License Transfer in OMA-DRM

Cheun Ngen Chong[1], Sorin Iacob[2], Paul Koster[1],
Javier Montaner[3], and René van Buuren[2]

[1] Philips Research, Prof. Holstlaan 100, 5656 AA Eindhoven, The Netherlands
{jordan.chong,r.p.koster}@philips.com
[2] Telematica Instituut, P.O. Box 589, 7500 AN Enschede, The Netherlands
{sorin.iacob,rene.van.buuren}@telin.nl
[3] Vodafone Group Research & Development, Gelissendomein 5, 6201 BM Maastricht, The Netherlands
javier.montaner@vodafone.com

Abstract. Many novel usage scenarios for protected digital content are emerging. Some of these are content transfer, by which users give away, borrow, or re-sell their protected content to other users. The transfer must abide by the rules defined by a rights issuer. The concept of authorized domain complicates the transfer operation. The main contribution of this paper is that we present a full-fledged design and implementation of the transfer operation for one of the major DRM standards, namely Open Mobile Alliance (OMA) DRM. The transfer operations include device compliance check, transfer between two devices (from two different domains), and license revocation within a domain. Our approach extends the user experience in using the DRM-protected digital content.

1 Introduction

Many novel digital content usage scenarios of Digital Rights Management (DRM) [1, 17] are emerging. The concept of authorized domain is proposed by DVB Consortium (http://www.dvb.org/) to support high usability in DRM [7, 10]. The user is able to use the digital content on any device, which belongs to her authorized domain, in any manner she likes. For instance, she can move the content freely or make and distribute replicas of the content within her authorized domain.

Nowadays, the users interact with each other more frequently in using the digital content. One of the user-to-user interaction is content *transfer*, as a gift, a loan or a resale. After transfer, the license (and the associated content) is bound to another user. Intuitively, this grants a user more freedom in using her digital content. There are potential benefits for the content providers as well. A content provider could distinguish between the costs of a license that can be transferred or cannot be transferred. Additionally, an increased distribution of the digital content by reselling might attract the users to buy original content from the content providers.

We envisage that the transfer operation in a DRM system especially in an authorized domain will become increasingly important. In this paper, we propose a theoretical approach and provide an implementation based on one of

D. Gollmann, J. Meier, and A. Sabelfeld (Eds.): ESORICS 2006, LNCS 4189, pp. 81–96, 2006.

the main DRM standards, namely Open Mobile Alliance (OMA) (`http://www.openmobilealliance.com`). We propose our approach based on OMA-DRM because OMA-DRM provides a full-fledged implementation to develop DRM systems on mobile devices. Thereby, by extending OMA-DRM to facilitate the transfer operation, we are able to develop a complete DRM system that is capable of supporting wide variety of usage scenarios.

Our approach improves user's experience of using the DRM-protected digital content in her authorized domain. As far as we know, this paper is the first proposal of a full-fledged design implementation of transfer operation in a DRM standard, namely OMA-DRM.

The remainder of the paper is organized as follows: Section 2 discusses our approach and its implementation in OMA-DRM. Section 3 analyzes the security of our approach. Section 4 describes our prototype. Section 5 briefly explains some related work. Finally, section 6 concludes the paper.

2 Our Approach

In this section, we elaborate on our approach to implement license and content transfer in OMA-DRM. There are several approaches available supporting transfer of license, as discussed in section 5. We propose an approach that can be seamlessly implemented in OMA-DRM at the same time fulfilling some reasonable requirements as listed in section 2.3.

2.1 Players

We identify four main players corresponding to the OMA-DRM v.2.0 [14], as shown in Figure 1, namely Compliance Server, Rights Issuer, and two users namely Giver and Taker. The Compliance Server acts as a trusted third party that proves the compliance of the devices a user possess in her domain. The Compliance Server has a direct (business) interaction with the Rights Issuer (RI).

The Rights Issuer manages Giver's and Taker's authorized domain. When a user adds a device in her domain, she has to initiate the ROAP Join Domain Protocol [14] from the Device or from a Website of a Rights Issuer that can send a trigger, etc. The management of OMA-DRM authorized domain is centralized, i.e., the rights issuer can keep a record of all the devices in the user's domain.

The Rights Issuer also generates and distributes licenses (and the associated digital content) to a user's device or domain. The Giver initiates the transfer of a license, and Taker receives the transferred license (and the associated digital content). Giver and Taker have their own domain. There are a number of devices whose compliance is checked and verified using proofs obtained from the Compliance Server. Transfer is performed between Taker's Device and Giver's Device, as shown in Figure 1.

