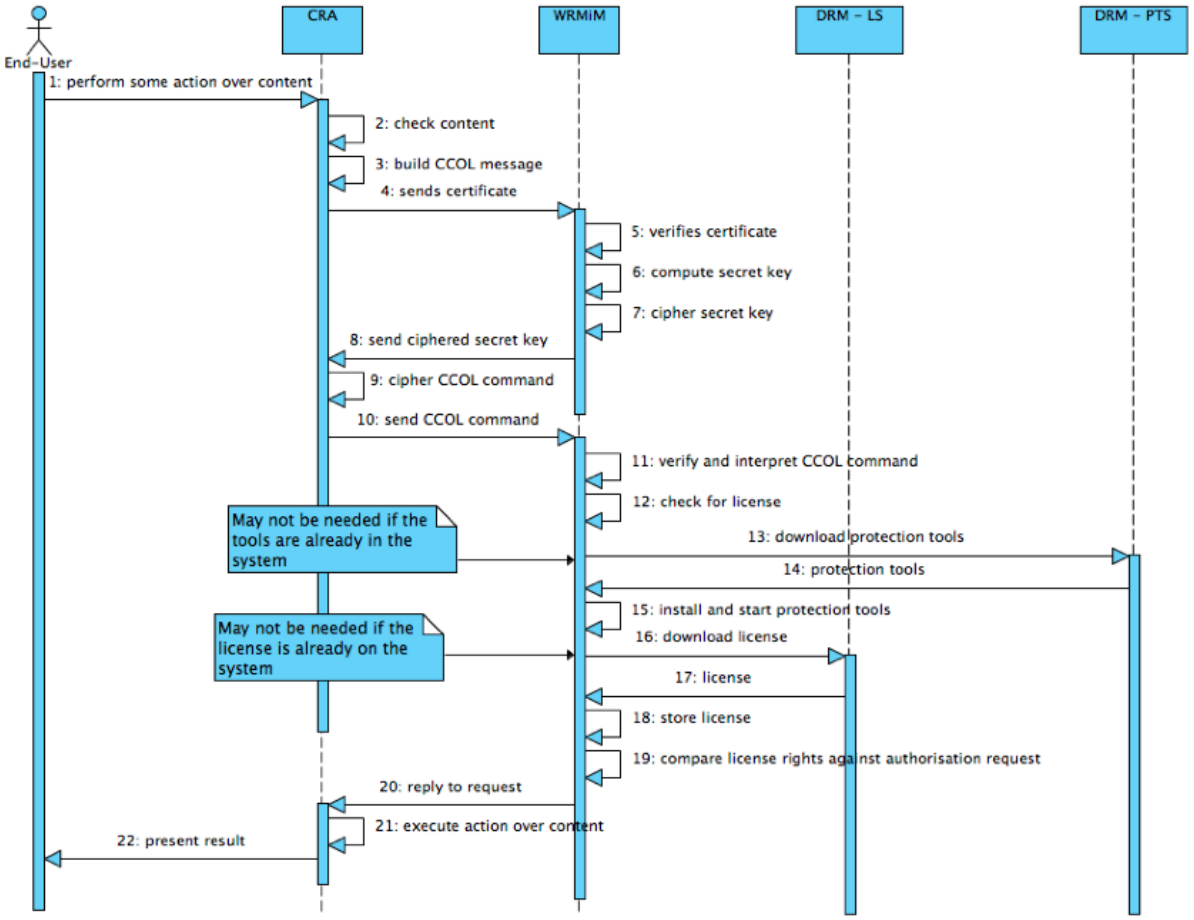

Figure III.7.10: Full authorisation protocol

This simple authorisation protocols allows that multiple CRA coexist on the end-user system without having to know any details about the rights management processes. Such processes are entirely handled by WRMiM. As it was presented, the only specific requirements for CRAs are for them to be registered and trusted by the WRMiM (Section 6), and know how to use

and implement the CCOL (Section 7.1).

8 License Template System

The rights management processes as they have been presented in this work require the usage of a mechanism that is able to express rights in such a way that any other system can use it to enforce them. Currently, in the DRM context, there is a major trend for expressing rights using XML. Two major rights expression languages emerge and are used to express licenses that are used to enforce the conceded rights: one is MPEG-21 REL (which is based on the formed XrML), used in MPEG-21 based solutions and the second is ODRL, used in OMA [Iannella, 2004]. Although these languages share XML as a major commonality they are different in terms of semantics and syntax.

Although these languages are a major trend today, more rights expression languages (XML-based or not) may appear in the future. The proliferation of such may lead to more complex, incompatible and non-interoperable rights management systems.

In a complex rights management ecosystem many content providers will have to coexist with many different ways of rights expression. Therefore this ecosystem should contemplate both the existence of different license rights providers on the system, and also the possibility that each of these providers issue more than one license type. Currently, all DRM platforms solutions implement vertical strategies, and a single license rights provider chooses their own license format that is only perceptible and valid in particular clients.

In many current real World cases, the license provider is strongly linked with the place where the content is obtained and with the implemented business model. This situation creates most of the times an unnecessary burden in licenses issuance and management, and also trends to work as an interoperability blocking force.

The contribution presented in this section minimises the rights definition complexity on the content provider's side while it increases the interoperability between different rights management approaches. This contribution foresees a model in which several content supply services can coexist while building business relationships with multiple license rights providers. These rights license providers can issue multiple license rights to many different

users – it is a many-to-many relationship (Figure III.7.11).

Therefore different content providers can use different license rights distributors, and at the same time, different users may obtain their licenses on different license rights distributors. This approach de-couples both the content provider from the license provider and the end-users from a particular license provider.

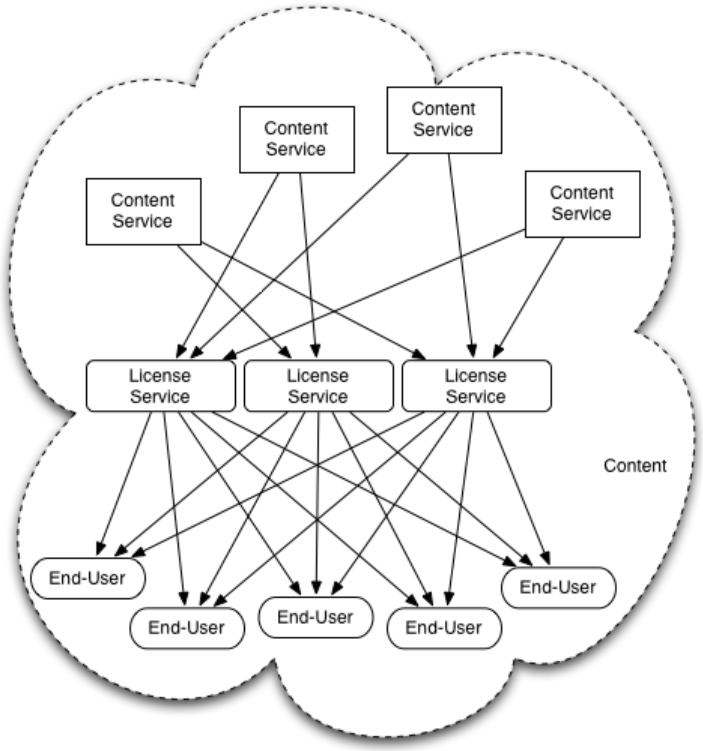

Figure III.7.11: License rights production and distribution schema

Based on this interoperability idea and on a concept that has been around for long in computing world, the creation of license rights templates was proposed. In general, a template is a model or pattern used for making multiple copies of a single object. Also in this case the concept applies. A model of a license is created (the template) from which many copies may be created (the different licenses, for the different types of content and the different users).

Using this approach, a content provider can define its business model (or a set of different business models) and, using tools provided by the license rights provider, define the conditions under which their digital content can be negotiated. Therefore the focus of the content provider is on its business and not on the technical decisions of choosing the

appropriate rights expression mechanism.

This thus gives origin to a license rights template, using one or several rights representation mechanism (for instance, the template may be represented in ODRL or MPEG-21 REL format). The content provider can define as many license rights templates as he finds adequate and choose as many of license rights providers as he wish.

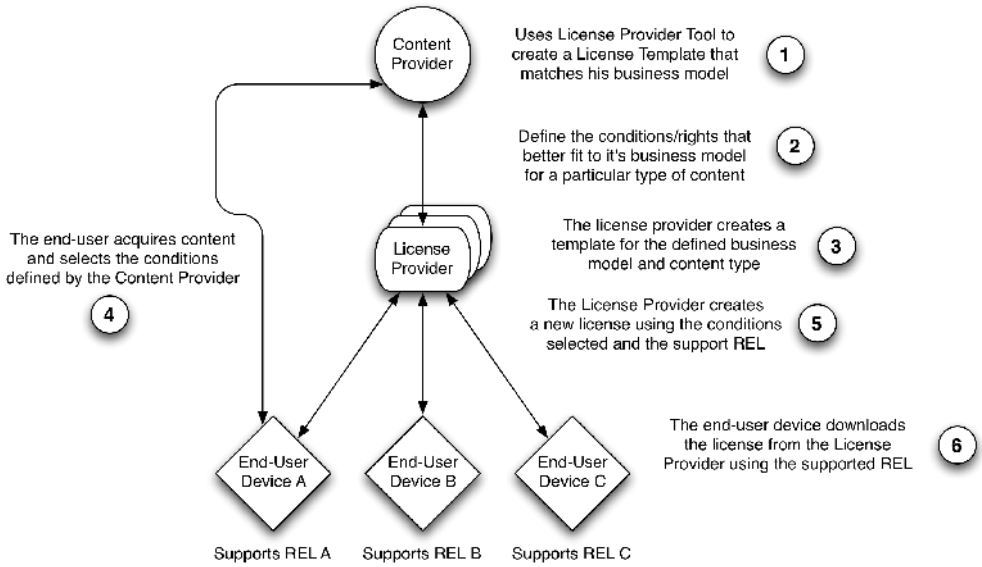

Figure III.7.12: Process to define the license templates

In particular, the process occurs in the following steps (Figure III.7.12):

1. A Content Provider, after establishing some contractual bound with some License Provider, uses a License Provider tool to design its business model, associated with a particular content identifier, or to a particular type of content. This tool allows the Content Provider to establish the terms and rights under which that particular content can be subject;

2. The Content Provider selects the different negotiable conditions and rights inside the license rights templates. The complexity of such templates depends on the complexity of the business model itself. If the business model for a particular type of content is simple and standardised, the generated license templates are quite simple;

3. After this, the Content Provider, based on the choices made by the Content Provider, creates a license rights template, using the REL (or different RELs) that are supported by him. This process creates a license rights meta-model, containing a set of

replaceable parameters that will be used to create instances of the final license in the specific REL format. These templates are stored by the License Provider and associated to a particular Content Provider and content type;

4. When the end-user acquires content on the Content Provider side, the license template (associated to the content) is presented to the end-user, via the License Provider, and the different negotiable conditions can be set-up. These conditions may or may not negotiable by the end-user. In some cases, the conditions and terms are just presented to the end-user;

5. Based on the selections made by the end-user and on the instructions provided by the Content Provider, the license template is instantiated, and a new license is produced by that end-user and content (and stored);

6. Finally, the end-user downloads the license in the appropriated format. This format (REL) depends on the capabilities that the end-user device has to process it. For instance, if the end-user device is able to understand and process just ODRL, than it will try to get the license rights from the License Provider in ODRL format.

The following demonstrates an example of a simplified ODRL license template that defines a particular business model for a content provider. This particular template defines a very common business case scenario for digital music. The template defines that the right to play is given to an individual, a certain number of times, for a given period of time.

```
<?xml version="1.0" encoding="UTF-8" ?>
<o-ex:rightsxmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xmlns:o-dd="http://odrl.net/1.1/ODRL-DD"
                  xmlns:ds="http://odrl.net/1.1/ODRL-DD"
                  xsi:schemaLocation="http://odrl.net/1.1/ODRL-EX
../schemas/ODRL-EX-11.xsd
                  http://odrl.net/1.1/ODRL-DD ../schemas/ODRL-DD-11.xsd">
      <o-ex:agreement>
            <o-ex:asset>
                  <ds:keyInfo>
                        <ds:keyValue>%KEY%</ds:keyValue>
                  </ds:keyInfo>
                  <o-ex:context>
                        <o-dd:uid>%CID%</o-dd:uid>
                        <o-dd:name>%PARAM_1%</o-dd:name>
                  </o-ex:context>
            </o-ex:asset>
            <o-ex:permission>
                  <o-dd:play>
```

```
                        <o-ex:constraint>
                              <o-dd:individual>%UID%</o-dd:individual>
                              <o-dd:count>%PARAM_2%</o-dd:count>
                              <o-dd:datetime>
                                    <o-dd:start>%SDATE%</o-dd:start>
                                    <o-dd:end>%EDATE%</o-dd:end>
                              </o-dd:datetime>
                        </o-ex:constraint>
                  </o-dd:play>
            </o-ex:permission>
      </o-ex:agreement>
</o-ex:rights>
```

On the license template all the parameters that can be replaced are represented using a specific notation (`%KEY%, %CID%, %UID%, %SDATE, %EDATE, %PARAM%`). There is an interactive tool at the content provider side that provides the means for the content provider graphically express it's several business models. Each of these business models is represented in a specific REL format, but without the proper realisation of the final rights license.

The license rights production process works in the following way (Figure III.7.13):

(a) Each of the content suppliers defines their own business models (translated into these license templates). These business models, describe the specific business rules and conditions for each of the templates. In an upper limit case, there may be as many license templates, as the number of digital content objects the content provider wishes to supply. In a lower limit there can be only one, where the content provider says that his business model is always the same independent of the content type;

(b) When an end-user obtains governed and protected content from some content supplier, a license is produced using the specific license template defined previously. This license contains both the content unique identifier and the user identification and deposited on a place for posterior download or sent directly with the content;

(c) After, the can be downloaded by the end user – not directly by the end-user but by the WRMiM and enforced over the governed content as it was previously presented (see Section 7).

Figure III.7.13: From license templates to license rights

Using this procedure the Content Provider can be independent of any of the particularities created by specific ways of representing rights. Moreover, the end-user devices will be able to get the license rights in a format that is format-friendly.

This contribution aims to enhance the interoperability of the different Content Providers, License Providers and End-Users devices, through the establishment of many-to-many rights management relationships between them

9 Conclusions

This chapter presented two contributions. The first contribution was about the establishment of a client-side DRM middle-ware to provide interoperability between different DRM-protected content rendering applications (CRA) [Serrão et al., 2006b]. The second refers to the creation and usage of license rights templates to offer rights management interoperability between multiple content providers, licence providers and end-user devices [Serrão et al., 2005a].

In the first contribution, a client-side interoperability middle-ware is created to provide the different CRA installed on the end-user device the capability to interoperate with different DRM regimes. This middle-ware (WRMiM) application provides the abstraction layer between the CRA and the rights management processes, which are specific of each of the different DRM platforms. This contribution also presented a way of requesting authorisations

to the rights management layer to perform operations over the DRM-governed digital content. This mechanism includes the registration of all the CRA installed on the end-user device, the establishment of trust relationships between the CRA and the WRMiM and the definition of a specific language to request permissions to the WRMiM.

The second contribution presented in this chapter refers to establishment of a mechanism that aims to facilitate the definition of rights by Content Providers. This mechanism will allow the different Content Providers to express their business model to the different License Providers, and to establish license rights templates, which can be instantiated in the future.

Both of these two contributions present mechanisms, both on the client-side and on the server-side that aim to improve the interoperability between the different DRM applications and the different DRM stake-holders. The contributions represent a step forward in the proposal of technical solutions that address directly the interoperability problems faced by DRM technologies, as they are presented today.

Chapter 8. OpenSDRM use cases and deployment scenarios

1 Introduction

The OpenSDRM platform, developed in the context of this thesis, was introduced on a previous chapter (see Part III, Chapter 6) as a platform that could be easily adaptable to any type of digital object or to any business model.

This chapter will provide some usage and deployment scenarios where OpenSDRM has been used. The following four different deployment scenarios will be present and described on this chapter:

1. Digital music e-commerce through an online service called Music-4You [Serrão, 2005];
2. Controlled access to video-surveillance data [Serrão et al., 2003d];
3. E-Commerce and controlled access to large Earth observation products [Serrão and Dias, 2002];
4. Home network music jukebox [Serrão et al., 2006g].

These deployment scenarios have been implemented and tested in different trials in different research projects where the author of this thesis has actively participated. Each of this projects involved digital object rights governance where the OpenSDRM platform, with some particular and specific adaptations was used.

In this particular chapter, the different experiences are presented and described, in particular on what concerns the usage of the OpenSDRM platform.

2 Music-4You, Digital music e-commerce

The Music-4You[20] web-site is a digital music B2C (business to consumer) e-commerce site that was developed to prove the technological concepts behind MOSES project (mainly the MPEG-4 player IPMP-X implementation and the OpenSDRM platform integration) [Kudumakis, 2003]. The content format adopted for this music web-site was MPEG-4 audio (IPMP-X had been developed primarily to target this format), and a specific MPEG-4 player, integrating the IPMP-X, was developed and distributed freely to users, during the project lifetime. MPEG-4 also presented the advantage of adding additional media information to the music tracks, such as images and the music lyrics. After this initial phase and after the MOSES project ended, the MP3 file format [Brandenburg, 1999] was also supported and a specific player was developed to be able to render MP3 protected files [Serrão, 2005].

Music-4You (Figure III.8.1) was developed with the purpose to become a music portal, targeted for two types of users: consumers that wanted to listen to music and for music bands (music providers) that wanted to promote their music [Serrão, 2005]. The portal aggregates the work of several amateur bands, allowing them to disseminate their work over the web with few effort and investment, with the possibility to protect and govern their own content according to a set of parameters previously specified and using OpenSDRM. On the other hand, consumers could access to a large set of free music and to non-free music previews and downloads.

20 http://www.music-4you.com

Figure III.8.1: Music-4You portal main web-page

The MOSES project targeted not only the PC as the final device, but also embedded devices, such as PDAs and Mobile Phones. The same architecture, with the proper adaptations, tackled all these devices to provide the same functionality – the possibility to listen to music and at the same time uphold the rights of the copyright owners. A specific version of the Music-4You portal was developed and tailored for small screen rendering devices, such as PocketPCs.

2.1 Users functionalities

Music-4You allows the portal users to access governed multimedia content – in this case music tracks. All the users are allowed to visualise the information about the bands on the web-site and to download all the free content from the portal. To access the protected and governed music tracks, the users need to register on the Music-4You portal. The registration process on Music-4You requires the installation of the WRMiM module (as presented in Part III, Chapter 7).

2.1.1 Users registration

The Music-4You registration process (Figure III.8.2) is performed directly on the WRMiM through an appropriate form (1). In this form the user supplies its user information, such as a selected username and a password, email address and also provides a valid payment method. All this information is stored centrally on the OpenSDRM platform (on the Authentication Server) and a new account is created. After this process is completed with success the user can go the portal main page and log-in using the appropriate fields and credentials.

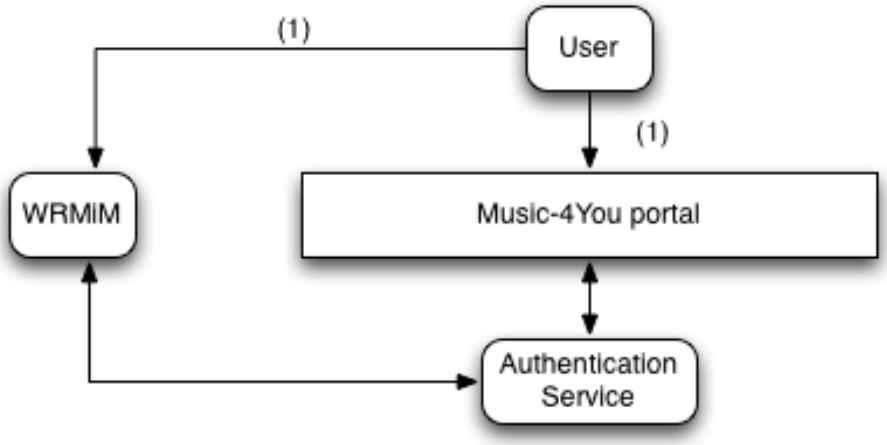

Figure III.8.2: User registration on the Music-4You portal

The Music-4You portal communicates with the OpenSDRM Authentication Server to validate the user account. This registration process is performed only once – this functionality provides a single sign-on mechanism for multiple digital content services.

2.1.2 License negotiation

OpenSDRM authenticated users are allowed to download governed content from Music-4you. The user selects the music track to be downloaded and establish the conditions to access the governed content. In this stage licensing conditions are established (in terms of the duration of the license and the number of play counts). The license is bounded to the user and to the specific content. In the case of portable devices there is also the possibility to lock content to just one device or a set of devices owned by the user (Figure III.8.3) [Serrão et al., 2006g].

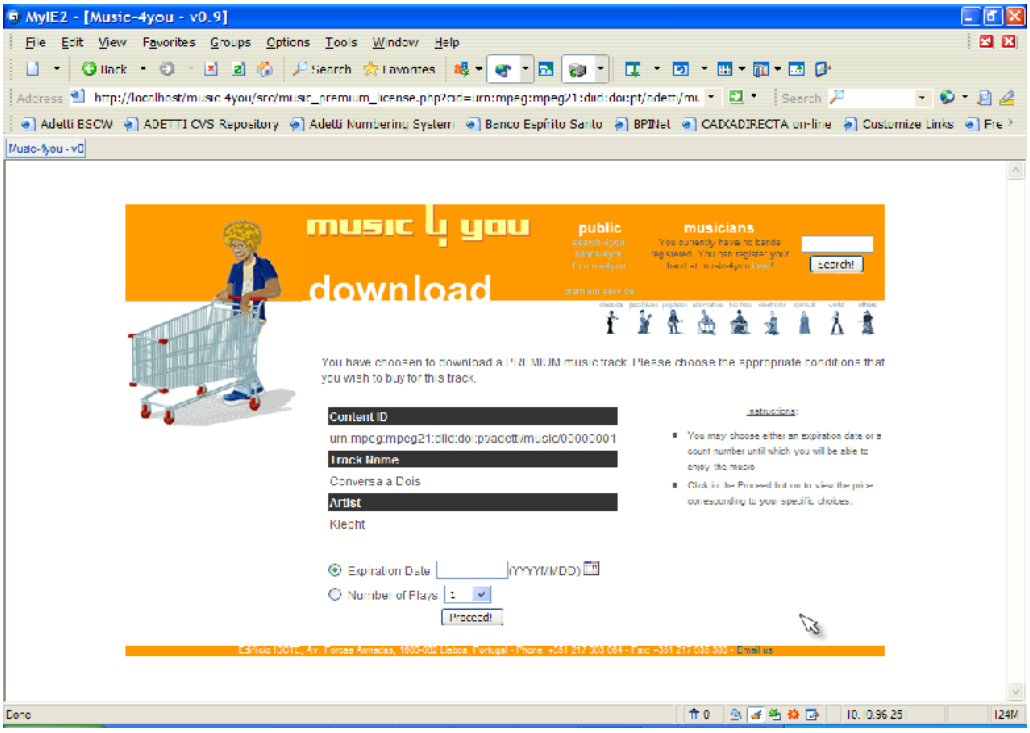

Figure III.8.3: License conditions negotiation at the Music-4You site

The final price varies according to the established licensing conditions.

