Interoperability Mechanisms for Registration and Authentication on Different Open DRM Platforms

Carlos Serrão†*, Victor Torres*††*, Jaime Delgado*††† *and Miguel Dias*††††

†*ISCTE/DCTI/ADETTI, Ed. ISCTE, Instituto Superior de Ciências do Trabalho e da Empresa, Dep. Ciências e Tecnologias de Informação, Av. Das Forças Armadas, 1600-082 Lisboa, Portugal*
††*UPF – Universitat Pompeu Fabra, Departament de Tecnologia, Pg. Circumvallació 8, E-08003 Barcelona, Spain*
†††*UPC – Universitat Polytècnica de Catalunya, Dept. AC, DMAG, Campus Nord Mòdul D6, E-08034 Barcelona, Spain*
††††**MLDC - Microsoft Language Development Center, Av. Dr. Anibal Cavaco Silva, Edificio qualidade C1-C2, Tagus Park, Porto Salvo, Portugal**

Summary
The DRM interoperability problem is a very complex problem. Even big software companies have already admitted that DRM as it is today is too complex – complex for end-users, complex for content providers and complex for content handling devices manufactures. There are different approaches to deal with this problem and there are different levels to address the problem. This article addresses the DRM interoperability issues from a security point of view, and as an example the authors take two open-specification DRM architectures – MIPAMS and OpenSDRM – to identify a strategy to interoperate some of the basic security mechanisms. In this article the authors will concentrate in the DRM components and user's registration, authentication and verification process and will derive a mechanism to handle and support both.

Key words:
DRM, interoperability, security, registration, authentication

1. Introduction

One of the most working examples of interoperability in the IT world is the Internet. The Internet is a fairly heterogeneous environment in terms of hardware, architectures and systems; however every hardware device or application can exchange information in a common and clear way. This is only possible because there is a single standard communication protocol (TCP/IP) on the network that bounds everything together. This means that any system or application willing to use the Internet has to implement the mechanisms to comply with the TCP/IP specifications.

This is a fairly straightforward way of providing interoperability; however a major requirement of it is that everyone agrees to follow a single standard. This approach works very well in the Internet case; however the same approach cannot be applied in some other situations. Multimedia is one of these. 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 has strictly implemented by everyone. Therefore interoperability in multimedia and in particular in the case of DRM is far more complex to handle than in the Internet scenario.

Some authors [1] have suggested a set of different approaches to achieve DRM interoperability, based on International Standards: full-format interoperability, connected interoperability and configuration driven interoperability. In the case of full-format interoperability all protected content conforms to some unique globally standardized format. This is hard to accomplish, since all of the content providers and all the DRM software manufacturers would have to have an agreement of the same file format to use. This is, nevertheless the strategy that's being followed by OMA DRM – in OMA, the DRM Content Format (DCF) is a format that each of the devices need to know and implement and even by Microsoft with the Windows Media Format (WMF). In the second 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 on the future, for instance implementing web-services that will allow the translation between different DRM functionalities to accomplish the same objective – to enable DRM interoperability between different DRM providers. In this approach a peer-to-peer architecture may need to be established in which each node allow an interface to its peers, and if it can't satisfy a direct request them redirects the search to other peers. Another approach is the "intermediated digital rights management" [2] where are identified four tasks to be carried 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 are executed by

Manuscript received December 8 2006.
Manuscript revised December 25 2006.

a third party server (the intermediary) on behalf of the content scripts and end-users. The third and final approach for DRM interoperability upholds that by downloading adequate tools any DRM system can get the ability to process protected content on end users devices. This is also a more valid and viable alternative for the DRM interoperability problem allowing each device and each digital content rendering application to "grow" its own 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 [3].

However, most of these approaches rely on the fact that the DRM platforms have either own specifications publicly open or that its source-code is available. Nevertheless there are many aspects where these different DRM platforms can differ. A first point of divergence might be their architecture or even the operations orchestration between the different components of that architecture. Another fundamental point of difference is the security aspects of the DRM platform, in particular those related to the components and users registration, validation and authentication. During this paper, the authors will focus especially in this security-related question on two open-specification and open-source DRM platforms and we will point out some directions for achieving interoperability between them. As we had the opportunity to refer previously DRM interoperability is a huge and complex problem and it needs to be solved by different stages – the work we are presenting in this paper is just one very small contribution for the overall problem.

2. Open DRM platforms

This section will focus 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 study, one open-source implementation (OpenSDRM) and one open-specification DRM platforms (DMAG/MIPAMS) were considered [17].