As shown in Figure 2, a connected Device can reach the Rights Issuer by establish a network connection with the Rights Issuer directly, e.g. via wired or wireless connection. On the other hand, a unconnected Device can reach the

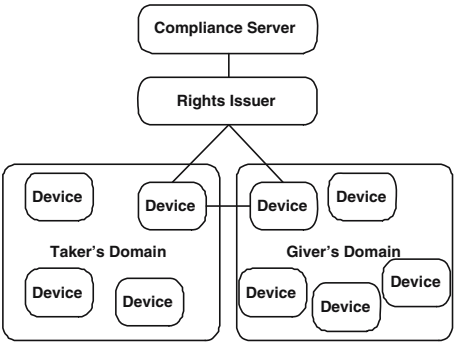

Fig. 1. The overview of the license transfer concept

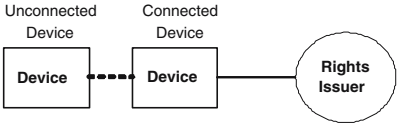

Fig. 2. The reachability of a device to a rights issuer

Rights Issuer via a connected Device, which acts as a proxy having a direct network connection to the Rights Issuer. This is as defined by OMA-DRM v.2.0 [14].

When a device is capable of reaching the rights issuer in real-time to perform transfer operation, we call it an *online transfer*. Otherwise, it is an *offline transfer*. The offline transfer can be performed when two devices can connect each other, e.g. via wired (e.g. USB) or wireless (e.g. Bluetooth) connection without connecting to the rights issuer. However, the rights issuer can still "participate" in offline transfer by using pre-defined usage rights (e.g. by specifying in the license that off-line transfer is not allowed).

2.2 Assumptions

We make the following assumptions under which our solution is valid:

- An OMA authorized domain has been established for the users. All devices belonging to the domain may maintain a replica of each domain-based license and therefore can render the domain-based digital content.
- Device compliance can be verified by the Compliance Server. After compliance verification, the device is trusted, i.e., it enforces the rights stated on the license correctly.
- All devices have OMA-DRM Agents [15].
- We assume that Giver's and Taker's domains are managed by the same Rights Issuer and the content stays under control of the Rights Issuer.
- We do not consider stateful license [14] in this paper.

2.3 Requirements

In order to support the rightful and secure transfer of licenses between users, we define the following requirements that have to be satisfied:

- Rights Issuer Control: The Rights Issuer shall be able to control the transfer operations, either in real-time or after the transfer is complete. The Rights Issuer should be able to keep track of the digital content before and after the transfer.
- Compliance Check of Participants: The Rights Issuer, the devices of Giver and Taker can achieve mutual compliance check to perform the transfer operations. This is only required in some specific cases, i.e., when transferring some secrets such as the content encryption keys.
- High Usability: The users, i.e., both Giver and Taker can perform the transfer operations with high convenience. The users do not need to connect their devices to the Rights Issuer in real-time (i.e., without network connection) to perform the transfer operations.
- Successful Completion: All transfer operations should be completed successfully. If errors occur during the transfer operation, the Giver, Taker and Rights Issuer will not suffer from any severe loss e.g. permanent loss of a license.

2.4 Four Steps

As shown in Figure 3, our approach has four steps. In this section, we elaborate on the protocols to achieve license and content transfer. In OMA-DRM, a license is called Rights Object (RO) [16]. Hereafter in this paper licenses will be referred to as Rights Objects, or RO. There are two types of RO, namely domain-based RO and device-based RO. A user can render the domain-based RO in every device that belongs to her domain, whereas the user can only render the device-based RO on a particular device.

Device Compliance Check. As mentioned in section 2.3, compliance check is only required for some specific cases. Both the Taker's and Giver's device connect to the Compliance Server via a Rights Issuer (RI) to obtain a proof of compliance, i.e., compliance token.

Online Certificate Status Protocol (OCSP) [11] is a method for determining the revocation status of X.509 certificates [9]. The mechanism requires a connection to an OCSP server (also called OCSP responder) in order to check the validity of a given certificate. OMA DRM clients (v.2.0) [15] already use this functionality to determine the revocation status of Rights Issuers.

Certificate Revocation Lists (CRLs) [4] are another mechanism that can be used to control the status of certificates. The advantage of OCSP over CRL is that it does not require the maintenance of long revocation lists in every device. An OCSP Responder (OR) returns a signed response indicating that the status of a certificate supplied in the request is 'good', 'revoked' or 'unknown'.

The compliance token is retrieved with the help of a RI as shown in Figure 3. It is assumed that the device has already registered with the RI with ROAP 4-Pass