Upon user acceptance, Music-4You contacts the OpenSDRM Authentication Server requesting the user payment authorisation. This payment authorisation is anonymised and passed to the OpenSDRM Payment Gateway that validates it and captures the user payment. After the payment is processed with success, the license is produced and stored on the OpenSDRM License Server. The user can afterwards download the protected music track from the OpenSDRM Media Distribution server.

2.1.3 License download

To access to the governed digital item the user needs to have a MPEG-4 IPMP-X enabled player (that can be obtained on the Music-4You portal), and the WRMiM installed and running and the appropriate protection tools needed to gain access to the content and to the license. If the governed content item needs a different set of protection tools they are automatically downloaded and installed (2) on the player when the user starts listening to the music (Figure III.8.4).

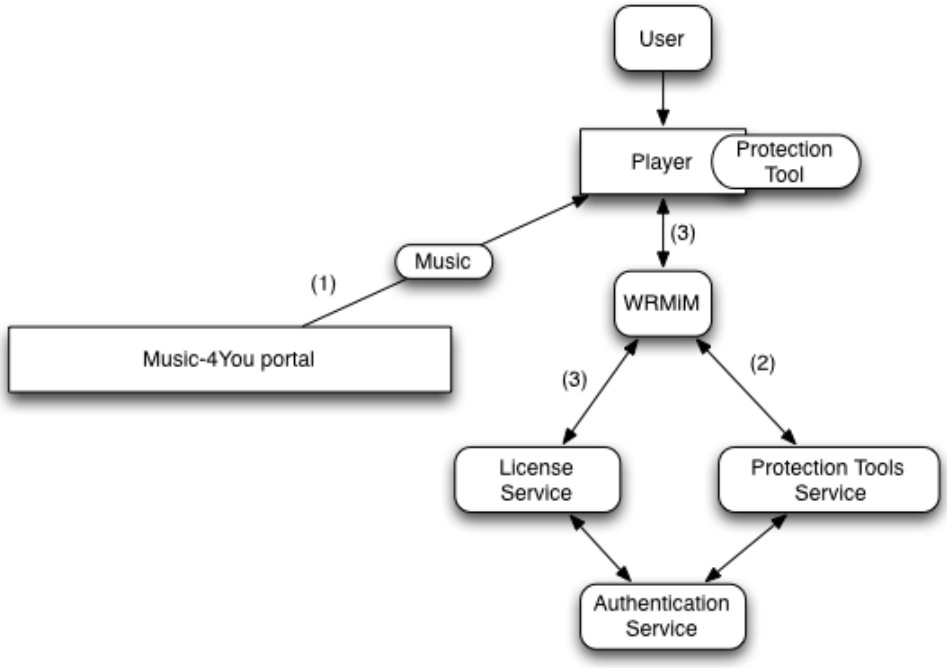

Figure III.8.4: License download

The user downloads the digital item from the Music-4You portal (1) and tries to open it with the player. The player checks that the music is protected and governed, retrieves its unique identifier and connects to the WRMiM requesting the appropriate license (3).

WRMiM can proceed in two different ways: either WRMiM has already a license stored for that content, it returns that license to the player and the music is rendered. If not, WRMiM contacts the OpenSDRM License Server and asks for the license that corresponds to the user and the content. If this license exists, it is downloaded by WRMiM and passed to the player that renders the content. If an appropriate license cannot be found, or if the license has already expired, the content is not rendered and the player warns the user about this fact.

2.2 Content providers functionalities

Music bands can be registered on the Music-4You portal creating automatically a personalised webpage containing band relevant information. The main functionalities provided by Music-4You to content providers are: content uploading and management and Band information registration and management.

2.2.1 Content uploading and management

A music band can be registered on the web-site and upload music to its personalised web-site (Figure III.8.5, Figure III.8.6). The band administrator can choose either to make the music available for free for the portal users or he may choose to used OpenSDRM to govern it. When the content is uploaded (1) it is automatically categorised (2, 3) and tagged (4).

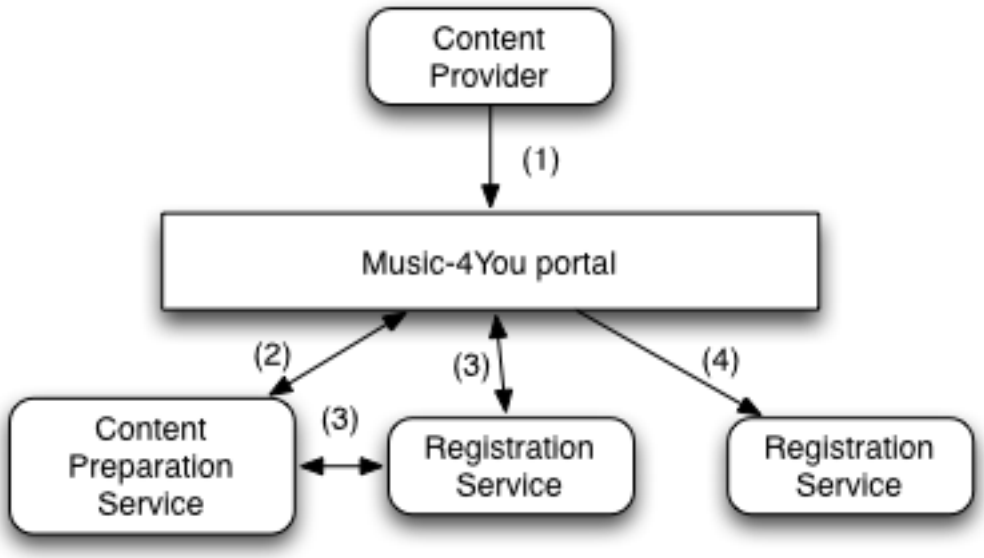

Figure III.8.5: Content uploading and registration

The Music-4You portal also allows the band's administrator to manage the band's content, editing the music title and further details.

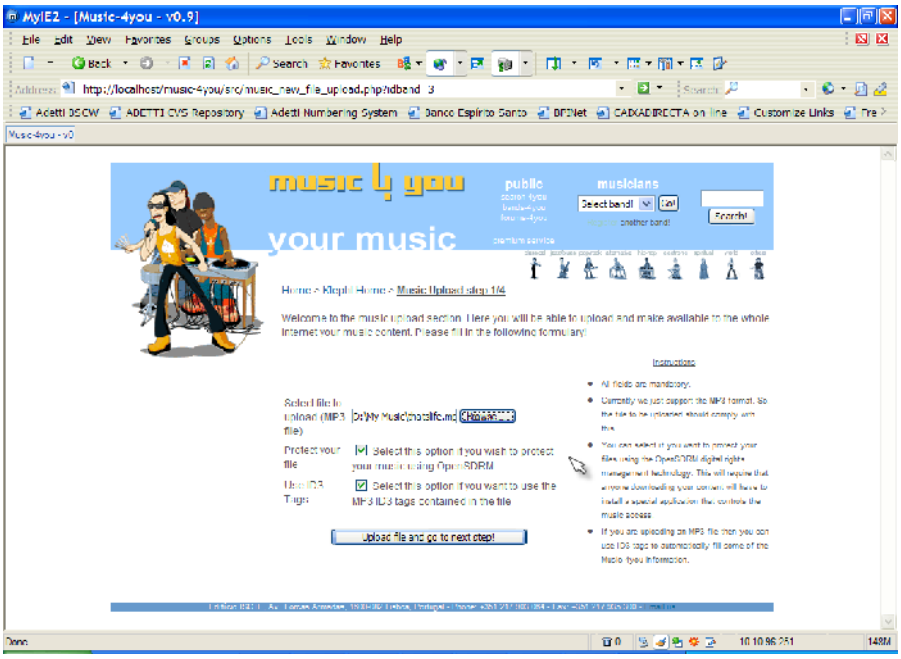

Figure III.8.6: Content uploading and registration at the Music-4You portal

2.2.2 Band information registration and management

Music-4You portal allows the registration and management of information related to the band news, information its agenda (public appearances, records releases, etc.), contact management (the contact point of the band that can be used to hire the band for concerts), links, discography, members and pictures. This information is available on the portal for the general public, and is managed by the band administrator registered on the portal.

3 Controlled access to video-surveillance data

The OpenSDRM system was also used on another RTD European project called WCAM[21] (Wireless Cameras and Audio-Visual Seamless Networking). The objective of the WCAM project was to study, develop and validate a wireless, seamless and secured end-to-end networked audio-visual system [Serrão et al., 2003d]. WCAM aimed to exploit the technology convergence between video surveillance and multimedia content distribution over the Internet. WCAM considered aspects such as real-time implementation, security of the delivery and scalability. The video content was encoded in emerging content formats: Motion

21 http://wcam.epfl.ch

JPEG 2000 and MPEG-4 AVC/H.264, and transmitted through Wireless LAN to different types of decoding platforms like PDA's and Set Top Boxes. The content is streamed on the network using the RTP protocol over UDP/IP.

Most of the governed digital content scenario was focused on the Motion JPEG 2000 format that is a basically a sequence of still JPEG 2000 images: there is no temporal compression involved like in MPEG video formats.

JPEG 2000 is a recent international standard developed by the Joint Photographic Expert Group, JPEG [JPEG2000, 2000a] [Taubman and Marcellin, 2002]. It defines an image compression system that allows great flexibility not only for the compression of images but also for accessing data in the code-stream. A key feature of JPEG2000 is the flexible bit stream representation of the images that allows to access different representations of images using its scalability features (resolution, quality, position and image component).

RTP (Real-time transport protocol) provides end-to-end network transport functions for applications transmitting real-time data, such as video data, over multicast or unicast network services. The data transport is augmented by a control protocol (RTCP) to allow monitoring of the data delivery in a manner scalable to large multicast networks, and to provide minimal control and identification functionality [Schulzrinne et al., 2003].

3.1 OpenSDRM in WCAM context

OpenSDRM was used to govern the WCAM protected video-surveillance data. In this section, the OpenSDRM components and its interaction with the WCAM platform are introduced and described

In order to implement the WCAM scenarios some of the components had to adapted while others had to be developed from scratch. This is the case of the specific modules that handle content specific operations, such as the the Content Preparation Server (CPS) and the Player.

3.1.1 WCAM content preparation

The OpenSDRM Content Preparation server (CPS) is responsible for WCAM content preparation and protection. It was used to perform all the required steps to receive raw

content as input and to produce as output any specific format that is consistent and coherent with the requirements of the platform being served (H264 and Motion JPEG-2000, in WCAM case).

WCAM CPS content production operation involves receiving raw video format feed from a given video capture device, adding metadata and protecting it, thus preparing the content to be injected in the OpenSDRM platform [Serrão et al., 2003d].

This component plays an important role on the registration of the content and associated metadata, as well as the active protection of the same content. The following image (Figure III.8.7) demonstrates how the CPS interacts with other components of the system.

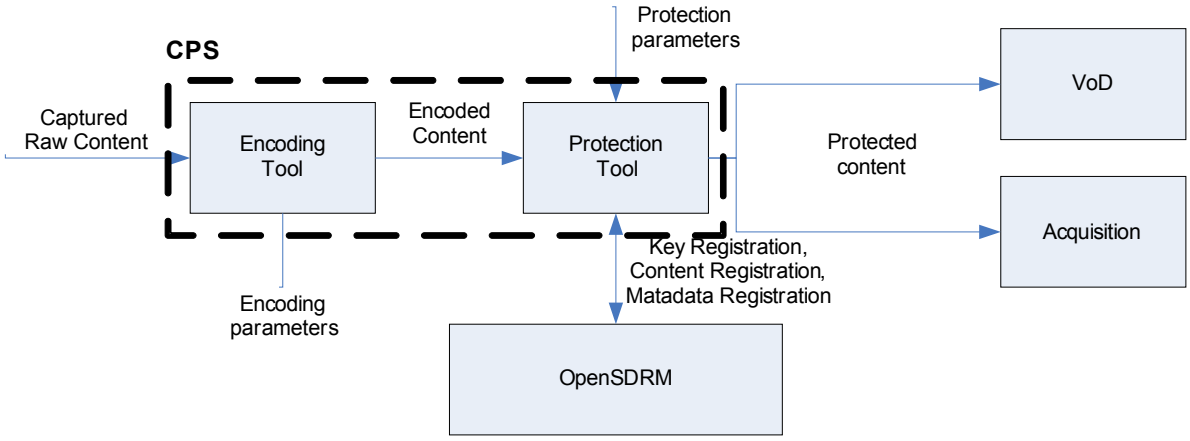

Figure III.8.7: Content preparation and protection

3.1.2 Select and access content

In the WCAM scenario, the users select the content through a server component that responsible for listing and displaying the content available and registered on the OpenSDRM platform.

The users are able to navigate, access to relevant metadata, and request access to the available content. Depending on the user credentials different clearance access levels are provided to the user. This credentials identify the type of user on the system and allow the production of a license that defines the type of access that the user will have to the content (Figure III.8.8).

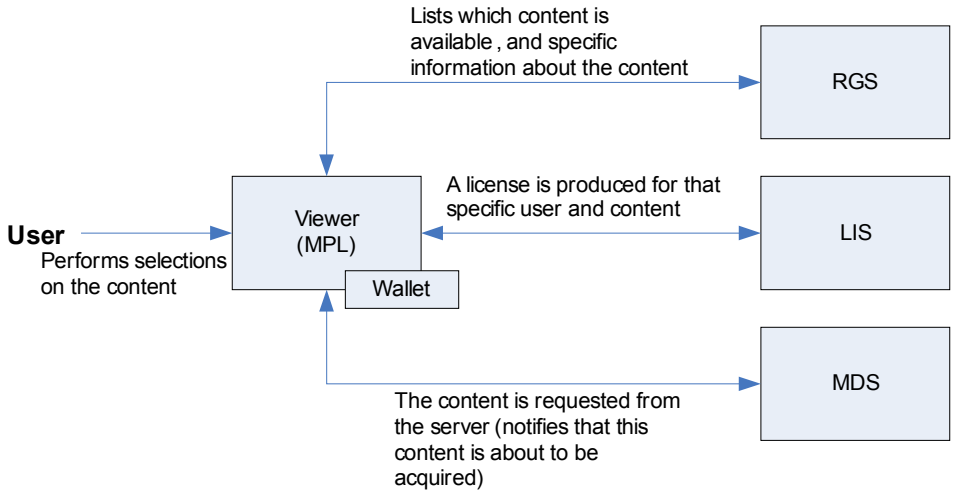

Figure III.8.8: Player interaction while requesting the content

3.1.3 Delivering content to the user

The OpenSDRM Media Distribution Server is responsible for keeping track of all registered content on the platform. MDS stores the location (URI) of all registered content (Figure III.8.8). MDS thus remains independent of content and delivery types (JPEG2000 streams and RTP).

3.1.4 Content registration

In WCAM scenario, the OpenSDRM Registration Server (RGS) is used to assign unique identifiers to the content that is governed by the DRM platform, as well as to register and keep metadata information for that specific content. Ultimately it is responsible for keeping a bullet-proof consistency in terms of content identification, assuring no ambiguities in protected content management (Figure III.8.8).

3.1.5 Users identification and authentication

The OpenSDRM Authentication Server (AUS) is used on the WCAM scenario to register and define the different user roles on this system. These different user roles allow the definition of different access rules to the system.

The WCAM users will have either full access to the governed content or some defined limited access just parts of the content or access to segments of content.

3.1.6 Defining and using licenses in WCAM

The OpenSDRM License Server (LIS) is the server component that responsible for storing and maintaining the rules associating WCAM governed content, an end-user, and his corresponding access rights. The LIS is also responsible for the establishment of license templates associated with particular types of content or business models. This component accepts connections from authenticated WRMiM components for the downloading of licenses, which will be applied to the protected content through an appropriate protection tool.

In WCAM, several license templates are defined according to the different video-surveillance content access scenarios and the type of user. When selecting and accessing to the governed video-surveillance content, the user credentials are analysed and the appropriate license template is set-up, allowing the user to access the content through his media player, according to the established conditions.

3.2 WCAM JPEG 2000 integrated DRM content protection

To protect the video surveillance data, WCAM uses a JPEG2000 scalable encryption approach that is integrated with the OpenSDRM framework [Serrão et al., 2003d].

One of the major enhancements provided by OpenSDRM to the WCAM scenario is the possibility to define under which conditions the governed video-surveillance content can be used, and also the possibility to enforce such conditions. These conditions may vary according to the type of content, the user, the device or many others.

The WCAM licenses allow the specification of the user and content identification, decryption key(s), number of usage and validity period. The player will temporarily remove the protections corresponding to the keys provided in the license.

JPEG2000 video streams can be protected in various ways, using this method. These are the two main approaches for ensuring confidentiality:

- The JPEG2000 code-streams are partially encrypted with one key: some parts of the code-stream are encrypted while others are not; in this case the same key is used to

encrypt the parts of the code-stream. In this situation, the user could be allowed to access freely a low resolution version of the image (whose corresponding packets are not encrypted), while the higher image resolutions are encrypted because they have some kind of value. The user is only allowed to access these higher resolutions levels with the appropriate license;

- The code-stream is partially encrypted with different keys: some parts of the code-stream are protected, and these parts are protected with different keys. This means that different users might have different license levels to access different image parts. This situation allows a more flexible solution, since different business models can be deployed and used in this case.

As part of the proposed security solution, each of the protected code-streams will need some piece of information indicating which protection was applied and how the protection tools can be obtained. This signalling information is necessary for the user in order to be able to properly decrypt or process the JPSEC-protected data.

The following images (Figure III.8.8, Figure III.8.9) display protected JPEG2000 frames illustrating this approach. The image contains three decomposition levels. Its two higher levels are encrypted with `Key1`, hence its blurry appearance. Additionally, the face zone has been encrypted with `Key2`, for all decomposition levels: that part of the image is completely encrypted and one cannot guess the hidden face.

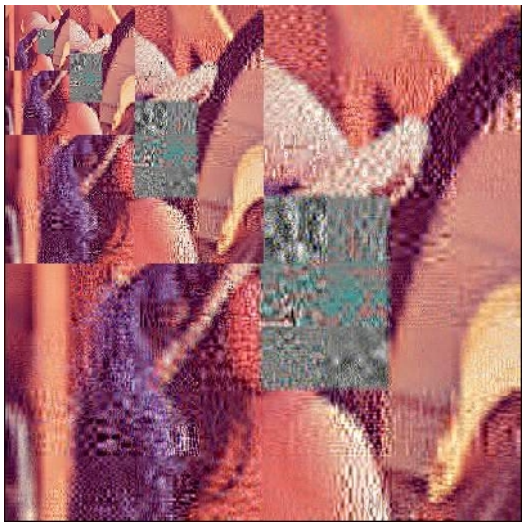

Figure III.8.9: Encrypted image at all resolution levels

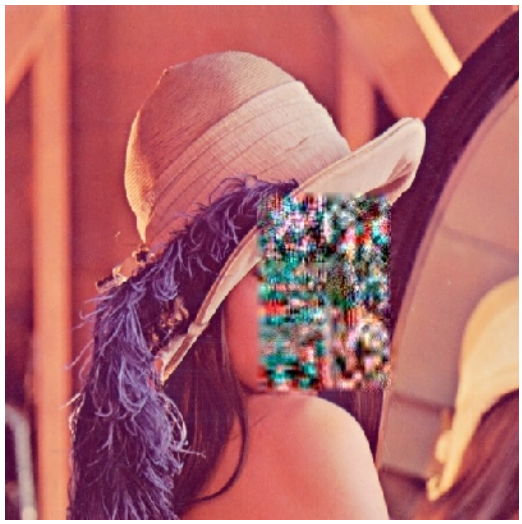

Figure III.8.10: Partially encrypted image – only the face is encrypted

- Figure III.8.8 shows the encrypted image at all four available resolutions. The lowest resolution is in clear, except for the face. However, a user won't be able to access the detailed image without `Key1`.
- Figure III.8.9 shows the partially decrypted image. In that case, the player has been granted a license with `Key1`, meaning that the user can access the highly detailed image, with the exception of the face. In WCAM, this feature was used to add protection levels for very sensitive data. For privacy reasons, a face can often be blurred or encrypted in video surveillance applications.

4 Controlled access to large Earth observation products

Another usage scenario for the OpenSDRM platform was the HICOD2000 project. HICOD2000 was a project conducted with the European Space Agency (ESA) to study the applicability of the new image encoding standard – JPEG2000 - to Earth Observation (EO) products [JPEG2000, 2000a]. In this project, a project was developed to ESA allowing, not only the encoding and decoding of EO products, but also allowing ESA to securely trade and manage the rights of these products over the Internet on a specific portal [Serrão et al., 2005f][Serrão et al., 2006a].

The work performed was mostly based on EO data captured by the ENVISAT [ENVISAT, 2002] and SPOT5 satellites. The ENVISAT satellite contains a payload of eight measuring instruments that store the measurement data in raw format, and send it back to ground stations at specific locations in their orbit path. This raw measurement data is afterwards combined to form different EO product levels. Each of the products is currently coded in a ESA proprietary format called Payload Data Segment (PDS). This data format does not feature any compression and so can usually result in quite large product files, not very suitable for Internet transfer.

Each PDS coded product is further subdivided into some distinct parts mostly indicated as headers and data-sets. The headers and the data sets in a PDS file store a large number of parameters with different data types. The parameters that are JPEG2000 compressible candidates are in the data sets of each PDS product, while the remaining components of a PDS product can be safely regarded as meta-data that does not need to be subject to JPEG2000 compression but that is still included in a JP2 file [JPEG2000, 2001]. The advantages of coding PDS data with JPEG2000 come not only from achieving good compression ratios but also from using a standard coding format that is recognised worldwide and that has a number of interesting features for scalable coding of data with progressive quality improvement [Serrão et al., 2006a].

One of the other major requirements of the HICOD2000 project consisted in the development of a new security technology to protect the EO products.

