Open Secure Infrastructure to control User Access to multimedia content

Carlos Serrão
carlos.serrao@adetti.iscte.pt
Adetti/ISCTE
Ed. ISCTE – Av. Das Forças Armadas, 1600-082 Lisboa, Portugal

Abstract

This paper will start by describing OpenSDRM an open-source framework developed for the IST project MOSES., OpenSDRM is used to control the multimedia content consumption in conjunction with the new IPMPX MPEG-4 proposed standard. This architecture, composed by several building blocks, protects the content flow from creation to final user consumption on a specific device.
This paper devotes a special attention to the security aspects of the OpenSDRM processes and functions, describing its communication protocols and message exchanges as well as it introduces the security details about the user's digital wallet.

Introduction

The modern content distribution business models such as Internet download and super distribution have showed massive flaws and drawbacks. Traditional approaches of content protection have to be reassessed, modified and extended. This change in the value chain affects mostly the music and film industry. Mass storage and copy systems have become cheaper, Internet bandwidth at the final user home has increased as prices for maintaining fell and the content compression technologies have evolved in such a way that created a greater interest for potential illegal use.
Different approaches to protect content exist [1]. Currently the music industry is trying to identify illegal copiers who are using the file sharing systems (P2P). Providers are forced by court to open their user logs and allow externals to prosecute their illegal users. The principle of anonymity and the protection of personal information become highly questionable. On the other hand this approach will never cope with the complete number of illegal distributions, since the number of users that are taking part in these systems, are diminishing the probability of being caught and consequently the prevention of illegal media distribution. Therefore other approaches must be found. These can only work by controlling either the distribution of the content via trusted channels or the protection of the content with access control (or the combination of both). A form of secure distribution would take place in a secured network, which is not a valid assumption since communication nowadays uses mostly insecure and open networks that can be sniffed and trapped. Access control hardware has been applied in broadcasting systems such as Pay-TV using set-top boxes, however most of these systems use hardware solutions, such as smart cards trusting them as a medium that is tamper resistant. Not all potential client devices provide an easy and built-in integration with smart cards: PCs, PDAs, PocketPCs, mobile phones are some examples of the absence of this integration, although in the mobile phones provide similar functionalities of the smart card. Software solutions to provide conditional access and rights management are implied to be used on these systems. The described approach requires a complete framework for content protection and conditional access. Protection methodologies, such as MPEG - Intellectual Property Management and Protection Extensions (IPMP-X) [2] and Windows Media (WM) [3] are two examples for content protection providing different approaches according to the disclosure of the software, the first is open while the second is mostly closed. Both are using sophisticated protection methods, such as watermarking and encryption to achieve a retraceable content distribution and protection. Both systems, regardless its closeness or openness rely on the underlying distribution platforms and finally on operating systems. An analysis of open and closed operating systems showed that these risks exist but are different in open and closed systems [4]. Using proprietary software to achieve the goals of content protection bears one problem: interoperability. This lack of interoperability may be extensible to the content itself. Open systems try to solve this problem by open standards, such as MPEG and may use open source software for their implementations. An example, which will be addressed in this paper, is the OpenSDRM system [5], which is currently being developed in the IST-Project MOSES.

The paper explains how this system facilitates a complete secure framework for content distribution over the Internet, focusing on the security features and protocols. OpenSDRM is independent from the type of content or protection applied. It uses multiple communication protocols and is based on the emerging web-services paradigm (WSDL, UDDI, SOAP) approach. The system shows how open software such as communication libraries (SOAP) can be combined with an open standard, such as MPEG-IPMPX. The aim shall be to create a complete DRM framework, which covers all phases of the content lifecycle, beginning from content authoring, distribution and management of the related rights for the final user.