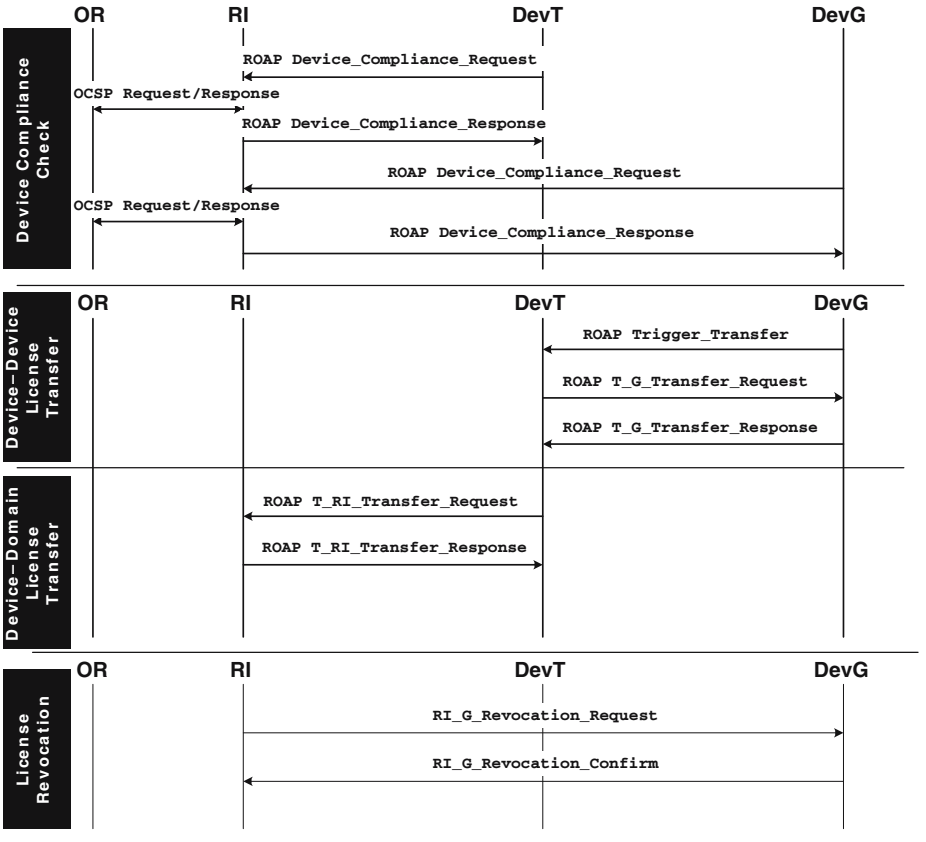

Fig. 3. OMA implementation of transfer

Registration Protocol [14]. The messages are similar to the ROAP Registration Request and Registration Response messages.

The processing at the RI will be exactly the same as for the ROAP registration protocol. The only difference is the certificate (chain) that is being validated. The ROAP registration protocol validates the RI certificate while the new ROAP device compliance protocol validates the device certificate. We elaborate on the messages of the protocol listed in Table 1:

M#1. The message is based on the ROAP Registration Request message. Since device and RI have already exchanged and validated their certificates at the ROAP Registration Protocol, it is not required to exchange this data again. The `trustedAuthorities` element is kept since the `DeviceCertificateChain` might differ from the RI certificate chain received at the ROAP 4-Pass Registration Protocol. Upon receipt of this message the RI will contact an OCSP responder as specified in the ROAP Registration protocol.

Table 1. ROAP protocol for Device Compliance Request, where the element DevX designates either Giver's or Taker's device

M#	Parties	ROAP Message
1	DevX → RI	`ROAP Device_Compliance_Request { trustedAuthorities }`
2	RI → DevX	`ROAP Device_Compliance_Response { DeviceCertificateChain, ocspResponse_DevX }`

M#2. The message is based on the ROAP Registration Response message. It includes a complete certificate chain for the device and the corresponding OCSP response, i.e., `ocspResponse_DevX`. The compliance token that is used in the Device–Device License Transfer is composed of the certificate chain and the OCSP response received in this message.

Device–Device License Transfer. To perform transfer, both the Taker and Giver's devices can be connected to each other (e.g. via wired connection with USB cable or wireless with Bluetooth). The Taker's device and Giver's device exchange messages, which include a valid compliance token. At the same time, the Taker's device checks its current status of reachability to the Rights Issuer. If the Taker's device is currently offline, the Giver's device transfers a device-based RO, i.e., one that the Taker can only access on her receiving device (for a period of time).

The permission for the Giver's device to issue a device-based RO for the Taker's device can be exclusively granted by the RI (e.g. through an additional usage rights specified in the original RO). The RO on the Giver's device is revoked temporarily through a process elaborated later. The RO replicas in the Giver's domain are then revoked permanently after the Taker requests for a domain-based RO from the RI.

Table 2 depicts the message exchange between the Taker's device (DevT) and the Giver's device (DevG) in transferring the RO.

M#1. The device DevG initiates the transfer operation by sending the ROAP Trigger Transfer message to DevT. The message includes the identity of the RO `RO_ID`, which indicates the RO that Giver wants to transfer to Taker.

M#2. After receiving the ROAP Trigger Transfer message, the device DevT sends a `ROAP T_G_Transfer_Request` message to the device DevG. The message must include the item `RO_ID` received from the device DevG earlier.

The message includes an indication of whether it requires a device-based RO it can use without contacting RI (typically DevT only sets this flag if it cannot reach RI in real-time – the decision is left to the device or the user if the device can reach RI and if it needs an RO for offline usage, respectively), i.e., `requestOfflineDeviceRO_DevT`. Compliance token must only be present if the item `requestOfflineDeviceRO_DevT` is set (i.e., `ocspResponse_DevT` and `certificateChain_DevT`).