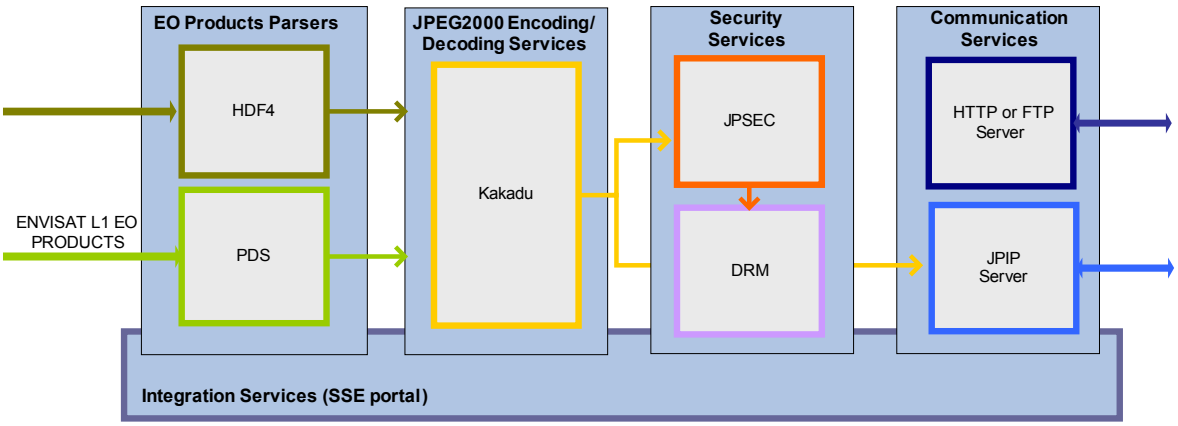

Figure III.8.11: Architecture of the HICOD2000 black-box (H2KBB)

HICOD2000 studied the possibility to integrate a better security model and at the same time

empower ESA, or ESA EO product retailers, to develop new business models based on this new paradigm. HICOD2000 designed the solution based on one of the parts of JPEG2000 standard, called JPSEC [JPEG2000, 2001] [JPSEC, 2004], which deals with code-stream protection. Therefore, a method for ciphering the EO product according to its resolution levels, integrating a DRM solution [Serrão et al., 2003b] to control the user access granularity to each of the EO product resolutions, was developed. Using this solution ESA could profit in terms of flexibility of its traditional business model, and allow the price differentiation of its own EO products according to the resolution level selected by the client [Serrão et al., 2003b]. The paying and registered client can use a viewer developed for the purpose to visualise the JPEG2000 EO products and that can allow the final user to access the image resolutions that he have paid for and also to recover the original product format [Serrão et al., 2006a].

4.1 JPEG2000 EO products coding and decoding

The HICOD2000 system created an environment for seamlessly converting between EO native file formats (PDS and HDF) to JPEG2000 and to perform the reverse conversion. The system defined a self-contained set of components that interact with each other to provide the conversion functionalities, as well as a set of additional functionalities:

- Security and Access Control;
- Integration Mechanisms;
- Access Interactivity.

This set of components, integrated, formed the HICOD2000 black-box (H2KBB). The H2KBB (Figure III.8.11) could be used either in stand-alone operation or integrated with other ESA services or applications, since it provides a public interface and an API that allowed it to be invoked from the outside [Serrão et al., 2006a].

Two of the H2KBB core sub-systems are the EO products parsing and JPEG2000 encoding [JPIP, 2004]. In a very generic way, the functionality of the EO products parser consisted in extracting from the original products format the JPEG2000 data and metadata to be encoded, as well as some additional information to be used in the original product

reconstruction.

4.2 JPEG2000 EO products protection

One of the requirements of the HICOD2000 system consisted in the capability of offering granular protection to JPEG2000 EO products – the capability to offer strong protection [Serrão et al., 2002] to the global EO product as well as the capability to protect just some parts of the product while others remained in clear.

The emerging JPEG2000 – Part 8 [JPEG2000, 2001], called JPSEC [JPSEC, 2004], aimed at the development of a protection scheme for JPEG2000 code-streams was the selected technology to protect the EO products. The HICOD2000 project specified the protection granularity of the EO products at the resolution level, up to a maximum of 6 different resolution levels [Serrão et al., 2003b].

Each of the JPEG2000-encoded EO product resolution level was ciphered with a different key using the AES (OFB mode) algorithm [Hongjun and Ma, 2004]. The JPEG2000 code-stream was properly signalled so that even the ciphered code-stream could maintain its integrity to be readable by any JPEG2000-compliant viewer [JPSEC, 2004].

The developed system had also the capability to control and enforce the user and content access control to protected EO products using a rights management system (OpenSDRM) [Serrão et al., 2003d].

The H2KBB was integrated with OpenSDRM using the publicly available WSDL interfaces [Serrão et al., 2003b], to provide DRM functionalities such as: user authentication, content registration, content encryption keys registration and license management.

Therefore when H2KBB is converting an EO product to JPEG2000 format a parallel DRM process is also triggered: the OpenSDRM registers the EO product, assigning a unique identifier that is inserted inside the JPEG2000 EO product. This identifier can be used after to establish the connection between two elements that allow the download of the necessary content keys to access the clear product – the user and the EO product [Serrão et al., 2003b].

During the H2KBB EO products protection, the content encryption keys are established and

registered on the OpenSDRM platform – a logical connection between the EO product unique identifier and the content keys.

Whenever a final user selects a product from the EO portal, and upon the payment is made, the OpenSDRM platform generates a license establishing specific usage conditions and containing the content keys needed to access to the contracted EO product resolution. At the client-side, using specific purpose software, the license rights are uphold on the content and the keys are applied allowing the user to browse the acquired product.

The OpenSDRM platform is not only responsible for managing the content and the rights, but at the same time is also controlling the user access to content itself.

4.3 Integration with ESA portal

One of the major objectives of ESA was the possibility to integrate the HICOD2000 system at two levels: on the service provider side and also at the final user side.

ESA has developed an EO portal[22] capable of integrating several services from different service providers – these could range from simple image provision to data conversion services. The H2KBB and the OpenSDRM platform had to be integrated with this portal. All these services were available at one unique point. This was accomplished by supplying each of the service providers with a standardised way to integrate their specific service with the EO portal (SSE portal). This mechanism is the SSE Toolbox – a software tool distributed freely by ESA to any service provider that allows it to be able to define their service business model and the specific inputs and outputs – using a descriptive language defined in XML – WSDL. The Toolbox, upon the definition of the service, creates the necessary data to be defined at the SSE portal when registering the service. These files allow the creation of an automatic business form that allows final users to specify the parameters for the service – in the specific case of HICOD2000 parameters are:

- EO product identifier: this is the ESA product identifier that identifies the EO products at the service provider side;
- User identifier: this is a unique user identifier assigned by the DRM platform;

22 http://services.eoportal.org/

- Order identifier: this value is generated automatically by the SSE portal when placing the user order on the system;
- Delivery Method: this is a selection field that allows the user to select how the final JPEG2000 EO product is going to be delivered to the user (HTTP, FTP or JPIP).

When the user places an order on the SSE portal, it contacts the SSE Toolbox at the service provider premises that invokes the H2KBB. The H2KBB converts the EO product into a protected JPEG2000 file that is placed on a specific output directory. The service provider SSE Toolbox returns the URL of the newly created JPEG2000 file, upon successful completion. This URL is formed with base on the user delivery method selection. The URL is presented to the user on the SSE portal that accesses the image on the service provider side, using one of the previously selected delivery methods (Figure III.8.12).

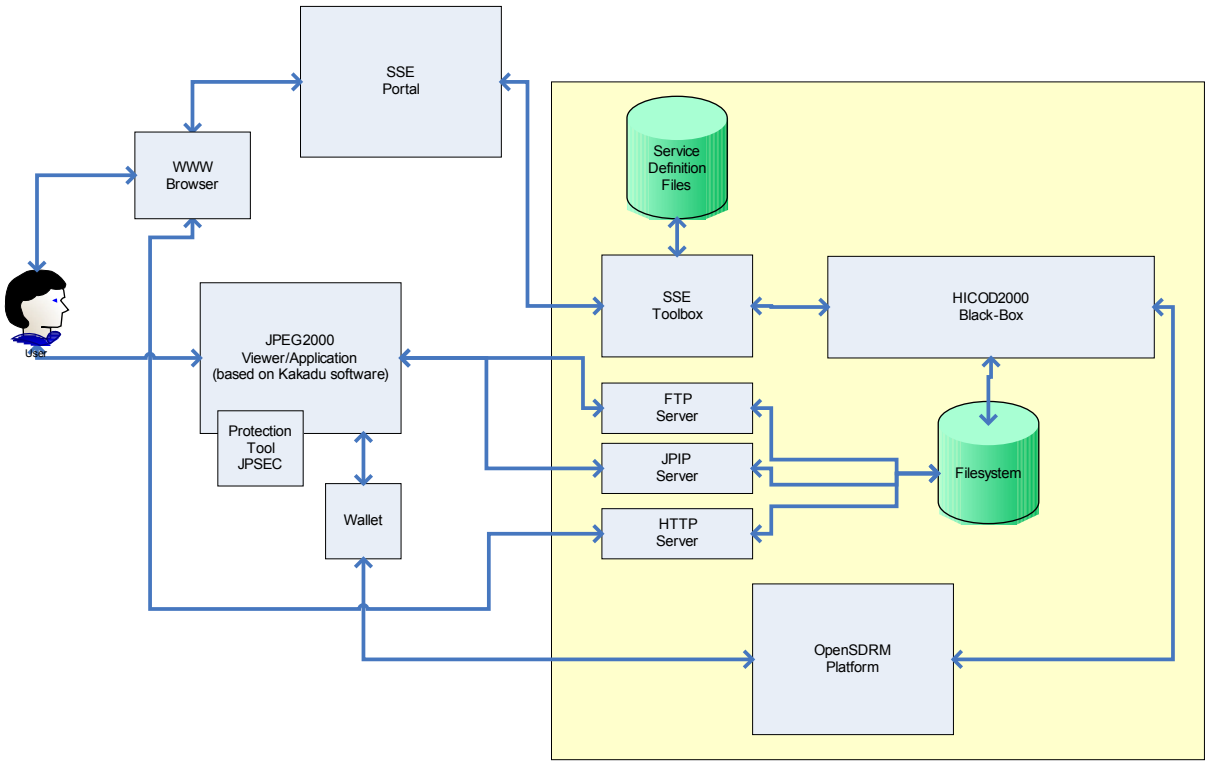

Figure III.8.12: HICOD2000 integrated architecture

At the client side, a specific application is launched allowing the user to navigate the JPEG2000 image. If the image is protected, the viewer requests the user authentication to download the appropriate keys from the OpenSDRM platform and deciphers the image, through the WRMiM [Serrão et al., 2006f].

5 Home network music jukebox

Another significant usage of the OpenSDRM platform was the MediaNet project. The MediaNet addressed the domain of digital multimedia personal communication and content distribution, as well as co-operation schemes between content owners, service providers, network access operators, and telecommunication, computer, components and consumer electronics industries [Travert and Lemonier, 2004]. The objective was to remove the obstacles to the end-to-end digital communications and content exchange, from content/service providers to customers and between persons, over shared broadband access and home network infrastructures at the same time [Serrão et al., 2005c].

Assuming an open system reference architecture model, MediaNet has studied a number of critical constituents of the on-line delivery chain (the e-media chain) [Jonker and Linnartz, 2004], made of various technologies, equipment or services, that were considered as pre-requisite elements for the creation of a myriad of new media services, supplied by multiple providers and vendors in Europe [Serrão et al., 2005c].

The MediaNet project integrated the OpenSDRM architecture in the MediaNet Reference Architecture, in order to provide rights management features for some selected MediaNet Use-Cases, which required DRM support [Travert and Lemonier, 2004].

OpenSDRM is a platform that integrates some critical DRM elements that were developed using a completely distributed architecture [Koster et al., 2006]. These critical elements include: content protection, rights expression, license management, content and metadata registration and payment [Serrão et al., 2003b]. This integration was being performed specially with another platform called Service Enabling Platform (SEP) [Serrão et al., 2005c], a middle-ware architecture which provided the necessary services to home networks applications provided by network operators. One of such services is rights management, provided by OpenSDRM as a SEP external functional capability [Popescu et al., 2004b].

5.1 Integration between OpenSDRM and the MediaNet SEP

In the scope of the MediaNet project, the OpenSDRM platform was used for the first time in the context of Home Networking linked with the Telecom Application Service platform. providing DRM capabilities to several identified use cases [Travert and Lemonier, 2004].

The MediaNet project aimed at enabling a myriad of multimedia applications over a common reference architecture. To define this common reference infrastructure as well as the interfaces between the stake-holders, one or more use cases were proposed by the project partners. These use cases were implemented and used to define and validate the MediaNet Systems Reference Architecture [Serrão et al., 2005c]. Within this Systems Reference Architecture, the n-Services Platform was a software platform aiming at accelerating the introduction of innovative, value-added services by application service providers [Serrão et al., 2005c]. This platform, which –was located in the network access provider domain, exposed generic capabilities, the so-called n-Services, through standardised interfaces and technology. These generic n-Services could be (re-)used by multiple application service providers [Travert and Lemonier, 2004]. The availability of an n-Services platform offered advantages to the network access providers as well as the application service providers. On the one hand, it enabled the network access provider to resell capabilities to multiple application service providers and to charge these providers for usage of the capabilities [Jonker et al., 2006]. On the other hand, the availability of generic capabilities enabled the application service providers to outsource certain generic aspects of their application to the network access provider, while focusing their own developments on the functionality that was specific for their applications. The Service Enabling Platform (SEP) was the instantiation within MediaNet of an n-Services Platform [Serrão et al., 2005c] (Figure III.8.13).

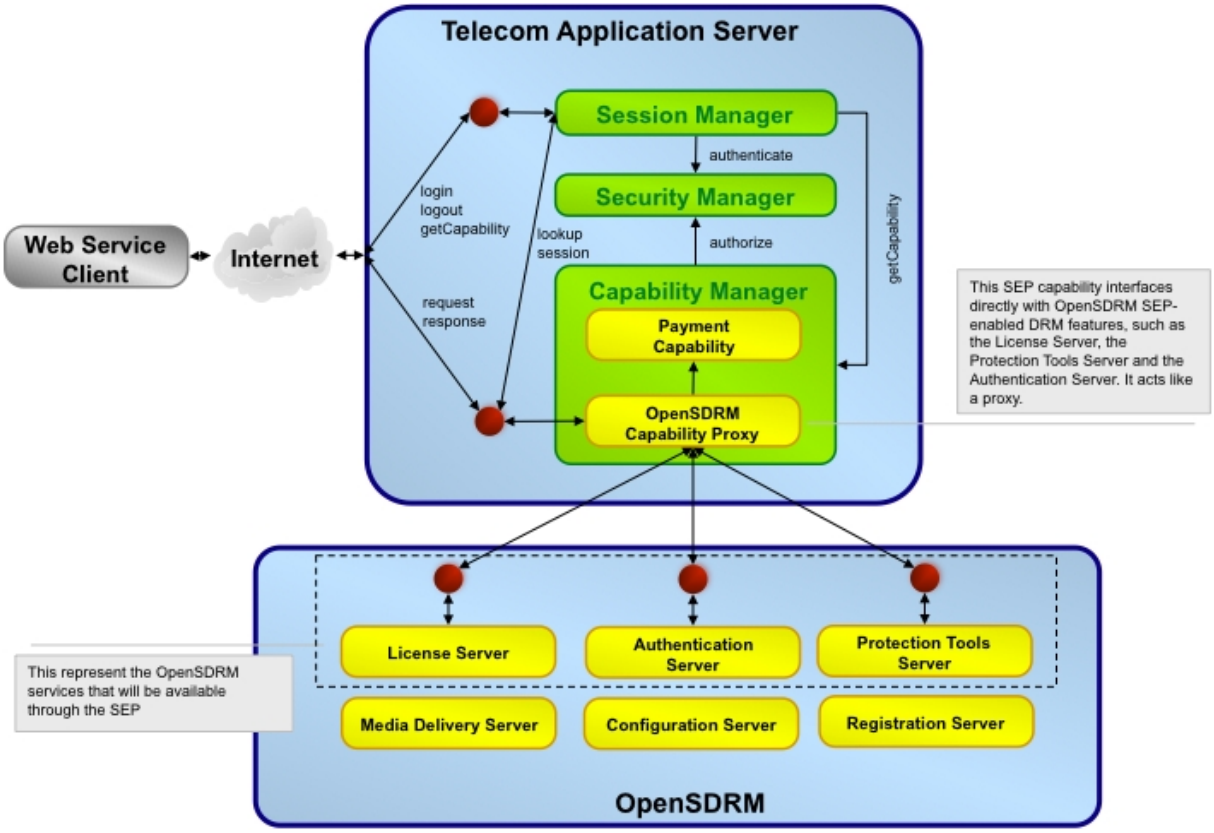

Figure III.8.13: Integration between SEP and OpenSDRM

The SEP was composed by a different set of components, but in terms of integration three of them were the most relevant: the Session Manager, the Security Manager and the Capability Manager. The Session Manager is responsible for establishing and maintaining sessions for clients (e.g. application service providers). The session manager is always active and its reference is registered in a Naming Service. This reference is the entry point for external applications/clients to obtain access towards the services provided by SEP. The Security Manager authenticates users and authorises operations and/or access towards resources based on the user's security profile, managed by external applications. Security profiles can be defined in a static "off-line" way or through a security administration client. The Capability Manager consisted of a capability repository, an interface to browse or discover supported capabilities and a register for capability addition/removal/change notification subscription. One of the capabilities that will be referenced by this Capability Manager will be the DRM support of the OpenSDRM platform.

Any client application and user on the Home Network is required first to establish a session with the SEP at the Telecom Application Server. This is the first entry point of any SEP-

enabled application. The SEP authenticates the user or application through the Security Manager and assigns a unique session identifier to it. After this process the client can request capabilities from the SEP.

One of the capabilities that may be requested by clients and application is DRM, provided by OpenSDRM, which is considered an external capability of SEP. The integration of both is made through the usage of an OpenSDRM Capability Proxy (OCP) which redirects the requests made by SEP clients to the corresponding service endpoint in the OpenSDRM platform. The OpenSDRM services which are directly connected to SEP and using this OCP, are the License Manager service, the Authentication service and the Protection Tools service. Whenever the client or application is trying to access to DRM-protected content, it connects to the SEP platform, authenticates and requests the DRM capability, to be able to access to the corresponding license to obtain the clearance to access to content. The capability reference is then returned to the client application, which in turn can connect to the services provided by OpenSDRM, through the SEP OCP. All the messages exchanged between the client application and the SEP, as well as the ones exchanged between the SEP and the OpenSDRM platform, are SOAP-based.

Using the integrated Service Enabling Platform and the OpenSDRM, it was possible to develop an home network Music Jukebox application (Figure III.8.14) that allowed to user to access to external governed digital content, acquire and enjoy it on the user's personal home network.

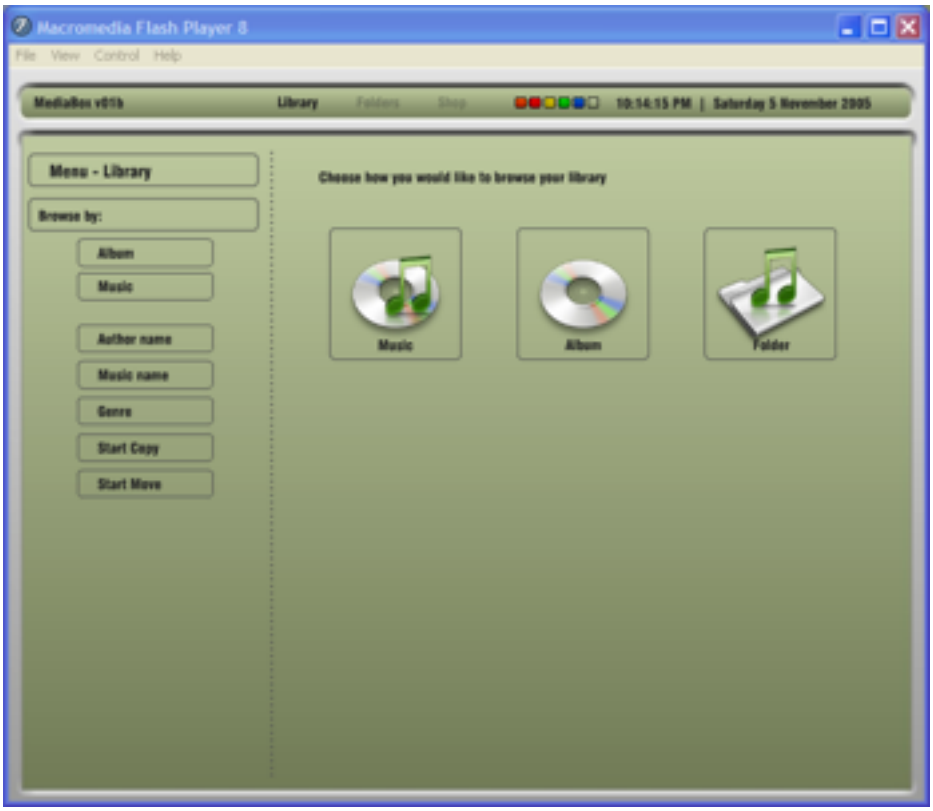

Figure III.8.14: MediaNet Music-Box, integrating the Services Enabling Platform and OpenSDRM

6 Conclusions

This chapter presented several use cases, where the OpenSDRM rights management was used and tested. It was presented a selection of four use cases, that demonstrated how OpenSDRM could be customised and adapted, supporting a multiplicity of content types and business models.

In this chapter it was possible to observe that most of the components from OpenSDRM can be used from use-case to use-case, and that there are some components that require some adaptation. The two components that require adaptation are the Media Player and the Content Preparation Server, because they are directly connected to the content type being used and the protection methods applied.

Part IV. Conclusions and Future Work

Chapter 1. Conclusions

1 Introduction

The digital World has introduced many important advantages for users in general but, at the same time, has created many challenges. In particular, on what concerns the digital content Intellectual Property (IP), the challenges are more than many, and need to be addressed in a proper manner. These challenges have been addressed, both legally and technologically [Arkenbout et al., 2004].

Legislation has been created, both by the European Union and the United States of America, mostly as a way to attempt to stop the growing number of IP violations in the digital content World. Also, from the technological point of view, rights management systems, known as Digital Rights Management (DRM), were created and deployed has a way to try to stop these digital IP violations [Bechtold, 2006].

From a technological point of view, the DRM solutions have worked up to a certain extent [Lesk, 2003]. These solutions were more or less effective, in sense that they have raised barriers to digital IP infringements at least less technically skilled users, that could not break the governing mechanisms they offered [Bradbury, 2007]. But, due to their closed and vertical nature, they started creating a sense of aversion on common users and even on content providers.

This sense of DRM aversion has many roots. Perhaps one of the most important ones is the incapacity of those DRM solutions to offer the same kind of content user experience by opposition to their analogue counterparts. It is not easily acceptable to the user, that the digital content that he has legally acquired (and paid for) can only be played on a specific

computer and on a specific type of device. It is typical of these DRM solutions to support just one player and just one device type. Current available rights management solutions have an interoperability problem. This problem was precisely the main addressed topic in this thesis.

The DRM interoperability problem is a vast and complex problem, that needs to be addressed from several different perspectives [Geer, 2004]. It is not the goal of this thesis to solve all the rights management interoperability problems. Instead, this thesis and its author present a study about some approaches and some mechanisms that could be used to address and solve, partly, the interoperability issues that trouble current DRM systems. These proposed mechanisms address also the future rights management systems, that need to be interoperable by default.

The work conducted on this PhD work has produced contributions in the following fields:

- The **emergence of the new service-oriented architectures** (SOA) as a catalyst for rights management interoperability;
- The **usage of PKI mechanisms as a way to establish interoperable trust environments** between the different rights management systems;
- Using **open rights management systems** as an approach, based on open-specifications, public interfaces and open-source, to promote the interoperability between different rights management systems;
- The **establishment of appropriate common mechanisms for the secure management of licenses and keys** within different rights management systems and analysing how this is handled by different systems;
- The **study, design and implementation of an open rights management platform** that has the potential to interoperate with other platforms, and that implements the different aspects uphold on this work;
- And the **study, design and implementation of a client-side rights management interoperability middle-ware** that allows the coexistence of multiple content rendering applications.

