Open Secure Infrastructure to control User Access to multimedia content

Carlos Serrão[1], Gregor Siegert[2]

[1] Adetti/ISCTE, Ed. ISCTE – Av. Das Forças Armadas, 1600-082 Lisboa, Portugal

[2] Avanti Communications, 28-30 Hoxton Square, London N16NN United Kingdom

Abstract. This paper describes the OpenSDRM security, based on an open-source framework developed for the IST project MOSES, OpenSDRM is used to control the multimedia content consumption in conjunction with the new MPEG-4 IPMPX proposed standard. This architecture, composed by several building blocks, protects the content flow from creation to final user consumption on a specific device.

1 Introduction

OpenSDRM deploys a secure and distributed DRM solution for content rights protection that can be applied for publishing and trading of digital multimedia content. This architecture started from the OPIMA international specifications [1], MPEG-4 IPMP Extensions [2] and the emerging MPEG-21 IPMP architecture [6]. OpenSDRM is being developed primarily in the scope of the MOSES project, an EC project joining companies from all over Europe, implementing the new MPEG-IPMP Extensions framework and at the same time developing business models and applications for secure content exchange between embedded devices [3, 5]. This 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 [3]. Figure 1 shows the architecture that will be explained in the next section in greater detail. The communication between the components will take place within insecure networks. This introduces special needs regarding the security of this communication. The concept behind the platform is the existence of two security layers. A first security layer established at the communication level, which provides the necessary secure and authenticated communication medium to components to communicate with each other and a second layer established at the application level, ensuring the security, integrity, authentication and non-repudiation mechanisms needed by the different components.

Serrão C. and Siegert G. (2004).
Open Secure Infrastructure to control User Access to multimedia content.
In *Proceedings of the 2nd International Workshop on Security in Information Systems*, pages 180-185
DOI: 10.5220/0002672501800185

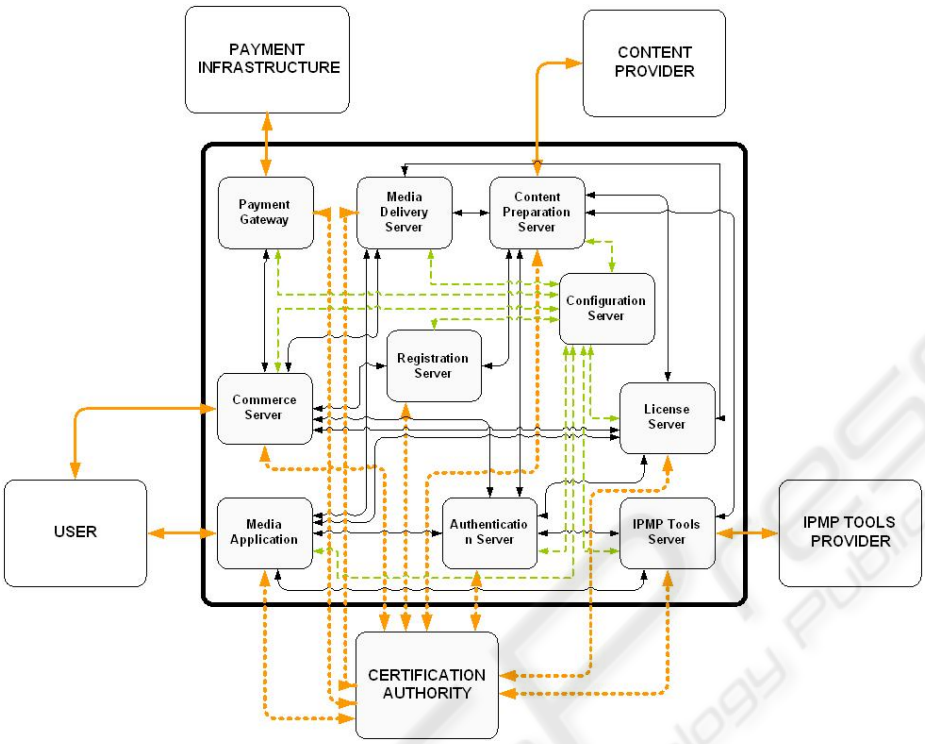

Fig. 1 - OpenSDRM platform architecture composed of several external (User, Payment Infrastructure, Content Provider, IPMP Tools Provider, Certification Authority) and internal components (Payment Gateway (PGW), Media Delivery Server (MDS), Content Preparation Server (CPS), Commerce Server (COS), Registration Server (RGS), Configuration Server (CFS), License Server (LIS), Media Application (MPL), Authentication Server (AUS) and IPMP Tools Server (ITS)).