OpenSDRM description

The Open SDRM architecture is adaptive [6], since it can be configured for use with several business models and different types of content. OpenSDRM deploys a traditional DRM solution for content rights protection and can be applied for publishing and trading of digital multimedia content. Additionally, the security architecture proposed started from the OPIMA international specifications [1], MPEG-4 IPMP Extensions [2] and the emerging MPEG-21 IPMP architecture [8] as well as with some of the proposals for JPEG2000 standard Part 8 – JPSEC – JPEG2000 security. OpenSDRM is being developed primarily in the scope of the MOSES project. MOSES is an EC project joining companies from all over Europe that is implementing the new MPEG-IPMP Extensions framework and at the same time developing business models and applications for secure content exchange between embedded devices [10]. This DRM solution is composed of several optional elements covering the content distribution value chain, from content production to content usage. It covers several major aspects of the content distribution and trading: content production, preparation and registration, content, interactive content distribution, content negotiation and acquisition, strong actors and user's authentication and conditional visualization/playback [11]. Even though the MOSES project refers explicitly to MPEG-4 file format as the content format, this infrastructure was designed with the concern to be adaptable and applicable to all types of content and business models (for download, streaming or even broadcasting). Figure 1 shows the architecture that will be explained in the next section in greater detail.

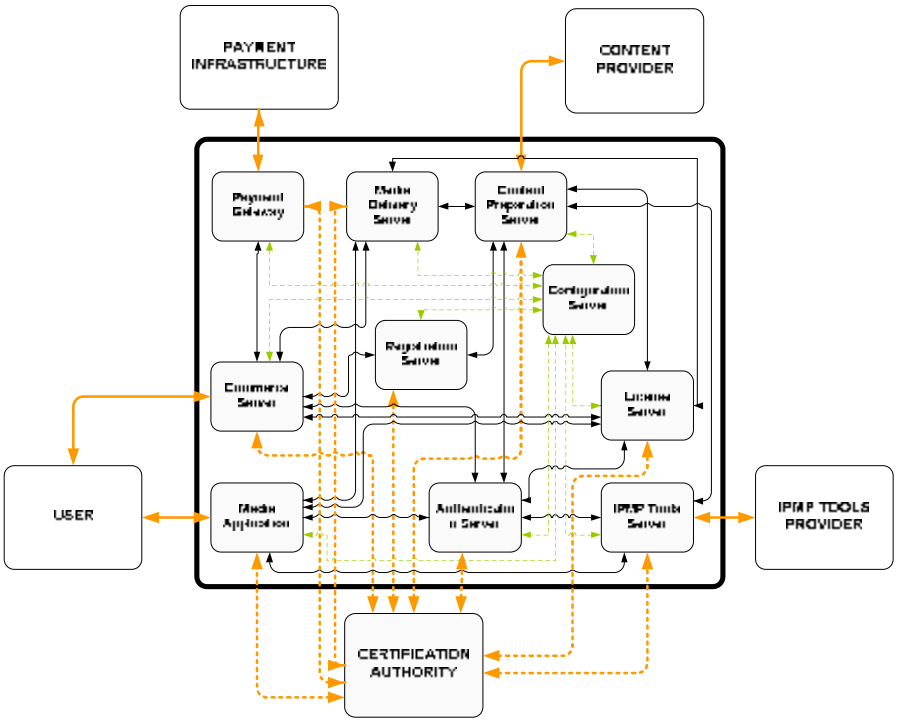

Figure 1 - OpenSDRM General Architecture

External Components & Interfaces

This part will present in detail the components and actors that interact externally with the OpenSDRM

architecture: User, IPMP Tools Provider, Content Provider, Payment Infrastructure and Certification Authority.

User represents a person who wishes to consume a piece of 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 Open SDRM in order to: identify him, download licenses and play multimedia using a web browser, EPG, Media Player. In a final analysis the User interaction with OpenSDRM will always result in one of two things: either the user can play/render the content and enjoy it or he can't; being then informed of the reason for this prevention.

IPMP Tools Provider is any organization that produces 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 rights protection. These tools will need to comply with some guidelines. These guidelines and a subscription translates into a business relation that must exist between a given Content Provider and the IPMP Tools Provider, since mostly, a given producer and/or distributor of content may want to choose which type of protection the content will have and respectively which tools can be applied to the content and from which supplier.

Content Provider is any multimedia content supplier that feeds OpenSDRM with content and optional metadata. The content can be complex multimedia content that is ready for distribution, or simple content, for example JPEG images, that can be edited and combined with other content. Currently, in the MOSES project we are addressing MPEG-4 content.

Payment Infrastructure facilitates OpenSDRM e-commerce features by providing services for handling electronic payments. The interface between OpenSDRM and the Payment Infrastructure is generic and independent of the payment method, allowing therefore a multiplicity of payment systems.