This thesis also proposes an open rights management platform that implements the different interoperability mechanisms and principles described on the thesis. This rights management platform was and continues to be used as a testbed for the implementation of governed digital content business models.

Therefore, this thesis, which work has officially started on 2003 (although the author had already some important experience on the field), presented some contributions to some standardisation initiatives and other joint international rights management related efforts. Also, the work developed on the context of this thesis has been published and disseminated through several refereed publications and communications.

Another interesting contribution of this thesis is the open-source rights management platform OpenSDRM [Serrão et al., 2003b]. This platform, developed in the context of this work, was contributed to the open-source community, and can be downloaded for free from the SourceForge.net[23] site, where it can be further developed and evolved.

2 Conclusions

This section presents the different specific conclusions of this work. For better understanding, the conclusions were divided according to the different contributions presented in this thesis.

1. Rights Management and Service-Oriented Architectures

The advent of Web Services is having an high impact on the Internet, pushing it forward to give birth to a network of services [Denaro et al., 2006]. The emergence of a Service-oriented Architecture (SOA) paradigm was the natural evolution, opening the way to more specialised and independent digital service providers. SOA has an important impact on the distributed programming field. The changes introduced by the SOA approach have also an important effect on rights management systems that can benefit from open and distributed service implementation and deployment as a way to improve service standardisation and interoperability [Jamkhedkar et al., 2007].

The SOA approach, as presented in this particular contribution, has the potential to create

23 http://sourceforge.net/projects/opensdrm/

heterogeneous rights management scenarios where different rights management service providers, through standardised and open interfaces, can establish co-operation scenarios. The presented contribution defends that is possible for rights management providers the de-coupling of their tightly integrated and vertical solutions into self contained services that expose their functionalities through a well described public interface [Serrão et al., 2005c].

Using the proposed approach, the different rights management providers can publish their services in UDDI repositories providing a description of their provided functionalities. This way, any content provider trying to make a digital governed content business model can select and integrate the services that best fit his needs. The UDDI repository, becomes a repository of rights management services, used to build truly open, standardised and interoperable rights management solutions [Serrão et al., 2005c].

SOA is definitely an interesting approach to the rights management. This contribution to rights management interoperability was implemented and tested in an open rights management platform that was implemented in the context of this thesis, called OpenSDRM (see Part III, Chapter 6).

2. PKI and rights management interoperability

Rights management systems are highly dependent from their trust establishment capabilities, which depend on the usage of cryptographic material. The effort to efficiently manage rights on digital objects would fall apart without the appropriate security mechanisms [Taban et al., 2006], and above all without the capabilities to establish trust relationships between the different elements that compose digital rights management solutions – actors, devices and software components [Arnab and Hutchison, 2007].

This work reached the conclusion that most of the current rights management solutions do not rely on already existing PKI services or vendors for that operations. Rather they implement their own security services, making rights management implementations more complex and non-interoperable. Rights management solutions perform PKI complex and time-consuming operations rather than focus on rights management [Serrão et al., 2007a].

In this field, the proposed contributions presented different novel approaches to handle the rights management interoperability issues from a trust establishment point of view. These

contributions allow the achievement of common trust environments between different non-interoperable solutions, making usage of existing PKI mechanisms [Serrão et al., 2007a].

The contributions presented here also included two different approaches for interoperability through PKI services were presented. One is based on the existence of a single PKI solution for all the available rights management while the other is based on the existence of a PKI broker capable of managing trust between different rights management PKI implementations [Serrão et al., 2006h].

As a conclusion on this contribution, an approach to the DRM interoperability between two different selected open rights management platforms was also presented. Both the registration and authentication processes were described and analysed, and commonalities and differences between them were identified. Based on this analysis and on the previous contributions on trust interoperability through PKI, an interoperable architecture was proposed to accommodate both rights management platform's security and trust requirements.

3. Open rights management towards interoperability

In this work it was possible to reach the conclusion that all the commercial rights management solutions present on the market are closed and vertical. This is one of the main sources that disable rights management interoperability. The alternative is to develop rights management solutions that follow an open model [Bar-El and Weiss, 2004].

Another conclusion on this topic is that there is a growing number of open rights management solutions that were presented and described in this thesis. These open rights management solutions include both open-source based rights management solutions as well as solutions that do not endorse the open-source software models but provide open specifications and or open interfaces.

Closed rights management systems will never be interoperable. Only the open model presented in this work will offer the conditions that need to be met to have interoperable rights management.

The major conclusion to retain from this part of the work is that a crucial approach has to be

followed in order to obtain truly interoperable rights management systems. Rights management systems should follow an open model to create the necessary mechanisms to provide interoperability points. From a simple SWOT analysis is easy to conclude that the open rights management model, although challenging, specially from the security point of view, presents many advantages over the traditional and actual closed model [Serrão et al., 2008a].

4. Secure key and license management for open rights management

Security is central to rights management on digital systems [Arnab and Hutchison, 2007]. Appropriate secure management of both rights representation and key material are of extreme importance [Taban et al., 2006].

In this specific topic of the work, different scenarios about the usage of rights expression in the rights management context, were identified and described. This scenarios describe the way digital rights expression languages [Guth, 2003b] are used for digital object licenses expression, creating a relationship between the presence of the content encryption key inside the licenses and the presence of such licenses inside the digital objects. From this work, six different scenarios have resulted, and the most relevant one (implemented in the most significant rights management solutions today) has been selected and the license management life cycle was also described [Serrão et al., 2007b].

Another important conclusion of this part of the work was the identification and description of the major processes in the selected scenario of license management life cycle model and the basic security procedures that make the license management processes effective on the digital objects rights management. Crucial aspects such as confidentiality, integrity and authentication are of extreme importance and therefore need to be used with care to offer trust across the entire license management life cycle [Serrão et al., 2007b].

Key management plays an important role in cryptography as the basis for securing cryptographic techniques providing confidentiality, entity authentication, data origin authentication, data integrity, and digital signatures. Therefore, systems providing cryptographic services, such as rights management solutions, require techniques for initialisation and key distribution as well as protocols to support on-line update of keying

material, key backup/recovery, revocation, and for managing certificates in certificate-based systems [Rafaeli and Hutchison, 2003].

Modern rights management solutions still lack some key management aspects. Although all of the analysed open rights management systems had some kind of key management mechanism considered or implemented, these mechanisms did not implemented the full key management life cycle. In most of the cases analysed only the pre-operational and part of the operational stages were considered.

Rights management solutions should consider the different aspects of key management as a way to reduce potential flaws and security risks. This aspect should be considered on the design of such rights management solutions [Serrão et al., 2007c].

5. OpenSDRM open rights management architecture

In this topic of the work, it was designed, specified and implemented an open rights management platform. This platform is based on a service-oriented approach to promote the integration and interoperability with other services and implements an end-to-end model that can be used to test and implement governed digital content business models, and its integration with other open rights management systems.

This open rights management platform (OpenSDRM) is composed by a set of entities, actors and services that were described and their interactions detailed. Each of these services provide an open interface based on a service-oriented approach to allow their internal and external integration [Serrão et al., 2003b]. This approach allows the interoperability between the different OpenSDRM service providers, in order to build end-to-end digital content business models. The main OpenSDRM architecture security mechanisms were also presented and detailed as well as some aspects of the security protocols that need to be handled by the OpenSDRM inner processes.

6. Wallet rights management interoperability middle-ware and license templates

Two different contributions were presented on this topic. The first was the establishment of a client-side DRM middle-ware to provide interoperability between different DRM-protected content rendering applications (CRA). The second refers to the creation and usage of license

rights templates to offer rights management interoperability between multiple content providers, licence providers and end-user devices [Serrão et al., 2006b].

The existence of a client-side interoperability middle-ware is important to provide the different CRA installed on the end-user device the capability to interoperate with different governing regimes. The objective of this middle-ware application is to provide an abstraction layer between the CRA and the rights management processes, which are specific of each of the different governing regimes.

This also contributed to the development of a way of requesting authorisations to the rights management layer to perform operations over the DRM-governed digital content. The designed mechanism includes the registration of all the CRA installed on the end-user device, the establishment of trust relationships between the CRA and the application middle-ware and the definition of a specific language to request permissions to the application middle-ware [Serrão et al., 2005a].

Another contribution of this work refers to the establishment of a mechanism that aims to facilitate the definition of rights by Content Providers. This mechanism allows the different Content Providers to express their business model to the different License Providers, and to establish license rights templates, which can be instantiated in the future, facilitating the design and implementation of new digital governed business models.

The contributions presented in this work aim to improve the interoperability between the different DRM applications and the different DRM stake-holders. Both contributions represent a step forward in the proposal of technical solutions that address directly the interoperability problems faced by current rights management technologies [Serrão et al., 2006b].

7. OpenSDRM use-cases

Finally, in this particular aspect of the work, it was possible to see how the open rights management solution developed can be easily adapted and used to establish a multiplicity of business models.

OpenSDRM, with small adaptations (that are dependent of the type of content) on the media

player and the content preparation system, can be used to establish different business models, such as the examples that were provided.

3 Publications

The research presented in this thesis has been validated against the specialised research community in the context of relevant scientific conferences and publications, as well as standardisation initiatives. This section presents the publications that result from the research that was carried out during this work.

3.1 Refereed Publications

These are the publications resulting from the contribution of results related to this work to international conferences and workshops. They are related to the different contributions of the carried work.

Rights Management and Service-Oriented Architectures

Serrão C., Dias M., Delgado J., "**Using Service-oriented Architectures towards Rights Management interoperability**", in Proceedings of the International Joint Conferences on computer, Information and Systems Sciences and Engineering (CISSE06), University of Bridgeport, USA, 4-14 December, 2006 [Serrão et al., 2006c]

Serrão C., Fonseca P., Dias M., Delgado J., "**The Web-Services growing importance for DRM interoperability**", in Proceedings of the IADIS International Conference WWW/Internet 2006, Múrcia, Spain, 5-8 October, 2006 [Serrão et al., 2006d]

Serrão C., Dias M., Delgado J., "**Using Web-Services to Manage and Control Access to Multimedia Content**", in Proceedings of The 2005 International Symposium on Web Services and Applications (ISWS05), Las Vegas, USA, 2005 [Serrão et al., 2005c]

PKI and rights management interoperability

Serrão C., Torres V., Delgado J., Dias M., "**Interoperability Mechanisms for**

registration and authentication on different open DRM platforms", in International Journal of Computer Science and Network Security, Vol. 6, Number 12, Pages 291-303, December, 2006 [Serrão et al., 2006h]

Serrão, C., Serra A., Dias M., Delgado J., "**PKI as a way to leverage DRM interoperability**", In Proceedings of the IADIS International Conference on Telecommunications, Networks and Systems 2007 (TNS2007), Lisboa, Portugal, 3-5 July, 2007 [Serrão et al., 2007a]

Open rights management towards interoperability

Serrão C., Torres V., Delgado J., Dias M., " **How Open DRM platforms can shape the future of DRM**", in IEEE Multimedia, 2008 (to be published) [Serrão et al., 2008a]

Serrão, C., Serra A., Dias M., Delgado J., "**Protection of MP3 Music Files Using Digital Rights Management and Symmetric Ciphering**", in the Proceedings of the 2nd. International Conference of Automated Production of Cross Media Content for Multi-channel Distribution (AXMEDIS2006), Leeds, United Kingdom, 13-15 December, 2006 [Serrão et al., 2006g]

Serrão C., Marques J., Dias M., Delgado J., "**Open-Source Software as a Driver for Digital Content E-Commerce and DRM interoperability**", in Proceedings of the Europe-China Conference on Intellectual Property in Digital Media – Optimisation of Intellectual Property in Digital Media (IPDM06), Shangai, China, 18-19 October, 2006 [Serrão et al., 2006e]

Pimenta F., Serrão C., "**Using OMA DRM 2.0 Protected Content – Ogg Vorbis protected audio under Symbian OS**", In Proceedings of the International Conference on Security and Cryptography (SECRYPT2006), Setúbal, Portugal, 7-10 August, 2006 [Pimenta and Serrão, 2006]

Serrão C., Siegert G., "**An Open-Source Approach to Content Protection and Digital Rights Management in Media Distribution Systems**", in Proceedings of the 8th. Annual international conference of the Center of Tele-Information, Technicall

University of Denmark (8th Annual CTI Conference), Denmark, 4-5 December, 2004 [Serrão and Siegert, 2004b]

Serrão C., Neves D., Trezentos P.,"**Open Source Security Analysis: Evaluating security of Open Source Vs Closed Source Operating Systems**", in Proceedings of the 5th. Internacional Conference on Enterprise Information Systems (ICEIS 2003), Angers, France, 23-26 April, 2003 [Serrão et al., 2003c]

Secure key and license management for open rights management

Serrão, C., Serra A., Dias M., Delgado J., "**Key Management in open DRM Platforms**", in the Proceedings of the 3rd. International Conference of Automated Production of Cross Media Content for Multi-channel Distribution (AXMEDIS2007), Barcelona, Spain, 28-30 November, 2007 [Serrão et al., 2007b]

Serrão, C., Serra A., Dias M., Delgado J., "**Secure License Management - Management of Digital Object Licenses in a DRM environment**", In Proceedings of the International Conference on Security and Cryptography (SECRYPT2007), Barcelona, Spain, 28-31 July, August, 2007 [Serrão et al., 2007c]

OpenSDRM open rights management architecture

Serrão C., Dias M., Kudumakis P., "**From OPIMA to MPEG IPMP-X - A standard's history across R&D projects**", in Special Issue on European Projects in Visual Representation Systems and Services, Image Communications, Volume 20, Issue 9-10, Pages 972-994, Elsevier, 2005 [Serrão et al., 2005d]

Serrão C., Marques J., "**Enabling Digital Content Protection on Super-Distribution Models**", in Proceedings of the International Workshop for Technology, Economy, Social and Legal Aspects of Virtual Goods (Virtual Goods 2004), Ilmenau, Germany, 27-29 May, 2004 [Serrão and Marques, 2004b]

Serrão C., Siegert G., "**Open Secure Infrastructure to control User Access to multimedia content**", in Proceedings of the Second International Workshop on Security In Information Systems (ICEIS2004-WOSIS2004), Porto, Portugal, 13 April,

2004 [Serrão and Siegert, 2004a]

Serrão C., Neves D., Kudumakis P., Barker T., Balestri M., "**OpenSDRM – An Open and Secure Digital Rights Management Solution**", in Proceedings of the IADIS International Conference e-Society 2003, Lisboa, Portugal, 3-6 June, 2003 [Serrão et al., 2003b]

Alberti, C., Romeo, A., Matavelli, M., Serrão, C., "**A Structured Description Language for Multimedia Content Protection**", in Proceedings of the 4th. Conference on Telecommunications (CONFETELE 2003), Aveiro, Portugal, May, 2003 [Alberti et al., 2003]

Wallet rights management interoperability middle-ware and license templates

Serrão, C., Dias L., Serra A., Dias M., "**JPEG2000 Image Compression and Visualization for Desktop and Mobile Clients**", in Proceedings of the Atlantic Europe Conference on Remote Imaging and Spectroscopy (AECRIS2006), International Journal of Internet Protocol Technology, Preston, United Kingdom, 11-12 September, 2006 [Serrão et al., 2006a]

Serrão C., Dias M., Delgado J., "**Digital Object Rights Management – Interoperable client-side DRM middleware**", In Proceedings of the International Conference on Security and Cryptography (SECRYPT2006), Setúbal, Portugal, 7-10 August, 2006 [Serrão et al., 2006b]

Serrão C., Dias M., Delgado J., "**Bringing DRM interoperability to digital content rendering applications**", in Proceedings of the CISSE05 – The International Joint Conferences on Computer, Information, and System Sciences, and Engineering, Springer, ISBN: 978-1-4020-5260-6, University of Bridgeport, USA, 10-20 Dezembro, 2005 [Serrão et al. 2005b]

Serrão C., Dias M., Delgado J., "**Using ODRL to express rights for different content usage scenarios**", in Proceedings of the ODRL2005 – 2nd International ODRL Workshop 2005, Lisboa, Portugal, 7-8 July, 2005 [Serrão et al., 2005a]

OpenSDRM use-cases

Carvalho H., Serrão C., Serra A., Dias M., "**Flexible Access to ESA Earth Observation data using JPEG2000 and DRM**", in Proceedings of the Fourth Conference on Imaging Information Mining (ESA-EUSC2006), Madrid, Spain, 27-28 November, 2006 [Carvalho et al., 2006]

Serrão, C., Dias M., Serra A., Carvalho H., "**Accessing Earth Observation data using JPEG2000**", in Proceedings of the Symposium on Computational Modelling of Objects Represented in Images (CompImage2006), Coimbra, Portugal, 20-21 October, 2006 [Serrão et al., 2006f]

Serrão C., "**Music-4you.com – Digital Music E-Commerce Case Study**", in IADIS International Journal on Internet/WWW, Volume 3, Issue 1, ISSN 1645-7641, 2005 [Serrão, 2005]

Serrão C., Fonseca P., "**Can Digital Content E-Commerce profit from P2P Networks**", in Proceedings of the IADIS e-Commerce 2005, Porto, Portugal, 15-17 Dezembro, 2005 [Serrão and Fonseca, 2005]

Serrão C., Serra A., Dias M., "**HICOD2000 – Integrated System for Coding, Protection and Trading of Earth Observation Products in JPEG2000**", in Proceedings of the Fifth IASTED International Conference on Visualization, Imaging and Image Processing (VIIP2005), Benidorm, Spain, September 2005 [Serrão et al, 2005e]

Serrão C., Serra A., Dias M., "**Using JPEG2000 to encode, protect and trade Earth Observation data**", in Proceedings of the 14th European Colloquium on Theoretical and Quantitative Geography (ECTQG'05), Tomar, Portugal, 9-13 September, 2005 [Serrão et al., 2005f]

Serrão C., Marques J., "**Digital Music Electronic Commerce - Addressing the super distribution model** ", in Proceedings of the 1st. International Conference on E-Business and Telecommunication Networks (ICETE2004), Setúbal, Portugal, 2004 [Serrão and Marques, 2004a]

Serrão C., Serra A., Fonseca P., Dias M., "**A Method for Protecting and Controlling Access to JPEG2000 Images**", in Proceedings of the International Society for Optical Engineering conference on Applications of Digital Image Processing XXVI (SPIE 2003) Anual Meeting, Volume 5203, pages 272-286, San Diego, California, United States of America, August, 2003 [Serrão et al., 2003d]

3.2 Contributions to Standardisation activities

This section lists the different contributions to different Standardisation activities (MPEG-21 IPMP [Kim et al., 2002] and JPEG2000 Part8 - JPSEC) and to other joint scientific activities on the field (Digital Media Project (DMP) and the Networked Audio Visual Systems and Home Platforms – Co-ordination Group 1 on DRM (NAVSHP-CG1)).

MPEG-21 IPMP

Serrão C., Kudumakis P., Alberti C., on behalf of MOSES, ENTHRONE and MEDIANET, "**Joint answer to the MPEG-21 IPMP CfP**", 2004 [Serrão et al., 2004a]

JPEG2000 Part 8 – JPEG 2000 Security (JPSEC)

Serrão, C., "**JPSEC – Introduction to OpenSDRM architecture and its applicability on JPSEC core experiments**", contribution to ISO/IEC JTC 1/SC 29/WG1, document N2723, 2002 [Serrão, 2002]

Serrão, C., Conan, V., Sadourny, Y., "**Proposal to JPSEC – Protecting the JPEG2000 code-stream**", contribution to ISO/IEC JTC 1/SC 29/WG1, document N2650, 2002 [Serrão et al., 2002]

Digital Media Project

Serrão C., "**Proposal for Registration, Authentication and Secure Communication Protocol between the Device and the External Device (XD)**", Contribution number DMP0468, 2005 [Serrão, 2005b]

Serrão C., "**Report on DMP Reference Software (Turin Meeting)**", Contribution

number DMP0467, 2005 [Serrão, 2005c]

Serrão C., "**DMP Reference Software**", Contribution number DMP0452, 2005 [Serrão, 2005d]

Serrão C., "**Proposal for detailing the interaction between the PAV and the External Device**", Contribution number DMP0418, 2005 [Serrão, 2005e]

Serrão C., "**Comments on 'Revised version of Chapter 6 of AD #3: Interoperable DRM Platform, Phase I'**", Contribution number DMP0417, 2005 [Serrão, 2005f]

Serrão C., Dias M., Delgado J., "**License Management for SAV**", Contribution number DMP0416, 2005 [Serrão et al., 2005e]

Serrão C., "**Integrated License Access Protocol**", Contribution number DMP0365, 2005 [Serrão, 2005f]

Serrão C., "**Local License Access protocol**", Contribution number DMP0364, 2005 [Serrão, 2005g]

Serrão C., "**Comments on AD #1 - IDP-1 WD 2.0**", Contribution number DMP0363, 2005 [Serrão, 2005h]

Serrão C., "**Contribution for the discussion about the DMP reference software (DMP0286)**", Contribution number DMP0311, 2005 [Serrão, 2005i]

Serrão C., "**License Access protocol**", Contribution number DMP0275, 2005 [Serrão, 2005j]

Serrão C. "**Device Identification and Authentication**", Contribution number DMP0250, 2004 [Serrão, 2004b]

Serrão C., Dias M., Trindade J., Fonseca P., Kudumakis P., Chiariglione F., "**Answer to the DMP CfP (DMP0145) for Portable Audio and Video Devices (OpenSDRM)**", DMP0216, 2004 [Serrão et al., 2004b]

Networked Audio Visual Systems and Home Platforms – Co-ordination Group 1 on DRM

NAVSHP CG1 Digital Rights Management, "**NAVSHP (FP6) DRM Requirements Report, version 1.0**", Networked Audiovisual Systems and Home Platforms strategic objective, 2006 [NAVSHP-CG1, 2006]

3.3 Resume

The following table resumes the dissemination activities and the publications produced within the period of this thesis and detailed previously.

	Communications on International Scientific Events	**Communications on International Journals**	**Contributions to Standards and others**	**Total**
Rights Management and Service-Oriented Architectures	3		1	4
PKI and rights management interoperability	1	1	1	3
Open rights management towards interoperability	5	1	2	8
Secure key and license management for open rights management	2		5	7
OpenSDRM open rights management architecture	4	1	4	9
Wallet rights management interoperability middle-ware and license templates	4		2	6
OpenSDRM use-cases	8			8
Total	27	3	15	45

Chapter 2. Future Work

1 Introduction

The objective of this thesis was to present contributions on different aspects to help solving the interoperability problems of the currents rights management systems. Although several, different and innovative contributions to the problem have been presented by this work, it is clear that this is just the “tip of the iceberg” and that more research and development work needs to be carried to progressively solve this problem.