2.1 OpenSDRM – Open and Secure Digital Rights Management

OpenSDRM is an open specification and an open-source implementation of a DRM platform. It started being developed for a project called MOSES [4], but since then it has being evolved and initial functionalities have been extended [5][6][7]. OpenSDRM relies on a distributed philosophy in which each of the different components is implemented in a self-contained way encapsulating a set of specific functionalities. The components are in fact web-services, with a public WSDL description, deployed either in the same hardware platform or in several ones remotely distributed. Messages exchanged between the different components are SOAP-based over SSL –secured and authenticated connections [13][16].

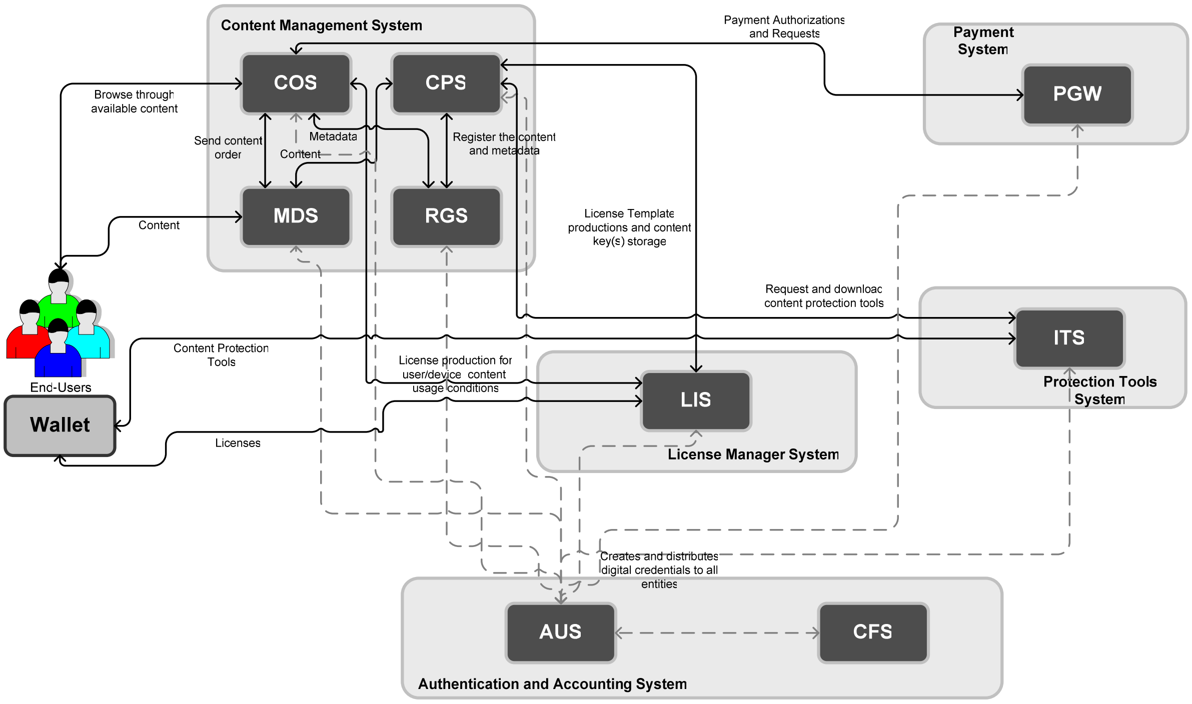
Content Management System
COS
CPS
MDS
RGS
Browse through available content
Send content order
Metadata
Content
Register the content and metadata
Content
Payment Authorizations and Requests
Payment System
PGW
License Template productions and content key(s) storage
Request and download content protection tools
ITS
Protection Tools System
End-Users
Wallet
Content Protection Tools
License production for user/device content usage conditions
LIS
License Manager System
Licenses
Creates and distributes digital credentials to all entities
AUS
CFS
Authentication and Accounting System

Fig. 1 OpenSDRM generic architecture.

The OpenSDRM platform (Fig. 1) uses two important concepts that will be largely referenced afterwards: Actors and Components. An **Actor** is a person or an organization that uses a Component. A **Component** is a set of software and hardware tools cooperating for offering a set of specific DRM-related functionalities.

2.1.1 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. 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 ($Kpriv^{Comp}$, $Kpub^{Comp}$) 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 ($Kpriv^{Comp}$, $Kpub^{Comp}$) and securely stores the $Kpriv^{Comp}$ 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.

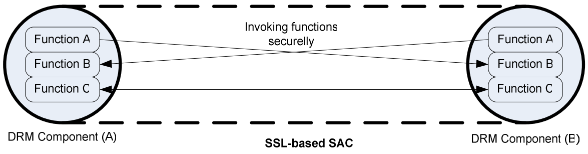

Fig. 2 Establishing a SAC between different DRM components.

2.1.2 Components Mutual Authentication

Each of the DRM components is certified by a CA. Each of the DRM components will have a list of trustworthy CAs, 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. All further communications within the DRM platform are handled over a mutually authenticated and secure SSL channel (Fig. 2). Whenever a function within a DRM component invokes another function on other DRM component this SAC process is repeated.