Table 2. The contents of each ROAP message exchange in the protocol of Device–Device License Transfer

M#	Parties	ROAP Message
1	DevG → DevT	`ROAP Trigger_Transfer { RO_ID }`
2	DevT → DevG	`ROAP T_G_Transfer_Request { RO_ID, certificateChain_DevT, requestOfflineDeviceRO_DevT, ocspResponse_DevT }`
3	DevG → DevT	`ROAP T_G_Transfer_Response { original_RO_G, transferContext, device_RO_T, certificateChain_DevG, ocspResponse_DevG, certificateChain_RI, ocspResponse_RI }`

M#3. The device DevG constructs the message `ROAP T_G_Transfer_Response`. The message contains the original RO that is given away, i.e., `original_RO_G`. The message must contain the transfer context, which asserts that G transfers this RO to Taker and has or will revoke the RO replicas in G's domain (`transferContext`). The `transferContext` has the following form:

```
transferContext =
     { RO_ID, DevG_ID, DevT_ID, transferTime,
       requestOfflineDeviceRO_DevT
     }sig-DevG_PriKey
```

The item `transferTime` indicates the moment Giver creates the `transferContext` and is used as part of the revocation process. By making `transferTime` mandatory the protocol is limited to the Giver's devices that support OMA-DRM v.2.0 secure time.

The device DevG includes the items `device_RO_T` and `certificateChain_DevG`, `ocspResponse_DevG` if the message M#2 includes the `requestOfflineDeviceRO_DevT` flag. The device DevG verifies that the original RO allows the creation of `device_RO_T` (which is decided by the RI when Giver acquires the RO from RI). The device DevG takes its own capabilities into account such as its capability to create `device_RO_T`. The device DevG also verifies in this case that DevT's request is authentic, and verifies the compliance of DevT, both using `certificateChain_DevT` and `ocspResponse_DevT`.

If DevT requires a `device_RO_T` for offline use, but it cannot be honoured due to lack of capabilities then `errorNoOfflineROCapabilities` is returned. If the reason is based on policy then `errorPolicyDeniesOfflineRO` is returned. If DevT prefers a `device_RO_T` for offline use but it cannot be honoured due to any reason then the protocol continues with the next message, but without `device_RO_T` and any other parameter that is only required for offline use.

In case the item `device_RO_T` is present DevT verifies that also the other optional fields, namely `certificateChain_DevG`, `ocspResponse_DevG`, `certificateChain_RI`, and `ocspResponse_RI` are present.

In case `device_RO_T` must be generated, the device DevG processes the original RO, i.e., `original_RO_G` to make it a RO to Taker, which Taker can only use on

DevT. DevG adds the limitations in the rights expression of device_RO_T if the transfer policy prescribes this. This policy can be hard-coded and prescribed by for example CMLA [15], or it is present as part of original_RO_G (stated in the REL [16], which is defined by RI).

Upon receipt of the M#3 DevT verifies that the DevT receives original_RO_G and transferContext that correspond with the requested RO_ID. If DevT requires device_RO_T but does not receive it within a specified timeout then DevT aborts the protocol. This leaves both the Giver's device and Taker's device in a state that equals to the state before initiation of the transfer protocol.

Upon receipt of the M#3 that includes the device-based RO device_RO_T, the device DevT attempts to access the content according to the following rules. The device DevT verifies the authenticity and integrity of the item original_RO_G using the items certificateChain_RI and ocspResponse_RI. The device DevT verifies the authenticity and integrity of the message transferContext using certificateChain_DevG and ocspResponse_DevG. The device DevT verifies the authenticity and integrity of device_RO_T using certificateChain_DevG and ocspResponse_DevG.

The device DevT verifies that RO_ID of original_RO_G and device_RO_T are identical. The device DevT evaluates the rights expression of device_RO_T as defined by OMA-REL [16] and must in addition check that this rights expression is not broader than the rights expression contained in original_RO_G. In case of successful evaluation DevT uses the content encryption key (CEK) of device_RO_T to decrypt the content.

Device–Domain License Transfer. Device–Domain License Transfer is performed for the Taker to obtain a domain-based RO from the Rights Issuer. The Taker must present the transfer context to the RI. The RI can use the transfer context to further process the revocation on the Giver's domain.

Table 3 depicts the message exchange between DevT and RI to perform Device–Domain License Transfer.

M#1. The device DevT sends the ROAP T_RI_Transfer_Request message to RI, which includes some message items received from the device DevG at the Device–Device License Transfer operation (Table 2). The ROAP T_RI_Transfer_Request message is very similar to an OMA DRM v.2.0 ROAP acquisition request message. The message must include the message transferContext, and the Giver's original RO, i.e., original_RO_G.