During this work, several aspects have been identified, that require further work, and that can be the ground basis for proposing new research topics for conducting more research. These topics are presented on the following section.

2 Future research and development topics

In this section, the different identified research and development topics are presented and described. These topics derive from the work conducted on this thesis, and can be the starting point for further works on this area.

2.1 Interoperable rights management brokerage

This is a topic that has been addressed on this thesis. Some steps have been given on the direction of creating a client-side middle-ware layer capable of handling the different content rendering application in a standardised and interoperable manner.

The idea proposed here is to extend the concepts presented on this thesis, to create a

generic rights management interoperability broker, that could handle the mapping between the different rights management systems. The approach to accomplish such objective should be analytical and systematic, through the identification of the different rights management systems processes and functionalities, formalising these processes (using the Web Services Business Process Execution Language (WS-BPEL)) and creating an orchestration between this processes and the corresponding ones on the broker.

WS-BPEL is an orchestration language that is used to define business processes describing the different Web Services interactions, thus providing a foundation for building SOA solutions based on Web services. To accomplish this using WS-BPEL the following requirements must be met:

- Build and then publish Web services to be used within an SOA solution
- Compose the Web services into business flows with WS-BPEL.

This is an innovative approach to the rights management interoperability problem that has already started in this thesis, but that can be further extended on future works.

2.2 Economic impact of OpenSDRM disintermediation

This thesis presented the analysis, design and implementation of an open rights management platform, whose design principles were based on the contributions of this thesis. It was also presented the multiplicity of use-cases where the platform was adapted and used on.

One of the potentialities that OpenSDRM offers is the capability of allowing the content producer to push its content directly to the end-user, without the intervention of any intermediary, such as a content provider (well, in fact the content producer assumes the content provider role).

This disintermediation is quite appealing for the current “mix culture”, allowing users to become content producers and vice-versa. This model imposes new business rules and a different logic on the current established digital multimedia content World.

The proposal is to study, from the economic point of view, which are the implications that

OpenSDRM may have digital content business scenarios in terms whenever this disintermediation takes place. It is important to understand such changes and to study and establish new revenue distribution models.

2.3 Key and license management on super-distribution

Another contribution that was presented on this thesis referred to the key and license management on rights management systems. This is an important topic for assuring the security and trustworthiness of any rights management solution.

In this particular contribution, the work performed on the key and license management aspects considered the most typical business scenario, in which the rights were expressed using a rights expression language, the content encryption key was placed inside the license and the content encryption key was ciphered for a particular user or device. This is the most common scenario that rights management systems have to deal with [Chen et al., 2006].

However, and since technology and devices are evolving in such a way that it is possible to receive content, re-author it, and re-distribute it (as many times as possible), this common scenario is not enough to handle this case of digital governed content super-distribution.

Therefore this is indeed a topic that requires further research. In particular, it is important to study how to handle key and license management in such super-distribution environments, in which end-users would have to have mechanisms to re-issue new licenses and new keys, inherited from the previous ones.

2.4 OpenSDRM development and improvement

OpenSDRM was developed with the purpose of being as open as possible, but at the same time secure. This is still a crucial aspect of every rights management system, and in particular of those that follow an open model.

The OpenSDRM platform, that is an open-source project, and that is accessible for free, to everyone, needs further developments and improvements.

First of all, it is important to conduct a security audit on the source-code, trying to find

vulnerabilities and design flaws that may compromise the overall security of the platform and of the governed content on the platform. With the results of such audit, it is possible to implement the appropriate patches to solve the security issues found.

Second, to develop and test the interoperability of the OpenSDRM platform with other open rights management systems, in such a way that it would be possible to have truly mixed and interoperable rights management platforms.

A common point should be taken into consideration. It is important that the new contributions to the platform remain as open as the original ones, so that others could continue and participate on its development. That is part of the beauty of adopting an open model.

Annex A. OpenSDRM WSDL interfaces

1 Authentication Server

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<definitions xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://opensdrm.adetti.pt/aus"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://opensdrm.adetti.pt/aus">
<types>
<xsd:schema targetNamespace="http://opensdrm.adetti.pt/aus"
>
 <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
 <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/" />
</xsd:schema>
</types>
<message name="AUSrequestModifyUserSubscriptionRequest">
  <part name="identification" type="xsd:string" />
  <part name="signature_algorithm_identifier" type="xsd:string" />
  <part name="name" type="xsd:string" />
  <part name="address" type="xsd:string" />
  <part name="email_address" type="xsd:string" />
  <part name="authentication_type" type="xsd:string" />
  <part name="uid" type="xsd:string" />
  <part name="username" type="xsd:string" />
```

```
  <part name="password" type="xsd:string" />
  <part name="arbitrary_data" type="xsd:string" />
  <part name="signature" type="xsd:string" /></message>
<message name="AUSrequestModifyUserSubscriptionResponse">
  <part name="result_message" type="xsd:string" /></message>
<message name="AUSrequestUserSubscriptionRequest">
  <part name="identification" type="xsd:string" />
  <part name="signature_algorithm_identifier" type="xsd:string" />
  <part name="name" type="xsd:string" />
  <part name="address" type="xsd:string" />
  <part name="email_address" type="xsd:string" />
  <part name="authentication_type" type="xsd:string" />
  <part name="username" type="xsd:string" />
  <part name="password" type="xsd:string" />
  <part name="public_key" type="xsd:string" />
  <part name="arbitrary_data" type="xsd:string" />
  <part name="signature" type="xsd:string" /></message>
<message name="AUSrequestUserSubscriptionResponse">
  <part name="result_message" type="xsd:string" />
  <part name="user_id" type="xsd:string" /></message>
<message name="AUSrequestAuthenticationRequest">
  <part name="identification" type="xsd:string" />
  <part name="signature_algorithm_identifier" type="xsd:string" />
  <part name="authentication_type" type="xsd:string" />
  <part name="username" type="xsd:string" />
  <part name="password" type="xsd:string" />
  <part name="signature" type="xsd:string" /></message>
<message name="AUSrequestAuthenticationResponse">
  <part name="result_message" type="xsd:string" />
  <part name="user_id" type="xsd:string" /></message>
<message name="AUSrequestDeleteUserSubscriptionRequest">
  <part name="identification" type="xsd:string" />
  <part name="signature_algorithm_identifier" type="xsd:string" />
  <part name="uid" type="xsd:string" />
  <part name="username" type="xsd:string" />
  <part name="password" type="xsd:string" />
  <part name="arbitrary_data" type="xsd:string" />
  <part name="signature" type="xsd:string" /></message>
<message name="AUSrequestDeleteUserSubscriptionResponse">
  <part name="result_message" type="xsd:string" /></message>
<message name="AUSrequestComponentSubscriptionRequest">
  <part name="arbitrary_data" type="xsd:string" />
  <part name="public_key" type="xsd:string" />
  <part name="password" type="xsd:string" /></message>
<message name="AUSrequestComponentSubscriptionResponse">
  <part name="result_message" type="xsd:string" />
  <part name="certificate" type="xsd:string" /></message>
<message name="AUSrequestComponentSubscription_Request">
  <part name="arbitrary_data" type="xsd:string" />
  <part name="public_key" type="xsd:string" />
  <part name="password" type="xsd:string" /></message>
<message name="AUSrequestComponentSubscription_Response">
  <part name="result_message" type="xsd:string" />
  <part name="certificate" type="xsd:string" /></message>
<message name="AUSrequestUserInfoRequest">
  <part name="identification" type="xsd:string" />
  <part name="signature_algorithm_identifier" type="xsd:string" />
  <part name="authentication_type" type="xsd:string" />
  <part name="uid" type="xsd:string" />
  <part name="username" type="xsd:string" />
  <part name="password" type="xsd:string" />
  <part name="signature" type="xsd:string" /></message>
<message name="AUSrequestUserInfoResponse">
```

```
  <part name="result_message" type="xsd:string" />
  <part name="uid" type="xsd:string" />
  <part name="name" type="xsd:string" />
  <part name="address" type="xsd:string" />
  <part name="email" type="xsd:string" />
  <part name="other_data_xml" type="xsd:string" />
  <part name="certificate" type="xsd:string" /></message>
<message name="AUSrequestUserPaymentInfoRequest">
  <part name="identification" type="xsd:string" />
  <part name="signature_algorithm_identifier" type="xsd:string" />
  <part name="pgw_identification" type="xsd:string" />
  <part name="user_identification" type="xsd:string" />
  <part name="pvalue" type="xsd:string" />
  <part name="signature" type="xsd:string" /></message>
<message name="AUSrequestUserPaymentInfoResponse">
  <part name="result_message" type="xsd:string" />
  <part name="payclearer" type="xsd:string" /></message>
<message name="AUSrequestListOfPGWRequest">
  <part name="identification" type="xsd:string" />
  <part name="signature_algorithm_identifier" type="xsd:string" />
  <part name="authentication_type" type="xsd:string" />
  <part name="signature" type="xsd:string" /></message>
<message name="AUSrequestListOfPGWResponse">
  <part name="result_message" type="xsd:string" />
  <part name="list_of_pgw" type="xsd:string" /></message>
<message name="AUSrequestListOfPGW_Request">
  <part name="identification" type="xsd:string" />
  <part name="signature_algorithm_identifier" type="xsd:string" />
  <part name="authentication_type" type="xsd:string" />
  <part name="signature" type="xsd:string" /></message>
<message name="AUSrequestListOfPGW_Response">
  <part name="result_message" type="xsd:string" />
  <part name="list_of_pgw" type="xsd:string" /></message>
<message name="AUSrequestMutualValidationRequest">
  <part name="uid" type="xsd:string" />
  <part name="login" type="xsd:string" />
  <part name="hash" type="xsd:string" /></message>
<message name="AUSrequestMutualValidationResponse">
  <part name="result_message" type="xsd:string" /></message>
<message name="AUSrequestWalletVerificationRequest">
  <part name="uid" type="xsd:string" /></message>
<message name="AUSrequestWalletVerificationResponse">
  <part name="result_message" type="xsd:string" /></message>
<portType name="opensdrm_aus_wsPortType">
  <operation name="AUSrequestModifyUserSubscription">
    <input message="tns:AUSrequestModifyUserSubscriptionRequest"/>
    <output message="tns:AUSrequestModifyUserSubscriptionResponse"/>
  </operation>
  <operation name="AUSrequestUserSubscription">
    <input message="tns:AUSrequestUserSubscriptionRequest"/>
    <output message="tns:AUSrequestUserSubscriptionResponse"/>
  </operation>
  <operation name="AUSrequestAuthentication">
    <input message="tns:AUSrequestAuthenticationRequest"/>
    <output message="tns:AUSrequestAuthenticationResponse"/>
  </operation>
  <operation name="AUSrequestDeleteUserSubscription">
    <input message="tns:AUSrequestDeleteUserSubscriptionRequest"/>
    <output message="tns:AUSrequestDeleteUserSubscriptionResponse"/>
  </operation>
  <operation name="AUSrequestComponentSubscription">
    <input message="tns:AUSrequestComponentSubscriptionRequest"/>
    <output message="tns:AUSrequestComponentSubscriptionResponse"/>
```

```
  </operation>
  <operation name="AUSrequestComponentSubscription_">
    <input message="tns:AUSrequestComponentSubscription_Request"/>
    <output message="tns:AUSrequestComponentSubscription_Response"/>
  </operation>
  <operation name="AUSrequestUserInfo">
    <input message="tns:AUSrequestUserInfoRequest"/>
    <output message="tns:AUSrequestUserInfoResponse"/>
  </operation>
  <operation name="AUSrequestUserPaymentInfo">
    <input message="tns:AUSrequestUserPaymentInfoRequest"/>
    <output message="tns:AUSrequestUserPaymentInfoResponse"/>
  </operation>
  <operation name="AUSrequestListOfPGW">
    <input message="tns:AUSrequestListOfPGWRequest"/>
    <output message="tns:AUSrequestListOfPGWResponse"/>
  </operation>
  <operation name="AUSrequestListOfPGW_">
    <input message="tns:AUSrequestListOfPGW_Request"/>
    <output message="tns:AUSrequestListOfPGW_Response"/>
  </operation>
  <operation name="AUSrequestMutualValidation">
    <input message="tns:AUSrequestMutualValidationRequest"/>
    <output message="tns:AUSrequestMutualValidationResponse"/>
  </operation>
  <operation name="AUSrequestWalletVerification">
    <input message="tns:AUSrequestWalletVerificationRequest"/>
    <output message="tns:AUSrequestWalletVerificationResponse"/>
  </operation>
</portType>
<binding name="opensdrm_aus_wsBinding" type="tns:opensdrm_aus_wsPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="AUSrequestModifyUserSubscription">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/AUS/AUS.ws.php/AUSrequestModifyUser
Subscription" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="AUSrequestUserSubscription">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/AUS/AUS.ws.php/AUSrequestUserSubscr
iption" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="AUSrequestAuthentication">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/AUS/AUS.ws.php/AUSrequestAuthentica
tion" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="AUSrequestDeleteUserSubscription">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/AUS/AUS.ws.php/AUSrequestDeleteUser
Subscription" style="rpc"/>
```

```
    <input><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="AUSrequestComponentSubscription">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/AUS/AUS.ws.php/AUSrequestComponentS
ubscription" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="AUSrequestComponentSubscription_">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/AUS/AUS.ws.php/AUSrequestComponentS
ubscription_" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="AUSrequestUserInfo">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/AUS/AUS.ws.php/AUSrequestUserInfo"
style="rpc"/>
    <input><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="AUSrequestUserPaymentInfo">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/AUS/AUS.ws.php/AUSrequestUserPaymen
tInfo" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="AUSrequestListOfPGW">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/AUS/AUS.ws.php/AUSrequestListOfPGW"
style="rpc"/>
    <input><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="AUSrequestListOfPGW_">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/AUS/AUS.ws.php/AUSrequestListOfPGW_
" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="AUSrequestMutualValidation">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/AUS/AUS.ws.php/AUSrequestMutualVali
dation" style="rpc"/>
```

```
    <input><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="AUSrequestWalletVerification">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/AUS/AUS.ws.php/AUSrequestWalletVeri
fication" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://opensdrm.adetti.pt/aus"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
</binding>
<service name="opensdrm_aus_ws">
  <port name="opensdrm_aus_wsPort" binding="tns:opensdrm_aus_wsBinding">
    <soap:address location="http://localhost/opensdrm.sf/server/drm/AUS/AUS.ws.php"/>
  </port>
</service>
</definitions>
```

2 Configuration Server

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<definitions xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://www.adetti.pt/opensdrmws"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://www.adetti.pt/opensdrmws">
<types>
<xsd:schema targetNamespace="http://www.adetti.pt/opensdrmws"
>
 <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
 <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/" />
</xsd:schema>
</types>
<message name="CFSrequestServerLocationRequest">
  <part name="id" type="xsd:string" /></message>
<message name="CFSrequestServerLocationResponse">
  <part name="result_message" type="xsd:string" />
  <part name="location" type="xsd:string" /></message>
<message name="CFSrequestLocationStorageRequest">
  <part name="id" type="xsd:string" />
  <part name="location" type="xsd:string" /></message>
<message name="CFSrequestLocationStorageResponse">
  <part name="result_message" type="xsd:string" /></message>
<message name="CFSrequestLocationDeleteRequest">
  <part name="id" type="xsd:string" /></message>
<message name="CFSrequestLocationDeleteResponse">
  <part name="result_message" type="xsd:string" /></message>
<portType name="opensdrmwsPortType">
  <operation name="CFSrequestServerLocation">
    <input message="tns:CFSrequestServerLocationRequest"/>
    <output message="tns:CFSrequestServerLocationResponse"/>
  </operation>
```

```
  <operation name="CFSrequestLocationStorage">
    <input message="tns:CFSrequestLocationStorageRequest"/>
    <output message="tns:CFSrequestLocationStorageResponse"/>
  </operation>
  <operation name="CFSrequestLocationDelete">
    <input message="tns:CFSrequestLocationDeleteRequest"/>
    <output message="tns:CFSrequestLocationDeleteResponse"/>
  </operation>
</portType>
<binding name="opensdrmwsBinding" type="tns:opensdrmwsPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="CFSrequestServerLocation">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/CFS/CFS.ws.php/CFSrequestServerLoca
tion" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="CFSrequestLocationStorage">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/CFS/CFS.ws.php/CFSrequestLocationSt
orage" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="CFSrequestLocationDelete">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/CFS/CFS.ws.php/CFSrequestLocationDe
lete" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
</binding>
<service name="opensdrmws">
  <port name="opensdrmwsPort" binding="tns:opensdrmwsBinding">
    <soap:address location="http://localhost/opensdrm.sf/server/drm/CFS/CFS.ws.php"/>
  </port>
</service>
</definitions>
```

3 Protection Tools Server

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<definitions xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://www.adetti.pt/opensdrmws"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://www.adetti.pt/opensdrmws">
<types>
<xsd:schema targetNamespace="http://www.adetti.pt/opensdrmws"
```

```
>
 <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
 <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/" />
</xsd:schema>
</types>
<message name="ITSrequestIPMPToolsListRequest"></message>
<message name="ITSrequestIPMPToolsListResponse">
  <part name="result_message" type="xsd:string" />
  <part name="ipmp_tools_list" type="xsd:string" /></message>
<message name="ITSrequestIPMPToolDownloadRequest">
  <part name="ipmp_tool_id" type="xsd:string" /></message>
<message name="ITSrequestIPMPToolDownloadResponse">
  <part name="result_message" type="xsd:string" />
  <part name="ipmp_tool_url" type="xsd:string" /></message>
<message name="ITSaddNewIPMPToolRequest">
  <part name="ipmptoolid" type="xsd:string" />
  <part name="ipmptoolurl" type="xsd:string" />
  <part name="ipmptooldesc" type="xsd:string" /></message>
<message name="ITSaddNewIPMPToolResponse">
  <part name="result_message" type="xsd:string" /></message>
<message name="ITSrequestIPMPToolDetailsRequest">
  <part name="ipmptoolid" type="xsd:string" /></message>
<message name="ITSrequestIPMPToolDetailsResponse">
  <part name="result_message" type="xsd:string" />
  <part name="ipmptoolid" type="xsd:string" />
  <part name="ipmptoolurl" type="xsd:string" />
  <part name="ipmptooldesc" type="xsd:string" /></message>
<portType name="opensdrmwsPortType">
  <operation name="ITSrequestIPMPToolsList">
    <input message="tns:ITSrequestIPMPToolsListRequest"/>
    <output message="tns:ITSrequestIPMPToolsListResponse"/>
  </operation>
  <operation name="ITSrequestIPMPToolDownload">
    <input message="tns:ITSrequestIPMPToolDownloadRequest"/>
    <output message="tns:ITSrequestIPMPToolDownloadResponse"/>
  </operation>
  <operation name="ITSaddNewIPMPTool">
    <input message="tns:ITSaddNewIPMPToolRequest"/>
    <output message="tns:ITSaddNewIPMPToolResponse"/>
  </operation>
  <operation name="ITSrequestIPMPToolDetails">
    <input message="tns:ITSrequestIPMPToolDetailsRequest"/>
    <output message="tns:ITSrequestIPMPToolDetailsResponse"/>
  </operation>
</portType>
<binding name="opensdrmwsBinding" type="tns:opensdrmwsPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="ITSrequestIPMPToolsList">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/ITS/ITS.ws.php/ITSrequestIPMPToolsL
ist" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="ITSrequestIPMPToolDownload">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/ITS/ITS.ws.php/ITSrequestIPMPToolDo
wnload" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
```

```
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="ITSaddNewIPMPTool">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/ITS/ITS.ws.php/ITSaddNewIPMPTool"
style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="ITSrequestIPMPToolDetails">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/ITS/ITS.ws.php/ITSrequestIPMPToolDe
tails" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
</binding>
<service name="opensdrmws">
  <port name="opensdrmwsPort" binding="tns:opensdrmwsBinding">
    <soap:address location="http://localhost/opensdrm.sf/server/drm/ITS/ITS.ws.php"/>
  </port>
</service>
</definitions>
```

4 License Server

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<definitions xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://www.adetti.pt/opensdrmws"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://www.adetti.pt/opensdrmws">
<types>
<xsd:schema targetNamespace="http://www.adetti.pt/opensdrmws"
>
 <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
 <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/" />
</xsd:schema>
</types>
<message name="LISrequestContentKeyStoreRequest">
  <part name="key" type="xsd:string" />
  <part name="cid" type="xsd:string" /></message>
<message name="LISrequestContentKeyStoreResponse">
  <part name="result_message" type="xsd:string" /></message>
<message name="LISrequestLicenseListRequest">
  <part name="uid" type="xsd:string" />
  <part name="cid" type="xsd:string" /></message>
<message name="LISrequestLicenseListResponse">
  <part name="result_message" type="xsd:string" />
  <part name="list_of_licenses" type="xsd:string" /></message>
<message name="LISrequestLicenseDownloadRequest">
  <part name="uid" type="xsd:string" />
```

```
  <part name="cid" type="xsd:string" />
  <part name="authData" type="xsd:string" /></message>
<message name="LISrequestLicenseDownloadResponse">
  <part name="result_message" type="xsd:string" />
  <part name="license" type="xsd:string" /></message>
<message name="LISrequestLicenseDownloadSpecialRequest">
  <part name="uid" type="xsd:string" />
  <part name="cid" type="xsd:string" />
  <part name="authData" type="xsd:string" /></message>
<message name="LISrequestLicenseDownloadSpecialResponse">
  <part name="result_message" type="xsd:string" />
  <part name="license" type="xsd:string" /></message>
<message name="LISrequestLicenseDeleteRequest">
  <part name="uid" type="xsd:string" />
  <part name="cid" type="xsd:string" /></message>
<message name="LISrequestLicenseDeleteResponse">
  <part name="result_message" type="xsd:string" /></message>
<message name="LISrequestLicenseCreationRequest">
  <part name="uid" type="xsd:string" />
  <part name="cid" type="xsd:string" />
  <part name="licdata" type="xsd:string" /></message>
<message name="LISrequestLicenseCreationResponse">
  <part name="result_message" type="xsd:string" /></message>
<message name="LISrequestLicenseUpdateRequest">
  <part name="uid" type="xsd:string" />
  <part name="cid" type="xsd:string" />
  <part name="licdata" type="xsd:string" /></message>
<message name="LISrequestLicenseUpdateResponse">
  <part name="result_message" type="xsd:string" /></message>
<message name="LISrequestLicensePassingRequest">
  <part name="uid_src" type="xsd:string" />
  <part name="uid_target" type="xsd:string" />
  <part name="cid" type="xsd:string" />
  <part name="rights" type="xsd:string" /></message>
<message name="LISrequestLicensePassingResponse">
  <part name="result_message" type="xsd:string" /></message>
<portType name="opensdrmwsPortType">
  <operation name="LISrequestContentKeyStore">
    <input message="tns:LISrequestContentKeyStoreRequest"/>
    <output message="tns:LISrequestContentKeyStoreResponse"/>
  </operation>
  <operation name="LISrequestLicenseList">
    <input message="tns:LISrequestLicenseListRequest"/>
    <output message="tns:LISrequestLicenseListResponse"/>
  </operation>
  <operation name="LISrequestLicenseDownload">
    <input message="tns:LISrequestLicenseDownloadRequest"/>
    <output message="tns:LISrequestLicenseDownloadResponse"/>
  </operation>
  <operation name="LISrequestLicenseDownloadSpecial">
    <input message="tns:LISrequestLicenseDownloadSpecialRequest"/>
    <output message="tns:LISrequestLicenseDownloadSpecialResponse"/>
  </operation>
  <operation name="LISrequestLicenseDelete">
    <input message="tns:LISrequestLicenseDeleteRequest"/>
    <output message="tns:LISrequestLicenseDeleteResponse"/>
  </operation>
  <operation name="LISrequestLicenseCreation">
    <input message="tns:LISrequestLicenseCreationRequest"/>
    <output message="tns:LISrequestLicenseCreationResponse"/>
  </operation>
  <operation name="LISrequestLicenseUpdate">
    <input message="tns:LISrequestLicenseUpdateRequest"/>
```

```
    <output message="tns:LISrequestLicenseUpdateResponse"/>
  </operation>
  <operation name="LISrequestLicensePassing">
    <input message="tns:LISrequestLicensePassingRequest"/>
    <output message="tns:LISrequestLicensePassingResponse"/>
  </operation>
</portType>
<binding name="opensdrmwsBinding" type="tns:opensdrmwsPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="LISrequestContentKeyStore">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/LIS/LIS.ws.php/LISrequestContentKey
Store" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="LISrequestLicenseList">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/LIS/LIS.ws.php/LISrequestLicenseLis
t" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="LISrequestLicenseDownload">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/LIS/LIS.ws.php/LISrequestLicenseDow
nload" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="LISrequestLicenseDownloadSpecial">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/LIS/LIS.ws.php/LISrequestLicenseDow
nloadSpecial" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="LISrequestLicenseDelete">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/LIS/LIS.ws.php/LISrequestLicenseDel
ete" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="LISrequestLicenseCreation">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/LIS/LIS.ws.php/LISrequestLicenseCre
ation" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
```

```
  <operation name="LISrequestLicenseUpdate">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/LIS/LIS.ws.php/LISrequestLicenseUpd
ate" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="LISrequestLicensePassing">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/LIS/LIS.ws.php/LISrequestLicensePas
sing" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
</binding>
<service name="opensdrmws">
  <port name="opensdrmwsPort" binding="tns:opensdrmwsBinding">
    <soap:address location="http://localhost/opensdrm.sf/server/drm/LIS/LIS.ws.php"/>
  </port>
</service>
</definitions>
```