2.1.3 Actors and components registration

While the authenticated and secure communication between the different DRM components is established like what it was presented in the previous sections, each of the DRM components can contain and represent different functions in the overall DRM architecture. In OpenSDRM, the interaction between the different functions of different DRM components demands a new layer of authentication, which is called application-level authentication. Also the different actors that interact with the different functions of the DRM components need to be registered and authenticated.

In OpenSDRM, there is a 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: `<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 that signed and issued the certificate;
(ii) `<identifier/>` is a unique identifier of the entity (either `<issuer>` our `<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 expiry date of the certificate;

(vi) Finally, `<signature/>` contains the digital signature of 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 may be optionally ciphered, using a secret password and contains the modulus, public-key exponent, private-key exponent, and the original prime numbers selected.

The process to register a functionality of a DRM component 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 ($Kpriv^{FComp}$, $Kpub^{FComp}$), and stores the private key protected by a secret key (AES): Sk [$Kpriv^{FComp}$]. 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: $Kpub^{FComp}$ and $FComp_{id}$;

(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 registration of Actors on the OpenSDRM system is mediated through a broker called Wallet [12]. This broker is the software responsible for interacting with the other DRM components functions and in particular with the AUS. The registration process of an Actor in 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, $Device_{Info}$) = Sk;

(b) The Wallet broker creates a key-pair, in XML format: ($Kpriv^{Actor}$, $Kpub^{Actor}$). The $Kpriv^{Actor}$ is securely stored on the database: Sk [$Kpriv^{Actor}$].

(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: $Kpub^{Actor}$ and $Actor_{id}$;

(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.

2.1.4 Components and Actors Authentication

In this section, we will describe how OpenSDRM handles Components (in terms of its functionalities) and Actors authentication.

Whenever a function in a DRM component wishes to use 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 check is important to assure that the requesting DRM component certificate has not been revoked by AUS (Fig. 3).

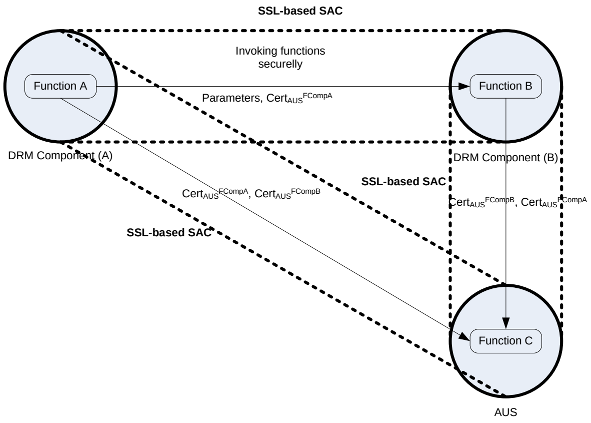
SSL-based SAC
Invoking functions securelly
Function A
Function B
Parameters, $Cert_{AUS}^{FCompA}$
DRM Component (A)
DRM Component (B)
SSL-based SAC
$Cert_{AUS}^{FCompA}$, $Cert_{AUS}^{FCompB}$
$Cert_{AUS}^{FCompB}$, $Cert_{AUS}^{FCompA}$
SSL-based SAC
Function C
AUS

Fig. 3 Establishing a SAC between different DRM components.

While invoking the DRM component function it may be possible to request also the certificate from the remote component, also to be sure 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.

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 OCSP (Fig.

4). The process that OpenSDRM uses to authenticate users is the following:

(i) The Actor, uses its Wallet broker to access his credentials: $Cert_{AUS}^{Actor}$;

(ii) The Actor requests the authentication or any other operation on a DRM component using $Cert_{AUS}^{Actor}$;

(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 ($Cert_{AUS}^{FComp}$) and the Actor certificate ($Cert_{AUS}^{Actor}$);

(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.