The message also includes the Taker's domain information, i.e., domainInfo_T. The device DevT should have user consent which domainInfo_T to use if multiple options are available. As stated in section 2.2, we assume that the domains of G and T are managed by the same RI. Thus, the device DevT can initiate M#1 (in Table 3). Note that if T has no prior relation with RI or if it has no domain context for this RI, the device DevT must initiate the required ROAP Registration and/or Join Domain protocol(s) as specified by OMA-DRM v.2.0 before sending M#1.

Table 3. The content of each ROAP message exchange in the protocol of Device–Domain License Transfer

M#	Parties	ROAP Message
1	DevT → RI	`ROAP T_RI_Transfer_Request{ original_RO_G, transferContext, certificateChain_DevG, domainInfo_T}`
2	RI → DevT	`ROAP T_RI_Transfer_Response{ domain_RO_T, certificateChain_RI, ocspResponse_RI}`

M#2. The Rights Issuer RI processes `original_RO_G` and produces `domain_RO_T`, i.e., the domain-based RO dedicated to the Taker's domain. Before generating M#2, RI should verify the authenticity and integrity of `ROAP T_RI_Transfer_Request`, `original_RO_G`, `transferContext` including verification of the revocation status of the signers.

RI should also verify that `transferContext` indicates the device DevT as the recipient of the transfer and signee of the request. RI should also verify that DevT is a member of the indicated domain in `domainInfo_T`. RI uses `transferContext` to determine if the RI wants to deliver the domain-based RO.

If RI decides not to honour the request it must return an error indicating the type, e.g. `errorWaitForSuccessfullRevocation` to indicate that revocation must be finished first, `errorTransferFailed` to indicate that transferring failed persistently. Otherwise, RI must create a RO as it would do for the OMA DRM v.2.0 ROAP Acquisition Protocol and send this in `ROAP T_RI_Transfer_Response` message to the device DevT. The device DevT must process this message as if it concerns a ROAP Acquisition Protocol, including verification of authenticity, integrity and compliance of `domain_RO_T`, RI, etc.

License Revocation. In this section, we describe a best-effort license revocation mechanism. The purpose of license revocation is to revoke the license replicas in a user's domain before or after the transfer of the license and the associated digital content. License revocation makes the license (and replicas thereof) unavailable to the current owner's domain. This ensures that the transferred license becomes unavailable to each device in the Giver's domain. Since the completion of the revocation process cannot be instantaneous due to the offline devices in the Giver's domain, it is required to ensure the completion of revocation within a time interval that would be set by the RI.

To support a correct license revocation, all devices must maintain a copy of the license revocation list, that each device uses to check the validity of stored licenses. The license revocation list indicates the domain licenses that cannot be further used within that domain. This may raise the storage problem of huge license revocation list. However, with the rapidly increasing capacity of current storage space with the relatively reasonable price, this can be solved to a certain extent. We can also apply some optimization techniques, such as a compression method to efficiently store the revocation list on small devices. This, however, we leave outside the scope of this paper.

The license revocation process involves updating the revocation list stored on the device. The update is triggered by all domain devices when they connect to the RI, through a revocation list refresh request message. The actual update can be done either by allowing the Rights Issuer to push the new revocation list, or to piggyback the new revocation list to the devices when they request for a fresh compliance token.

During the Device–Device License Transfer operation discussed earlier, the revocation is initiated by the Giver's device (DevG). The item `transferContext` is generated at DevG, asserting that DevG has transferred the RO and has/will revoke the RO replicas in the Giver's domain. Both the DevG and Taker's device (DevT) possess the transfer context.

When either device connects to the RI, the transfer context is sent to RI as a notification that the RO has been transferred from Giver to Taker. Thereby, RI can initiate the license revocation – the RI sends a request for refreshing their RO revocation list (RORL) to all the Giver's devices in her domain (that happen to be online), i.e., `RI_G_Revocation_Request`.

All devices receiving the revocation request must replace their RORLs with the new one and where possible must delete the newly revoked RO. Other devices that are offline will receive the revocation request from the RI whenever they are online. Until then, the RO remains usable on those devices. However, the RI notifies Giver which of her device has not updated the RORL, as the RI keeps a record of devices that belong to her domain.

Until the new RORL is updated on all the devices in the domain (for some specific time) the RI maintains the following state information for the Giver's domain, as shown in Table 4. The timestamps (`Time_n`) is used to indicate the last time the corresponding device updates the RORL.

Each device receiving the request from the RI for revocation must update its RORL and must respond with the message `G_RI_Revocation_Confirm`, so that the RI can update the state information (as shown in Table 4). The message structure of the request and confirm messages are listed in Table 5. Note that the label DevG here indicates each of the devices belong to the Giver's domain.

M#1. The `RI_G_Revocation_Request` message is initiated by RI to DevG, i.e., one of the devices in Giver's domain, which connects to the RI to request fresh compliance token. When the user requests for a fresh compliance token, the user's device initiates the `RORLRefreshTrigger`. The `RI_G_Revocation_Request` message includes a new RORL, i.e., `RORL_new`, signed by the RI for integrity and authenticity.