Certification Authority is responsible for receiving requests for and issuing credentials to, entities. These credentials will be used by entities to authenticate themselves to each other, allowing the establishment of secure and authenticated communication channels between them. All the components in the OpenSDRM architecture communicate using the channel security provided by the SSL/TLS protocol. This Certification Authority may be internal to OpenSDRM, and therefore entirely managed by some entity, or it may be an external commercial Entity.

Internal Components & Interfaces

In this part, the internal components of the OpenSDRM platform and the corresponding interfaces are presented. These components include: Media Application, Media Delivery Server, Commerce Server, Authentication Server, License Server, IPMP Tools Server, Registration Server, Content Preparation Server and the Payment Gateway.

Content Preparation server (CPS) this server component is responsible for the content preparation. It receives raw content from a specified source or sources and encodes it on a specified format, adds metadata and protects it. Currently, and under the MOSES project, content will be encoded in MPEG-4 format, according to some pre-established templates. These templates will allow the creation of MPEG-4 files containing music files in MP3 or AAC format together with some JPEG images about the album and artist.

Payment Gateway (PGW) is a server component responsible for verifying and validating the payment methods provided by the User for a Commerce Server;

Commerce server (COS) is a server component responsible for trading the content with the users. Normally, content is chosen via web browser, some very generic metadata might be consulted, information about the price is also available, and especially the content usage conditions might be established.

Media Delivery server (MDS) is a server component responsible for exchanging pieces of content with the client. This Media Delivery server will implement a specific protocol (download (FTP, HTTP, or other), streaming (RTSP, other), broadcast) to exchange protected content with the client application.

Registration server (RGS) is a server component whose role is to assign unique identifiers to content and to register metadata information for that specific content. This architecture will be as close as possible to standards and therefore for this unique ID, it follows the MPEG-21 directives about Digital Item Identification (DII), using a reduced version of the MPEG-21 DII Digital Object Identifiers (DOI) [12].

Authentication server (AUS) is responsible for authenticating all the entities, internal and external to the DRM system. It validates the access rights of all the entities and components in the system working as a SSO point, 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 (XML Encryption and XML Signatures).

License server (LIS) is a server component responsible for house-keeping the rules associating a user, the

content and his/her corresponding access rights. This component will accept connections from authenticated client Media Players for downloading of licenses, which will be applied to the protected content through an appropriate IPMP tool. The licenses are XML formatted using Open Digital Rights Language (ODRL/OMA profile) [13], and, in the future, they will migrate to the Rights Expression Language (REL) [14], currently being developed by MPEG-21.

IPMP tools server (ITS) is the server component responsible for registering new IPMP tools and for receiving authenticated client Media Player requests for the downloading of a specific IPMP tool. It is also responsible for making IPMP tools available to the Content Preparation Server to allow the protection of content.

Media Application (MPL) This component represents the software that will be used to render the content. This is a generic component with the particularity of being able to display/playback the appropriate 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 work with one or several IPMP tools in order to control how the content is accessed by a particular user. This component works on the client side of the general architecture; however it plays an important role in the DRM functions.

In the next sections some security functions are described to implement the open DRM approach used on the system.

OpenSDRM Security

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.

An underlying concept behind the OpenSDRM platform is the existence of two security layers as displayed in Figure 2. A first security layer is established at the communication level, which will provide the necessary secure and authenticated communication medium 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.

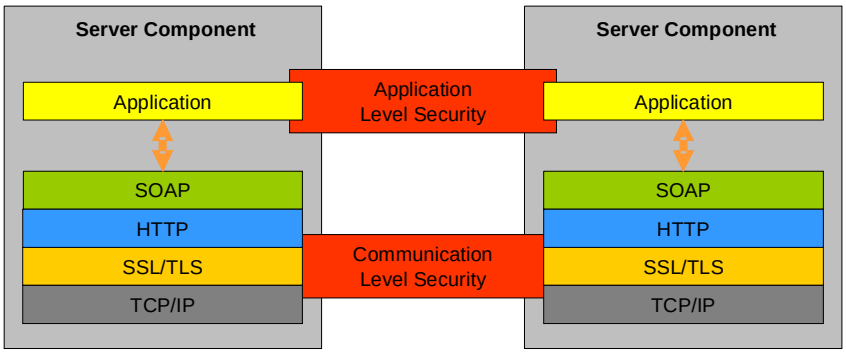

Figure 2 - OpenSDRM Security Layers

Each of the messages exchanged between the different components share a common structure. The format as shown in Figure 3 is composed of:

- **Component ID**: a 128 bits identifier (generated by an MD5 hashing algorithm) that identifies uniquely this component. This identifier was issued by the AUS;
- **Message Contents**: this is a set of different fields of the message, and typically it is different from component to component;
- **Digital Signature**: the digital signature of the message, to avoid the message contents tampering.