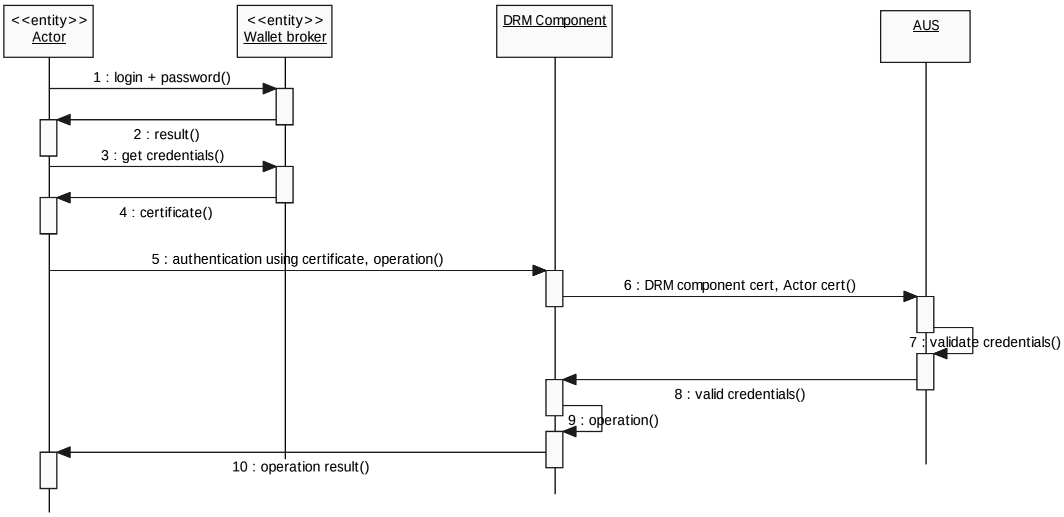

Fig. 4 Actor's authentication through the AUS

2.2 MIPAMS – Multimedia Information Protection and Management System

The MIPAMS architecture was already described and presented in several papers: [8], [9], [10]. This architecture is used to manage multimedia information taking into account digital rights management (DRM) and protection. The architecture, also called DMAG-MIPAMS, which includes the DMAG [11] research group acronym, 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.

DMAG-MIPAMS encompasses an important part of the whole content value chain, from content creation and distribution to its consumption by final users. The DMAG-MIPAMS architecture is depicted in Fig. 5.

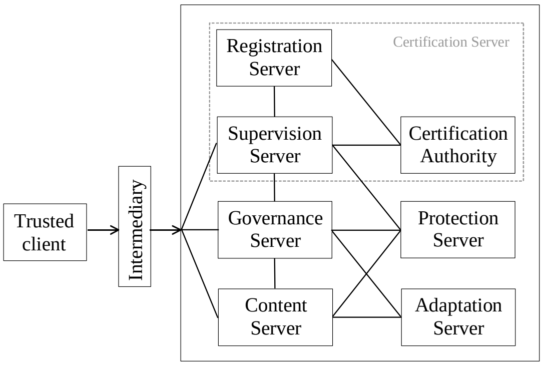

Fig. 5 MIPAMS generic architecture

In this section of the paper we will also follow the same approach as in OpenSDRM platform description referring to the concepts of Component and Actor.

2.2.1 Component Certification

From a security point of view, MIPAMS requires that each component to be verified and certified in the system before being able to deploy it. Once certified, the component will have a particular X.509 server digital certificate and a private key that will be useful to establish secure communication with other components or actors in the system. The certification process works in the following manner:

(i) An actor's 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;

(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 requestor. The PKCS#12 is protected with the password chosen by the requestor;

(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.

2.2.2 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.

Every actor has associated a 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 that uses 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.

The communication between the client tool and the server part of the system is performed by means 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, as explained in next sections. 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 we could provide a second security layer at the application level.

2.2.3 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 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, as we will see in further sections.

2.2.4 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.

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, depicted in Fig. 6, 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) Supervisor Server verifies the Actor credentials and tool software fingerprint against registered tool fingerprint; (5) Supervisor Server generates a tool unique identifier; (6) 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) Supervisor stores the installed tool fingerprint for future verification purposes; (10) 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 we have already explained in previous sections, the communication between the client tool and the server part of the system is performed by means of a secure channel established by means of the tool certificate and the server component certificate.