5 Media Delivery Server

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<definitions xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://www.adetti.pt/opensdrmws"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://www.adetti.pt/opensdrmws">
<types>
<xsd:schema targetNamespace="http://www.adetti.pt/opensdrmws"
>
 <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
 <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/" />
</xsd:schema>
</types>
<message name="MDSrequestContentStorageRequest">
  <part name="cid" type="xsd:string" />
  <part name="filetype" type="xsd:string" />
  <part name="protocol" type="xsd:string" />
  <part name="location" type="xsd:string" /></message>
<message name="MDSrequestContentStorageResponse">
  <part name="result_message" type="xsd:string" /></message>
<message name="MDSrequestContentDeliveryRequest">
  <part name="content_id" type="xsd:string" />
  <part name="user_id" type="xsd:string" /></message>
<message name="MDSrequestContentDeliveryResponse">
  <part name="result_message" type="xsd:string" /></message>
<message name="MDSrequestContentDownloadRequest">
  <part name="content_id" type="xsd:string" />
  <part name="user_id" type="xsd:string" /></message>
```

```
<message name="MDSrequestContentDownloadResponse">
  <part name="result_message" type="xsd:string" />
  <part name="url" type="xsd:string" /></message>
<message name="MDSrequestContentDeleteRequest">
  <part name="content_id" type="xsd:string" /></message>
<message name="MDSrequestContentDeleteResponse">
  <part name="result_message" type="xsd:string" /></message>
<message name="MDSrequestDirectoryListRequest"></message>
<message name="MDSrequestDirectoryListResponse">
  <part name="result_message" type="xsd:string" />
  <part name="clist" type="xsd:string" /></message>
<message name="MDSrequestUpdateRequest">
  <part name="cid" type="xsd:string" />
  <part name="filetype" type="xsd:string" />
  <part name="protocol" type="xsd:string" />
  <part name="location" type="xsd:string" /></message>
<message name="MDSrequestUpdateResponse">
  <part name="result_message" type="xsd:string" /></message>
<portType name="opensdrmwsPortType">
  <operation name="MDSrequestContentStorage">
    <input message="tns:MDSrequestContentStorageRequest"/>
    <output message="tns:MDSrequestContentStorageResponse"/>
  </operation>
  <operation name="MDSrequestContentDelivery">
    <input message="tns:MDSrequestContentDeliveryRequest"/>
    <output message="tns:MDSrequestContentDeliveryResponse"/>
  </operation>
  <operation name="MDSrequestContentDownload">
    <input message="tns:MDSrequestContentDownloadRequest"/>
    <output message="tns:MDSrequestContentDownloadResponse"/>
  </operation>
  <operation name="MDSrequestContentDelete">
    <input message="tns:MDSrequestContentDeleteRequest"/>
    <output message="tns:MDSrequestContentDeleteResponse"/>
  </operation>
  <operation name="MDSrequestDirectoryList">
    <input message="tns:MDSrequestDirectoryListRequest"/>
    <output message="tns:MDSrequestDirectoryListResponse"/>
  </operation>
  <operation name="MDSrequestUpdate">
    <input message="tns:MDSrequestUpdateRequest"/>
    <output message="tns:MDSrequestUpdateResponse"/>
  </operation>
</portType>
<binding name="opensdrmwsBinding" type="tns:opensdrmwsPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="MDSrequestContentStorage">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/MDS/MDS.ws.php/MDSrequestContentSto
rage" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="MDSrequestContentDelivery">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/MDS/MDS.ws.php/MDSrequestContentDel
ivery" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
```

```
  </operation>
  <operation name="MDSrequestContentDownload">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/MDS/MDS.ws.php/MDSrequestContentDow
nload" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="MDSrequestContentDelete">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/MDS/MDS.ws.php/MDSrequestContentDel
ete" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="MDSrequestDirectoryList">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/MDS/MDS.ws.php/MDSrequestDirectoryL
ist" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="MDSrequestUpdate">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/MDS/MDS.ws.php/MDSrequestUpdate"
style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
</binding>
<service name="opensdrmws">
  <port name="opensdrmwsPort" binding="tns:opensdrmwsBinding">
    <soap:address location="http://localhost/opensdrm.sf/server/drm/MDS/MDS.ws.php"/>
  </port>
</service>
</definitions>
```

6 Payment Gateway Server

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<definitions xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://www.adetti.pt/opensdrmws"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://www.adetti.pt/opensdrmws">
<types>
<xsd:schema targetNamespace="http://www.adetti.pt/opensdrmws"
>
```

```
 <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
 <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/" />
</xsd:schema>
</types>
<message name="PGWrequestPaymentClearenceRequest">
  <part name="identification" type="xsd:string" />
  <part name="signature_algorithm_identifier" type="xsd:string" />
  <part name="data" type="xsd:string" />
  <part name="signature" type="xsd:string" /></message>
<message name="PGWrequestPaymentClearenceResponse">
  <part name="result_message" type="xsd:string" />
  <part name="transaction_number" type="xsd:string" /></message>
<message name="PGWrequestPaymentCaptureRequest">
  <part name="identification" type="xsd:string" />
  <part name="signature_algorithm_identifier" type="xsd:string" />
  <part name="tid" type="xsd:string" />
  <part name="signature" type="xsd:string" /></message>
<message name="PGWrequestPaymentCaptureResponse">
  <part name="result_message" type="xsd:string" />
  <part name="transaction_number" type="xsd:string" /></message>
<message name="PGWrequestCOSSubscribeRequest">
  <part name="identification" type="xsd:string" />
  <part name="signature_algorithm_identifier" type="xsd:string" />
  <part name="certificate" type="xsd:string" />
  <part name="signature" type="xsd:string" /></message>
<message name="PGWrequestCOSSubscribeResponse">
  <part name="result_message" type="xsd:string" />
  <part name="cert" type="xsd:string" /></message>
<portType name="opensdrmwsPortType">
  <operation name="PGWrequestPaymentClearence">
    <input message="tns:PGWrequestPaymentClearenceRequest"/>
    <output message="tns:PGWrequestPaymentClearenceResponse"/>
  </operation>
  <operation name="PGWrequestPaymentCapture">
    <input message="tns:PGWrequestPaymentCaptureRequest"/>
    <output message="tns:PGWrequestPaymentCaptureResponse"/>
  </operation>
  <operation name="PGWrequestCOSSubscribe">
    <input message="tns:PGWrequestCOSSubscribeRequest"/>
    <output message="tns:PGWrequestCOSSubscribeResponse"/>
  </operation>
</portType>
<binding name="opensdrmwsBinding" type="tns:opensdrmwsPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="PGWrequestPaymentClearence">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/PGW/PGW.ws.php/PGWrequestPaymentCle
arence" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="PGWrequestPaymentCapture">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/PGW/PGW.ws.php/PGWrequestPaymentCap
ture" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="PGWrequestCOSSubscribe">
```

```
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/PGW/PGW.ws.php/PGWrequestCOSSubscri
be" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
</binding>
<service name="opensdrmws">
  <port name="opensdrmwsPort" binding="tns:opensdrmwsBinding">
    <soap:address location="http://localhost/opensdrm.sf/server/drm/PGW/PGW.ws.php"/>
  </port>
</service>
</definitions>
```

7 Registration Server

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<definitions xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://www.adetti.pt/opensdrmws"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://www.adetti.pt/opensdrmws">
<types>
<xsd:schema targetNamespace="http://www.adetti.pt/opensdrmws"
>
 <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
 <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/" />
</xsd:schema>
</types>
<message name="RGSrequestContentRegistrationRequest">
  <part name="identification" type="xsd:string" />
  <part name="signature_algorithm" type="xsd:string" />
  <part name="hash" type="xsd:string" />
  <part name="file" type="xsd:string" />
  <part name="additional_data" type="xsd:string" />
  <part name="aus_cert" type="xsd:string" />
  <part name="signature" type="xsd:string" /></message>
<message name="RGSrequestContentRegistrationResponse">
  <part name="status_message" type="xsd:string" />
  <part name="content_id" type="xsd:string" /></message>
<message name="RGSrequestMetadataRegistrationRequest">
  <part name="identification" type="xsd:string" />
  <part name="signature_algorithm" type="xsd:string" />
  <part name="content_id" type="xsd:string" />
  <part name="metadata" type="xsd:string" />
  <part name="aus_cert" type="xsd:string" />
  <part name="signature" type="xsd:string" /></message>
<message name="RGSrequestMetadataRegistrationResponse">
  <part name="status_message" type="xsd:string" /></message>
<message name="RGSrequestListAvailableContentRequest">
  <part name="identification" type="xsd:string" />
  <part name="signature_algorithm_identifier" type="xsd:string" />
  <part name="criteria" type="xsd:string" />
  <part name="aus_cert" type="xsd:string" />
```

```
  <part name="signature" type="xsd:string" /></message>
<message name="RGSrequestListAvailableContentResponse">
  <part name="status_message" type="xsd:string" />
  <part name="content" type="xsd:string" /></message>
<message name="RGSrequestListMetadataRequest">
  <part name="identification" type="xsd:string" />
  <part name="signature_algorithm_identifier" type="xsd:string" />
  <part name="content_id" type="xsd:string" />
  <part name="aus_cert" type="xsd:string" />
  <part name="signature" type="xsd:string" /></message>
<message name="RGSrequestListMetadataResponse">
  <part name="status_message" type="xsd:string" />
  <part name="content_id" type="xsd:string" />
  <part name="metadata" type="xsd:string" /></message>
<message name="RGStestFunctionRequest">
  <part name="param1" type="xsd:string" />
  <part name="param2" type="xsd:string" /></message>
<message name="RGStestFunctionResponse">
  <part name="status_message" type="xsd:string" />
  <part name="real_message" type="xsd:string" /></message>
<portType name="opensdrmwsPortType">
  <operation name="RGSrequestContentRegistration">
    <input message="tns:RGSrequestContentRegistrationRequest"/>
    <output message="tns:RGSrequestContentRegistrationResponse"/>
  </operation>
  <operation name="RGSrequestMetadataRegistration">
    <input message="tns:RGSrequestMetadataRegistrationRequest"/>
    <output message="tns:RGSrequestMetadataRegistrationResponse"/>
  </operation>
  <operation name="RGSrequestListAvailableContent">
    <input message="tns:RGSrequestListAvailableContentRequest"/>
    <output message="tns:RGSrequestListAvailableContentResponse"/>
  </operation>
  <operation name="RGSrequestListMetadata">
    <input message="tns:RGSrequestListMetadataRequest"/>
    <output message="tns:RGSrequestListMetadataResponse"/>
  </operation>
  <operation name="RGStestFunction">
    <input message="tns:RGStestFunctionRequest"/>
    <output message="tns:RGStestFunctionResponse"/>
  </operation>
</portType>
<binding name="opensdrmwsBinding" type="tns:opensdrmwsPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="RGSrequestContentRegistration">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/RGS/RGS.ws.php/RGSrequestContentReg
istration" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="RGSrequestMetadataRegistration">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/RGS/RGS.ws.php/RGSrequestMetadataRe
gistration" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="RGSrequestListAvailableContent">
```