SOAP message

Figure 3- SOAP Message Structure

Server Components Certification

In order to establish the secure transport layer, the software components of the OpenSDRM architecture, use the SSL/TLS protocol to ensure such functionality. Each of the servers, on which the software components are installed, need to have a X.509 certificate issued by a Certification Authority (CAU). If more than one of the OpenSDRM components are installed on the same server then such one of this certificates are necessary. The CAU can be operated 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.

- Each component computes a key pair (public and private) , K_{pub}^{Server}, K_{priv}^{Server}, using the RSA algorithm and create Certificate Signing Request (CSR) using its public key and some additional information sending it after to the CAU
- The CAU verifies the CSR validity and issues the X.509 SSL certificate to the appropriate component, $Cert_{X.509}^{Server}$;
- The X.509 SSL certificate is installed and the components can use SSL/TLS to communicate, establishing therefore the secure transport layer.

Registration of Components on OpenSDRM

The architecture requires that both components and Users on the OpenSDRM architecture to be registered, in order to establish the Application/Transaction level security.
Concerning components (COS, PGW, RGS, LIS, ITS, 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 uses a 1024 bit length RSA keys, but higher key lengths are also possible): $Kpub^{Component}$, $Kpriv^{Component}$ (respectively the public and private keys);
- The component administrator selects a login and a password, and ciphers the $Kpriv^{Component}$, using AES, with the key (K_{AES}) deduced from the hash of the concatenation of the login and password selected: K_{AES} := MD5(login+password). The ciphered component private key gets then protected from unauthorized usage: $K_{AES}[Kpriv^{Component}]$.
- The component then connects to the AUS and sends some registration information together with the $Kpub^{Component}$.

AUS verifies the information sent by the component, validates and registers it, and issues a certificate for the component: $Cert_{AUS}^{Component}$. This certificate contains, among other information, a unique identifier of the component and the public key. This certificate is return to the component.
With these component certificates, each of the components will be able to establish trust relationships among them and sign and authenticate all the transactions – this establishes then the Application Level security.

User's registration on the OpenSDRM platform

In OpenSDRM three components interact directly with external users/entities – MPL, CPS and ITS. These users, respectively Content Users, Content Providers and IPMP Tools Providers are registered on the platform, through the AUS.
Content Providers and IPMP Tools Providers, subscribe respectively on the CPS and ITS, relying on the registration and authentication functionalities of the AUS. 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 components (ITS and CPS) gather the new registrant information (Info) and request the registration of a new user on the AUS;
- The components build a new message: $SignKpriv^{Component}\{Component_{ID}, Info\}$. This message is send to AUS;
- AUS verifies and validates the message, registering the new User and returning a unique $User_{ID}$ to the component.

Registering a Content User is a more complex process. This is due to the fact that while both Content Providers and IPMP Tool Providers have their information stored on remote servers, Content Users rely on

their own platforms to store their data. In order to provide some additional degree of security, OpenSDRM provides a digital wallet, capable of storing sensitive information such as cryptographic data and licenses in a secure way. The process to register new Content Users can be described in the following steps:

- When the user runs the wallet for the first time, it creates the User a RSA key pair ($Kpriv^{User}$, $Kpub^{User}$) and asks the user to enter a login and a password;
- 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 ($Person_{Data}$) and also some payment data (Pay_{Data}) 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 $Kpub^{AUS}$: $Kpub^{AUS}[Person_{Data}, Pay_{Data}, KPriv^{User}, Kpub^{User}]$;
- AUS receives the data, deciphers it and registers the User. AUS responds to the Wallet with a new certificate generated for the User: $Cert_{AUS}{}^{User}$, containing among other information the unique identifier of the User, its public key, the identification of the AUS its signature;
- The wallet stores all the relevant information on the secure repository: $K_{AES}[Cert_{AUS}{}^{User}]$.