1.1 Server Components Certification and Registration on OpenSDRM

To establish the secure transport layer, the software components use SSL/TLS protocol. Each of the servers, have a X.509 certificate issued by a Certification Authority (CAU). The CAU can be operated internally by OpenSDRM itself or can be an external and commercial one. OpenSDRM establishes an underlying secure and authenticated transport channel that allows messages to flow from component to component securely. The process works this way: (a) 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; (b) The CAU verifies the CSR validity and issues the X.509 SSL certificate to the component, $Cert_{X.509}^{Server}$; (c) The certificate is installed and the components can use SSL/TLS to communicate, establishing the secure transport layer. The architecture requires both components and Users to be registered, in order to

establish the Application/Transaction level security. Concerning components 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: (a) Each component computes a key-pair (1024 bit length RSA keys, but higher key lengths are also possible): $Kpub^{Component}$, $Kpriv^{Component}$ (respectively the public and private keys); (b) 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: $K_{AES}[Kpriv^{Component}]$; (c) 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 is returned 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.

1.2 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: (a) The components (ITS and CPS) gather the new registrant information (Info) and request the registration of a new user on the AUS; (b) The components build a new message: $SignKpriv^{Component}\{Component_{ID}, Info\}$. This message is send to AUS; (c) 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: (a) 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; (b) 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: $K_{AES}[Info]$; (c) 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; (d) The wallet requests the AUS to register a new User, sending all the information ciphered with the AUS $Kpub^{AUS}$: $Kpub^{AUS}[Person_{Data}, Pay_{Data}, KPriv^{User}, Kpub^{User}]$; (e) 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; (f) The wallet stores all the relevant information on the secure repository: $K_{AES}[Cert_{AUS}^{User}]$.

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

1.4 Payment information and services

Payment of content usage is one of the questions that OpenSDRM also deals and incorporates mechanisms for payment. To provide this functionality a direct trust relationship must be established between the COS and the PGW. Therefore the COS needs to subscribe a PGW. The process to subscribe a PGW can be described as the following: (a) 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; (b) AUS verifies the message, and returns an answer to the COS: $SignKpriv^{AUS}\{<ListOfAvailablePGWs, Cert_{AUS}^{PGW}>\}$; (c) 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. Using the payment service provided in OpenSDRM involves two steps: validating the payment instrument and capturing the payment. Validating the payment instrument allows the COS to trust the payment method supplied by the user, and that the transaction can be conducted without problems. Validating the payment involves the following steps: (a) The COS sends information about the payment details, namely information about the User order and the price to pay for it, to AUS: $SignKpriv^{COS}\{COS_{ID}, U_{ID}, PGW_{ID}, PayData\}$; (b) 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: $Kpub_{PGW}[PaymentClearence_U]$; (c) The AUS returns this

information for the COS, signing it: $SignKpriv_{AUS}\{Kpub_{PGW}[PaymentClearence_U]\}$; (d) This information is then passed by the COS to the PGW, requesting it to validate the payment transaction: $SignKpriv_{COS}\{COS_{ID}, Kpub_{PGW}[PaymentClearence_U]\}$; (e) 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, assuring the COS that the services supplied to the User will be 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 owns a $Transaction_{ID}$. The capture process can be described in the following: (a) COS sends a message to PGW: $SignKPriv_{COS}\{COS_{ID}, Transaction_{ID}\}$; (b) PGW validates the message and verifies the $Transaction_{ID}$, in order to evaluate if that transaction is in fact pending, and processes the payment; (c) PGW returns and a result status to the COS: $SignKPriv_{PGW}\{PGW_{ID}, Transaction_{ID}, Result\}$.

1.4 License Production, download and expiry

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: (a) 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; (b) 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, $Cert_{AUS}^{COS}\}$; (c) 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: $SignKpriv_{LIS}\{License\}$; (d) AUS receives the data, retrieves the $Kpub_{User}$ and ciphers the received data: $Kpub_U[License]$, returning it afterwards to the LIS: $SignKpriv_{AUS}\{Kpub_{User}[License]\}$; (e) LIS stores $Kpub_{User}[License]$. 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: (a) The player contacts the wallet to obtain the license for the ContentID and UserID; (b) 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: $SignKpriv_U\{Cert_{AUS}^{User}, Content_{ID}\}$; (c) LIS receives the data, validates it and retrieves the license from the database, passing it to the wallet: $SignKpriv_{LIS}\{Kpub_{User}[License], Cert_{AUS}^{LIS}\}$; (d) 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. 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.

2 Conclusions

This paper presented and discussed the security aspects of an open platform for the multimedia content IPR. The focus was mostly in the description of the security protocol and secure message exchanging which is established among the different components [4]. OpenSDRM relies on text-based communication. Therefore it defines a two layered security protocol [3]. OpenSDRM, contrarily to the normal operation followed by other DRM solutions, addresses DRM using an open approach, following open standards and open-source software. 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. 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
4. "Digital Rights: Background, Systems, Assessment", Commission Staff Working Draft, Commision of the European Communities, 2002
5. Open Mobile Alliance "Generic Content Download Over The Air Specification", v1.0 December 2002
6. Multimedia Description Schemes (MDS) Group, "MPEG-21 Rights Expression Language WD V3", ISO/IEC JTC1/SC29/WG11/N4816, 2002