M#2. Upon receipt the message, DevG updates the stored RORL, by replacing the old one with the newly received RORL from the `RI_G_Revocation_Request` message. To confirm the revocation, the Giver's device DevG responds to the RI with the `G_RI_Revocation_Confirm` message. The message includes the version, identifier, and update time of the RORL, i.e., `RORL_version`, `RORL_identifier`, and `RORL_time`, respectively. RI verifies if the `RORL_version`, `RORL_identifier`, and `RORL_time` received are correct to ensure the revocation has been done properly.

Table 4. State information of the Giver's domain maintained by the RI

Domain Composition	Active RORL
Current RI RORL	`RORL(Time_n)`
Device 1	`RORL(Time_n)`
...	...
Device n	`RORL(Time_some-time-ago)`

Table 5. The message structure of revocation request and confirm

M#	Parties	ROAP Message
1	RI → DevG	`RI_G_Revocation_Request { { RORL_new }sig-RI_PriKey }`
2	DevG → RI	`G_RI_Revocation_Confirm { { RORL_version, RORL_identifier, RORL_time }sig-DevG_PriKey }`

RI checks if all the recorded devices (see Table 4) have already updated the RORL.

We can also implement revocation locally prior to the transfer. In other words, the user *must* revoke all the license replicas within her domain before transferring. This can be expanded easily with our approach – the user must connect all devices in her domain to the Rights Issuer for revocation prior to the transfer. However, this is more inconvenient than our best-effort approach from the user's perspective. The user normally cannot anticipate which content she wants to transfer especially when the user cannot connect the Rights Issuer.

3 Threat Analysis

In this section, we analyze several threats to our proposed solution. Section 3.1 and section 3.2 are considered as a form of replay attacks.

3.1 Store on Non-compliant Storage

The user can move a license (or its replica) to a (non-compliant) storage medium. After the transfer and revocation, the user moves the license back to the device in her domain.

We propose using a license revocation list (or rights object revocation list, i.e., RORL) to keep a record of *all* the revoked license (see section 2.4). However, this solution seems to be confined by some limitations, which we believe can be overcome:

– The RORL may grow to a size that is unmanageable by devices with limited resources. Therefore, we need an optimization mechanisms to manage the RORL.

– The device must be *obliged* to re-connect the rights issuer for an up-to-date RORL. We must make sure that without an up-to-date RORL, the device (with the DRM agent built-in) cannot further distribute and/or render any content.

3.2 Illegitimate Transfer Proliferation

The user can perform *multiple* offline transfers from every device in her domain to all the devices of another user's domain. For instance, the Giver can use Device_1 to transfer a license to Taker's Device offline. After that, without connecting to the the Rights Issuer, the Giver could use Device_2 to transfer the replica of the transferred license to Taker's *another* Device. Instead of the same Taker (say Taker_1), the user can offline transfer to another Taker (Taker_2) device. These threats are caused by the license revocation approach discussed in section 2.4.

As discussed in section 2.4, to perform the transfer, the device must request a fresh compliance token from the Rights Issuer. Additionally, the Giver and Taker must connect to the Rights Issuer for the new digital content, etc. on a regular basis. Thus, the Rights Issuer can detect this form of replay attack if there are more than one transfer context for the same license has been generated (Note: only one transfer context is generated to transfer the license). Furthermore, there is a limited number of devices allowed in a user's domain (as defined by OMA-DRM [15]), which further alleviates this threat.

To tackle the threat of offline transfer to Taker_1 and Taker_2, if either Taker contacts the Rights Issuer requesting for the domain-based license, the Rights Issuer can notice there are two *same* transfer context. Thereby, the Rights Issuer can then detect the Giver's unauthorized multiple offline transfer. Moreover, the device-based license can only be rendered in within a short period.

Additionally, the Rights Issuer in our approach (based on OMA-DRM architecture [15]) keeps a record of all the available devices in the user's domain. Therefore, it is easy for the Rights Issuer to check if a particular device has been offline for a long period of time.

4 Prototype

To serve the purpose of proof-of-concept, a prototype is built implementing the proposed solution for content transfer. In the prototype, we have defined a use case as follows: Aaron and Jasmine love collecting music tracks from the 1950s. Each has a mobile terminal that contains the some music tracks and the associated rights. They want to trade some tracks using their mobile. They connect the mobiles and swap the content. After trading, both Aaron and Jasmine are not able to access their original content and licenses anymore.