Components message exchange

The process for the components to exchange messages and to verify the authenticity and validity of such messages is composed of the following steps:

- The sender component (CSender) composes a message using the following syntax: $SignKpriv^{CSender}\{CSender_{ID}, Payload, Cert_{AUS}{}^{CSender}\}$;
- The receiver component (CReceiver) receives the message and verifies the trust on the message. This trustability is assured in the following way:
- CReceiver gets $Cert_{AUS}{}^{CSender}$ and checks if it was issued by a AUS in which CReceiver trusts.
- This verification can be conducted if CReceiver has also a certificate issued by AUS: $Cert_{AUS}{}^{CReceiver}$.
- After the trust is established, the message signature can be verified and validated and CReceiver can trust its contents, and also in the component who has sent this message;
- CReceiver can then process the message payload and return its results for the CSender;
- CReceiver returns the following message to CSender: $SignKpriv^{CReceiver}\{CReceiver_{ID}, Results, Cert_{AUS}{}^{CReceiver}\}$.

Payment information

Payment of content usage is one of the questions that OpenSDRM also deals and incorporates mechanisms for payment, although the payment method is outside the scope of the OpenSDRM 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:

- The COS connects to the AUS and asks the AUS which are the PGW available on the system. COS sends $SignKpriv^{COS}\{COS_{ID}, RequestAvailablePGWs\}$ to AUS;
- AUS verifies the message, and returns an answer to the COS: $SignKpriv^{AUS}\{<ListOfAvailablePGWs, Cert_{AUS}{}^{PGW}>\}$;
- The COS selects one available PGW and sends to it a subscription request: $SignKpriv^{COS}\{AUS_{ID}, SubscribePGW, Cert_{AUS}{}^{COS}\}$;

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.

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:

- The COS sends information about the payment details, namely information about the User order and the price to pay for it, to AUS: $SignKpriv^{COS}\{COS_{ID}, U_{ID}, PGW_{ID}, PayData\}$;
- AUS verifies and validates the COS request and checks the UID 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: $Kpub_{PGW}[PaymentClearence_U]$;
- The AUS returns this information for the COS, signing it: $SignKpriv_{AUS}\{Kpub_{PGW}[PaymentClearence_U]\}$;
- This information is then passed by the COS to the PGW, requesting it to validate the payment transaction: $SignKpriv_{COS}\{COS_{ID}, Kpub_{PGW}[PaymentClearence_U]\}$;
- PGW validates the message and deciphers the User payment clearance, using this information to communicate to the corresponding Payment Infratructure, validating it. After, the PGW returns the result of the payment validation to the COS: $SignKpriv_{COS}\{PGW_{ID}, Transaction_{ID}\}$;

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.
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 $Transaction_{ID}$. The capture process can be described in the following:

- COS sends a message to PGW: $SignKPriv_{COS}\{COS_{ID}, Transaction_{ID}\}$;
- PGW validates the message and verifies the $Transaction_{ID}$, in order to evaluate if that transaction is in fact pending, and processes the payment;
- PGW returns and a result status to the COS: $SignKPriv_{PGW}\{PGW_{ID}, Transaction_{ID}, Result\}$.

License Production

One of the major functionalities of the OpenSDRM platform resides on the fact that it can control 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 client player by the appropriate set of IPMP tools. These licenses are produced and stored securely by the LIS, according to the choices made by the User and after the payment has been performed. The process can be described in the following steps:

- 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;
- COS sends a message to the LIS, requesting the production of a new license, for a specific content, and for a given User: $SignKpriv_{COS}\{U_{ID}, Content_{ID}, LicenseConditions, Cert_{AUS}^{COS}\}$;
- LIS receives the request, verifies it and validates it. LIS generated the license using the appropriate language and parameters, contacting after the AUS for ciphering the license data for the User: $SignKpriv_{LIS}\{License\}$;
- AUS receives the data, retrieves the $Kpub_{User}$ and ciphers the received data: $Kpub_U[License]$, returning it afterwards to the LIS: $SignKpriv_{AUS}\{Kpub_{User}[License]\}$;
- LIS stores $Kpub_{User}[License]$;

License Download

When the User tries to access the content on the client side the player verifies that a license is needed to access the content. The player contacts the wallet to try to obtain the required licenses and corresponding keys to access the content. This process can be described in the following steps:

- The player contacts the wallet to obtain the license for the ContentID and UserID;
- The wallet checks on its secure repository if a license for that specific ContentID is already there. If that is true than this license is returned for the player in order for the content to be deciphered and accessed, controlled by a set of IPMP tools. If the wallet doesn't contain the license, it will request it from the LIS: $SignKpriv_U\{Cert_{AUS}^{User}, Content_{ID}\}$;
- LIS receives the data, validates it and retrieves the license from the database, passing it to the wallet: $SignKpriv_{LIS}\{Kpub_{User}[License], Cert_{AUS}^{LIS}\}$;
- The wallet receives the data from the LIS, validates the message and deciphers the license that is passed to the player. Also the license is stored on the wallet 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.

License Expiry

Depending on the rights specified, a license will eventually expire. Rights such as a play count or a 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 digital wallet. Upon expiration, for example when a play count reaches zero, the wallet automatically checks at the LIS for a new license for that particular content. If there is no license available and the user wants to continue with the consumption of the content he has 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 will be removed from the LIS.

Conclusions

This paper presented and discussed an open platform for the multimedia content IPR, introducing some complexities and difficulties about digital IPR management and also the components which compose it. Throughout it we presented some of main aspects, in terms of security, of the platform. The focus was mostly in the description of the security protocol and secure message exchanging which is established among the different components, leaving out security aspects such as network configuration, server configuration and management [6]. OpenSDRM relies on text-based communication. Therefore it defines a two layered security protocol: (a) a secure transport protocol (SSL/TLS) assuring the different components exchanging messages the security and authenticity of the communication channel and (b) an application/transaction security layer, which is specific of the OpenSDRM operations [5].

OpenSDRM, contrarily to the normal operation followed by other DRM solutions, addresses DRM using an open approach, following open standards and open-source software. We strongly believe that protection of digital multimedia IPR can be achieved through strong cryptography and authentication techniques such as the one presented on this paper .

Finally, it is important to stress the fact that although OpenSDRM is mostly an open-standards and open-source based solution, this doesn't prevent that some parts of the system may be closed. An example of this is the fact that the authoring tools used to protect the content itself may be closed. Protection tools, such as watermarking algorithms and specific scrambling or encryption algorithms may be closed, although they are used on an open environment such as OpenSDRM.

References

[1] Chiariglione, L., "Intellectual Property in the Multimedia Framework", Management of Digital Rights, Berlin, 2000

[2] Jack Lacy, Niels Rump, Panos Kudumakis, "MPEG-4 Intellectual Property Management & Protection (IPMP) - Overview & Applications Document", ISO/IEC JTC1/SC29/WG11/N2614, 1998

[3] Andrea Pruneda: Windows Media Technologies: Using Windows Media Rights Manager to Protect and Distribute Digital Media, MSDN Magazine, December 2001

[4] Carlos Serrão, Daniel Neves, Paulo Trezentos,"Open Source Security Analysis: Evaluating security of Open Source Vs Closed Source Operating Systems", ICEIS 2003, Angers, France, 23-26 April 2003

[5] Carlos Serrão, Daniel Neves, Panos Kudumakis, Trevor Barker, Massimo Balestri, "OPEN SDRM – AN OPEN AND SECURE DIGITAL RIGHTS MANAGEMENT SOLUTION", IADIS 2003, Lisboa,

[6] Gregor Siegert, Carlos Serrão, "An Open-Source Approach to Content Protection and Digital Rights Management in Media Distribution Systems", ICT Conference 2003, Copenhagen December 2003

[8] Jan Bormans, Keith Hill , "MPEG-21 Overview v.4", ISO/IEC JTC1/SC29/WG11/N4801, 2002

[10] J.Duhl, S.Keroskian "Understanding DRM Systems", Whitepaper 2001

[11] "Digital Rights: Background, Systems, Assessment", Commission Staff Working Draft, Commision of the European Communities, 2002

[12] Bill Rosenblatt, "Solving the Dilemma of Copyright Protection Online", JEP Journal December 1997 vol3, ISSN 1080-2711

[13] Open Mobile Alliance "Generic Content Download Over The Air Specification", v1.0 December 2002

[14] Multimedia Description Schemes (MDS) Group, "MPEG-21 Rights Expression Language WD V3", ISO/IEC JTC1/SC29/WG11/N4816, 2002