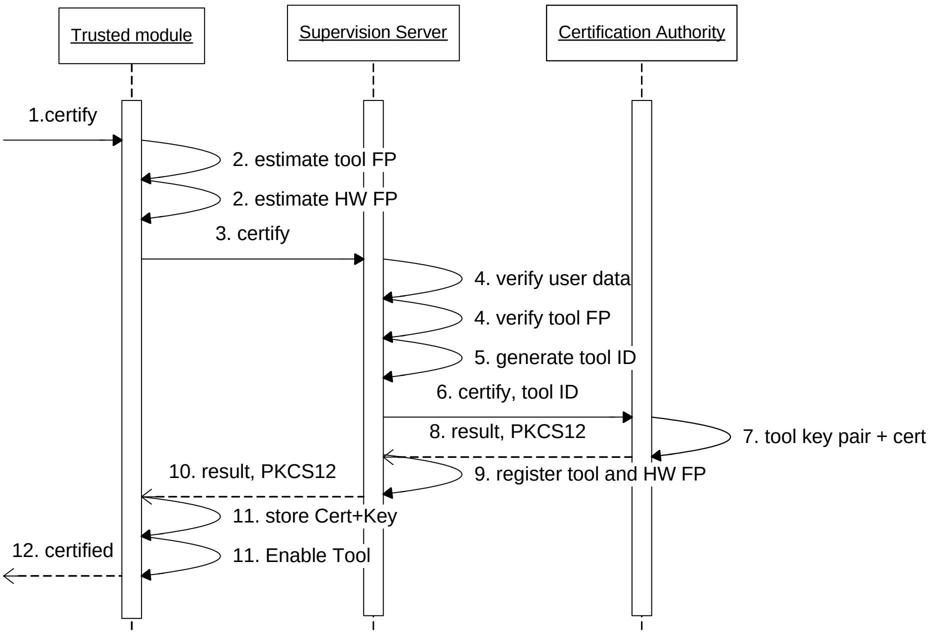

Fig. 6 MIPAMS client tool certification process

2.2.5 Actors and Client components authentication

Any actor in the system is authenticated in two manners, as we have already mentioned: 1) by being able to select its user certificate in the client application; 2) by means of it user identifier, extracted by the application form the certificate.

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

In 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 client application trusted module or the intermediary are responsible for centralizing 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.

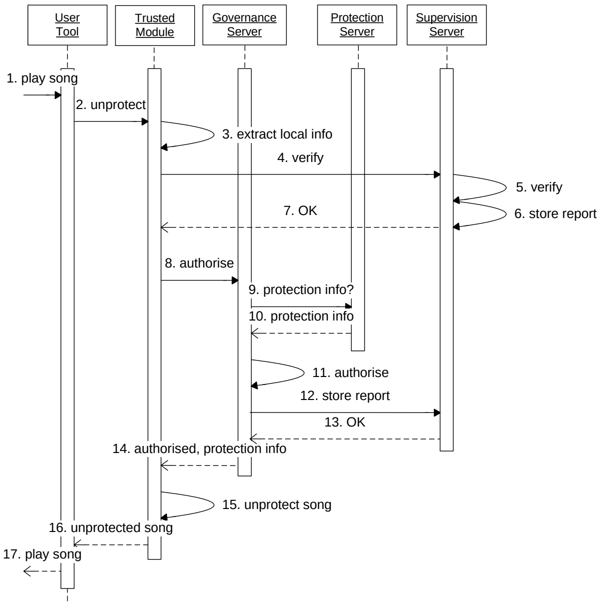

Fig. 7 MIPAMS Content consumption Use Case

Fig. 7 depicts a content consumption scenario, where a user wants to play a protected song. At step 5, 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 song would not be unprotected.

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

2.3 A comparison between OpenSDRM and MIPAMS

On the previous sections the authors have analysed some of the security features of two different open DRM systems. In this new section we can determine some common points and differences and provide some means for them to interoperate at this level.

Regarding component registration and certification, we have seen that both systems enable 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. This point is not a difficult aspect to be overcome, as the result of the process ends to be the same: a X.509 certificate for the component, although in OpenSDRM X.509 certificates are used to certify DRM components, while the different DRM components functionalities are certified by another different certificate [15]. In this way, independently of the registration and certification processes, a component will own a X.509 digital certificate in both systems. We just need to issue compatible certificates for having compatible components at this level.

Regarding component mutual authentication processes, we have presented and explained how both systems perform a client-server mutual authentication based on their component digital certificates. 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, we just need to be sure 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. Something different happens with client components in MIPAMS. Client tools, as we have already explained, 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 authentication process Actor and Component functions authentication, 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 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.

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 XML mapping of X.509 user certificates for DRM 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 authentication processes are somehow shortcut.

3. Interoperability between open DRM systems

During the last few paragraphs we have described and identified some security related aspects of two open DRM architectures (open specification and/or open-source): OpenSDRM and MIPAMS. Across this description the authors have identified some common points and some differences between architectures, which were also pointed in the last section.

In this section, we introduce some directions in terms of interoperability between both DRM systems, on what concerns the registration and authentication aspects. Basically, what we will accomplish 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.

3.1 Registration Interoperability

Both the DRM systems require both the registration of DRM components and Actors. Additionally, OpenSDRM requires also that the DRM components functionalities to the 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 in a common Certification Authority. Nevertheless this is may not be a possible scenario in the real world – many commercially available CAs 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 CAs 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, like web-browsers – if the deployment scenario is more global. If the deployment, in terms of interoperability is restricted or local, an alternative solution may be used (Fig. 8).