Figure 4 illustrates part of the prototype that facilitates transfer operation between two mobile phones. We use the state-of-the-art broadband network protocols, Bluetooth protocols, Near-Field Communication (NFC) and General Packet Radio Service (GPRS) to facilitate the communications between these components. There are three main components in the prototype, namely:

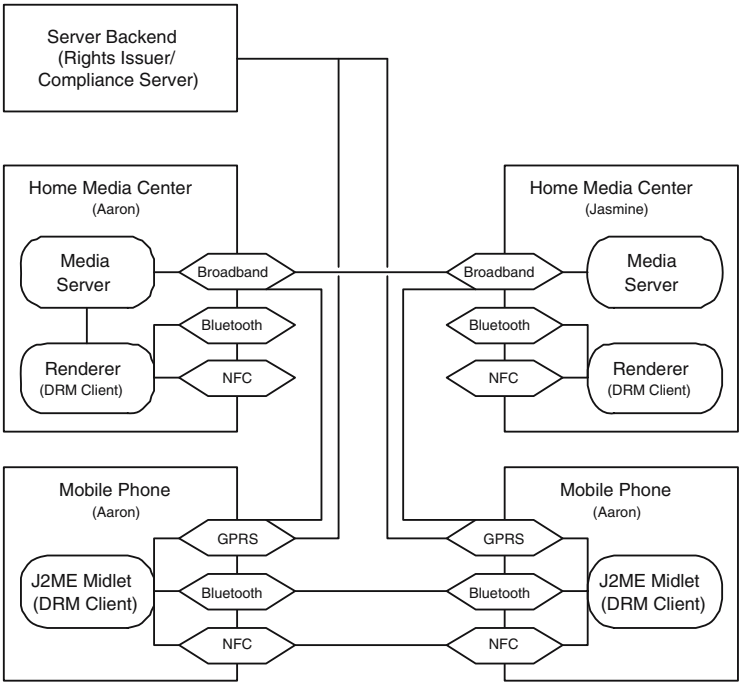

Fig. 4. The architecture of the prototype that implements transfer

- Backend Server: It implements the Rights Issuer and Compliance Server.
- Home Media Center (HMC): It implements the Media Server, which manages the user's content; and Renderer, which implements the OMA DRM Client.
- Mobile Phone: It implements J2ME Midlet, which is an OMA DRM Client.

Aaron and Jasmine each has his/her own Home Media Center and Mobile Phone. When Aaron initiates transfer operation to Jasmine on their mobile phones, the Device–Device License Transfer operation is initiated. After the Device–Device License Transfer operation, Jasmine can only render the transferred content on her Mobile Phone. She can connect her mobile phone to the Backend requesting for a domain-based RO so that she can render the content on her HMC. If the request is granted, the Backend contacts Aaron's HMC (assuming now that Aaron's HMC is connected) to initiate the revocation process. As the Backend Server keeps a record of all Aaron's devices in his domain, thus, the Backend Server can contact Aaron (e.g. via emails) for updating the revocation list on all his devices.

5 Related Work

The transfer rights can already be expressed in the available rights expression languages (REL) [6], such as XrML [5], ODRL [8], OMA-REL [16], and Licens-

eScript [2]. However, the underlying DRM architecture enforcing the aforementioned RELs does not support explicitly transfer.

Nair et al. [12] have proposed a set of protocols, which can achieve redistribution of the content from one user to another. However, they merely focus on the device-to-device distribution, i.e., they do not consider the concept of authorized domain. They also propose several payment methods for the redistribution, which considers the benefits of the rights issuer. However, the rights issuer does not participate in the process of redistribution in real-time.

Conrado et al. [3] have proposed an approach that supports anonymous transfer of the digital content (and the associated license) to another user. To perform transfer operation, the user must initiate a connection with the rights issuer requesting an anonymous license, which is a license that is not associated to any user. The user can thus transfer the anonymous license to another user. Thereby, the privacy of the user who receives the license is protected from the rights issuer. Prior to transfer, the user must first revoke all the license replicas in her authorized domain, i.e., pre-revocation.

Our approach does not consider privacy protection for user due to the requirement of Rights Issuer Control (as described in section 2.3). The Rights Issuer would like to keep track of the digital content before and after the transfer. However, our approach can be easily customized to support pre-revocation.

6 Conclusions

Digital Rights Management (DRM) attempts to control the user's access over the digital content, protecting the benefits of the rights issuers. The drawback is that the users are constrained in the usage of digital content. Thus, the concept of authorized domain is proposed to grant the users more freedom - the user is allowed to make replicas of the license (and the associated digital content) and use the content on all devices that belong to her domain.

The transfer operation is useful from the user's and also the rights issuer's perspective. The user can transfer a license and the associated content to another user as a gift, loan or resale. However, license transfer between two users becomes complicated due to the user's freedom granted by the concept of authorized domain. There are a number of license replicas in a user's domain, which have to be taken care of for the transfer operations.

We propose an approach for the transfer of license and content. The approach facilitates the compliance check of devices performing transfer, the secure license transfer between users' devices, and the efficient license revocation in the user's domain. We design a set of protocols, which extend the OMA-DRM interacting the Giver, Taker and Rights Issuer. The proposed approach is able to fulfill the following requirements:

– Rights Issuer Control: The Rights Issuer can decide if the Giver can issue a device-based license by stating the transfer rights on the license for offline transfer. When the Taker requests for a domain-based license for the received content, the Rights Issuer can determine to grant the Taker's request.