```
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/RGS/RGS.ws.php/RGSrequestListAvaila
bleContent" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="RGSrequestListMetadata">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/RGS/RGS.ws.php/RGSrequestListMetada
ta" style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="RGStestFunction">
    <soap:operation
soapAction="http://localhost/opensdrm.sf/server/drm/RGS/RGS.ws.php/RGStestFunction"
style="rpc"/>
    <input><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="http://www.adetti.pt/opensdrmws"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
</binding>
<service name="opensdrmws">
  <port name="opensdrmwsPort" binding="tns:opensdrmwsBinding">
    <soap:address location="http://localhost/opensdrm.sf/server/drm/RGS/RGS.ws.php"/>
  </port>
</service>
</definitions>
```

References

[Adams et al., 2000] Adams, C., Burmester, M., Desmedt, Y., and Reiter, M., “Which PKI (Public Key Infrastructure) is the right one? (panel session)”, in ACM Proceedings of the 7th ACM conference on Computer and Communications Security, Pages 98–101, CCS2000, Athens, Greece, 2000.

[Ahamed and Ravinuthala, 2007] Ahamed, I. and Ravinuthala, K., “Secure Media Player”, ICCE 2007, in Proceedings of the International Conference on Consumer Electronics, Digest of Technical Papers, Pages 1–2, 2007.

[Alberti et al., 2003] Alberti, C., Romeo, A., Matavelli, M., Serrão, C., "A Structured Description Language for Multimedia Content Protection", in Proceedings of the 4th. Conference on Telecommunications (CONFETELE 2003), Aveiro, Portugal, May, 2003

[Arkenbout et al., 2004] Arkenbout, E., van Dijk, F., and van Wijck, P., “Copyright in the Information Society: Scenario’s and Strategies”, in European Journal of Law and Economics, Volume 17, Issue 2, Pages 237–249, 2004.

[Arnab and Hutchison, 2007] Arnab, A. and Hutchison, A., “Persistent Access Control: a formal model for DRM”, in DRM ’07: Proceedings of the 2007 ACM workshop on Digital Rights Management, Pages 41–53, New York, NY, USA, ACM, 2007.

[AXMEDIS, 2007] AXMEDIS Project Homepage, http://www.ist-axmedis.org, visited on 12

December 2007.

[Bar-El and Weiss, 2004] Bar-El, H. and Weiss, Y., “DRM on open platforms may be possible after all”, In Proceedings of the 2nd IEE (Ref. No. 2004/10660) Secure Mobile Communications Forum: Exploring the Technical Challenges in Secure GSM and WLAN, 2004, Pages 2/1– 2/5, 2004.

[Bechtold, 2006] Bechtold, S., “The present and future of Digital Rights Management”, in Proceedings of the Second International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution, AXMEDIS ’06, Pages 6–7, 2006.

[Bellovin, 2007] Bellovin, S. “DRM, complexity, and correctness”, in Security & Privacy Magazine, Volume 5, Issue 1, Page 80, 2007.

[Bormans and Hill, 2002] Bormans, J. and Hill, K., “Mpeg-21 overview v.5”, Technical Report N5231, ISO/IEC JTC1/SC29/WG11, 2002.

[Bormans et al., 2003] Bormans, J., Gelissen, J., and Perkis, A., “Mpeg-21: The 21st century multimedia framework”, in IEEE Signal Processing Magazine, Issue 20, Pages 53–62, 2003.

[Bradbury, 2007] Bradbury, D., “Decoding Digital Rights Management”, in Computers & Security, Volume 26, Issue 1, Pages 31–33, 2007.

[Brandenburg, 1999] Brandenburg, K., “Mp3 and AAC explained”, in Proceedings of the AES 17th International Conference on High Quality Audio Coding, 1999.

[Buhse and van der Meer, 2007] Buhse, W. and van der Meer, J., “The Open Mobile Alliance Digital Rights Management [standards in a nutshell]”, in Signal Processing Magazine, IEEE, Volume 24, Issue 1, Pages 140–143, 2007.

[Bumett et al., 2005] Bumett, I., Davis, S., and Drury, G., “MPEG-21 Digital Item Declaration and Identification: Principles and Compression”, in Special section on IEEE Transactions on Multimedia, Volume 7, Issue 3, Pages 400–407, 2005.

[Buyens et al., 2007] Buyens, K., Michiels, S., and Joosen, W., “A software architecture to facilitate the creation of DRM systems”, in Proceedings of the IEEE Consumer

Communications and Networking Conference, CCNC2007, Las Vegas, Nevada, USA, 2007.

[Camp, 2003] Camp, L., “First principles of Copyright for DRM design”, in IEEE Internet Computing, Volume 7, Pages 59–65, 2003.

[Carvalho et al., 2006] Carvalho, H., Serrão, C., Serra, A., and Dias, M., “Flexible access to ESA Earth observation data using JPEG2000 and DRM”, in Proceedings of the Fourth Conference on Imaging Information Mining, ESA-EUSC2006, Madrid, Spain, 2006.

[Chen et al., 2006] Chen, C., Wu, Z., Tang, L., and Ran, C., “Encryption and program realisation of information resources in DRM”, in Proceedings of the Web Information Systems - Wise 2006 Workshops, Volume 4256, Pages 251–258, 2006.

[Chen, 2007] Chen, X., Huang, T., “Interoperability issues in DRM and DMP solutions”, in Proceedings of the 2007 IEEE International Conference on Multimedia and Expo, Pages 907–910, 2007.

[Cheng and Rambhia, 2005] Cheng, S. and Rambhia, A., “DRM and Standardization—can DRM be Standardized?”, Lecture Notes on Computer Science, Springer, Volume 2770/2003, 2005.

[Chiariglione et al., 2004] Chiariglione, F., and Kudumakis, P. on behalf of MOSES EC IST project, "Distribution of digital content", Digital Media Project GA2, Contribution No 56, Los Angeles, U.S.A., April 2004.

[Chiariglione et al., 2005] Chiariglione, L. et al., “NAVSHP (FP6) DRM Requirements Report v1.0”, Technical Report, Networked Audiovisual Systems and Home Platforms Strategic objective, 2005.

[Chiariglione and Liao, 2006] Chiariglione, F. and Liao, Y., “Chillout - the open source DRM software”, in Proceedings of the Europe-China Conference on Intellectual Property in Digital Media – Optimisation of Intellectual Property in Digital Media, IPDM06, Shangai, China, 2006.

[Chiariglione and Merrill, 2006] Chiariglione, L. and Merrill, P., “Intellectual Property in Digital Media – there is more than meets the eye”, in Proceedings of the Europe-China Conference on Intellectual Property in Digital Media – Optimisation of Intellectual Property

in Digital Media, IPDM06, Shangai, China, 2006.

[Chiariglione et al., 2007] Chiariglione, F., Kim, T., and Wang, X., "ISO/IEC 21000-5/FDAM, Rights Expression Language: the DAC profile", contribution numbered as ISO/IEC JTC 1/SC 29/WG 11/N8812, ISO/IEC JTC 1/SC 29/WG 11, 2007.

[Chiariglione, 2000] Chiariglione, L., "Intellectual property in the Multimedia Framework", in Management of Digital Rights, Berlin, 2000.

[Chiariglione, 2005] Chiariglione, L., "Balancing protection of intellectual property and its use", in Proceedings of the First International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution, AXMEDIS 2005, 2005.

[Choi et al., 2007] Choi, B., Byun, Y., Nam, J., and Hong, J., "A tool pack mechanism for DRM interoperability", ETRI Journal, Volume 29, Number 4, Pages 539–541, 2007.

[Chong and Deng, 2006] Chong, D. and Deng, R., "Privacy-enhanced superdistribution of layered content with trusted access control", in Proceedings of the ACM workshop on Digital rights management, Pages 37–44, Alexandria, Virginia, USA, ACM Press, 2006.

[Christopher, 2006] Christopher, M., "Digital Rights Management: The Problem of Expanding Ownership Rights", Chandos Publishing (Oxford) Ltd., 2006.

[ContentGuard, 2001] ContentGuard, "Extensible Rights Markup Language (XrML) 2.0 specification", Technical specification, ContentGuard, 2001.

[Cooper and Martin, 2006a] Cooper, A. and Martin, A., "Towards an open, trusted Digital Rights Management platform", In IEEE Consumer Communications and Networking Conference, 2006.

[Cooper and Martin, 2006b] Cooper, A. and Martin, A., "Towards an open, trusted Digital Rights Management platform", in Proceedings of the ACM workshop on Digital rights management - DRM06, Pages 79–88, New York, NY, USA, ACM Press, 2006.

[Coral, 2006a] Coral, "Coral Core Architecture, version 3.0", Coral consortium specification, Coral Consortium, 2006.

[Coral, 2006b] Coral, "Coral consortium Core Architecture overview", Coral consortium

informative document, Coral Consortium, 2006.

[Coral, 2006c] Coral, “Coral consortium whitepaper”, White paper, Coral Consortium, 2006.

[Coral, 2006d] Coral, “Ecosystem-A specification”, Coral consortium specification, Coral Consortium, 2006.

[Coral, 2006e] Coral, “Providing interoperability with Windows Media DRM”, Coral consortium application note, Coral Consortium, 2006.

[Cotroneo et al., 2004] Cotroneo, D., Graziano, A., and Russo, S., “Security requirements in service oriented architectures for ubiquitous computing”, in Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing (MPAC ’04), Pages 172–177, New York, NY, USA. ACM Press, 2004.

[Craig and Graham, 2003] Craig, C. and Graham, R., “Rights management in the digital world”, in Computer Law & Security Report, Volume 19, Issue 5, Pages 356–362, 2003.

[Dalziel, 2004] Dalziel, J., “DOI in a DRM environment”, White paper, Copyright Agency Limited, 2004.

[Delgado et al., 2005] Delgado, J., Torres, V., Llorente, S., and Rodriguez, E., “Rights and trust in multimedia information management”, in Proceedings of the International Conference on Communications and Multimedia Security, Lecture Notes in Computer Science, Volume 3677, Pages 55–64, 2005.

[Delgado et al., 2006] Delgado, J., Martini, T., Nesi, P., Rodríguez, E., Rogai, D., and Vallotti, A., “Definition of mechanisms that enable the exploitation of governed content”, in Proceedings of the Second International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS 2006), 2006.

[Denaro et al., 2006] Denaro, G., Pezze, M., Tosi, D., and Schilling, D., “Towards self-adaptive service-oriented architectures”, in Proceedings of the 2006 workshop on Testing, analysis, and verification of web services and applications (TAV-WEB ’06), Pages 10–16, New York, NY, USA, ACM Press, 2006.

[DMP, 2007] DMP, “IDP3 - Interoperable Digital rights management Platform, version 3.0”,

Technical specification, Digital Media Project, 2007.

[Duhl and Keroskian, 2003] Duhl, J. and Keroskian, S., "Understanding DRM systems", White paper, IDC, 2003.

[Ellison, 2002] Ellison, C., "Improvements on conventional PKI wisdom", in Proceedings of the 1st Annual PKI Research Workshop, Pages 165-175, 2002.

[ENVISAT, 2002] ESA, "ENVISAT RA2/MWR Product Handbook, Issue 1.0", ESA, 2002.

[Erickson, 2003] Erickson, J., "Fair use, DRM, and Trusted Computing", Communications of the ACM, Volume 46, Issue 4, Pages 34–39, 2003.

[Fan et al., 2006] Fan, K., Pei, Q., Mo, W., Zhao, X., and Li, X., "A novel authentication mechanism for improving the creditability of DRM system", in Proceedings of the International Conference on Communication Technology (ICCT '06), Pages 1–4, 2006.

[Fernando et al., 2005] Fernando, G., Jacobs, T., and Swaminathan, V., "Project DreaM, an architectural overview", White paper, Sun Microsystems Laboratories, 2005.

[Fetscherin and Schmid, 2003] Fetscherin, M. and Schmid, M., "The application of digital rights management systems in the music industry-an empirical investigation", in Proceedings of the Third International Conference on Web Delivering of Music (WEDELMUSIC 2003), Pages 115– 121, 2003.

[Fetscherin, 2006] Fetscherin, M., "Economics of online music and consumer behavior", in Proceedings of the 8th international conference on Electronic commerce: The new e-commerce: innovations for conquering current barriers, obstacles and limitations to conducting successful business on the internet, Pages 599–604, Fredericton, New Brunswick, Canada, ACM Press, 2006.

[FIPS, 2001] FIPS, "Announcing the Advanced Encryption Standard (AES)", International Specification, National Institute of Standards and Technology, 2001.

[Foroughi et al., 2002] Foroughi, A., Albin, M., and Gillard, S., "Digital rights management: a delicate balance between protection and accessibility", in Journal of Information Science, Volume 28, Number 5, Pages 389–395, 2002.

[Geer, 2004] Geer, D., “Digital rights technology sparks interoperability concerns”, in Computer, Volume 37, Issue 12, Pages 20–22, 2004.

[Gooch, 2003] Gooch, R., “Requirements for DRM Systems Introduction – The Requirement for DRM”, in Digital Rights Management – Technological, Economical, Legal and Political Aspects, Lecture Notes in Computer Science, Volume 2770, Pages 16-25, 2003.

[Groenenboom et al., 2006] Groenenboom, M., Helberger, N., Orwat, C., and Schaub, M. “ Consumer’s guide to Digital Rights Management”, Report, INDICARE, 2006.

[Guth and Ianella, 2005] Guth S., Ianella R., “Critical review of MPEG LA software patent claims”, INDICARE, http://www.indicare.org/tiki-read_article.php?articleId=90, 2005

[Guth et al., 2003] Guth, S., Neumann, G., and Strembeck, M., “Experiences with the enforcement of access rights extracted from ODRL-based digital contracts”, in Proceedings of the 3rd ACM workshop on Digital Rights Management, Pages 90 – 102. Washington, DC, USA, ACM Press, 2003.

[Guth et al., 2005] Guth, S., Ianella, R., and Serrão, C., “ODRL workshop 2005 report”, Report, 2005.

[Guth, 2003a] Guth, S., “A sample DRM system”, in Digital Rights Management, Lecture Notes in Computer Science, Volume 2770/2003, Springer, 2003.

[Guth, 2003b] Guth, S., Book Chapter on “Rights Expression Languages”, in Digital Rights Management, Lecture Notes in Computer Science, Volume 2770/2003, Pages 101–112, Springer, 2003.

[Hassen et al., 2007] Hassen, R., Bouabdallah, A., Bettahar, H., and Challal, Y., “Key management for content access control in a hierarchy”, in Computer Networks: The International Journal of Computer and Telecommunications Networking, Volume 51, Issue 11, Pages 3197–3219, 2007.

[Heileman and Jamkhedkar, 2005] Heileman, G. and Jamkhedkar, P., “DRM interoperability analysis from the perspective of a layered framework”, in Proceedings of the 5th ACM workshop on Digital Rights Management, Pages 17 – 26, Alexandria, VA, USA, ACM Press, 2005.

[Herzog, 2006] Herzog, P., “Open-Source Security Testing Methology Manual (OSSTMM) 2.2”, Specification, Institute for Secure and Open Methologies (ISECOM), 2006.

[Hibbert, 2005] Hibbert, C., “A copy protection and content management system from the DBV”, Technical report, DVB - DVB-Copy Protection Technology Group, 2005.

[Hongjun and Ma, 2004] Hongjun Wu, Ma, D., "Efficient and secure encryption schemes for JPEG2000", in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2004), Canada, 2004

[hwan Suh et al., 2006] Suh, J., Jin, J.-H., Park, S.-J., Bae, S.-Y., and Park, S.-C., “U-DRM: A Framework of Digital Rights Management based on Ubiquitous Computing”, in Proceedings of the 16th IEEE International Conference on Artificial Reality and Telexistence--Workshops (ICAT'06), Pages 570–575, 2006.

[Hwang et al., 2004] Hwang, S., Yoon, K., Jun, K., and Lee, K., “Modeling and Implementation of Digital Rights” in The Journal of Systems & Software, Volume 73, Issue 3, Pages 533–549, 2004.

[Iannella, 2004] Iannella, R., “The Open Digital Rights Language: XML for Digital Rights Management”, in Information Security Technical Report, Volume 9, Issue 3, Pages 47–55, 2004.

[IFPI, 2004] IFPI, “The Recording Industry commercial Piracy Report 2004”, Technical report, International Federation of the Phonographic Industry (IFPI), 2004.

[IFPI, 2005] IFPI, “IFPI:05 Digital Music Report”, Technical Report, International Federation of the Phonographic Industry (IFPI), 2005.

[ISMA-DRM, 2005] Internet Streaming Media Alliance, “Internet Streaming Media Alliance: Encryption and Authentication – Version 1.1”, External Proposed Specification, ISMA, DRM Task Force, http://www.isma.tv/technology/TD00084.pdf, 2005

[ISO/IEC-2382-20, 1993] ISO/IEC-2382-20, “Information technology - Vocabulary - Part 20: System development”, ISO/IEC-2382-20:1993, International Standard, ISO/IEC, 1993

[ISO/IEC14496-12, 2005] ISO/IEC14496-12, “Information technology – Coding of Audio-

Visual Objects – Part 12: ISO Base Media File Format", ISO/IEC 14496-12:2003, International Standard, ISO/IEC, 2005.

[Jamkhedkar and Heileman, 2004] Jamkhedkar, P. and Heileman, G., "DRM as a layered system", in Proceedings of the 4th ACM Workshop on Digital Rights Management, Pages 11–21, Washington DC, USA, ACM Press, 2004.

[Jamkhedkar et al., 2007] Jamkhedkar, P. A., Heileman, G. L., and Martinez-Ortiz, I., "Middleware services for DRM", in Proceedings of the 2nd International Conference on Communication Systems Software and Middleware (COMSWARE 2007), Pages 1–8, 2007.

[Jonker and Linnartz, 2004] Jonker, W. and Linnartz, J., "Digital Rights Management in consumer electronics products", in IEEE Signal Processing Magazine, Volume 21, Pages 82–91, 2004.

[Jonker and Mauw, 2004] Jonker, H. and Mauw, S., "Core security requirements of DRM systems", in 25th Symposium on Information Theory in the Benelux, 2004.

[Jonker et al., 2006] Jonker, H., Nair, S., and Dashti, M., "Nuovo DRM Paradiso: Towards a verified fair DRM protocol", in International Symposium on Fundamentals of Software Engineering, Lecture Notes in Computer Science, Volume 4767/2007, Pages 33-48, Springer, 2006.

[JPEG2000, 2000a] JPEG2000, "JPEG2000 Image Coding System - Part 1: Core coding system", International Standard, ISO/IEC 15444-1/ IUT-T T.800, 2000.

[JPEG2000, 2001] JPEG2000, "JPEG 2000 Image Coding System: Extensions", International Standard, ISO/IEC 15444-2, 2001

[JPIP, 2004] JPEG2000, "JPEG 2000 image coding system — Part 9: Interactivity tools, APIs and protocols", Final Draft Internation Standard, ISO/IEC FDIS 15444-9:2004(E), ISO/IEC JTC 1/SC 29, 2004

[JPSEC, 2004] JPEG2000, "JPSEC Committee Draft - Version 2.0", International Standard Committee Draft, ISO/IEC JTC 1/SC 29/WG 1, 2004

[Kalker et al., 2007] Kalker, T., Carey, K., Lacy, J., and Rosner, M., "The Coral DRM

interoperability framework", in Proceedings of the 4th IEEE Consumer Communications and Networking Conference 2007 (CCNC2007), Pages 930-934, 2007.

[Kamperman et al., 2007] Kamperman, F., Szostek, L., and Baks, W., "Marlin common domain: Authorized domains in Marlin technology". in Proceedings of the 4th IEEE Consumer Communications and Networking Conference 2007 (CCNC2007), Pages 935-939, 2007.

[Kelly et al., 2002] Kelly, S. U., Sung, C., and Farnham, S., "Designing for improved social responsibility, user participation and content in on-line communities", in Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '02), Pages 391–398, New York, NY, USA, ACM Press, 2002.

[Kerckhoffs, 1883] Kerckhoffs, A., "La cryptographie militaire", in Journal des Sciences Militaires, Volume IX, Pages 161–191, 1883.

[Keukelaere et al., 2005] Keukelaere, F. D., Zutter, S. D., and de Walle, R. V., "MPEG-21 digital item processing", in IEEE Transactions on Multimedia, Volume 7, Issue 3, Pages 427–434, 2005.

[Kim et al., 2002] Kim, J., Hwang, S., Yoon, K., and Park, C., "MPEG-21 IPMP", in International Conference on Information Technology and Applications (ICITA2002), Bathurst, Australia, 2002.

[Kim et al., 2006] Kim, J., Jeong, Y., Yoon, K., and Ryou, J., "A trustworthy end-to-end key management scheme for Digital Rights Management", in Proceedings of the 14th annual ACM International conference on Multimedia, Pages 635 – 638, ACM Press, Santa Barbara, CA, USA, 2006.

[King et al., 2001] King, J. and Kudumakis, P., "MPEG-4 IPMP Extensions", in Proceedings of the Workshop on Security and Privacy in Digital Rights Management held as part of the 8th ACM Conference on Computer and Communications Security (CCS-8), Philadelphia, Pennsylvania, USA, Nov. 5, 2001.

[Kleijnen and Raju, 2003] Kleijnen, S. and Raju, S., "An Open Web Services Architecture", ACM Queue, Volume 1, Issue 1, Pages 38–46, 2003.

[Koenen et al., 2004] Koenen, R., Lacy, J., Mackay, M., and Mitchell, S. "The long march to

interoperable Digital Rights Management", in Proceedings of the IEEE, Volume 92, Issue 6, Pages 883–897, 2004.

[Koenen, 2002] Koenen, R., "Overview of the MPEG-4 Standard", Technical Report N4668, ISO/IEC JTC1/SC29/WG11, 2002.

[Koster et al., 2006] Koster, P., Kamperman, F., Lenoir, P., and Vrielink, K., "Identity-based DRM: Personal entertainment domain", in Transactions On Data Hiding and Multimedia Security I, Lecture Notes in Computer Science, Volume 4300/2006, Pages 104–122, Springer, 2006.

[Ku and Chi, 2004] Ku, W. and Chi, C., "Survey on the technological aspects of Digital Rights Management". in Proceedings of the International Supercomputer Conference (ICS2004), Lecture Notes in Computer Science , Volume 3225/2004, Pages 391–403, Heidelberg, Germany, Springer, 2004.

[Kudumakis, 2000a] Kudumakis, P., "Liaison to MPEG concerning version 1.1 of the OPIMA Specification", Ref. as ISO/IEC JTC1/SC29/WG11/M6130 Beijing, China, July 2000.

[Kudumakis et al., 2000b] Kudumakis P., Balestri M., Guimaraes J., Lenoir P., Voukelatos S. and Dal Lago S., "OPIMA: A response to Call for Proposals for IPMP Solutions", Ref. as ISO/IEC JTC1/SC29/WG11/M6452 LaBaule, France, Oct. 2000.

[Kudumakis, 2003] Kudumakis, P., "MOSES: MPEG Open Security for Embedded Systems", in Proceedings of the 4th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS'03), London, UK, April 9-11, 2003.

[Kudumakis et al., 2001] Kudumakis, P., Dal Lago, S., Balestri, M., Lenoir, P., Guimarães, J., Serrão, C., "OPIMA/OCCAMM: A solution to DVB Call for Proposals for Content Protection & Copy Management Technologies", DVB-CPT-722, Geneva, Switzerland, Nov. 13 - 15, 2001.

[Kwok, 2002] Kwok, S., "Digital Rights Management for the online music business", in ACM SIGecom Exchanges, Volume 3, Issue 3, Pages 17 – 24, 2002.

[Lacy et al., 1999] Lacy, J., Rump, N., Shamon, T., and Kudumakis, P., "MPEG-4 Intellectual Property Management & Protection (IPMP) Overview & Aplications", in Proceedings of the AES 17th International Conference on High Quality Audio Coding, Volume 47, Page 392,

1999.

[Lamonica, 2006] Lamonica, M., "New open-source license targets DRM, Hollywood", CNET News.com, visited on January 18, 2006, http://news.com.com/New+open-source+license+targets+DRM%2C+Hollywood/2100-7344_3-6028284.html?tag=nl, 2006

[Lee et al., 2004] Lee, K., Park, J., Lee, K., Lee, J., and Kim, H., "The design of a DRM system using PKI and a licensing agent", in Proceedings of the IFIP International Conference, Network and Parallel Computing 2004, Lecture Notes in Computer Science, Volume 3222, Pages 611–617, Wuhan, China, Springer, 2004.

[Lee et al., 2007] Lee, W.-B., Wu, W.-J., and Chang, C.-Y., "A portable DRM scheme using smart cards", in Journal of Organizational Computing and Electronic Commerce, Volume 17, Pages 247–258, 2007.

[Lenzi et al., 2003] Lenzi, R., Schmucker, M., and Spadoni, F., "Technical report: Apple iTunes music store", Technical Report, Interactive MusicNetwork, 2003.

[Lesk, 2003] Lesk, M., "The good, the bad, and the ugly: what might change if we had good DRM", in IEEE Security & Privacy Magazine, Volume 1, Issue 3, Pages 63–66, 2003.

[Lifshitz, 2003] Lifshitz, Z., "Digital Rights Management - a zero-sum game?", in Proceedings of the IEEE Region 8 EUROCON 2003 - Computer as a Tool, Volume 1, Pages 29–32 (vol.1), 2003.

[Liu et al., 2003] Liu, Q., Safavi-Naini, R., and Sheppard, N. P., "Digital Rights Management for content distribution", in Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003 (ACSW Frontiers '03), Pages 49–58, Darlinghurst, Australia, Australia, Australian Computer Society, 2003.

[Marlin, 2006a] Marlin, "Marlin architecture overview", Technical paper, Marlin Developer Community, 2006.

[Marlin, 2006b] Marlin, "The role of NEMO in Marlin", Technical paper, Marlin Developer Community, 2006.

[Marlin, 2006c] Marlin, "The role of Octopus in Marlin", Technical paper, Marlin

Developer Community, 2006.

[Marques et al., 2006a] Marques, F., Filho, F., de Albuquerque, J. P., and de Geus, P. L., “A service-oriented framework to promote interoperability among DRM systems”, in Proceedings of the Autonomic Management of Mobile Multimedia Services, Volume 4267, Pages 124–127, 2006.

[Marques et al., 2006b] Marques, F., Filho, F., de Albuquerque, J. P., and de Geus, P. L., “A service-oriented framework to promote interoperability among DRM systems”, in Proceedings of the 9th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services, Lecture Notes in Computer Science, Volume 4267, Pages 124–127, Dublin, Ireland, Springer, 2006.

[Marson, 2005] Marson, Ingrid., “Sony settles 'rootkit' class action lawsuit”, CNET News.com, visited on December 29, 2005, http://news.com.com/Sony+settles+rootkit+class+action+lawsuit/2100-1002_3-6012173.html

[Media-S, 2003] Media-S web-site, http://www.sidespace.com/products/medias/

[Menezes, 1996] Menezes, A., van Oorschot, P., and Vanstone, S., “Handbook of Applied Cryptography”, CRC Press, 1996.

[Michael and Rosenblatt, 2005] Michael, E. and Rosenblatt, B., “Peer-to-peer networking and Digital Rights Management - How market tools can solve copyright problems”, in Policy Analysis 534, Cato Institute, 2005.

[Michiels et al., 2005] Michiels, S., Verslype, K., Joosen, W., and Decker, B. D., “Towards a software architecture for DRM”, in Proceedings of the 5th ACM Workshop on Digital Rights Management, Pages 65 – 74, Alexandria, VA, USA, ACM Press, 2005.

[Microsoft, 2004a] Microsoft, “Microsoft Windows Media - Architecture of Windows Media Rights Manager”, Whitepaper, 2004.

[Microsoft, 2004b] Microsoft, “Scenarios for Windows Media DRM”, Whitepaper, 2004.

[Montaner et al., 2007] Montaner, J., Koster, P., Najib, K., and Iacob, S., “Introduction of