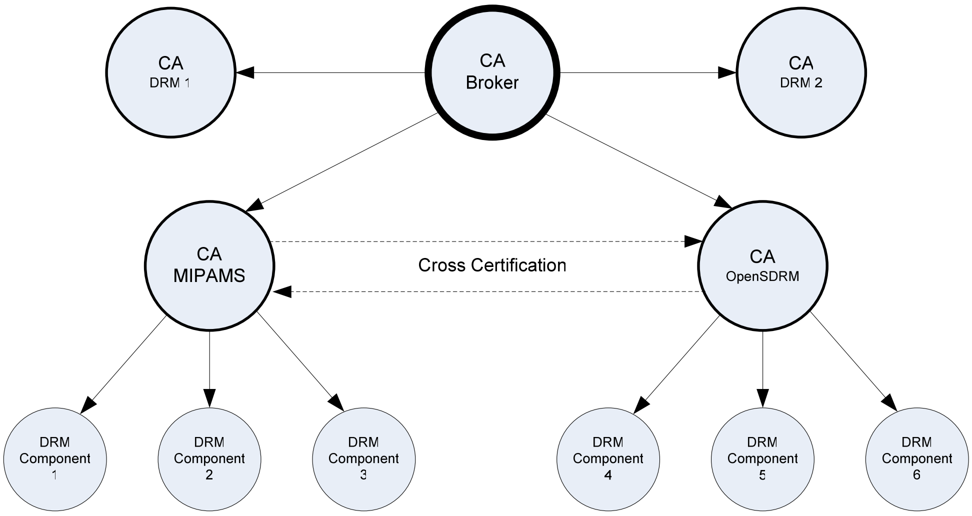

Fig. 8 Scenarios for CA registration interoperation

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

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) that will issue certificates to each of the DRM CAs. Any DRM component registered and certified on one CA would have a common trust point on the certificate certification path: $\mathrm{Cert}_{\mathrm{CABroker}}$. Since the certificates share a common CA, trust will be possible between different DRM components: $\mathrm{Cert}_{\mathrm{CABroker}}^{\mathrm{CA\text{-}MIPAMS}} \Rightarrow \mathrm{Cert}_{\mathrm{CA\text{-}MIPAMS}}^{\mathrm{DRMComponent}}$ and $\mathrm{Cert}_{\mathrm{CABroker}}^{\mathrm{CA\text{-}OpenSDRM}} \Rightarrow \mathrm{Cert}_{\mathrm{CA\text{-}OpenSDRM}}^{\mathrm{DRMComponent}}$.

In the specific case of OpenSDRM, the DRM 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 DRM 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 (Fig. 9).

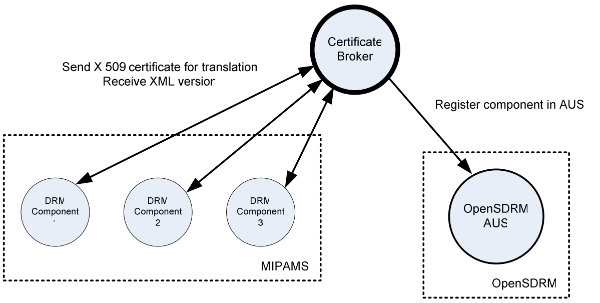

Fig. 9 Bridge between the X.509 and XML certificates

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.

3.2 Authentication Interoperability

In terms of DRM components authentication, what was referred in the previous section also applies here. So there is the need to have a common authentication point between both X.509 certificates that identify the different DRM components (Fig. 8). Again, three possibilities are presented:

1. Both open DRM platforms components have an internal database of the trusted CAs, so that $\mathrm{Cert}_{\mathrm{TrutedCA}}^{\mathrm{MIPAMS\text{-}DRMComponent}}$ and $\mathrm{Cert}_{\mathrm{TrutedCA}}^{\mathrm{OpenSDRM\text{-}DRMComponent}}$ 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 CAs. This would allow that trust between $\mathrm{Cert}_{\mathrm{OpenSDRM}}^{\mathrm{CA\text{-}MIPAMS}} \Rightarrow \mathrm{Cert}_{\mathrm{CA\text{-}MIPAMS}}^{\mathrm{DRMComponent}}$ and $\mathrm{Cert}_{\mathrm{MIPAMS}}^{\mathrm{CA\text{-}OpenSDRM}} \Rightarrow \mathrm{Cert}_{\mathrm{CA\text{-}OpenSDRM}}^{\mathrm{DRMComponent}}$ because they both share the same CA;

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

An important aspect in the DRM component authentication is certificates revocation (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. Has we had the opportunity to enlighten on the previous section, OpenSDRM and MIPAMS differ on this aspect – both in terms of process and credentials format. In terms of format, while in the MIPAMS platform, 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 (Fig. 10).

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 ($\mathrm{Cert}_{\mathrm{AUS}}^{\mathrm{Actor}}$). 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.