– Compliance Check of Participants: The Giver's and Taker's devices must possess an up-to-date compliance token to perform offline transfer. Thus, the devices must connect to the Rights Issuer to request for an up-to-date compliance tokens on a regular basis for offline transfer.
– High Usability: The Giver and Taker can perform offline transfer when there is no network connection, and the Taker can use the transferred license on the receiving device (within a limited time period).
– Successful Completion: We make the protocols for the transfer operations context-free. If errors happen, the user can resume the protocols by resending the messages (within a timeout).

As discussed in section 2, we assume that the Giver and Taker's domain are maintained by the same Rights Issuer. We envision the possibility to separate the domain management from the Rights Issuer, which may provide more business opportunities and high scalability. We consider this as our future work.

Acknowledgement

We would like to thank James Irwin, Tim Kerins, Milan Petkovic, Andrew Tokmakoff, Koen Vrielink and Tim Wright for their valuable comments.

References

[1] Chong, C.N., van Buuren, R., Hartel, P.H., Kleinhuis, G.: Security attribute based digital rights management (SABDRM). In Boavida, F., Monteiro, E., Orvalho, J., eds.: Joint Int. Workshop on Interactive Distributed Multimedia Systems/Protocols for Multimedia Systems (IDMS/PROMS). Volume 2515 of LNCS., Springer-Verlag (2002) pp. 339–352
[2] Chong, C.N., Corin, R., Etalle, S., Hartel, P.H., Jonker, W., Law, Y.W.: LicenseScript: A novel digital rights language and its semantics. In Ng, K., Busch, C., Nesi, P., eds.: 3rd International Conference on Web Delivering of Music (WEDELMUSIC), Los Alamitos, California, United States, IEEE Computer Society Press (2003) pp. 122–129
[3] Conrado, C., Petkovic, M., Jonker, W.: Privacy-preserving digital rights management. In: Secure Data Management 2004. Volume 3178., Springer-Verlag (2004) pp. 83–99
[4] Feghhi, J., Williams, P.: Digital Certificates: Applied Internet Security. Addison-Wesley (1998)
[5] Guo, H.: Digital rights management (DRM) using XrML. In: T-110.501 Seminar on Network Security 2001. (2001) Poster paper 4
[6] Guth, S.: Rights expression languages. In Becker, E., Buhse, W., Günnewig, D., Rump, N., eds.: Digital Rights Management: Technological, Economic, Legal and Political Aspects. Volume 2770 of LNCS., Springer-Verlag (2003) pp. 101–112
[7] Heuvel, S. van den, Jonker, W., Kamperman, F., Lenoir, P.: Secure content management in authorised domains. In: Int. Broadcasting Convention (IBC), Amsterdam, The Netherlands, Broadcastpapers Pty Ltd, PO Box 259, Darlinghurst, NSW, 1300, AUSTRALIA (2002) pp. 467–474

[8] Iannella, R.: Open digital rights management. In: World Wide Web Consortium (W3C) DRM Workshop. (2001) Position paper 23
[9] ITU: ITU-T Recommendation X.509, Information Technology, Open Systems Interconnection, The Directory: Authentication Framework. International Telecommunication Union, Place des Nations 1211 Geneva 20 Switzerland. (1997)
[10] Koster, R. P., Kamperman, F., Lenoir, P., Vrielink, K.: Identity based drm: Personal entertainment domain. In: Communications and Multimedia Security (CMS), 9th IFIP TC-6 TC-11 International Conference. Volume 3677 of LNCS., Springer-Verlag (2005) pp. 42–54
[11] Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C.: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol (OCSP). Technical Report RFC 2560, VeriSign, CertCo, ValiCert, My CFO, and Entrust Technologies (1999)
[12] Nair, S., Popescu, B., Gamage, C., Crispo, B., Tanenbaum, A.: Enabling drm-preserving digital content redistribution. In: CEC 2005. Seventh IEEE International Conference on E-Commerce Technology (CEC 2005), IEEE (2005) pp. 151–158
[13] Niehüser, L.: The right to sell. INDICARE Monitor **1**(3) (2004)
[14] OMA: DRM specification: Candidate version 2.0. Technical Report OMA-TS-DRM-DRM-V2_0-20060303-A, Open Mobile Alliance (2004)
[15] OMA: DRM architecture: version 2.0. Technical Report OMA-DRM-ARCH-V2_0-20060303-A, Open Mobile Alliance (2004)
[16] OMA: DRM rights expression language v.2.0: version 2.0. Technical Report OMA-TS-DRM-REL-V2_0-20060303-A, Open Mobile Alliance (2005)
[17] Rosenblatt, B., Trippe, B., Mooney, S.: 4. In: Digital Rights Management: Business and Technology. M&T Books, New York, United States (2002)