the domain issuer in OMA DRM", in Proceedings of the 4th IEEE Consumer Communications and Networking Conference, Pages 940–944, 2007.

[MOSES, 2007] MOSES, Moses web-site, http://atlantis.tilab.com/projects/moses/index.htm, as visited on December 2007, 2007.

[MPEG-21-DID, 2005] MPEG-21-DID, "ISO/IEC IS 21000-2 Digital Item Declaration", International Standard, ISO/IEC JTC 1/SC 29/WG 11, 2005.

[MPEG-21-ER, 2006] MPEG-21-ER, "ISO/IEC IS 21000-15 Event Reporting", International Standard, ISO/IEC JTC 1/SC 29/WG 11, 2006.

[MPEG-21-IPMP, 2006] MPEG-21-IPMP, "ISO/IEC IS 21000-4 Intellectual Property Management and Protection", International Standard, ISO/IEC JTC 1/SC 29/WG 11, 2006.

[MPEG-21-RDD, 2004] MPEG-21-RDD, "ISO/IEC IS 21000-6 Rights Data Dictionary", International Standard, ISO/IEC JTC 1/SC 29/WG 11, 2004.

[MPEG-21-REL, 2004] MPEG-21-REL, "ISO/IEC IS 21000-5 Rights Expression Language", International Standard, ISO/IEC JTC 1/SC 29/WG 11, 2004.

[Mulligan et al., 2003] Mulligan, D., Han, J., and Burstein, A., "How DRM-based content delivery systems disrupt expectations of 'personal use'", In Proceedings of the 3rd ACM workshop on Digital Rights Management, Pages 77 – 89, Washington, DC, USA, ACM Press, 2003.

[NAVSHP-CG1, 2006] NAVSHP CG1 Digital Rights Management, "NAVSHP (FP6) DRM Requirements Report, version 1.0", Networked Audiovisual Systems and Home Platforms strategic objective, IST FP6, Coordination Group 1, Digital Rights Management, 2006

[Nuetzel and Beyer, 2006] Nuetzel, J. and Beyer, A., "Towards trust in Digital Rights Management systems", Proceedings of Trust, Privacy, and Security In Digital Business, Lecture Notes in Computer Science, Volume 4083, Pages 162–171, Springer, 2006.

[Nutzel and Beyer-b, 2006] Nützel, J., and Beyer, A., "How to Increase the Security of Digital Rights Management Systems Without Affecting Consumer's Security", in Proceedings of the Emerging Trends in Information and Communication Security, Volume 3995, Pages 368-380,

Springer, 2006.

[Odlyzko, 2007] Odlyzko, A., “Digital Rights Management: desirable, inevitable, and almost irrelevant”, in Proceedings of the 2007 ACM workshop on Digital Rights Management (DRM ’07), Pages 39–40, New York, NY, USA, ACM Press, 2007.

[ODRL, 2002] ODRL, “Open Digital Rights Language 1.1”, Specification, ODRL, 2002.

[OHA, 2007] OHA, “Open Handset Alliance, Android overview”, Whitepaper, OHA, 2007.

[OMA, 2006a] OMA, “Digital Rights Management”, Technical specification, Open Mobille Alliance, 2006.

[OMA, 2006b] OMA, “DRM architecture”, Technical specification, Open Mobille Alliance, 2006.

[OMA, 2006c] OMA, “DRM content format”, Technical specification, Open Mobille Alliance, 2006.

[OMA, 2006d] OMA, “DRM Rights Expression Language”, Technical specification, Open Mobille Alliance, 2006.

[OMA, 2007] OMA, Open Mobile Alliance web-site, http://www.openmobilealliance.org/, as visited on December 2007.

[OpenIPMP, 2003] OpenIPMP, “OpenIPMP overview”, Whitepaper, ObjectLab, 2003.

[OPIMA, 2000] OPIMA, “OPIMA specification - version 1.1”, Technical report, Open Platform Initiative for Multimedia Access, 2000.

[Owens and Akalu, 2004] Owens, R. and Akalu, R., “Legal policy and Digital Rights Management”, in Proceedings of the IEEE, Volume 92, Issue 6, Pages 997–1003, 2004.

[Papazoglou and Heuvel, 2007] Papazoglou, M. P. and Heuvel, W.-J., “Service oriented architectures: approaches, technologies and research issues”, in the VLDB Journal - International Journal on Very Large Data Bases, Volume 16, Issue 3, Pages 389–415, 2007.

[Pereira and Ebrahimi, 2002] Pereira, F. and Ebrahimi, T., “The MPEG-4 Book”, Prentice Hall, 2002.

[Petitcolas et al., 1998] Petitcolas, F., Anderson, R., and Kuhn, M., "Attacks on copyright marking systems", in Proceedings Proceedings of the Second International Workshop on Information Hiding, Lecture Notes in Computer Science, Springer-Verlag, London, UK, 1998

[Picot, 2003] Picot, A., "Digital Rights Management", 1st. Edition, Springer, 2003.

[Pimenta and Serrão, 2006] Pimenta, F. and Serrão, C., "Using OMA DRM 2.0 protected content – Ogg Vorbis protected audio under Symbian OS", in Proceedings of the International Conference on Security and Cryptography (SECRYPT2006), Setúbal, Portugal, 2006.

[Pinkas, 2004] Pinkas, B., "Efficient state updates for Key Management", in Proceedings of the IEEE, Volume 92, Issue 6, Pages 910-917, 2004.

[PKCS-12, 1999] RSA Laboratories, "PKCS 12 v1.0: Personal Information Exchange Syntax", Technical Specification, RSA Laboratories, 1999

[Popescu et al., 2004b] Popescu, B., Crispo, B., Tanenbaum, A., and Kamperman, F., "A DRM security architecture for home networks", in Proceedings of the 4th ACM workshop on Digital Rights Management, Pages 1 – 10, Washington DC, USA, ACM Press, 2004.

[Porter, 2004] Porter, M., "Competitive Advantage - Creating and Sustaining Superior Performance", Free Press, 2004.

[Prados et al., 2005] Prados, J., Rodriguez, E. and Delgado, J., "Interoperability between different rights expression languages and protection mechanisms", in Proceedings of the Automated Production of Cross Media Content for Multi-Channel Distribution, 2005 (AXMEDIS 2005), Page 145, IEEE Computer Society, 2005.

[Prunela, 2001] Prunela, A., "Windows Media technologies: Using Windows Media Rights Manager to protect and distribute digital media", in MSDN Magazine, Microsoft, 2001.

[Rafaeli and Hutchison, 2003] Rafaeli, S. and Hutchison, D., "A survey of Key Management for secure group communication", in ACM Computing Surveys, Volume 35, Issue 3, Pages 309–329, ACM, 2003.

[Raymond, 2004] Raymond, E. S., "If Cisco ignored Kerckhoffs's Law, users will pay the price", LWN.net, 19 May 2004, http://lwn.net/Articles/85958/

[RealNetworks, 2007] Real Networks, “Rhapsody - Premium legal music subscription service”, Real Networks, http://www.real.com/rhapsody, as visited on December 2007.

[Rescorla, 2000] Rescorla, E., “SSL and TLS: Designing and Building Secure Systems”, Addison-Wesley Professional, 1st Edition, 2000.

[Rodriguez and Delgado, 2007a] Rodriguez, E. and Delgado, J., “Trust in Event Reporting mechanisms for DRM”, in Proceedings of the 3rd IEEE International Workshop on Digital Rights Management Impact on Consumer Communications (CCNC 2007), Pages 1058 – 1062, 2007.

[Rodriguez and Delgado, 2007b] Rodriguez, E. and Delgado, J., “Verification algorithms for governed use of multimedia content”, in Online Information Review, European Workshop on Technological & Security Issues on Digital Rights Management (EuDiRights'06), Volume 31, Issue 1, Pages 38–58, 2007.

[Rosenblatt et al., 2001] Rosenblatt, B., Trippe, B., and Mooney, S., “Digital Rights Management: Business and Technology”, Wiley, 1st Edition, 2001.

[Rosenblatt, 2002] Rosenblatt, B., “Enterprise content integration with the digital object identifier: A business case for information publishers”, in Technical Report, GiantSteps, 2002.

[Rump, 2004] Rump, N., “Can Digital Rights Management be Standardized?”, in IEEE Signal Processing Magazine, Volume 21, Issue 2, Pages 63– 70, IEEE, 2004.

[Russinovich, 2005] Russinovich, M., "Sony, Rootkits and Digital Rights Management Gone Too Far", Mark's Sysinternals Blog, October 31, 2005, http://www.sysinternals.com/blog/2005/10/sony-rootkits-and-digital-rights.html, as visited on December 2007

[Safavi-Naini and Yung, 2006] Safavi-Naini, R. and Yung, M., “Digital Rights Management: Technologies, Issues, Challenges and Systems”, in First International Conference on Digital Rights Management: Technology, Issues, Challenges and Systems (DRMTICS 2005), Lecture Notes in Computer Science, Volume 3919, Sydney, Australia, 2006.

[Sander, 2002]Sander, T., “Security and Privacy in Digital Rights Management”, in Proceedings of the ACM CCS-8 Workshop DRM 2001, Lecture Notes in Computer Science, Volume 2320,

Philadelphia, USA, Springer, 2002.

[Saunders et al., 2006] Saunders, S., Ross, M., Staples, G., and Wellington, S., "The software quality challenges of service oriented architectures in e-commerce", in Software Quality Journal, Volume 14, Issue 1, Pages 65–75, Springer, 2006.

[Schmidt et al., 2004] Schmidt, A., Tafreshi, O., and Wolf, R., "Interoperability challenges for DRM systems", in Proceedings of the International Workshop for Technology, Economy, Social and Legal aspects of Virtual Goods, Pages 125-136, Ilmenau, Germany, 2004.

[Schneier, 2001] Schneier, B., "The futility of digital copy prevention", in Crypto-Gram Newsletter, http://www.schneier.com/crypto-gram-0105.html, published on May 2001, visited on December 2007.

[Schulzrinne et al., 2003] Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V., "RTP: A Transport Protocol for Real-Time Applications", in IETF RFC3350, 2003.

[Seki and Kameyama, 2003] Seki, A. and Kameyama, W., "A proposal on open DRM system coping with both benefits of rights-holders and users", in Proceedings of the IEEE Global Telecommunications Conference 2003 (GLOBECOM '03), Volume 7, pages 4111– 4115, 2003.

[Senoh et al., 2004] Senoh, T., Kogure, T., Jing, T., Schultz, L., and et al., "DRM renewability & interoperability", in Proceedings of the First IEEE Consumer Communications and Networking Conference 2004 (CCNC2004), pages 424– 429, 2004.

[Serrão and Dias, 2002] Serrão, C. and Dias, M., "Space and planetary imaging using JPEG2000", in Proceedings of the 7th International Workshop on Simulation for European Space Programmes, ESA/ESTEC, Netherlands, 2002.

[Serrão and Fonseca, 2005] Serrão, C. and Fonseca, P., "Can digital content e-commerce profit from P2P networks", in Proceedings of the IADIS e-Commerce 2005, Porto, Portugal, 2005.

[Serrão and Marques, 2004a] Serrão, C. and Marques, J., "Digital music electronic commerce - addressing the super distribution model", in Proceedings of the 1st. International Conference on E-Business and Telecommunication Networks (ICETE2004), Volume 3, Pages 375-378, Setúbal, Portugal, INSTICC Press, 2004.

[Serrão and Marques, 2004b] Serrão, C. and Marques, J., "Enabling digital content protection on super-distribution models", in Proceedings of the International Workshop for Technology, Economy, Social and Legal Aspects of Virtual Goods (Virtual Goods 2004), Pages 101-112, Ilmenau, Germany, 2004.

[Serrão and Siegert, 2004a] Serrão, C. and Siegert, G.,"Open secure infrastructure to control user access to multimedia content", in Proceedings of the Second International Workshop on Security In Information Systems (WOSIS2004) integrated on the 6th International Conference on Enterprise Information Systems (ICEIS), Pages 433-440, Porto, Portugal, 2004.

[Serrão and Siegert, 2004b] Serrão, C. and Siegert, G., "An open-source approach to content protection and digital rights management in media distribution systems", in Proceedings of the 8th Annual CTI Conference Copyright and Software Patents, Open vs. Proprietary Developments Paths, Center for Tele-Information, Technical University of Denmark, Kongens Lyngby, Denmark, 2004.

[Serrão et al., 2002] Serrão, C., Conan, V., and Dadourny, Y., "Proposal to JPSEC – Protecting the JPEG2000 code-stream", in Contribution N2650, ISO/IEC JTC 1/SC 29/WG1, 2002.

[Serrão et al., 2003a] Serrão, C., Marques, J., and Fonseca, P., "E-commerce payment systems: An overview", in Proceedings of the 5th. Internacional Conference on Enterprise Information Systems (ICEIS 2003), Pages 486-493, Angers, France, 2003.

[Serrão et al., 2003b] Serrão, C., Neves, D., Barker, T., Kudumakis, P., and Balestri, M., "OpenSDRM – an Open and Secure Digital Rights Management solution". In Proceedings of the IADIS International Conference e-Society (e-Society 2003), Lisbon, Portugal, 2003.

[Serrão et al., 2003c] Serrão, C., Neves, D., and Trezentos, P., "Open source security analysis: Evaluating security of open source vs closed source operating systems", in Proceedings of the 5th. Internacional Conference on Enterprise Information Systems (ICEIS 2003), Angers, France, 2003.

[Serrão et al., 2003d] Serrão, C., Serra, A., Fonseca, P., and Dias, M., "A method for protecting and controlling access to JPEG2000 images", in Proceedings of the International Society for Optical Engineering conference on Applications of Digital Image Processing XXVI (SPIE 2003),

Anual Meeting, Volume 5203, Pages 272–286, San Diego, USA, 2003.

[Serrão et al., 2004a] Serrão, C., Kudumakis, P., and Alberti, C., "Joint answer to the MPEG-21 IPMP CfP" on behalf of MOSES, ENTHRONE and MEDIANET EC IST projects, Ref. as ISO/IEC JTC1/SC29/WG11/M10854, Seattle, U.S.A., July 2004.

[Serrão et al., 2004b] Serrão, C., Dias, M., Trindade, J., Fonseca, P., Kudumakis, P., Chiariglione, F., "Answer to the DMP CfP (DMP0145) for Portable Audio and Video Devices (OpenSDRM)", Digital Media Project GA4, Contribution No 0216, Barcelona, Spain, Oct. 2004.

[Serrão et al., 2005a] Serrão, C., Delgado, J., and Dias, M., “Using ODRL to express rights for different content usage scenarios”, in Proceedings of the ODRL2005 – 2nd International ODRL Workshop 2005, Lisbon, Portugal, 2005.

[Serrão et al., 2005b] Serrão, C., Dias, M., and Delgado, J., “Bringing DRM interoperability to digital content rendering applications”, in Advances in Computer, Information, and Systems Sciences, and Engineering, Proceedings of the International Conference on Telecommunications and Networking, Pages 323–329, 1st Edition, Kluwer Academic Publishers, 2005.

[Serrão et al., 2005c] Serrão, C., Dias, M., and Delgado, J., “Using web-services to manage and control access to multimedia content”, in Proceedings of The 2005 International Symposium on Web Services and Applications (ISWS 2005), Pages 27-30, Las Vegas, Nevada, USA, 2005.

[Serrão et al., 2005d] Serrão, C., Dias, M., and Kudumakis, P., “From OPIMA to MPEG IPMP-X - A standard’s history across R&D projects”, in Special Issue on European Projects in Visual Representation Systems and Services, Signal Processing: Image Communications, Volume 20, Issues 9-10, Pages 972–994, Elsevier, 2005.

[Serrão et al., 2005e] Serrão, C., Serra, A., and Dias, M., “HICOD2000 – Integrated system for coding, protection and trading of earth observation products in JPEG2000”, in Proceedings of the Fifth IASTED International Conference on Visualization, Imaging and Image Processing (VIIP2005), Volume 480, Acta Press, Benidorm, Spain, 2005.

[Serrão et al., 2005f] Serrão, C., Serra, A., and Dias, M., “Using JPEG2000 to encode, protect

and trade earth observation data", in Proceedings of the 14th European Colloquium on Theoretical and Quantitative Geography (ECTQG05), Tomar, Portugal, 2005.

[Serrão et al., 2006a] Serrão, C., Dias, L., Serra, A., and Dias, M., "JPEG2000 image compression and visualization for desktop and mobile clients", in Proceedings of the Atlantic Europe Conference on Remote Imaging and Spectroscopy (AECRIS2006), Pages 8–14, InderScience, University of Central Lancashire, Preston, UK, 2006.

[Serrão et al., 2006b] Serrão, C., Dias, M., and Delgado, J., "Digital object Rights Management – interoperable client-side DRM middleware", in Proceedings of the International Conference on Security and Cryptography (SECRYPT2006), SECRYPT is part of ICETE - The International Joint Conference on e-Business and Telecommunications, Pages 229-236, INSTICC Press, Setúbal, Portugal, 2006.

[Serrão et al., 2006c] Serrão, C., Dias, M., and Delgado, J., "Using service-oriented architectures towards Rights Management interoperability", in Innovations and Advanced Techniques in Computer and Information Sciences and Engineering, Proceedings of the International Joint Conferences on computer, Information and Systems Sciences and Engineering (CISSE06), Pages 519-524, Springer, University of Bridgeport, USA, 2006.

[Serrão et al., 2006d] Serrão, C., Fonseca, P., Dias, M., and Delgado, J., "The web-services growing importance for DRM interoperability", in Proceedings of the IADIS International Conference WWW/Internet 2006. IADIS ICWI2006, Múrcia, Spain, 2006.

[Serrão et al., 2006e] Serrão, C., Marques, J., Dias, M., and Delgado, J., "Open-source software as a driver for digital content e-commerce and DRM interoperability", in Proceedings of the Europe-China Conference on Intellectual Property in Digital Media – Optimisation of Intellectual Property in Digital Media (IPDM06), Shangai, China, 2006.

[Serrão et al., 2006f] Serrão, C., Serra, A., Carvalho, H., and Dias, M., "Accessing Earth Observation data using JPEG2000", in Proceedings of the Symposium on Computational Modelling of Objects Represented in Images (CompImage2006), Coimbra, Portugal, 2006.

[Serrão et al., 2006g] Serrão, C., Serra, A., Dias, M., and Delgado, J., "Protection of MP3 music files using Digital Rights Management and symmetric ciphering", in Proceedings of the

2nd. International Conference of Automated Production of Cross Media Content for Multi-channel Distribution (AXMEDIS2006), Pages 128-135, Leeds, UK, 2006.

[Serrão et al., 2006h] Serrão, C., Torres, V., Dias, M., and Delgado, J., "Interoperability mechanisms for registration and authentication on different open DRM platforms", in International Journal of Computer Science and Network Security, Volume 6, Issue 12, Pages 291–303, 2006.

[Serrão et al., 2007a] Serrão, C., Serra A., Dias M., Delgado J., "PKI as a way to leverage DRM interoperability", In Proceedings of the IADIS International Conference on Telecommunications, Networks and Systems 2007 (TNS2007), Lisboa, Portugal, 3-5 July, 2007

[Serrão et al., 2007b] Serrão, C., Serra A., Dias M., Delgado J., "Key Management in open DRM Platforms", in the Proceedings of the 3rd. International Conference of Automated Production of Cross Media Content for Multi-channel Distribution (AXMEDIS2007), Barcelona, Spain, 28-30 November, 2007

[Serrão et al., 2007c] Serrão, C., Serra A., Dias M., Delgado J., "Secure License Management - Management of Digital Object Licenses in a DRM environment", In Proceedings of the International Conference on Security and Cryptography (SECRYPT2007), Barcelona, Spain, 28-31 July, August, 2007

[Serrão et al., 2008a] Serrão C., Torres V., Delgado J., Dias M., " How Open DRM platforms can shape the future of DRM", in IEEE Multimedia, 2008 (to be published)

[Serrão, 2002] Serrão, C., "JPSEC – Introduction to OpenSDRM architecture and its applicability on JPSEC core experiments", in Technical Report N2723, ISO/IEC JTC 1/SC 29/WG1, 2002.

[Serrão, 2004] Serrão, C., "Open secure infrastructure to control user access to multimedia content", in Proceedings of the 4th. International Conference on Web Delivering of Music (WEDELMUSIC2004), Pages 62-69, Barcelona, Spain, 2004.

[Serrão, 2004b] Serrão C. "Device Identification and Authentication", Contribution number DMP0250, Digital Media Project, 2004

[Serrão, 2005] Serrão, C., "Music-4you.com – digital music e-commerce case study", in IADIS

International Journal on Internet/WWW, Volume 3, Issue 1, IADIS, 2005.

[Serrão, 2005b] Serrão C., “Proposal for Registration, Authentication and Secure Communication Protocol between the Device and the External Device (XD)”, Contribution number DMP0468, Digital Media Project, 2005

[Serrão, 2005c] Serrão C., “Report on DMP Reference Software (Turin Meeting)”, Contribution number DMP0467, Digital Media Project, 2005

[Serrão, 2005d] Serrão C., “DMP Reference Software”, Contribution number DMP0452, Digital Media Project, 2005

[Serrão, 2005e] Serrão C., “Proposal for detailing the interaction between the PAV and the External Device”, Contribution number DMP0418, Digital Media Project, 2005

[Serrão, 2005f] Serrão C., “Comments on 'Revised version of Chapter 6 of AD #3: Interoperable DRM Platform, Phase I'”, Contribution number DMP0417, Digital Media Project, 2005

[Serrão et al., 2005e] Serrão C., Dias M., Delgado J., “License Management for SAV”, Contribution number DMP0416, Digital Media Project, Digital Media Project, 2005

[Serrão, 2005f] Serrão C., “Integrated License Access Protocol”, Contribution number DMP0365, Digital Media Project, Digital Media Project, 2005

[Serrão, 2005g] Serrão C., “Local License Access protocol”, Contribution number DMP0364, Digital Media Project, 2005

[Serrão, 2005h] Serrão C., “Comments on AD #1 - IDP-1 WD 2.0", Contribution number DMP0363, Digital Media Project, 2005

[Serrão, 2005i] Serrão C., “Contribution for the discussion about the DMP reference software (DMP0286)”, Contribution number DMP0311, Digital Media Project, 2005

[Serrão, 2005j] Serrão C., “License Access protocol”, Contribution number DMP0275, Digital Media Project, 2005

[Shapiro and Vingralek, 2002] Shapiro, W., and Vingralek, R., “How to Manage Persistent State in DRM Systems”, in Proceedings of the Security and Privacy in Digital Rights

Management : ACM CCS-8 Workshop DRM 2001, Pages 176-191, Philadelphia, PA, USA, 2001.

[Sharpe and Arewa, 2007] Sharpe, N. F. and Arewa, O. B., “Is apple playing fair? Navigating the iPod Fairplay DRM controversy”, in NORTHWESTERN JOURNAL OF TECHNOLOGY AND INTELLECTUAL PROPERTY, Volume 5, Issue 2, Pages 332-350, Northwestern University School of Law, 2007.

[Sheppard, 2007] Sheppard, N. P., “On implementing MPEG-21 Intellectual Property Management and Protection”, in Proceedings of the 2007 ACM Workshop on Digital Rights Management (DRM ’07), Pages 10–22, New York, NY, USA, ACM, 2007.

[Stamp, 2002] Stamp, M., “Risks of Digital Rights Management”, in Communications of the ACM, Volume 45, Issue 9, Page 120, ACM, 2002.

[Sun et al., 2007] Sun, H.-M., Hung, C.-F., and Chen, C.-M., “An improved Digital Rights Management system based on smart cards”, in Proceedings of the Inaugural IEEE-IES Digital EcoSystems and Technologies Conference (DEST ’07), pages 308–313, 2007.

[SunLabs, 2006] SunLabs, “DReaM - MMI specification, version 0.8”, in Technical specification, Sun Microsystems Laboratories, 2006.

[SunLabs, 2007] SunLabs, “DReaM CAS - client specification, version 1.0 Rev A”, Technical specification, Sun Microsystems Laboratories, 2007.

[Taban et al., 2006] Taban, G., Cardenas, A. A., and Gligor, V. D., “Towards a secure and interoperable DRM architecture”, in Proceedings of the ACM Workshop on Digital Rights Management, Pages 69–78, New York, NY, USA, ACM, 2006.

[Taubman and Marcellin, 2002] Taubman, D. and Marcellin, M., “JPEG 2000: Image Compression Fundamentals, Standards and Practice”, Kluwer Academic Publishers, 2002.

[Thomas, 2000] Thomas, S. A., “SSL & TLS Essentials: Securing the Web”, Wiley, 2000.

[Torres et al., 2005] Torres, V., Rodríguez, E., et al., “Use of standards for implementing a Multimedia Information Protection and Management System”, in First International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution

2005 (AXMEDIS 2005), Pages 197-204, IEEE Computer Society, 2005.

[Torres et al., 2006] Torres, V., Delgado, J., and Llorente, S., "An implementation of a trusted and secure DRM architecture", in Proceedings of the On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops, Lecture Notes in Computer Science, Volume 4277/2006, Pages 312–321, 2006.

[Travert and Lemonier, 2004] Travert S., Lemonier M., "The MediaNet Project", in Proceedings of the 5th International Workshop on Image Analysis for Multimedia Interactive Services 2004 (WIAMIS2004), Lisbon, 2004

[Venkataramu, 2007] Venkataramu, R., "Analysis and enhancement of Apple's Fairplay Digital Rights Management", in Master Thesis, The Faculty of the Department of Computer Science, San Jose State University, 2007.

[Verslype and De Decker, 2006] Verslype, K. and De Decker, B., "A flexible and open DRM framework", in Communications and Multimedia Security 2006, Lecture Notes in Computer Science, Volume 4237, Pages 173–184, 2006.

[Wang et al., 2002] Wang, X., Lao, G., DeMartini, T., Reddy, H., Nguyen, M., and Valenzuela, E., "XRML – eXtensible Rights Markup Language", in Proceedings of the 2002 ACM workshop on XML security (XMLSEC '02), Pages 71–79, New York, NY, USA, ACM, 2002.

[Wang et al., 2005] Wang, X., DeMartini, T., Wragg, B., Paramasivam, M., and Barlas, C., "The MPEG-21 Rights Expression Language and Rights Data Dictionary", in IEEE Transactions On Multimedia, Volume 7, Pages 408–417, 2005.

[Weber, 2007] Weber, R. M., "Is the iPhone right for you?", in Journal of Financial Service Professionals, Volume 61, Issue 5, Pages 40–41, 2007.

[Wegner, 2003] Wegner, S., "Opera - Interoperability of Digital Rights Management (DRM) technologies, an Open DRM architecture", in Project Report, OPERA Project, 2003.

[White, 2000] White, A., "Convergence of P2P and B2B: New economy business models", in Short Brief, Logility, 2000.

[Wyant, 2002] Wyant, J., "Applicability of public-key cryptosystems to Digital Rights

Management applications", in Proceedings of the 5th International Conference on Financial Cryptography, Lecture Notes in Computer Security, Volume 2339, Pages 75-78, Springer, 2002.

[Zeng et al., 2006] Zeng, W., Yu, H., and Lin, C.-Y., "Multimedia Security Technologies for Digital Rights Management", 1st Edition, Academic Press, 2006.

[Zhang et al., 2004] Zhang, J., Chung, J.-Y., and Chang, C. K., "Migration to web services oriented architecture: a case study", in Proceedings of the 2004 ACM Symposium on Applied Computing (SAC '04), Pages 1624–1628, New York, NY, USA, ACM, 2004.

[Zhang et al., 2005] Zhang, J., Li, B., Zhao, L., and Yang, S.-Q., "License management scheme with anonymous trust for Digital Rights Management", in Proceedings of the IEEE International Conference on Multimedia and Expo, 2005 (ICME 2005), IEEE, 2005