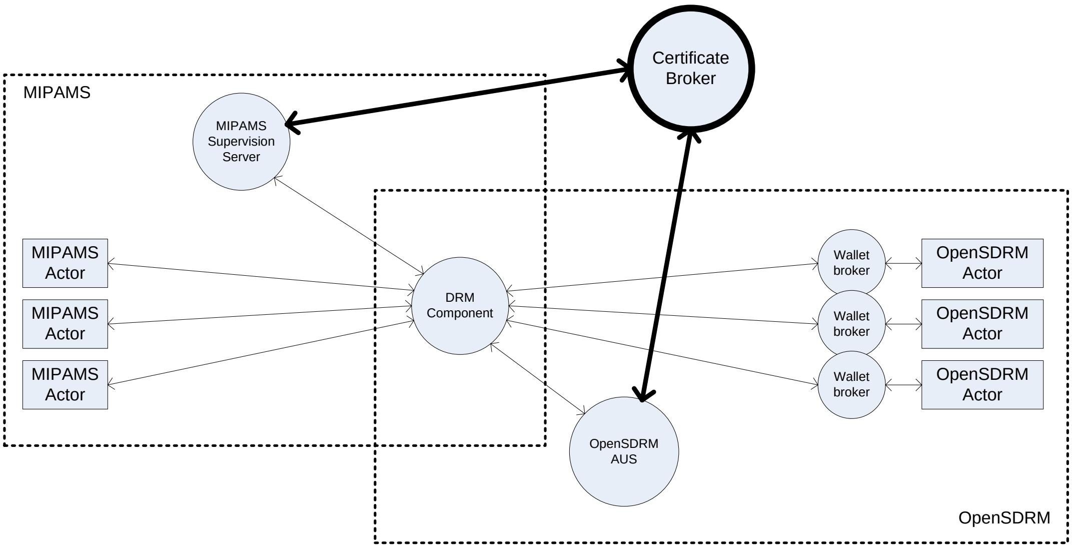

Fig. 10 Interoperability in the Actors authentication

4. Conclusion

DRM is one complex technology that handles the management of digital content usage. DRM is particular complex especially by the fact that many different DRM systems exist and they are incompatible among each other. This is one of the aspects that make DRM as one of the most “non-grata” technologies.

This paper does not try to solve the problem. However it presents some important directions on what concerns the interoperability of certain DRM security-related functions, such as registration/certification and authentication between open DRM platforms. This is just one of the many aspects that need to be solved before we have a truly interoperable DRM scenario.

For this article, we have considered two different open DRM platforms: OpenSDRM is an open specification and open-source implementation of a DRM platform, and MIPAMS an open specification DRM platform.

In this paper we have described the registration and authentication processes in both platforms, and identified the commonalities and differences between them. We have concentrated our work mostly on the differences to derive some processes to make them vanish and to allow interoperability between them.

Although some of the interoperability mechanisms that we have identified refer only to the two studied open DRM platforms, they can be generalized and applied to a broader range of DRM platforms interoperability.

We would like also to acknowledge that this work is just a small step towards a more global and wider DRM interoperability scenario that covers all the different aspects of DRM interoperability and not only the ones considered here.

References

[1] Koenen, R.H., et al, “The long march to interoperable digital rights management”, in Proceedings of the IEEE, 92:883-897, 2004

[2] Schmidt, A, U,.et al “Interoperability challenges for DRM systems”, in International Workshop for technology, Economy, Social and Legal aspects of virtual Goods, Ilmenau, Germany, 2004

[3] ISO/IEC 14496-1:2003, Information technology -- Coding of Audio-Visual Objects -- Part 1: Systems, Amendment 3: IPMP Extensions

[4] MOSES web-site, http://atlantis.tilab.com/projects/moses/, as visited in 30.11.2006

[5] HICOD2000 web-site, http://www.hicod2000.org, as visited in 30.11.2006

[6] WCAM web-site, http://www.ist-wcam.org, as visited in 30.11.2006

[7] MediaNet web-site, http://www.ist-medianet.org, as visited in 30.11.2006

[8] Torres, V., Rodríguez, E., Llorente, S., Delgado, J. Use of standards for implementing a Multimedia Information Protection and Management System. In Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS 2005) (Florence, Italy, Nov. 30 – Dec. 2, 2005). IEEE Computer Society, Los Alamitos Washington Brussels Tokyo, 2005, 197-204

[9] Torres, V., Delgado, J., Llorente, S. An implementation of a trusted and secure DRM architecture. In On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops (IS'06) (Montpellier, France, Oct. 30 – Nov. 1, 2006). Lecture Notes in Computer Science (LNCS) 4277. Springer-Verlag, Berlin Heidelberg New York, 2006, 312-321

[10] Delgado, J., Torres, V., et al. Rights and Trust in Multimedia Information Management. In 9th IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2005) (Salzburg, Austria, September 19-21, 2005). Lecture Notes in Computer Science, vol. 3677. Springer-Verlag, Berlin Heidelberg New York, 2005, 55-64

[11] Distributed Multimedia Applications Group (DMAG), http://dmag.upf.edu, as visited in 30.11.2006

[12] Serrão C., Dias M., Delgado J., "Bringing DRM interoperability to digital content rendering applications", In CISSE05 – The International Joint Conferences on Computer, Information, and System Sciences, and Engineering, Univ. Bridgeport, USA, 10-20 December 2005

[13] Serrão C., Dias M., Delgado J., "Using Web-Services to Manage and Control Access to Multimedia Content", In the 2005 International Symposium on Web Services and Applications (ISWS05), Las Vegas, USA, 2005

[14] 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, Elsevier, 2005

[15] Serrão C., Serra A., Dias M., Delgado J., "Protection of MP3 Music Files Using Digital Rights Management and Symmetric Ciphering", Proceedings of the 2nd International Conference on Automated Production of Cross Media Content for Multi-channel Distribution (AXMEDIS2006), Leeds, UK, 13-15 December, 2006

[16] Serrão C., Fonseca P., Dias M., Delgado J., "The Web Services growing importance for DRM interoperability", In Proceedings of the IADIS International Conference Internet/WWW (ICWI2006), Murcia, Spain, 5-8 October, 2006

[17] 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 (IPDM06), Shangai, China, 18-19 October, 2006

Carlos Serrão received his B.Sc. (5 years degree) in Management and Computer Science from ISCTE (Portugal) in 1996 and his M.Sc. from ISCTE in 2001 (Information Systems Management). He is currently lecturing at ISCTE and participates in several national and international R&D projects and on standardization initiatives. Current research interests involve information security, public-key infrastructures, secure access to multimedia content, access control and digital rights management. Author and co-author of several papers both in Portuguese and international conferences. Co-author of multiple European project deliverables. Author of several books about PHP.

Victor Torres received his B.Sc. in Telecommunication Engineering from Universitat Politècnica de Catalunya (Spain) in 2003 and his M.Sc. from Universitat Pompeu Fabra (Spain) in 2005. Since 2003 he is a member of the Distributed Multimedia Applications Group (DMAG) research group. He has participated in the standardization of MPEG-21 and in several European projects (IP and NoE) in the area of digital rights management (DRM). His research interest includes digital rights management architectures, security, multimedia content and distributed applications.

Jaime Delgado has a Ph. D. in Telecommunication Engineering since 1987. Telecommunication Engineer since 1983. Since September 2006, Professor at the Computer Architecture Department of the Universitat Politècnica de Catalunya (UPC) in Barcelona (Spain). Previously, Professor of Computer Networks and Computer Architecture at the Technology Department, Universitat Pompeu Fabra (UPF), also in Barcelona, since 1999. Head and founder of the Distributed Multimedia Applications Group (DMAG) of the UPF and the UPC. Project Manager of several European and national research projects in the areas of electronic commerce, Digital Rights Management, metadata, multimedia content, security and distributed applications. Active participation, since 1989, in International standardisation, as editor of standards and chairman of groups in ISO/IEC, EWOS, ETSI, ITU-T and CEN/ISSS. Evaluator and reviewer for the European Commission in different research programs since 1989. Advisor for the Spanish Ministry of Science. Author of more than 100 published papers and books, and member or chairman of many Conference International Programme Committees.

Miguel Dias holds a PhD (1998) in Sciences and Information Technologies, field of Computer Graphics and Multimedia, at ISCTE. Holds a Msc (1988) in Electrical and Computer Engineering, area of Electronics, at IST-UTL, Instituto Superior Técnico, Universidade Técnica de Lisboa (Technical University of Lisbon). Holds a B.Sc. (1985) in Electrical Engineering, field of Telecommunications and Electronics, at IST-UTL. Director of MLDC, Microsoft Language Development Center (since November 2005) Associated professor of the Department of Sciences and Information Technologies at ISCTE; Past President of ADETTI; Past Vice-President since 1988. Lectures undergraduates and graduates at ISCTE in variety of courses. His main research interests are: Multimodal User Interfaces, Speech and Natural Language Computer Graphics namely, Augmented and Mixed Reality, Digital Rights Management, JPEG2000 Digital Image Standard, Computer Vision. He is a member of the editorial board of the on-line Virtual Journal, and of the Eurographics Computer Graphics Educational Materials Source, all dealing with Computer Graphics. He is the Vice-President of the Eurographics Portuguese Chapter and he is member of several Programme Committees of National and International conferences in Computer Graphics, Virtual and Augmented Reality. He is regularly commissioned by the EC for R&D project evaluations and reviews. Since 1992, he has participated in National and International R&D International research projects. Author of 1 patent and more than 90 papers in national and international conferences and journals, being 17 indexed in ISI. Co-author of more than 70 European project deliverables.